The Hong Kong Polytechnic University

Department of Computing

Energy-Efficient Interference-Free Link Scheduling in Wireless Sensor Networks

by

Junchao MA

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

June 2013

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

[illegible]	(Signed)
MA Junchao	(Name of student)

Abstract

Wireless sensor networks (WSNs) consist of thousands of tiny, inexpensive and low-power sensor nodes that perform collaborative tasks and organize themselves into multi-hop networks. These sensor nodes are deployed over a given area, including both the terrestrial and underwater environments. Consequently, WSNs can be divided into terrestrial wireless sensor networks (TWSNs) and underwater wireless sensor networks (UWSNs). In this thesis, we mainly focus on the design of energy-efficient interference-free link schedulings in WSNs. In a link scheduling, each transmission link is assigned a set of time slots such that it can transmit without interferences.

In TWSNs, sensor nodes use radio waves to communicate, and the major source of energy wastage is the idle listening state in the radio modules. To reduce the energy consumption, sensors are generally scheduled to sleep when the radio is not in use. In a traditional sleep scheduling, however, sensors have to start up numerous times in a period, and thus consume extra energy due to the state transitions (e.g. from the sleep state to the listening state or transmitting state). We first identify a novel energy-efficient interference-free link scheduling problem called contiguous link scheduling, in which links incident to one node are scheduled together to obtain consecutive time slots so that the node needs to start up only once to monitor the channel in a period. We prove that the problem is NP-complete, and propose efficient centralized and distributed algorithms that use time slots at most a constant factor of the optimum in both homogeneous and heterogeneous networks. Our proposed distributed scheduling with efficient delay algorithm can also efficiently reduce the network delay. Extensive simulations are conducted to demonstrate the effectiveness of our proposed algorithms.

To further reduce the frequency of state transitions in TWSNs, we address a novel energy-efficient interference-free link scheduling problem called compact wakeup scheduling. In the scheduling, a node needs to wake up only once to communicate bidirectionally with all its neighbors. However, not all communication graphs have valid compact wakeup schedulings, and it is NP-complete to decide whether a valid compact wakeup scheduling exists for an arbitrary graph. In particular, trees and grid graphs, which are commonly used in WSNs, have valid compact wakeup schedulings. Polynomial-time algorithms using the optimum number of time slots in a period are proposed for these topologies. The simulation results illustrate the efficiency of our proposed algorithms.

In UWSNs, acoustic communication is commonly used unlike that in TWSNs. The long propagation delay of acoustic signals causes the spatio-temporal uncertainty, which makes the link scheduling in UWSNs a challenging problem. To describe the propagation delays of the transmission links and deal with the spatio-temporal uncertainty, we construct a so-called slotted spatio-temporal conflict graph. We propose centralized and distributed scheduling algorithms with performance guarantee to the optimum solutions under both unified and weighted traffic load scenarios. In the weighted traffic load scenario, we consider the scheduling with and without the consecutive constraint. Simulations validate our theoretical results, and show the efficiency of our proposed algorithms.

Publications

Journal Papers

1. **Junchao Ma**, Wei Lou, and Junmei Yao, “On Providing Interference-Aware Spatio-Temporal Link Scheduling for Long Delay Underwater Sensor networks”, submitted to *IEEE Transactions on Computers (TC).*

2. **Junchao Ma**, Wei Lou, and Xiang-Yang Li, “Contiguous Link Scheduling for Data Aggregation in Wireless Sensor Networks”, accepted to appear in *IEEE Transactions on Parallel and Distributed Systems (TPDS).*

3. Zhibo Wang, Wei Lou, Zhi Wang, **Junchao Ma**, and Honglong Chen, “A Hybrid Cluster-Based Target Tracking Protocol for Wireless Sensor Networks”, in *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 494863, 16 pages, 2013.

4. **Junchao Ma** and Wei Lou, “Interference-Free Wakeup Scheduling with Consecutive Constraints in Wireless Sensor Networks”, in *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 525909, 17 pages, 2012.

5. Xianlong Jiao, Wei Lou, **Junchao Ma**, Jiannong Cao, Xiaodong Wang, and Xingming Zhou, “Minimum Latency Broadcast Scheduling in Duty-Cycled Multi-Hop Wireless Networks”, in *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 1, pp. 110-117, 2012.

6. Xianlong Jiao, Wei Lou, Xiaodong Wang, **Junchao Ma**, Jiannong Cao, and Xingming Zhou, "On interference-aware gossiping in uncoordinated duty-cycled multi-hop wireless networks", in *Elsevier Ad Hoc Network Journal*, 2011.

Conference Papers

1. **Junchao Ma** and Wei Lou, " Interference-aware Spatio-Temporal Link Scheduling for Long Delay Underwater Sensor Networks", in the *8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Salt Lake City, Utah, USA, June 2011.

2. **Junchao Ma** and Wei Lou, " Compact Wakeup Scheduling in Wireless Sensor Networks", in the *IEEE Global Communications Conference (GLOBECOM)*, Miami, Florida, USA, December 2010.

3. Xianlong Jiao, Wei Lou, Xiaodong Wang, **Junchao Ma**, Jiannong Cao, and Xingming Zhou, "Interference-Aware Gossiping Scheduling in Uncoordinated Duty-Cycled Multi-hop Wireless Networks", in the *International Conference on Wireless Algorithms, Systems and Applications (WASA)*, Beijing, China, August 2010.

4. Zhibo Wang, Wei Lou, Zhi Wang, **Junchao Ma**, and Honglong Chen, "A Novel Mobility Management Scheme for Target Tracking in Cluster-Based Sensor Networks", in the *6th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Santa Barbara, California, USA, June 2010.

5. Xianlong Jiao, Wei Lou, **Junchao Ma**, Jiannong Cao, Xiaodong Wang, and Xingming Zhou, "Duty-Cycle-Aware Minimum Latency Broadcast Scheduling in Multi-hop Wireless Networks", in the *30th International Conference on Distributed Computing Systems (ICDCS)*, Genoa, Italy, June 2010.

6. Zhen Jiang, **Junchao Ma**, Wei Lou, and Jie Wu, "A Straightforward Path Routing in Wireless Ad Hoc Sensor Networks", in the *IEEE ICDCS workshop WWASN*, Montreal, Quebec, Canada, June 2009.

7. **Junchao Ma**, Wei Lou, Yanwei Wu, Xiang-Yang Li, and Guihai Chen, "Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks", in the *28th Annual IEEE Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, April 2009.

8. Honglong Chen, Wei Lou, **Junchao Ma**, and Zhi Wang, "TSCD: A Novel Secure Localization Approach for Wireless Sensor Networks", in the *Second International Conference on Sensor Technologies and Applications (SensorComm)*, Cap Esterel, France, August 2008.

9. Zhen Jiang, **Junchao Ma**, Wei Lou, and Jie Wu, "An Information Model for Geographic Greedy Forwarding in Wireless Ad-Hoc Sensor Networks", in the *27th Annual IEEE Conference on Computer Communications (INFOCOM)*, Phoenix, USA, April 2008.

Acknowledgements

I would link to express my appreciation to all those who helped me during my Ph.D. study. First and foremost, I would like to take this opportunity to thank Dr. Wei Lou, my chief supervisor, for his continuous encouragement and rigorous supervision of my research. He always trains me to be a good researcher, such as how to find research issues, how to solve thorny problems, how to express the ideas clearly, and how to write high-quality academic papers. His motivation, enthusiasm, patience, and immense knowledge make him a terrific supervisor, and inspire me a lot. What I have learned and experienced will always help and encourage me in the future.

Second, I would like to thank the members of my thesis committee, Prof. Xiaohua Jia, Dr. Alvin Chan and Dr. Yu Wang for their valuable comments. I am grateful to my co-supervisor, Prof. Jiannong Cao, for his encouragement and advice. I would like to thank Dr. Xianlong Jiao and Dr. Zhen Jiang for our collaborations on the research. We explored ideas and wrote papers together. I am also very grateful to all my co-authors, Dr. Honglong Chen, Zhibo Wang, Dr. Zhi Wang and Prof. Xiangyang Li, for their insightful comments and constructive suggestions.

Third, I would like to convey my thanks to Jin Zhang, Tao Xiong, Junmei Yao, Xice Sun, Libin Yang, Nianbo Liu, Bing Bai, Lei Wang, Yanli Cai and Cheng Jia of our research group. I thank them for their kind help in my research, such as interesting discussions and suggestions on my work. I also would like to thank Qingjun Xiao, Yuhong Feng, Gang Yao, Daqiang Zhang, Kai Bu and Binbin Zhou for sharing with me the pain and pleasure of study at the university. Furthermore, I would like to thank all my teachers in my study, especially Dr. Bin Xiao, Dr. Dan Wang and Dr. Zhijun Wang, from whom I have learned a lot.

Last but not least, I would link to thank my family, including my parents, my brother and my sister. I thank them for their love, understanding, and support. Their encouragement has convinced me that I can achieve anything I put mind to. Without them, this work would not have been possible.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BFS	Breadth First Search
CDMA	Code Division Multiple Access
CIST-U	Centralized Spatio-Temporal Link Scheduling with Unified Traffic Load
CIST-W	Centralized Spatio-Temporal Link Scheduling with Weighted Traffic Load
CIST-WC	Centralized Spatio-Temporal Link Scheduling with Weighted Traffic Load under the Consecutive Constraint
CSMA	Carrier Sense Multiple Access
DAG	Directed Acyclic Graph
DIST-U	Distributed Spatio-Temporal Link Scheduling with Unified Traffic Load
DIST-W	Distributed Spatio-Temporal Link Scheduling with Weighted Traffic Load
DIST-WC	Distributed Spatio-Temporal Link Scheduling with Weighted Traffic Load under the Consecutive Constraint
FDMA	Frequency Division Multiple Access
GPS	Global Positioning System
LPL	Low-power Listensing
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network
MIS	Maximal Independent Set
PLL	Phase-locked Loop
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
SIDS	Sudden Infant Death Syndrome
SINR	Signal to Interference-Plus-Noise Ratio
S-STC	Slotted Spatio-temporal Conflict
STC	Spatial-temporal Conflict
TDMA	Time Division Multiple Access

TTL	Time-to-live
TWSN	Terrestrial Wireless Sensor Network
UWSN	Underwater Wireless Sensor Network
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

The main objective of this research is to investigate energy-efficient interference-free link schedulings in wireless sensor networks (WSNs), and design novel efficient scheduling algorithms. In this chapter, we first introduce the background of WSNs in Section 1.1. Then we describe the link scheduling issue in Section 1.2. In Section 1.3, we explain the motivations of this thesis. In Section 1.4, we summarize the main contributions of this research work. Finally, the organization of this thesis is presented in Section 1.5.

1.1 Wireless Sensor Networks

In recent years, wireless sensor networks (WSNs) [65, 87] have received a lot of interest in both academia and industry. WSNs consist of a variable number of small, inexpensive and low-powered sensor nodes, which have the capabilities of sensing, processing and communication. The sensor nodes may be equipped with various sensing devices to measure the different physical phenomena, such as temperature, humidity, vibration, light intensity and acidity. By organizing themselves, these sensor nodes form a wireless multi-hop radio network.

WSNs, in general, can be classified into two broad categories [58], terrestrial wireless sensor networks (TWSNs) and underwater wireless sensor networks (UWSNs). TWSNs are deployed in terrestrial areas, such as buildings, factories and forests, while UWSNs are deployed in underwater areas, such as lakes, rivers and oceans.

1.1.1 Terrestrial Wireless Sensor Networks

In TWSNs, sensor nodes are typically powered by limited non-rechargeable batteries, and it is often difficult and expensive to replace the batteries once they are deployed. When the battery of a sensor node runs down, it will die and disconnect from the network. In a multi-hop sensor network, each node plays the dual role of data source and forwarder. The run-down of batteries in few nodes can affect the network topologies significantly and even affect the network connectivity. Therefore, increasing the battery life time is a prime consideration of TWSNs [98], which differentiates the network design from other wireless networks, such as WLAN [30], WPAN [49], WMN [6] and MANET [50].

In TWSNs, sensor nodes use radio waves for data transmission and the maximum energy consumption of a sensor node is consumed by the radio module in the data communication, which involves transmit, listen and receive. Note that sensors not only consume different amounts of energy in different states (transmit, receive, listen and sleep), but also consume energy for state transitions (e.g. from the sleep state to the listening state or transmitting state).

TWSNs also have other basic features that are different from other wireless networks, which are listed as follows:

- Sensor nodes are densely deployed;

- Sensor nodes have limited computational capacities and memory;
- Sensor nodes collaborate to perform a common task;
- The communication range of sensor nodes is short and thus multi-hop routing is necessary;
- The topology of TWSNs changes frequently due to both node failures and mobilities.

1.1.2 Underwater Wireless Sensor Networks

More than two thirds of the Earth's surface is covered by water, it is desirable to study UWSNs in order to explore and observe the ocean. In the underwater environment, radio waves decay rapidly. For example, the underwater transmission range of Berkeley Mica2 Motes [29] is 120 cm at 433 MHz in experiments [4]. Long-wave radio can be used underwater, but it requires large antennae and high transmission power. Optical communication using light waves has also been investigated in [112]. As light is strongly absorbed and scattered underwater, transmitting optical signals requires either high power or high precision laser beams. Thus, acoustic communication is commonly used in underwater networks due to the low attenuation of acoustic waves.

The speed of acoustic waves depends on the water properties of temperature, salinity and pressure [74]. The propagation speed of acoustic waves is approximately 1500 m/s, which is several orders of magnitude slower than the 3×10^8 m/s wireless propagation. The transmission range of acoustic waves (2-4 km) is much larger than that of radio waves (150 m), thus, the propagation delay can be quite long in UWSNs. In contrast, the propagation delay is negligible in terrestrial networks. This unique feature is the central problem in the network design of UWSNs.

The data rate of underwater sensors depends on both transmission range and frequency [31]. For long-range communication (e.g., several kilometers) in UWSNs, the frequency of most acoustic systems operates below 30 kHz. Since the frequency band of acoustic channels is much narrower than that of terrestrial sensors, the data rate of underwater sensors is much smaller, and thus the packet transmission time is much longer.

In summary, UWSNs are significantly different from TWSNs in the aspects of low propagation speed, long transmission range, and limited bandwidth due to the underwater acoustic communication channel. Moreover, UWSNs are usually with the feature of three-dimensional topology. These characteristics present a challenge to the researchers in the field of UWANs.

1.1.3 Applications

By collecting, processing and spreading data in WSNs, information can be easily obtained from anywhere at every time. Thus, WSNs can help the end user to have a better understanding of the environment. Based on such a technological vision, WSNs have a multitude and ever-increasing number of applications. Figure 1.1 shows an overview of the applications of WSNs, we can see that WSNs can be applied to areas as diverse as environmental monitoring, military defense, health care, smart home, as well as business and industry. Next, we give a brief survey of the applications of WSNs in specific areas.

Environment monitoring: The applications of WSNs for environment monitoring include habitat monitoring [130], fishery monitoring [112], precision agriculture [21], seismic monitoring [52], forest fire detection [128], pollution monitoring [97], and so on. Ze-

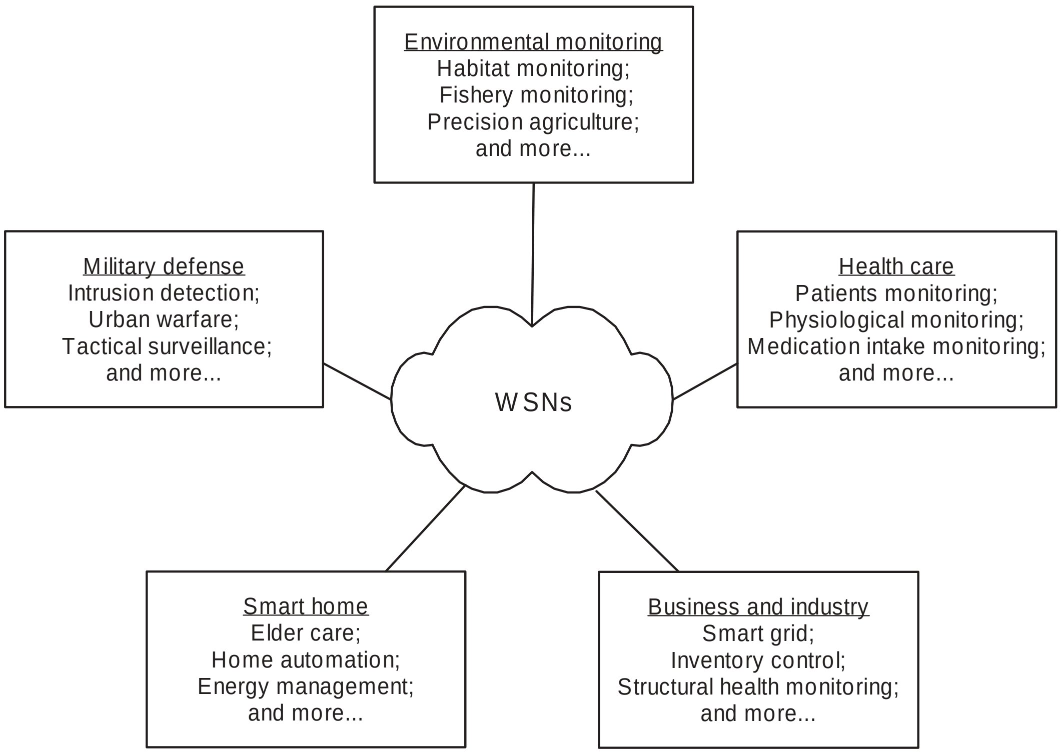


Figure 1.1: An overview of the applications of WSNs.

braNet [130] system, as one of the most typical examples, uses the technology of Global Positioning System (GPS) to track animal migrations. In ZebraNet, sensor nodes are equipped on the collars of wild zebras. ZebraNet generates the logs of the position readings of zebras using GPS, and the position data are propagated from zebra to zebra until received by the base station. By recovering and analyzing these logs, the biologists can have a better understanding of the movement of zebras. Moreover, the system can run autonomously for months at a time. In [112], Vasilescu et al. develop an UWSN platform to monitor the coral reefs and fisheries in the underwater environment. The nodes are equipped with several types of sensing devices, including cameras, water temperature sensors, and pressure sensors. Several nodes have the ability of mobility, and can hover above the static nodes to collect data. These mobile nodes can also perform network maintenance functions, e.g., deployment, relocation and recovery. In the system, two communication modalities are used, acoustic and optical. The acoustic communication is for low data rate broadcast, while the optical communication is for high data rate point-to-point data transfer. In [21], Burrell et al. study the potential of sensor networks in the grape monitoring to help the people working in a vineyard. The deployed sensors monitor the maximum and minimum temperature of the day to calculate heat units, which can assist vineyard managers to have a sense of the ripeness of the grapes.

Military defense: The applications of WSNs for military defense include intrusion detection [12], urban warfare [70], tactical surveillance [22], enemy tracking [124], chemical attack detection [5], and so on. In [12], Arora et al. investigate a project called "A Line in the Sand" for the distributed intrusion detection, where the intruding object may be a soldier with a ferrous weapon or a vehicle. When an intrusion is detected, the intrusion information is processed locally and shared with neighboring nodes. Collaborative processing of the average across time and space makes the system to achieve better sensitivity and noise rejection than an individual node. In [70], Ledeczi et al. present a WSN-based counter-sniper system that aims to reduce the injury caused by a sniper to a soldier in urban environments. By detecting and analyzing the muzzle blast and ballistic shockwave of a gun shot's acoustic signals, the acoustic multi-path effects in urban areas can be mitigated and thus the snipers can be detected and accurately located. In [22], a three-dimensional underwater sensor network is designed for underwater surveillance, where sensors, connecting to the surface buoys with cables, are randomly deployed at different depths. In the system architecture, sensor nodes communicate with the surface buoys through the wired medium, and they communicate with each other through the wireless medium by using the antenna at the surface buoys.

Health care: The applications of WSNs for health care include patients monitoring [15], physiological monitoring [15], medication intake monitoring [75], patient tracking [42], and so on. In [15], Baker et al. develop several prototypes that use the technologies of WSNs for health care, including SleepSafe and Heart@Home. As babies from one month to one year old may die from the sudden infant death syndrome (SIDS) without any warning, a SleepSafe prototype is built to detect the sleeping positions of the babies. Once the babies are detected to be lying on their stomachs, their parents will be alerted. Consequently, the parents do not need to watch their children all the time when the babies sleep. To help people to keep track of their blood pressure, Baker et al. also develop a blood pressure monitoring and tracking system called Heart@Home. After getting the user's blood pressure by the sensors, the information is sent to the computer and processed by a software application. The software application can generate a graph of the blood pressure over time, and also provide helpful medical suggestions. iCabiNET [75] is a prototype designed to monitor the intake of prescription and over-the-counter drugs from anywhere. iCabiNET can track drug names and available doses, check expiration dates, notify the user the time and the recommended dose to take a pill, etc.

Smart home: The applications of WSNs for smart home include elder care [102], home automation [55], energy management [1], and so on. In [102], Skubic et al. develop a sensor network with smart home technologies to monitor older adults in their home. The sensing devices used include video sensors, motion sensors, and a bed sensor that detects presence, sleep restlessness, pulse, and respiration in the bed. By extracting typical activity patterns and identifying alert conditions (e.g. falls), potential problems can be detected in the early stage. In [55], Hussain et al. integrate radio frequency identification (RFID) into WSNs in a smart home environment. To track occupants, RFID can be used to identify the occupant with a RFID tag, while the received signal strength indicator (RSSI) can be used to identify the presence of a person without a RFID tag using a WSN technology. In [1], Agarwal et al. design and implement a WSN platform to reduce the energy consumption of the heating, ventilation, and air-conditioning systems in buildings. Rather than fixed schedules, the platform accurately detects the occupancy at the level of individual offices and then makes smart decisions.

Business and Industry: WSNs also have many potential applications for business and industry, including smart grid [47], inventory control [37], structural health monitoring [121], traffic light control [131], interactive games [116], mine reconnaissance [3], and so on. Smart grid [47], as a modernized electric power system for the production and distribution of electricity, can utilize WSNs as a vital component. The applications of WSNs for smart grid include remote system monitoring, equipment fault diagnostics, and wireless automatic meter reading. SensorScheme [37] is a supply chain management platform, which uses WSNs for active transport tracking. In SensorScheme, sensor nodes are attached to returnable transport items, such as crates, pallets, roll containers and shipping containers. The sensors act as active transport tracking devices, which can monitor the transportation process, verify suitable conditions of goods and detect damage (e.g. opening of containers due to sudden shocks). Wisden [121] is a wireless sensor network system designed for structural health monitoring. In Wisden, the data packages of the vibration measurements are transmitted hop by hop to a base station for processing.

1.2 Link Scheduling in WSNs

In wireless networks, the packets transmitted by a node may be received by all the nodes within its transmission range due to the broadcast nature of the wireless medium. Therefore, the transmission of one link may interfere with the reception of another link. To avoid the interferences among the communication links, we adopt the time division multiple access (TDMA) MAC protocols, such as TRAMA [90], DCQS [24] and DRAND [95]. TDMA protocols have the natural advantages of having no contention-introduced overhead or collisions [126]. TDMA protocols are also more energy-efficient since nodes switch to the active state (transmit, receive or listen) in the allocated time slots and switch to the sleep state in the other time slots. Moreover, the TDMA MAC protocols can guarantee a deterministic delay bound.

In the TDMA MAC protocols, time is divided into equal intervals referred to as *time slots*, which are grouped into frames. We assume that time is synchronized, and previous work [26, 72, 105, 120] has provided some time synchronization mechanisms for WSNs. A link scheduling is to assign each link a set of time slots $t \subset [1, T]$ to transmit, where T is the scheduling period. A link scheduling is said to be interference-free if a scheduled transmission will not result in a collision at both sender and receiver. Figure 1.2 shows a frame (or a scheduling period T) consists of N time slots in TDMA, which repeats periodically in the scheduling.

In order to be interference-free, a simple approach in the link scheduling is to assign each communication link a time slot, and then the number of time slots assigned is equal to the number of communication links of the network. This link scheduling scheme results

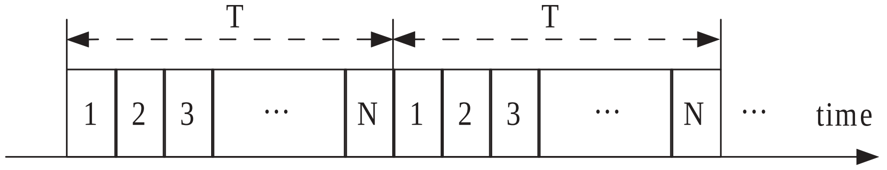


Figure 1.2: Time slots in TDMA.

in many more time slots than necessary, considering that multi-hop networks are able to make space reuse in the shared channel and multiple transmissions can be scheduled in one time slot without any interference. By reducing the time slots assigned, the channel utilization and network throughput can be improved.

Though the TDMA link scheduling suffers from the clock drifting and topological flexibility [95], there are various approaches to solve these issues. Clock drifting can be handled by the guard time, which is two time intervals at the beginning and the end of each time slot. Topological flexibility can be solved by rescheduling the time slots assigned to the links at certain intervals. Moreover, nodes drift on their tethers in UWSNs can also be handled by the guard time [64]. The TDMA link scheduling aims to minimize the number of time slots assigned while producing an interference-free link scheduling, and it has been proven to be NP-complete [11, 35]. Several approximation algorithms have been proposed for the link scheduling problem [9, 34, 40, 92, 114, 118, 122, 132].

1.3 Motivations of the Thesis

In this section, we identify the problems that need to be further investigated in the link schedulings of WSNs.

First, reducing the energy consumption is critical in TWSNs, as the sensor nodes are powered by limited batteries. Apart from the energy consumption in the active states (transmit, receive and listen), sensor nodes also consume energy in the state transitions. However, this kind of energy cost was not considered in the previous studies of the TDMA link schedulings in TWSNs. After a link scheduling, each communication link is assigned a time slot to transmit. If a node starts up to transmit or receive in the time slot assigned and turns to the sleep state after the time slot, the node may start up numerous times in a scheduling period T as it has multiple neighbors to communicate. Note that the typical startup time is on the order of milliseconds, while the transmission time may be less than the startup time if the packets are small [115]. Consequently, the transient energy consumption during the startup process can be higher than the energy consumption during the actual transmission. If a node starts up too frequently, it not only needs extra time, but also consumes extra energy for state transitions. Moreover, it reduces the battery capacity due to the current surges in the state transitions. Therefore, the startup frequency of sensor nodes should be reduced to save both energy and time.

Second, UWSNs use acoustic communications for transmitting data, which is different from TWSNs. Due to the long propagation delay of acoustic signals, current terrestrial MAC approaches are not suitable for UWSNs. Generally speaking, the arrival time of a packet is uncertain owning to the uncertainty introduced at both the transmission and reception sides. In traditional terrestrial MACs, as the propagation delay is commonly neglected, the reception time of a packet is assumed to be the same as the transmission time. Thus, the uncertainty of the arrival of a packet only considers the transmission time uncertainty, i.e., the reception time uncertainty is removed. In UWSNs, however, the reception time uncertainty depends on both the transmission time and relative propagation delay to the receiver. This phenomenon is called "spatio-temporal uncertainty" in [107], which should be considered to design efficient link schedulings in UWSNs.

In this thesis, we will analyze the aforementioned problems in detail and then design methodologies for them.

1.4 Contributions of the Thesis

In this section, we briefly summarize the contributions of this thesis. Our contributions mainly lie in the following three topics: contiguous link scheduling, compact wakeup scheduling and spatio-temporal link scheduling.

1.4.1 Contributions in Contiguous Link Scheduling

To reduce the frequency of state transitions in TWSNs, we identify a novel energy-efficient interference-free link scheduling problem called contiguous link scheduling. In the scheduling, links directed to a node are scheduled together to obtain consecutive time slots such that each node can start up only once to receive data from its neighbors in a period.

We prove that the contiguous link scheduling problem is NP-complete, and then it is necessary to design efficient approximation algorithms with performance guarantee. To solve the problem, we propose a novel conflict graph called merged conflict graph. By formulating the contiguous link scheduling as an interval vertex-coloring of the merged conflict graph, we present a centralized scheduling algorithm that uses time slots at most a constant factor of the optimum. Especially, if the topology is a data gathering tree, each node can start up only twice in a scheduling period: once for receiving data from its children nodes and once for transmitting its data to its parent node.

We further improve the centralized scheduling by re-arranging the time slots in a so-called interference matrix to achieve a higher efficiency, including recursive backtracking and minimum conflicts heuristic. We also develop two efficient distributed algorithms, one only aims to reduce the frequency of state transitions and the other also has the consideration of how to efficiently reduce the network delay. Interestingly, all these algorithms have theoretical performance bounds to the optimum, not only in homogeneous networks, but also in heterogeneous networks. The simulation studies corroborate the theoretical results, and show the efficiency of our proposed centralized and distributed algorithms.

1.4.2 Contributions in Compact Wakeup Scheduling

To minimize the frequency of state transitions in TWSNs, we identify another novel energy-efficient interference-free link scheduling problem called compact wakeup scheduling. In the scheduling, all the links incident to a node are scheduled together to obtain consecutive time slots such that each node can start up only once to communicate bidirectionally with all its neighbors in a period. Note that compact wakeup scheduling has the consecutive constraint in both the incoming and outgoing links, while the contiguous link scheduling only requires the consecutive constraint in the incoming links.

Unfortunately, not all communication graphs have valid compact wakeup schedulings, and it is NP-complete to decide whether a valid compact wakeup scheduling exists for an arbitrary graph. In particular, tree and grid topologies, which are commonly used in WSNs, have valid compact wakeup schedulings. We propose polynomial-time algorithms using the optimum number of time slots in a period for trees and grid graphs. In tree graphs, we first obtain an interval edge-coloring, and then attempt to assign time slots to each edge by the direction of transmission assignment to make sure interference-free. In grid graphs, we present all the possible coloring patterns, and analyze the lower bound as well as the upper bound of the compact wakeup scheduling. Simulations further validate our theoretical results.

1.4.3 Contributions in Spatio-Temporal Link Scheduling

In UWSNs, the long propagation delay of acoustic signals causes spatio-temporal uncertainty. We identify the spatio-temporal link scheduling problem to deal with the spatio-temporal uncertainty. The spatio-temporal link scheduling is also NP-hard, same as the link scheduling in terrestrial wireless networks.

Considering the link transmission delay, we propose a new graph called improved spatial-temporal conflict graph to enhance the existing spatial-temporal conflict graph in [53]. We further transform this conflict graph to a novel three-dimensional conflict graph called slotted spatio-temporal conflict graph, which is constructed based on the network topology, conflict relationship, propagation delay and link transmission delay. We propose both centralized and distributed scheduling algorithms that have theoretical performance bounds to the optimum. We also consider both unified and weighted traffic load scenarios when designing the scheduling algorithms. In the weighted traffic load scenario, we further consider the scheduling whether it has the constraint of consecutive time slots or not. The simulation results show the effectiveness of our proposed algorithms.

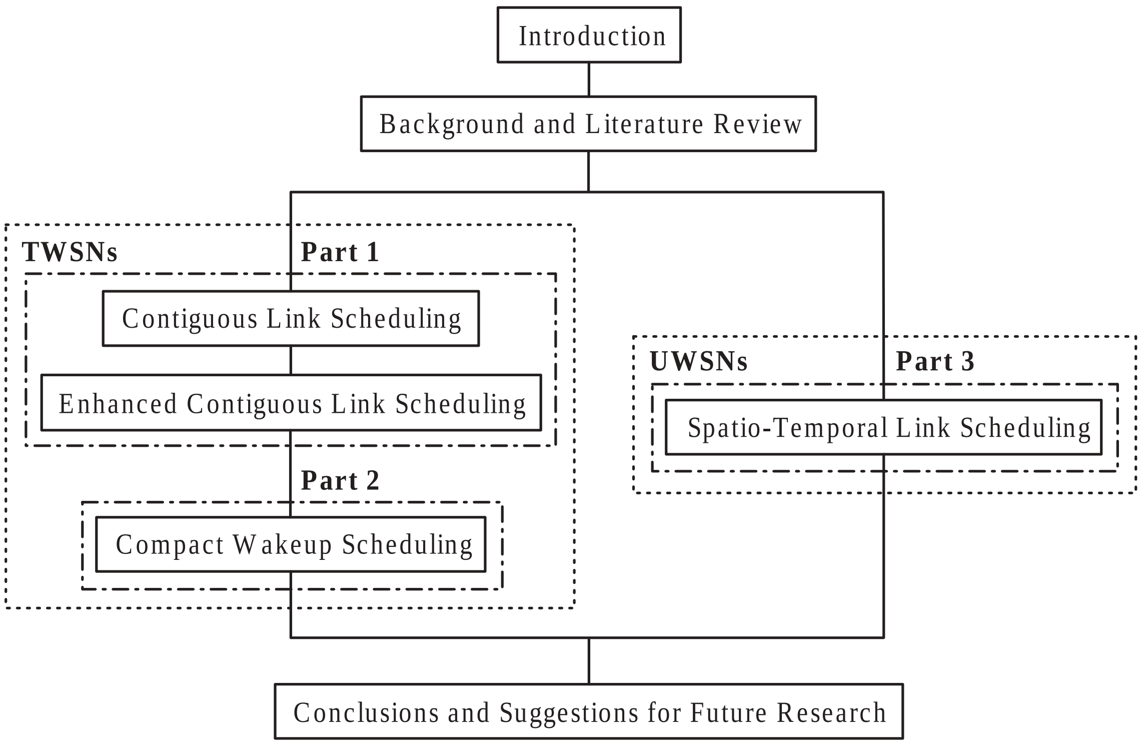

Introduction
Background and Literature Review
TWSNs
Part 1
Contiguous Link Scheduling
Enhanced Contiguous Link Scheduling
Part 2
Compact Wakeup Scheduling
UWSNs
Part 3
Spatio-Temporal Link Scheduling
Conclusions and Suggestions for Future Research

Figure 1.3: The structure of this thesis.

1.5 Organization of the Thesis

The structure of this thesis is illustrated in Figure 1.3. Chapter 1 is the introduction to this thesis, with a summary of the research background, motivations, contributions and the organization of the thesis. Chapter 2 presents the literature review and some background knowledge related to the research issues in this thesis. The main body of this thesis, from Chapter 3 to Chapter 6, is divided into three parts and organized as follows.

In the first part, we discuss our research on the contiguous link scheduling in TWSNs, and this part consists of two chapters. In Chapter 3, we identify the contiguous link scheduling problem and prove it to be NP-complete. To deal with the problem, we propose an efficient centralized scheduling algorithm with a theoretical performance bound. In

Chapter 4, we further investigate the contiguous link scheduling problem to enhance the performance. We propose efficient centralized and distributed algorithms with theoretical bounds in both homogeneous networks and heterogeneous networks.

In the second part, we discuss our research on the compact wakeup scheduling in TWSNs. In Chapter 5, we identify the compact wakeup scheduling problem, and prove that the problem is NP-complete. Especially, for trees and grid graphs, we propose polynomial-time algorithms using the optimum number of time slots in a period.

In the third part, we discuss our research on the spatio-temporal link scheduling in UWSNs. In Chapter 6, we consider the spatio-temporal uncertainty, and propose a slotted spatio-temporal conflict graph. We also design centralized and distributed scheduling algorithms with performance guarantee to the optimum solutions for both unified and weighted traffic loads.

Finally, we conclude this thesis and provide directions for future research in Chapter 7.

Chapter 2

Background and Literature Review

In this chapter, we provide the literature review of the related topics and some necessary background knowledge for the research in this thesis. The organization of this chapter is as follows. We first review the related work about MAC protocols in WSNs in Section 2.1. Then we review the related work about TDMA scheduling in WSNs in Section 2.2. Finally, we make a brief introduction to the graph coloring theory, which is the mathematical tool used in this thesis.

2.1 Existing Work about MAC Protocols in WSNs

In WSNs, Medium Access Control (MAC) is one of the most critical issues, and its primary task is to control when and how each node can transmit in order to avoid collisions. According to the underlying mechanism for collision avoidance, MAC protocols in WSNs can be classified into two categories: contention-based protocols and scheduled protocols [125]. In the contention-based protocols, a sensor contends for the medium to decide whether it can access the channel or not. In the scheduled protocols, the most common method uses time division multiple access (TDMA) to schedule the activities of a sensor, as other forms of multiple access, e.g. frequency division multiple access (FDMA) and code division multiple access (CDMA), would increase the cost and power requirements of the sensors [60].

2.1.1 Contention-based MAC Protocols

We first review the contention-based MAC protocols in TWSNs, and then review the contention-based MAC protocols in UWSNs.

2.1.1.1 Contention-based MAC Protocols in TWSNs

In S-MAC [126], Ye et al. try to reduce the energy consumption of the idle listening state, and propose a mechanism to establish a low-duty-cycle operation of each node, where the duty cycle is defined as the ratio of the listen interval to a complete cycle of listen and sleep periods. In the mechanism, nodes periodically turn to the sleep state for some time, and then wake up to monitor the channel whether it needs to communicate with other nodes. The listen interval is normally of a fixed size and long enough to handle the highest expected traffic load, while the sleep interval can be varied according to different applications. T-MAC [111] improves S-MAC by adopting a dynamic duty cycle instead of a fixed duty cycle. The basic idea is to transmit all messages in bursts of variable sizes and sleep between the bursts. The length of an active period is determined by adaptively ending the listening period when nothing is heard within a limited time, such that the active period can be optimized under different traffic loads.

B-MAC [89] reduces the power consumption by employing an adaptive preamble sampling scheme to decrease the duty cycle, where the transmission duration of a preamble is slightly longer than the sleep time of the receiver. Each node periodically wakes up to check the channel for activity using low-power listening (LPL). If the node detects a preamble, it stays awake to receive the data. Otherwise, it turns to the sleep state. In PW-MAC [108], each node operates a pseudo-random wakeup schedule, and senders try to predict the wakeup time of receivers so as to wake up right before the predicted time. PW-MAC reduces the duty cycle for both senders and receivers, and achieves a near-optimal energy efficiency. To enable accurate predictions considering the hardware and operating system latency and clock drift, PW-MAC also presents an on-demand prediction-error correction mechanism.

The aforementioned contention-based MAC protocols use active/sleep duty cycles to increase the network life, but they suffer from the cost of the packet delivery latency as a forwarding node has to wait until its next-hop receiver wakes up before it can forward a packet. DMAC [76] is an energy-efficient and low-latency MAC for data gathering trees. To allow continuous data forwarding, DMAC schedules the activities of nodes according to their depths in the tree. After that, the nodes along a multi-hop path can wake up sequentially like a chain reaction. In addition, a data prediction is used to request active sending time slots, when multiple children of a sensor have data to send in the same time slot. Keshavarzian et al. [63] consider the design of efficient wakeup scheduling schemes to reduce the bidirectional end-to-end overall delay in sensor networks. They analyze different wakeup patterns and derive their corresponding delay distribution. They also propose a novel cross-layer method, called multi-parent technique, where multiple parents are assigned with different wakeup schedules to each node in the network.

DW-MAC [104] wakes up nodes on demand during the sleep interval of a period, and adaptively increases the effective channel capacity as the traffic load increases. DW-MAC can reduce the packet delivery latency under both unicast and broadcast traffic loads. CyMAC [88] provides a mechanism to decrease the idle listening time through establishing rendezvous times between neighbors and adjust the duty cycles of nodes dynamically based on the traffic conditions. The mechanism also provides a delay bound guarantee for data delivery services.

2.1.1.2 Contention-based MAC Protocols in UWSNs

Traditional contention-based protocols in terrestrial wireless networks perform poorly in underwater networks due to the long propagation delay and the low data rate. Many researchers have attempted to modify them to be suitable for UWSNs. In [25], Chirdchoo et al. study the Aloha-based protocols for UWSNs, and propose two enhanced schemes. In the schemes, the long propagation delay in underwater networks is considered. Each node attempts to use the sender-receiver information from overhearing the packet headers, since the information is helpful in determining whether transmitting a packet is likely to result in a collision at an intended receiver. In [106], Syed et al. firstly identify the unique features of underwater networks, such as spatio-temporal uncertainty and deafness. They propose a tone-based MAC mechanism called T-Lohi, which exploits the spatio-temporal uncertainty and high latency to detect collisions and count contenders. To be energy-efficient, T-Lohi requires each sensor to equip a low power tone receiver and reserve the channel by short wake-up tones.

In [86], Noh et al. propose the delay-aware opportunistic transmission scheduling (DOTS) algorithm to increase the opportunity of concurrent transmissions while reducing collisions. In DOTS, each node has the propagation delay information and expected transmission schedules of its neighbors through overhearing packet transmissions. By compensating for the propagation delay, each node is scheduled to transmit its upcoming packages (RTS/CTS/DATA/ACK), which do not interfere with the current transmissions. In [133], Zhou et al. propose a MAC protocol called CUMAC for long delay multi-channel UWSNs, which makes use of a tone pulse sequence technique for the distributed collision notification. In CUMAC, each node cooperates with its neighbors to select a suitable data channel and detect potential collisions on the selected channel. In [51], Han et al. propose the Multi-session FAMA algorithm to improve the DOTS protocol by allowing multiple outgoing sessions from each source and pipelined packets on each session.

2.1.2 TDMA MAC Protocols

Compared to contention-based MAC protocols, TDMA protocols are lack of scalability, have limited adaptability to network changes and require strict time synchronization. However, TDMA has the advantage of the interference-free communication, which can significantly improve the energy efficiency under a high traffic load. Moreover, TDMA can directly support the low-duty-cycle operations on sensors and allows sensors to sleep if they are not in the active state.

2.1.2.1 TDMA MAC Protocols in TWSNs

In [90], Rajendran et al. propose a TDMA-based MAC protocol called TRAMA, which achieves an energy efficiency improvement by ensuring interference-free data transmissions and allowing nodes to switch to the sleep state when they are not transmitting or receiving. In TRAMA, each node exchanges the two-hop neighbor information and their schedules using a neighbor protocol and a schedule exchange protocol. To achieve an adequate throughput, TRAMA improves the channel reuse using the traffic load information. To achieve fairness, TRAMA uses an adaptive election algorithm that is inherently fair.

Z-MAC [94] is a hybrid MAC, which uses CSMA (carrier sense multiple access) as the baseline MAC scheme and uses a TDMA scheduling as a hint to improve the contention resolution. Under low contention scenarios, Z-MAC behaves like CSMA to achieve a high channel utilization and a low latency. Under high contention scenarios, Z-MAC behaves like TDMA to achieve a high channel utilization and reduce collisions among two-hop neighbors at a low cost. To obtain the time slot assignments that any two nodes within a two-hop neighborhood are assigned to different time slots, Z-MAC uses DRAND [95]. DRAND is a distributed, robust and scalable TDMA implementation for RAND [91]. DRAND incurs $O(\delta)$ running time and message complexities, where δ is the maximum number of two-hop neighbors for any node in the network.

Funneling-MAC [2] is a hybrid TDMA and CSMA protocol to deal with the funneling effect, where the nodes closer to the sink are heavily congested since all data generated in the sensor field are forwarded to the sink through those nodes. Funneling-MAC mitigates the funneling effect by using a TDMA scheduling in the funneling region and implementing the CSMA/CA network-wide. This MAC is sink-oriented as the burden of managing the TDMA scheduling of sensor events in the funneling region is performed by the sink node rather than by the other sensor nodes.

TreeMAC [103] is a localized TDMA MAC protocol, and designed to achieve a high throughput and a low congestion with low overheads. In TreeMAC, a scheduling period is divided into frames, which are further divided into three slots. A node calculates its children's frame assignment by their relative bandwidth demands, and each node determines its own slot assignment by its hop count to the sink. PIP [39] is a TDMA based MAC to achieve a high throughput bulk transfer of large amounts of sensed data in WSNs. PIP uses multiple channels for better spatial reuse and further throughput enhancement. PIP, as a cross-layer design, has a transport layer to handle the end-to-end reliability.

2.1.2.2 TDMA MAC Protocols in UWSNs

To obtain the optimal energy-efficient MAC in UWSNs, TDMA approaches have attracted much attention. In [59], Kredo and Mohapatra propose a hybrid protocol, in which a slotted frame consists of a scheduled period and an unscheduled period. The scheduled portion divides time into scheduled slots using TDMA, and has the advantages of no collisions and low energy consumption. The unscheduled portion uses a contention-based approach, and allows nodes to adapt to changing traffic loads.

UW-FLASHR [123] is a distributed TDMA MAC protocol in UWSNs, which does not require an accurate propagation delay estimation or a tight time synchronization. In UW-FLASHR, each frame is divided into two portions, an experimental portion and an established portion. In the experimental portion, control frames and requests are made to request new transmission time slots. In the established portion, nodes can transmit in the assigned time slots. UD-TDMA [73] is also a distributed TDMA protocol in UWSNs. In UD-TDMA, each node collects the information of its 2-hop neighbors, and assigns itself an initial time slot by a distributed algorithm. A maximal independent set (MIS) can be formed by the nodes with the same initial time slot.

In [53], Hsu et al. construct the spatial-temporal conflict graph to describe the conflict delays among the transmission links and model the link scheduling problem as a vertex coloring problem of the spatial-temporal conflict graph. They propose a new link scheduling called ST-MAC to overcome the spatio-temporal uncertainty. In [64], Kredo et al. propose a scheduled, interference-free TDMA-based MAC protocol called STUMP that increases the channel utilization by leveraging node position diversity and the low propagation speed of the underwater channel. In STUMP, it segments the area around a node into concentric rings and then the uncertainty of node's location would lead to the overestimation of the effect of interferences.

In [54], Huang et al. consider the problem of link scheduling in a single broadcast domain UWSN, which is proven to be NP-complete. They focus on low-complexity distributed, randomized and topology-independent (DRT) schemes, and then find the optimal DRT scheduler by using a nonlinear programming algorithm. In [23], Chen et al. propose a TDMA-based MAC protocol with Dynamic Slot Scheduling Strategy (DSSS) for UWSNs. DSSS improves the channel utilization by using grouping, ordering decision, scheduling, and shifting heuristic strategies. DSSS also considers the sink-to-node, node-to-node, and node-to-sink transmissions to increase the network scalability.

2.2 Existing Work about TDMA Scheduling

TDMA scheduling, which provides an interference-free channel access, can be classified into two types [92], broadcast scheduling and link scheduling. Broadcast scheduling and link scheduling are time slot assignments to nodes and links respectively. In a broadcast scheduling, the transmission of a node must be received by all its one-hop neighbors without interferences. In a link scheduling, the transmission of a node must be received by one particular node of its neighbors without interferences. Several approximation algorithms have been proposed in both broadcast scheduling and link scheduling.

2.2.1 Broadcast Scheduling

Ramaswami and Parhi [93] propose an efficient interference-free broadcast scheduling in a multi-hop packet radio network. They present a centralized polynomial-time algorithm based on a sequential graph coloring heuristic. They also present a distributed implementation of the centralized algorithm, based on circulating a token through the nodes in the network. In [66], the broadcast scheduling problem is modeled as a distance-2 coloring problem, and Krumke et al. propose approximation heuristic algorithms for various geometric graphs. In [84], Ngo et al. formulate the broadcast scheduling to a within-two-hop connectivity matrix. Based on the matrix, they present a centralized genetic-fix algorithm to reduce the search space.

In [41], Gandhi et al. aim to minimize the latency and redundancy in the broadcast scheduling. They present efficient distributed algorithms that have performance bounds in both the number of retransmissions and latency. In [78], Mahjourian et al. further consider the carrier sensing range in the broadcast scheduling, and develop a new conflict-aware network model. Based on the model, they present a constant approximation algorithm and also present an efficient greedy heuristic algorithm.

However, the performance of the broadcast scheduling is worse than the link scheduling in WSNs, especially in terms of energy conservation. In a broadcast scheduling, when a node wants to transmit, all the neighbors have to turn on their radio and start up, no matter whether they are the intended receiver or not. In contrast, only the intended receiver needs to start up in a link scheduling.

2.2.2 Link Scheduling

Ramanathan and Lloyd [92] consider both broadcast scheduling and link scheduling. In tree networks, the scheduling can be optimal. In arbitrary networks, and the performance of the proposed algorithms is bounded by the thickness of a network. In [91], Ramanathan develops a unified framework and uses a generalization based on the constraints for TDMA, FDMA and CDMA based multi-hop wireless networks. Based on the framework, Ramanathan presents a unified algorithm for efficient scheduling, which allows the constraints characterizing the problem and the network topology to be the input parameters.

In [40], Gandham et al. propose a link scheduling algorithm involving two phases. In the first phase, a valid edge coloring is obtained in a distribution fashion. In the second phase, each color is mapped to a unique time slot, and the hidden terminal problem as well as the exposed terminal problem are avoided by assigning each edge a direction of transmission. The overall scheduling requires at most $2(\Delta + 1)$ time slots when the topologies are acyclic, where Δ is the maximum degree of a graph. In [118], Wang et al. propose both centralized and distributed algorithms with performance guarantee to obtain a good interference-free link scheduling that maximizes the throughput of the network. In the algorithms, the sensors are scheduled individually in a predefined order without the consecutive assignment of time slots, and each node is assigned the best possible time slot to transmit or receive without causing interferences to the already-scheduled sensors.

Djukic and Valaee [34] aim to find schedules with minimum round trip delay in the link scheduling. They formulate the problem as a network flow problem in a conflict graph and propose an efficient min-max delay scheduling method. Ergen and Varaiya [36] formulate a many-to-one scheduling problem in sensor networks to find the smallest length interference-free assignment of time slots during which the packets generated at each node reach the sink. They prove the problem to be NP-complete, and propose efficient centralized and distributed algorithms.

In [57], Jolly et al. provide a TDMA-based MAC to minimize the energy consumption which includes the energy consumed by the state transitions, and a tabu search heuristic to assign contiguous transmission/reception slots to each sensor. In [79], Mao et al. consider a multi-objective TDMA scheduling problem to minimize the total time for data collection and to minimize the energy consumed in the state transitions, where a genetic algorithm and a particle swarm optimization algorithm are hybridized for searching the optimal solution. In [119], Wu et al. propose efficient centralized and distributed scheduling algorithms that reduce the energy cost of state transitions, and also propose an efficient method to construct an energy-efficient data gathering tree. After the scheduling, the energy consumption for both homogeneous and heterogeneous networks have performance guarantee.

In [114], Wan et al. propose a robust link scheduling without the information of positions, and only use the information on whether a given pair of links has interferences or not. Moreover, the approximation bound of the proposed algorithm is no worse than the existing methods. Zhou et al. [132] study the distributed link scheduling under the physical interference model that can achieve a constant fraction of the optimal capacity. The proposed distributed algorithm uses the graph partition and shifting techniques, and can achieve a capacity that is at least a constant factor of the optimum.

Xu et al. [122] propose an efficient interference-free link scheduling for data aggregation with Signal to Interference-Plus-Noise Ratio (SINR) constraints in WSNs. They construct a routing tree, and propose two algorithms with efficient delay. In [9], Alsulaiman et al. formulate the link scheduling problem as a distance-2 edge coloring of a bi-directed graph to avoid the hidden terminal problem. They propose two efficient distributed algorithms under two communication models, including the synchronous message passing model and the asynchronous message passing model.

2.3 Background Knowledge about Graph Coloring

Link scheduling can be reduced to the problem of graph coloring, including vertex coloring and edge coloring.

A vertex coloring is an assignment of colors to each vertex in a graph G so that no two adjacent vertices share the same color. The smallest number of colors needed to color a graph in the vertex coloring is called the chromatic number $\chi(G)$, and the chromatic number problem is one of Karp's 21 NP-complete problems [62]. If graph G contains a clique with size ω, to color the clique requires at least ω colors and then $\chi(G) \geq \omega$. For any greedy coloring, the number of colors needed in the vertex coloring is at most $\Delta + 1$, that is, $\chi(G) \leq \Delta + 1$, where Δ is the maximum degree. Brooks' theorem [18] states that the number of colors needed is at most Δ except for complete graphs and odd cycles. By mapping each communication link to a vertex and the interference relationship of the links to edges, we can construct a conflict graph [56, 118]. In the conflict graph, there is an edge between the vertices if the corresponding links interfere with each other. Consequently, we can formulate the link scheduling problem as a vertex coloring of the conflict graph.

The interval vertex-coloring [33] is a special vertex coloring in a vertex-weighted graph, in which each node is assigned a set of w_i consecutive colors and no two adjacent nodes share the same color. The interval vertex coloring problem is proven to be NP-hard in [67]. We apply the interval vertex-coloring to address the contiguous link scheduling problem.

An edge coloring is an assignment of colors to each edge in a graph G so that no two adjacent edges share the same color. By Vizing's theorem [18], the number of colors needed to color a simple graph in the edge coloring is either Δ or $\Delta + 1$, where Δ is the maximum degree. However, it is NP-complete to determine whether the number of colors needed is Δ or $\Delta + 1$ for an arbitrary graph. As a node cannot transmit or receive simultaneously due to the half-duplex radio in WSNs, no two adjacent links can transmit at the same time. By mapping each node to a vertex and each communication link to an edge in a graph, we can see that the edge coloring is a natural method to model the link scheduling problem.

The interval edge-coloring [13, 14], is a special edge coloring in which the colors of edges incident to the same vertex must be contiguous, i.e., the colors must be composed of an integer interval. Not every graph has an interval edge-coloring, since a graph G with an interval edge-coloring belongs to Class 1 graphs where the chromatic number of edge coloring is equal to the maximum degree Δ [14]. Sevastjanov [100] proves that the problem of determining the existence of an interval edge-coloring is NP-complete, even for bipartite graphs, and Kubale [68] proves that the interval edge-coloring problem with forbidden colors is also NP-complete. However, experiments [43] with small and sparse graphs show that the existence of an interval edge-coloring is with a high probability. Some examples of graphs with interval edge-colorings are trees, complete bipartite graphs and grid graphs [13, 14, 44]. Giaro and Kubale give several polynomially solvable graphs in [45]. We apply the interval edge-coloring to address the compact wakeup scheduling problem.

Chapter 3

Contiguous Link Scheduling in TWSNs

In this chapter, we investigate a novel energy-efficient interference-free link scheduling problem called contiguous link scheduling. The contiguous link scheduling assigns a sensor with consecutive time slots to receive data from its neighbors in order to reduce the frequency of state transitions. The organization of this chapter is as follows. Section 3.1 is the overview of this work. Section 3.2 describes the system model and formulates the contiguous link scheduling problem. Section 3.3 presents the centralized scheduling algorithm for the problem. Section 3.4 describes the simulation settings and analyzes the simulation results of the proposed algorithm. Finally, Section 3.5 summarizes this chapter.

3.1 Overview

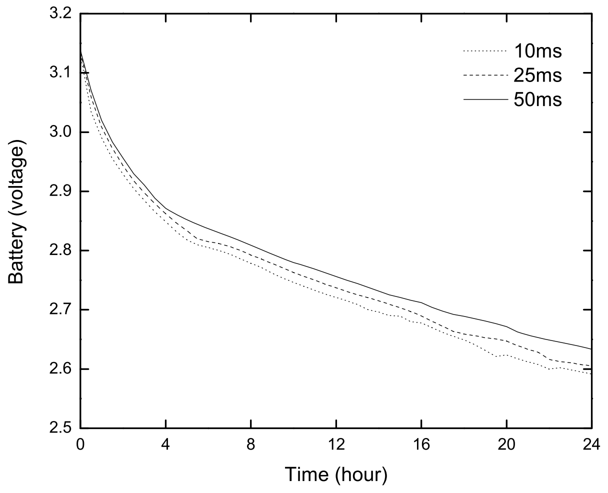
As the batteries of most sensor nodes are non-rechargeable, one key challenging issue in TWSNs is to schedule the activities of nodes to minimize the energy consumption. The major source of energy wastage [10, 111, 126] in TWSNs is the idle listening state in the radio modules, which in fact consumes almost as much energy as receiving. Therefore, nodes are generally scheduled to sleep when the radio is not in use [17, 104], and wake up when necessary. By using such a sleep scheduling, nodes could operate in a low-duty-cycle mode, and periodically start up to check the channel for activity.

The previous studies [63, 118] in the sleep scheduling did not, however, consider all possible energy consumption, especially the energy consumed in the *state transitions*, for example, from the sleep state to the listening state or transmitting state. After such a scheduling, a node may start up numerous times in a period to communicate with its neighbors. Note that the typical startup time is on the order of milliseconds, while the transmission time may be less than the startup time if the packets are small. Take Tmote Sky [99] as an example, the time and energy consumption to activate a node are about 2.1 ms and 32.9 μJ respectively; whereas the time and energy consumption to transmit 1 byte are about 0.032 ms and 1.7 μJ respectively.

Figure 3.1 shows the battery voltage of Tmote Sky sensors [99] with different startup frequencies but with the same duty cycle (50%): One starts up every 20 ms, stays in the receive state for 10 ms and turns to the sleep state for the rest period; one starts up every 50 ms, stays in the receive state for 25 ms and turns to the sleep state for the rest period; another one starts up every 100 ms, stays in the receive state for 50 ms and turns to the sleep state for the rest period. We can see that about 8% and 5% battery voltage can be saved by reducing the startup frequency from five times and twice to once in every 100 ms respectively. Therefore, the state transitions should be considered for an energy-efficient sleep scheduling in TWSNs.

Effective sleep schedulings should allow every node to start up and transmit or receive

(a) Tmote Sky sensor



(b) Battery voltage curve

Figure 3.1: (a) Tmote Sky sensor. (b) The battery voltage of Tmotes with different startup frequency. AA Carbon-Zinc batteries are used in the experiment. $10ms$, $25ms$ and $50ms$ are the active time in each period.

its messages without interferences. One popular way to achieve this is to adopt the time division multiple access (TDMA) MAC protocols, which can directly support the low-duty-cycle operations and have the natural advantages of having no contention-introduced overhead or collisions [126]. In such protocols, time is divided into equal intervals referred to as *time slots*. Correspondingly, nodes turn on the radio during the assigned time slots, and turn off the radio when they are not transmitting or receiving in the wakeup scheduling. For multiple transmission links can communicate at the same time in wireless networks, several nodes can wake up to transmit their packets simultaneously when they do not interfere with each other. Hence, we attempt to minimize the number of time slots assigned to each node while guaranteeing interference-free among the communication links.

In this chapter, we identify a novel interference-free TDMA sleep scheduling problem called *contiguous link scheduling* to reduce the frequency of state transitions in TWSNs. In this scheduling, links incident to one node are scheduled together to obtain consecutive time slots so that each node can start up only once to monitor the channel in a scheduling period T. Especially, if the topology is a data gathering tree, each node needs to start up only twice in a period, once for receiving packets from its children nodes and once for transmitting its packet to its parent node. The objective of the contiguous link scheduling is to find a link scheduling with the minimum period, i.e. the number of time slots assigned in a period, by maximizing the spatial reuse of concurrent transmissions without interferences. This can increase the network throughput and reduce the network delay.

Note that, contiguous link scheduling can not only reduce the state transitions of transceivers, but also reduce the state transitions of other components in the nodes, such as external memory and sensing devices. Moreover, for the data aggregation application [61] in WSNs, each intermediate node can easily collect the data from all its children nodes and then immediately compute the aggregated value, rather than waiting for a long delay until all the data can be received and computed.

The main contributions of this chapter are summarized as follows:

- We address the scheduling problem using a new energy model, which is closer to realistic sensor nodes.
- We identify the contiguous link scheduling problem in WSNs and prove that the problem is NP-complete.
- We present a centralized scheduling algorithm that has a theoretical performance

bound to the optimum of the contiguous link scheduling problem.

- We develop simulations to show the efficiency of the proposed algorithm.

3.2 System Model and Problem Formulation

In this section, we first present the system model consisting of a network model, an interference model and an energy model, then we formulate the contiguous link scheduling problem and prove it to be NP-complete.

3.2.1 System Model

Network Model. We assume that a TWSN has n static sensor nodes, which are all equipped with single omni-directional antennas, and there exists a sink node to collect the data from other sensor nodes. With the assumption that all the sensors have the same communication range r, the network can be represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and E denotes the set of edges referred to all the communication links. If $\{v_i, v_j\} \subseteq V$, the edge $e = (v_i, v_j) \in E$ if and only if v_j is located within the transmission range of v_i. In a directed graph, the edge e is called incident from v_i, and incident to v_j. Similar to most TDMA protocols, we assume that the time is synchronized in the network and the clock drift of a node can be overlooked.

We consider two types of network topologies in this chapter, data gathering tree and directed acyclic graph (DAG), which are commonly used for data collection and aggregation in TWSNs. A data gathering tree is a tree rooted at a sink node, where each intermediate node collects the data from its children nodes and then forwards the data

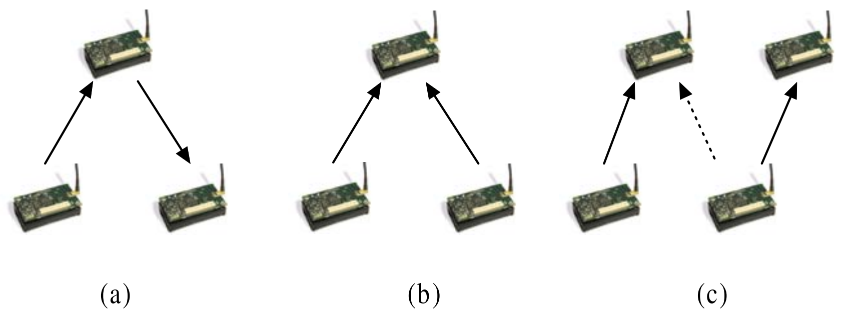


Figure 3.2: Types of interferences: (a) Sending and receiving at one time, (b) Receiving from two different nodes, (c) Secondary interference.

to its parent node. In [109], an approach on how to construct an efficient data gathering tree is proposed. A DAG is a graph with no directed cycles, that is, there is no path that starts and ends at the same node. The depth of a node in a DAG is the length of the longest path from that node to the sink. DAG topologies are also of interest for TWSNs due to their route redundancy to the sink, which can provide reliable data gathering and aggregation. In [69], an approach on how to construct a DAG is proposed.

Interference Model. In wireless networks, the packets transmitted by a node may be received by all the nodes within its transmission range due to the broadcast nature of the wireless medium. Therefore, the transmission of one link may interfere with the reception of another link. The interferences in the network can be divided into two types: primary interference and secondary interference [92]. The primary interference occurs when a node has more than one communication task in a single time slot. Typical examples are sending and receiving at the same time and receiving from two different transmitters, as shown in Figure 3.2 (a) and (b). The secondary interference occurs when a node tuned to a particular transmitter is also within the transmission range of another transmission intended for other nodes, as shown in Figure 3.2(c). Both primary interference and

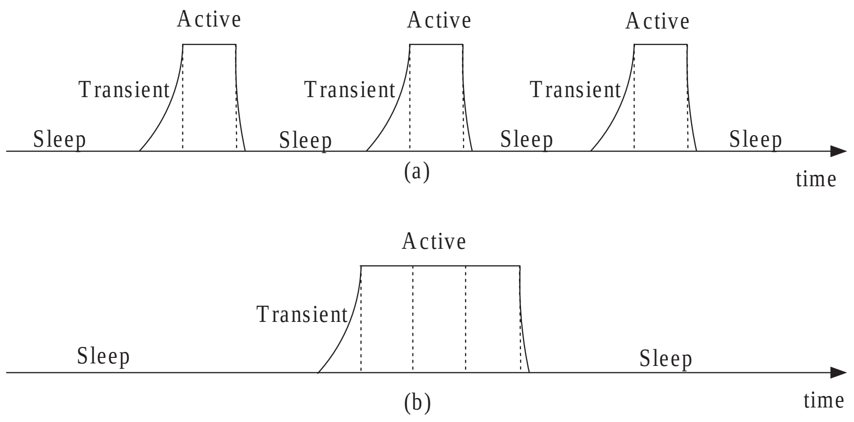


Figure 3.3: The energy model: (a) Before active time slots merged, (b) After active time slots merged.

secondary interference are considered in this chapter.

The interference between two links in the network depends on the interference model, and we use the protocol model [7, 48, 56] in this chapter. In the protocol model, each node v_i has a fixed transmission range r and an interference range R, where $R > r$. We denote the ratio between the interference range and the transmission range as $\gamma = \frac{R}{r}$. In practice, $2 \leq \gamma \leq 4$ [118]. A transmission from v_i to v_j is successful if any node v_k located within a distance R from v_j is not transmitting during the same time interval.

Energy Model. In our energy model, we assume that each node operates in three states: active state (transmit, receive and listen), sleep state, and transient state (state transition) [32]. A node in any active state (i.e., transmit, receive or listen) consumes the power at the same level, compared to the extreme low power consumption in the sleep state. A significant energy saving can be achieved if the sleep state is employed during the periods of inactivity. Hence, the design of a low-duty-cycle mode is essential in the resource constrained TWSNs.

The transient state comprises two processes: startup (from the sleep state to the active state), and turndown (from the active state to the sleep state). Compared to the startup, the turndown process is rapid enough to be negligible. Therefore, we only consider the startup process in the transient state. In [89], Polastre et al. present the energy consumption of the startup process, which includes radio initialization, radio and its oscillator startup, and radio switching to receive/transmit state. The startup process is slow due to the feedback loop in the phase-locked loop (PLL) [19], and the typical setting time of the PLL-based frequency synthesizer is on the order of milliseconds. The startup time and energy consumption in the startup process can also be found in the channel polling in [127].

Our energy model is illustrated in Figure 3.3(a), and there is a significant energy consumption and time overhead when the sensor's radio powers on and off. Figure 3.3(b) shows that merging the sensor's active time slots together can reduce the startup frequency so as to save both energy and time, which benefits the duty cycle network design.

Table 3.1: Time and power consumption in the startup process for a Tmote sky sensor [28].

Operation process	**Time**		**Power consumption**	
Sleep	—	—	P_{sleep}	$0.063mW$
Radio initialization	t_{init}	$0.47ms$	P_{init}	$42mW$
Turn on radio	t_{on}	$1.42ms$	P_{on}	$3mW$
Switch to RX/TX state	t_{sw}	$0.212ms$	P_{sw}	$42mW$
Listen	—	—	P_{ls}	$59.1mW$
Receive 1 byte	t_{rx}	$0.032ms$	P_{rx}	$59.1mW$
Transmit 1 byte	t_{tx}	$0.032ms$	P_{tx}	$52.2mW$

Table 3.1 lists the typical values of time and power consumption for a Tmote Sky sensor in the startup process. Note that the energy consumption of receiving is higher

than transmitting, as the radio CC2420 used in Tmote Sky sensors is designed for low power and short range wireless applications (such as ZigBee [8]) and the drain efficiency is small [117]. The time to activate a sensor is $t_{init} + t_{on} + t_{sw} \approx 2.1ms$, the energy consumption to activate a sensor is $P_{init}t_{init} + P_{on}t_{on} + P_{sw}t_{sw} \approx 32.9\mu J$, and the energy consumption to transmit a packet (e.g. 36 bytes) is $P_{tx}t_{tx}L_{packet} \approx 60.1\mu J$. We can see that the transient energy consumption during the startup process is over 50% compared to that of transmitting a packet.

3.2.2 Problem Formulation

In a TDMA sleep scheduling, each link $l_{i,j}$ is assigned a time slot, in which both sender node v_i and receiver node v_j should start up to communicate. After the allocated time slot, nodes v_i and v_j change to the sleep state. When using traditional link scheduling algorithms (e.g. [118], called *degree-based heuristic* in this chapter) which schedule the communication links one by one, node v_j may start up w_j times to monitor the channel in a period T, where w_j is the number of directed links incident to node v_j. As addressed before, the frequent startup would consume a large amount of extra energy and time. To deal with this problem, we assign consecutive time slots to all the directed links incident to the same node, and then a node needs to start up only once to receive all the packets from its neighbors. We refer to such an interference-free scheduling as the *contiguous link scheduling*.

Definition 3.1. The *contiguous link scheduling problem* is to find an interference-free link scheduling with the minimum period, in which each link is scheduled a time slot to transmit satisfying the consecutive constraint that all the links incident to one node are

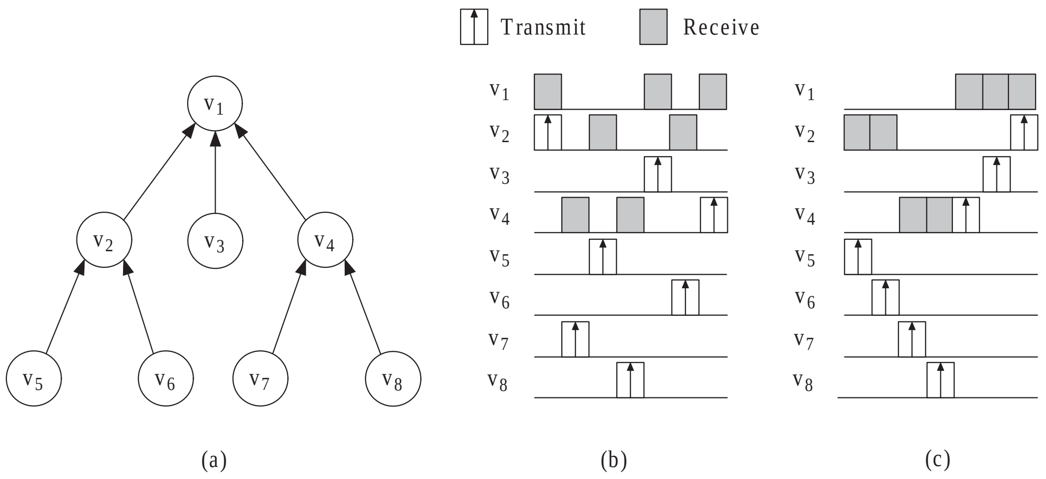


Figure 3.4: Link scheduling and contiguous link scheduling: (a) Communication graph, (b) Link scheduling, (c) Contiguous link scheduling.

assigned consecutive time slots. A contiguous link scheduling is said to be *valid* if the scheduling satisfies the consecutive constraint.

Figure 3.4 illustrates the link scheduling and contiguous link scheduling. In Figure 3.4(a), the given network is a data gathering tree rooted at node v_1, in which any two links interfere with each other. Figure 3.4(b) shows an interference-free link scheduling, where a node starts up numerous times in a period. Figure 3.4(c) shows a contiguous link scheduling that a node can start up only once for receiving data from its neighbors. Note that the contiguous link scheduling can be applied not only to trees, but also to other topologies, such as DAGs. In particular, if the network is a data gathering tree where each node has only one parent, a node just needs to start up at most twice in a period: once for receiving data from its children nodes and once for transmitting its data to its parent node.

Given the protocol interference model, the interference of the links in the communication graph $G = (V, E)$ can be represented as a conflict graph G_c [56]. Corresponding to each directed link from node v_i to node v_j in G, the conflict graph G_c contains a vertex $l_{i,j}$. There is an edge between the two vertices in G_c if the corresponding links interfere with each other in G. In [118], the link scheduling problem is modeled as a vertex coloring of the conflict graph. For the sample communication graph shown in Figure 3.5(a), where dashed lines represent the interference relationship, the corresponding conflict graph is shown in Figure 3.5(b).

To solve the contiguous link scheduling, we propose a *merged conflict graph* G_{mc} corresponding to a graph G, and formulate the contiguous link scheduling problem as an interval vertex-coloring of the graph G_{mc}. In G_{mc}, the w_i directed links incident to node v_i in G correspond to a vertex $L_i(w_i)$, and w_i is the weight of vertex $L_i(w_i)$. There is an edge between the two vertices $L_i(w_i)$ and $L_j(w_j)$ in G_{mc} if and only if at least one link incident to node v_i interferes with a link incident to node v_j in G. We can see that G_{mc} is a vertex-weighted graph. For the communication graph in Figure 3.5(a), the corresponding merged conflict graph is shown in Figure 3.5(c). For example, the two links $l_{3,2}$ and $l_{4,2}$ incident to node v_2 shown in Figure 3.5(a) correspond to vertex $L_2(2)$ in Figure 3.5(c). Similarly, link $l_{6,4}$ corresponds to vertex $L_4(1)$. There is one edge between $L_2(2)$ and $L_4(1)$ in Figure 3.5(c) since $l_{6,4}$ interferes with $l_{3,2}$ and $l_{4,2}$ in Figure 3.5(b). From the communication graph in Figure 3.5(a) and the merged conflict graph in Figure 3.5(c), we can see that the number of vertices in G_{mc} is equal to the number of receiving nodes in G.

Next, we show the NP-completeness of the contiguous link scheduling problem. As the NP-completeness does not apply to an optimization problem but to a decision problem, we define the decision version of the contiguous link scheduling problem as follows.

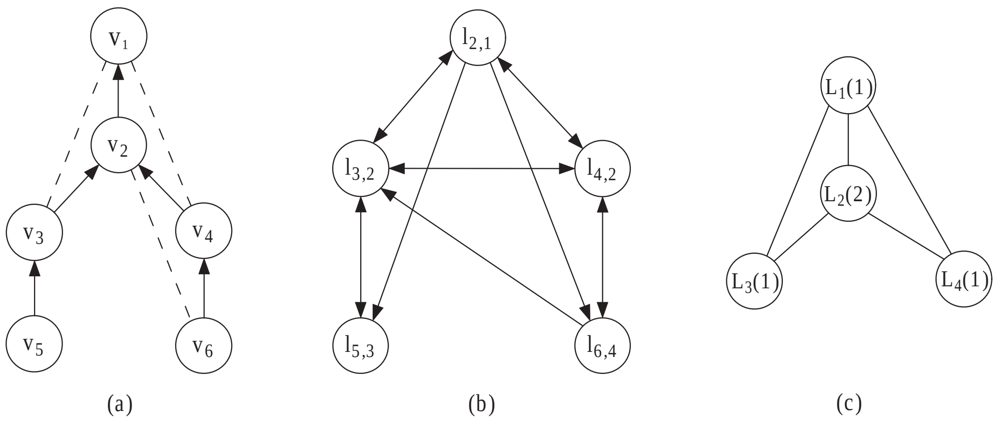


Figure 3.5: Conflict graph and merged conflict graph: (a) Communication graph, (b) Conflict graph, (c) Merged conflict graph. Dashed lines represent the interference relationship.

Definition 3.2. The *decision version of the contiguous link scheduling problem* is to determine, when given a network $G = (V, E)$ and an integer T^*, whether there exists a valid contiguous link scheduling that uses at most T^* time slots.

Theorem 3.1. *The contiguous link scheduling problem is NP-complete.*

Proof. We first show that the contiguous link scheduling problem is in NP. Suppose we are given a scheduling with T time slots. To verify whether the scheduling is a solution to the contiguous link scheduling problem, we need to check (i) whether T is not larger than T^*; (ii) whether the links incident to each node are assigned consecutive time slots; (iii) whether the scheduling is interference-free. Verifying (i), (ii) and (iii) require one operation, $O(|E|)$ operations and $O(|E|^2)$ operations respectively, where $|E|$ is the number of links. It is clearly that this verification can be done in polynomial time.

To prove that the contiguous link scheduling problem is NP-hard, we show a polynomial-time transformation from the problem of finding the chromatic number in the interval vertex-coloring for unit disk graphs to the contiguous link scheduling problem. An *in-*

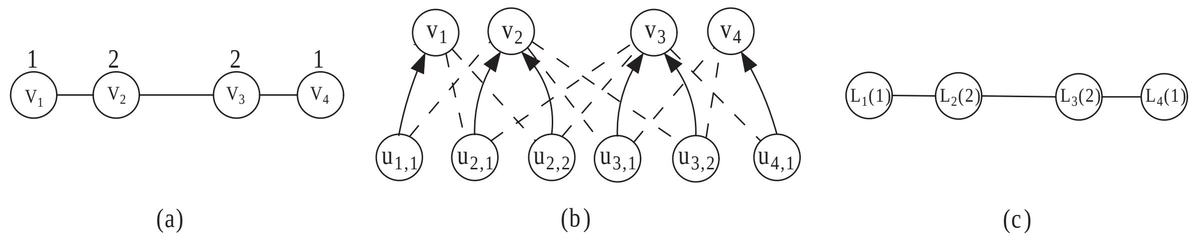


Figure 3.6: (a) A sample of vertex-weighted unit disk graph $G_w = (V_w, E_w)$, (b) Communication graph $G = (V, E)$, (c) Merged conflict graph $G_{mc} = (V_{mc}, E_{mc})$. Dashed lines represent the interference relationship.

terval vertex-coloring is an assignment of w_i consecutive colors to each node v_i satisfying the constraint that no two adjacent nodes share the same color. The chromatic number of a graph G in the interval vertex-coloring is the smallest number c such that vertices in G can be colored using c different colors. The vertex coloring problem for unit disk graphs is proven to be NP-hard in [27]. As the vertex coloring is a special case of the interval vertex-coloring where the weight of each vertex w_i is equal to 1, the interval vertex-coloring for unit disk graphs is NP-hard.

Consider an instance of a vertex-weighted unit disk graph $G_w = (V_w, E_w)$, where $V_w = \{v_1, \cdots, v_n\}$, and each vertex v_i corresponds to a unit disk and has a weight of w_i. There is an edge between two vertices in G_w when the corresponding disks contains the center of each other. Figure 3.6(a) illustrates a sample vertex-weighted graph, where the weights of vertices v_1, v_2, v_3 and v_4 are 1, 2, 2 and 1 respectively. Based on G_w, we construct a communication graph $G = (V, E)$. The vertex set V of graph G first includes all the vertices in G_w. Then, for each vertex v_i in G_w, we add w_i new vertices $u_{i,1}, u_{i,2}, \cdots, u_{i,w_i}$ into V. The communication edges of graph G are $E = \{(u_{i,j}, v_i) | 1 \leq i \leq n, 1 \leq j \leq w_i\}$, i.e., every vertex $u_{i,j}$ has a directed link to its parent v_i. Vertices in G also have the interference relationship: $u_{i,j}$ $(1 \leq j \leq w_i)$ and v_k $(k \neq i)$ are in the interference range of each other if and only if there is an edge between v_i and v_k in G_w. For the vertex-weighted graph shown in Figure 3.6(a), the corresponding communication graph is shown in Figure 3.6(b), in which the dashed lines represent the interference relationship. The children of v_1, v_2, v_3 and v_4 are $\{u_{1,1}\}$, $\{u_{2,1}, u_{2,2}\}$, $\{u_{3,1}, u_{3,2}\}$ and $\{u_{4,1}\}$ respectively, and the number of children of each node is same as the weight of the corresponding vertices in Figure 3.6(a). Note that each constructed communication graph G is a deployable graph (i.e., there exists a deployment of sensor nodes in WSNs), and a simple method is to place each node v_i as well as its children to the same location and the derived layout of the network is then same as the unit disk graph G_w. We can further construct the merged conflict graph $G_{mc} = (V_{mc}, E_{mc})$ based on the communication graph G. Figure 3.6(c) shows the corresponding merged conflict graph of the communication graph in Figure 3.6(b). The two links $l_{u_{2,1},v_2}, l_{u_{2,2},v_2}$ shown in Figure 3.6(b) correspond to vertex $L_2(2)$ in Figure 3.6(c). Similarly, link $l_{u_{1,1},v_1}$ corresponds to vertex $L_1(1)$. There is an edge between $L_2(2)$ and $L_1(1)$ in Figure 3.6(c), because $l_{u_{1,1},v_1}$ interferes with $l_{u_{2,1},v_2}$ and $l_{u_{2,2},v_2}$ in Figure 3.6(b).

For graph G, in the contiguous link scheduling, the w_i links incident to v_i should be scheduled consecutive time slots. Suppose that the chromatic number in the interval vertex-coloring of the unit disk graph G_w is c. Since G_w is same as the merged conflict graph G_{mc}, the chromatic number in the interval vertex-coloring of G_{mc} is c. According to the definition of the merged conflict graph, it takes c time slots to transmit the packets in each link. Thus, the minimum scheduling period is exactly c.

As we can transform the interval vertex-coloring problem for unit disk graphs to the contiguous link scheduling problem and this transformation can be done in polynomial time, the contiguous link scheduling problem is NP-hard.

Since the contiguous link scheduling problem is both NP and NP-hard, we conclude the problem is NP-complete. □

Notice that we assume nodes have the ability of data aggregation, and all data can be transmitted in one time slot, as we consider a low data-rate TWSN environment. All our results can be easily extended to the case where nodes need multiple time slots to transmit data to their neighbors as discussed in [119]. To deal with that, we can assign each link with multiple time slots and change the weight information from the number of incident links to the number of time slots required by the incident links.

As the contiguous link scheduling problem is NP-complete, it is impossible to find a polynomial-time algorithm with the optimal solution if $P \neq NP$. In the following section, we propose an approximation algorithm with performance guarantee.

3.3 Contiguous Link Scheduling Algorithm

In this section, we propose a *centralized scheduling* algorithm for the contiguous link scheduling problem. Instead of scheduling a time slot individually for each communication link, the links incident to a node are scheduled consecutive time slots, and then each node can start up only once to receive all the data from its neighbors. The centralized scheduling algorithm is described in Algorithm 3.1: We first construct a merged conflict graph G_{mc} based on the communication graph G, and each vertex $L_i(w_i)$ in G_{mc} has a weight of w_i. The scheduling is proceeded in the decreasing order of the weights, i.e., the node which has more incident links is scheduled earlier. In the assignment, each node v_i

is assigned the smallest w_i consecutive time slots using the first-fit heuristic, and these time slots are not yet assigned to any node v_j if $L_j(w_j)$ is adjacent to $L_i(w_i)$ in G_{mc}. After that, the w_i time slots are assigned sequentially to the w_i links incident to v_i.

Algorithm 3.1 Centralized scheduling (Centralized)

```
Input: A communication graph $G = (V, E)$.
Output: A valid contiguous link scheduling.
1: Construct the merged conflict graph $G_{mc}$, and initialize an empty stack $S$.
2: Push the vertices in $G_{mc}$ in the non-decreasing order of weights to the stack $S$.
3: while $S$ is not empty do
4:     Pop a vertex $L_i(w_i)$ from $S$. Assign the smallest $w_i$ consecutive time slots to $v_i$,
       which are not yet assigned to any node $v_j$ if $L_j(w_j)$ is adjacent to $L_i(w_i)$ in $G_{mc}$.
5:     Schedule the $w_i$ time slots sequentially to the $w_i$ links that are incident to node $v_i$
       in $G$.
6: end while
```

We then prove the theoretical bound of the centralized scheduling algorithm. We rely on the disk coverage concept to prove this. For ease of presentation, we use $D(v_i, x)$ to denote the disk centered at node v_i and with radius x, and $d_{i,j}$ to denote the distance between two nodes v_i and v_j.

In the centralized scheduling algorithm, if $L_i(w_i)$ is adjacent to $L_j(w_j)$ in G_{mc}, any link incident to v_i will be assigned a time slot different from the links incident to v_j. Suppose link e is incident to v_i, we define $I(e)$ to be the set of links that cannot be assigned the same time slot with link e including e itself, and $|I(e)|$ to be the number of links in $I(e)$. Take Figure 3.6(b) as an example, let $l_{u_{2,1},v_2}$ be link e, the links in $I(e)$ include the link incident to node v_1, the two links incident to v_2 and the two links incident to v_3. Thus, $|I(e)| = 2 + 2 + 1 = 5$.

Lemma 3.1. *Under the protocol model, for each link e, there are at least $|I(e)|/C'$ time*

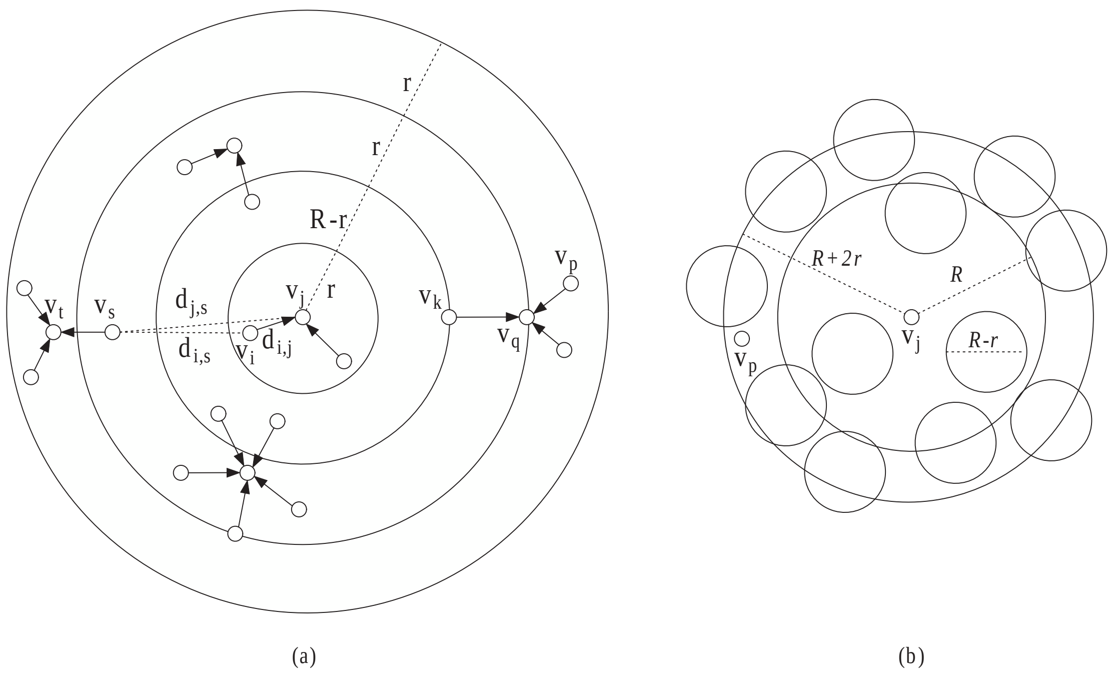


Figure 3.7: (a) Interference of links in the protocol model, (b) Bounding C'.

slots needed if all links in $I(e)$ are interference-free, where $C' = \frac{9(\gamma+1)^2}{(\gamma-1)^2}$.

Proof. Let link $l_{i,j}$ be the link e incident to node v_j. As v_i can communicate with v_j, node v_i must be in $D(v_j, r)$, i.e., $d_{i,j} \leqslant r$. Figure 3.7(a) shows that, for any link l_{pq} with transmitter v_p belongs to $I(e)$, v_q must be in $D(v_j, R{+}r)$, and v_p must be in $D(v_j, R{+}2r)$, because there is at least one adjacent node of v_q in $D(v_j, R)$ to interfere with v_j, such as v_k.

We observe that the distance between two nodes transmitting simultaneously without interferences should be at least $R - r$. For example, if l_{ij} and l_{st} are interference-free, the distance between v_j and v_s is larger than R, and the distance between node v_i and v_s should be $d_{i,s} \geqslant d_{j,s} - d_{i,j} \geqslant R - r$. This means that, to be interference-free, each disk with a diameter $R - r$ can have at most one transmitter. In other words, these disks

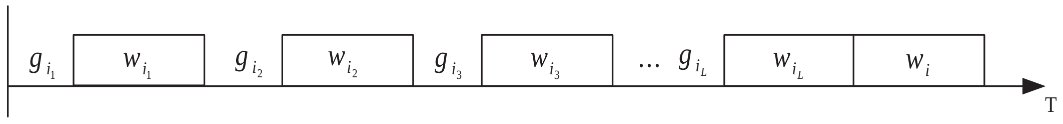


Figure 3.8: The time slots that cannot be assigned to node v_i when v_i is scheduled.

must be non-overlapped. Considering these non-overlapped disks must be placed within disk $D(v_j, R + 2r + 0.5(R - r))$, there are at most $C' = \frac{\pi[R+2r+0.5(R-r)]^2}{\pi[0.5(R-r)]^2} = \frac{9(\gamma+1)^2}{(\gamma-1)^2}$ disks that can be placed at the same time as shown in Figure 3.7(b).

Thus, there exists a link set with the size at least $|I(e)|/C'$ such that each pair of links in the set interferes with each other. Therefore, at least $|I(e)|/C'$ time slots are needed to schedule all links in $I(e)$. □

Theorem 3.2. *The number of time slots assigned by the centralized scheduling algorithm is at most a constant factor of the optimum.*

Proof. For a given communication graph G, the centralized scheduling algorithm first constructs the merged conflict graph G_{mc}, and then schedules each node a number of consecutive time slots in the non-increasing order of weights. In G_{mc}, the vertices adjacent to $L_i(w_i)$ are denoted as $L_{i_1}(w_{i_1})$, $L_{i_2}(w_{i_2})$, $\cdots$, $L_{i_L}(w_{i_L})$, where L is the number of vertices adjacent to $L_i(w_i)$. Suppose that node v_i is the node to be scheduled in the latest w_i time slots and all the other nodes have already been scheduled in some earlier time slots. The numbers of time slots assigned to nodes $v_{i_1}, v_{i_2}, \cdots, v_{i_L}$ are $w_{i_1}, w_{i_2}, \cdots, w_{i_L}$ respectively, and these time slots cannot be assigned to node v_i as shown in Figure 3.8. As the links are scheduled in the non-increasing order of weights, $w_{i_L}(1 \leq l \leq L)$ is not smaller than w_i.

There may exist some gaps, which are the time slots not assigned to nodes $v_{i_1}, v_{i_2}, \cdots, v_{i_L}$

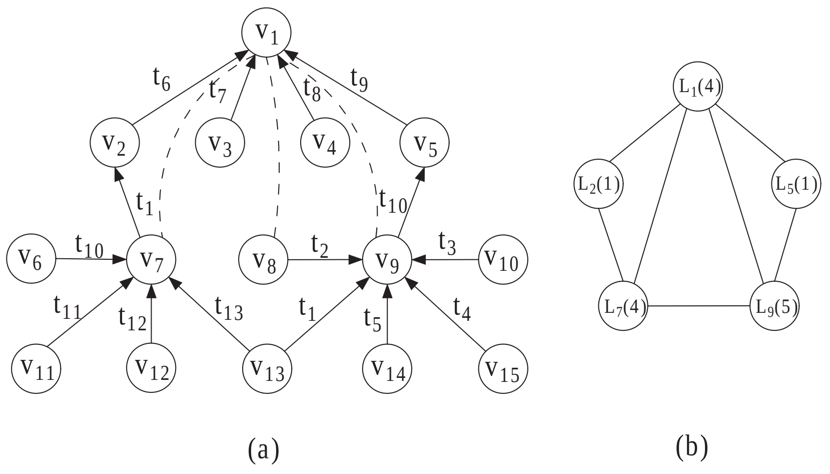


Figure 3.9: (a) Centralized scheduling in the contiguous link scheduling, (b) Merged conflict graph (t_1, t_2, $\cdots$, t_{13} are the assigned time slots, and dashed lines represent the interference relationship).

when node v_i is scheduled. These gaps are denoted as $g_{i_1}, g_{i_2}, \cdots, g_{i_L}$, as shown in Figure 3.8. Since v_i is assigned the smallest w_i consecutive time slots using the first-fit heuristic, g_{i_l} $(1 \leq l \leq L)$ is smaller than w_i, i.e., these gaps are not large enough to be assigned to the links incident to v_i.

The total number of time slots assigned by the centralized scheduling algorithm is

$$\begin{aligned} T = \sum_{l=1}^{L} g_{i_l} + \sum_{l=1}^{L} w_{i_l} + w_i &< L \cdot w_i + \sum_{l=1}^{L} w_{i_l} + w_i \\ &< 2(\sum_{l=1}^{L} w_{i_l} + w_i). \end{aligned}$$

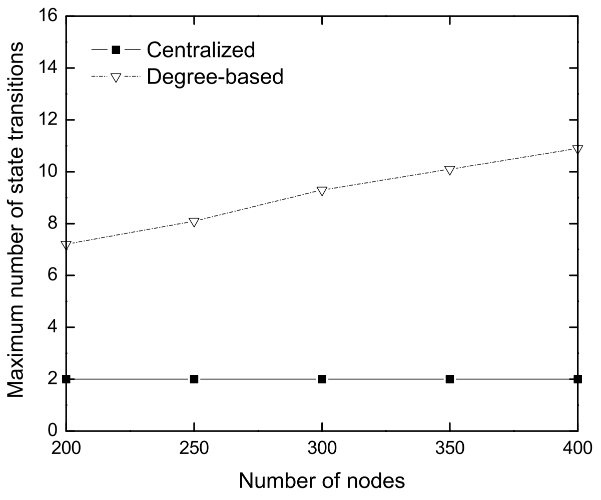
Suppose link e is incident to v_i, the number of links in $I(e)$ is $|I(e)| = \sum_{l=1}^{L} w_{i_l} + w_i$. We denote T_{opt} as the minimum number of time slots (i.e., the minimum scheduling period) that can be achieved by any scheduling. From Lemma 3.1, we know that $T_{opt} \geq \frac{|I(e)|}{C'} = \frac{\sum_{l=1}^{L} w_{i_l} + w_i}{C'}$. Then, we get $T \leq 2C' \cdot T_{opt} = C \cdot T_{opt}$, where $C = 2C'$. □

Example 3.1. The sample network, as shown in Figure 3.9(a), is a directed acyclic graph (DAG) where dashed lines represent the interference relationship. The corresponding merged conflict graph of the network is shown in Figure 3.9(b). When using the centralized scheduling algorithm (Algorithm 3.1), all the nodes that have incident links (i.e., nodes v_1, v_2, v_5, v_7 and v_9) will be scheduled according to their weights. Since the weights of nodes v_1, v_2, v_5, v_7 and v_9 are 4, 1, 1, 4 and 5 respectively, the scheduling order is v_9, v_1, v_7, v_2 and v_5. As $L_1(4)$, $L_7(4)$ and $L_9(5)$ are pair-wise adjacent in the merged conflict graph G_{mc} as shown in Figure 3.9(b), v_9 is assigned consecutive time slots from t_1 to t_5, v_1 is assigned consecutive time slots from t_6 to t_9, and v_7 is assigned consecutive time slots from t_{10} to t_{13}. For v_2 and v_5, they are assigned time slots t_1 and t_{10} respectively. After the scheduling, the total number of time slots assigned is 13. The time slot assigned to each link is indicated in Figure 3.9(a).

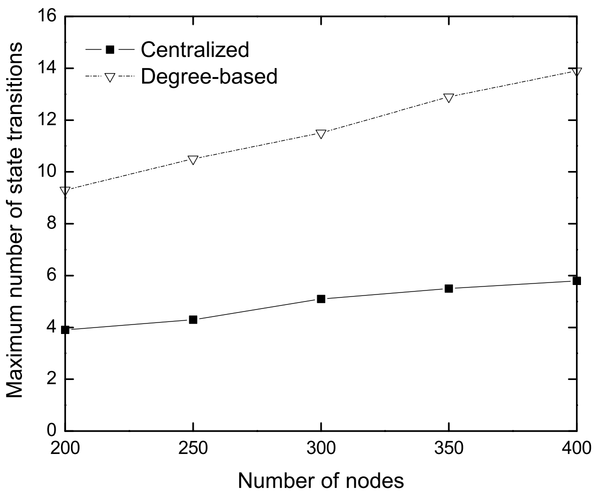
Complexity Analysis. As the number of vertices in the merged conflict graph is no more than the number of nodes n in the network, it takes $O(n^2)$ time to construct the merged conflict graph. To sort the vertices in the decreasing order of their weights takes $O(nlogn)$ time, and to use the first-fit heuristic takes $O(n\Delta')$ time, where Δ' is the maximum degree in G_{mc} and $\Delta' \leq n$. Therefore, the time complexity of the centralized scheduling algorithm (Algorithm 3.1) is $O(n^2) + O(nlogn) + O(n\Delta') = O(n^2)$.

3.4 Simulation Results

In this section, we evaluate the performance of the contiguous link scheduling using a simulator built in C++. We compare our proposed algorithm with the *degree-based heuristic* in [118], where the contiguous link scheduling is not used and the communication links are scheduled one by one. The performance metrics used in the evaluation are maximum



(a) BFS tree



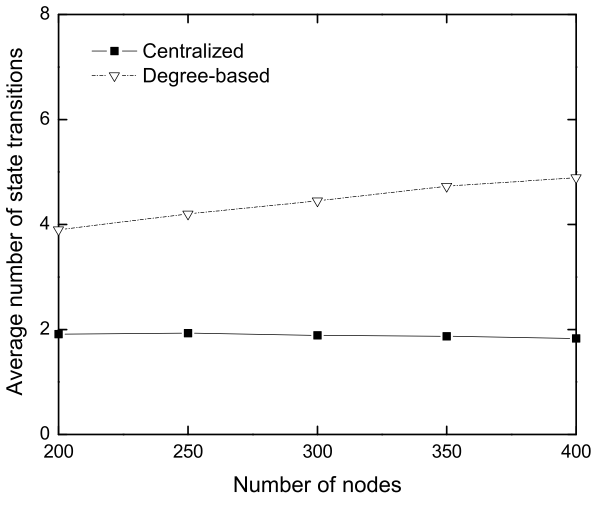
(b) DAG graph

Figure 3.10: Maximum number of state transitions.

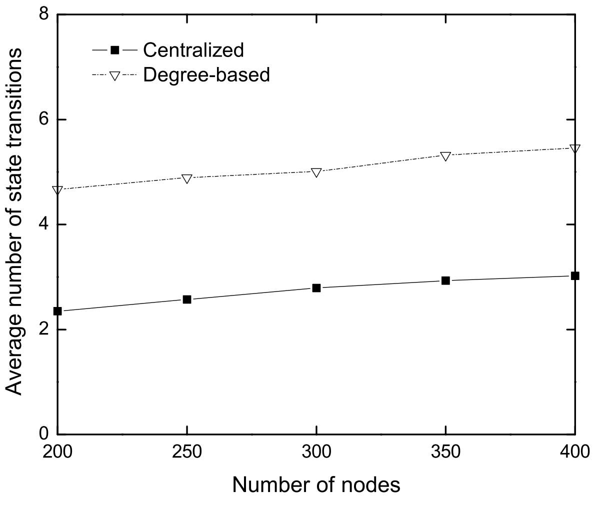
number of state transitions, average number of state transitions, and the number of time slots assigned.

In the simulations, nodes with a transmission range of 15 m and an interference range of 30 m are deployed in a square area of 100 $m \times 100$ m. We test the networks when the number of nodes varies from 200 to 400 with a step of 50. We construct a breadth first search (BFS) tree and a directed acyclic graph (DAG) rooted at the sink node as the topologies of the network. For each case, 50 network topologies are randomly generated where nodes are deployed uniformly, and the average performances over all these randomly generated networks are reported.

Figure 3.10 (a) and (b) show the maximum number of state transitions of the centralized scheduling algorithm (centralized) and degree-based heuristic (degree-based). With the BFS tree topology, the maximum number of state transitions is two in the centralized scheduling, while many nodes need to start up numerous times under the degree-based heuristic, which is proportional to the total time slots required by this node. With the



(a) BFS tree



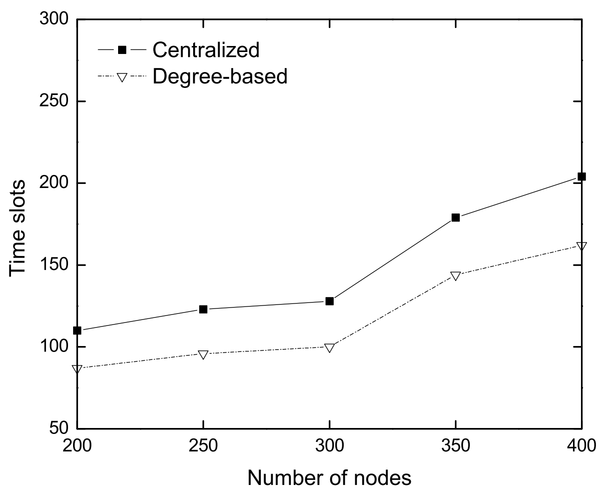
(b) DAG graph

Figure 3.11: Average number of state transitions.

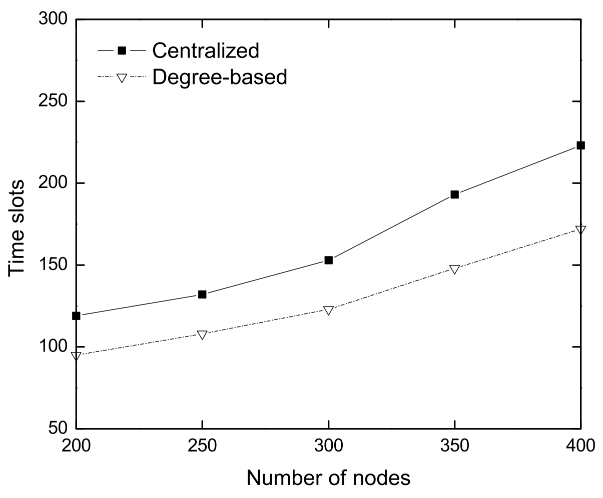
DAG topology, the maximum number of state transitions in the centralized scheduling is still much less than the number of the degree-based heuristic.

Figure 3.11 (a) and (b) show the average number of state transitions. With the BFS tree topology, the average number of state transitions for each node is less than two in the centralized scheduling, compared to the case that many nodes need to start up several times in the degree-based heuristic. With the DAG topology, the average number of state transitions in the centralized scheduling is much less than that in the degree-based heuristic, and thus the transient energy cost can be reduced.

The average number of time slots assigned in the scheduling is shown in Figure 3.12. In both the BFS tree and DAG topologies, the number of time slots assigned increases as the number of nodes increases, as the number of interfering links increases when the number of nodes increases. The number of time slots assigned in the centralized scheduling is comparable with that in the degree-based heuristic.



(a) BFS tree



(b) DAG graph

Figure 3.12: Average number of time slots assigned.

We summarize observations from the simulation results as follows:

- The centralized scheduling algorithm proposed can reduce the number of state transitions, and thus achieve a better energy efficiency.
- If the topology is a tree, the nodes can start up only twice in a period in the contiguous link scheduling.

3.5 Summary

In this chapter, we identify a new interference-free TDMA sleep scheduling problem, called contiguous link scheduling. In the scheduling, a sensor node starts up only once to receive all the data from its neighbors, and thus can reduce the energy consumption and time overhead in the state transitions. We also propose a centralized scheduling algorithm that use time slots at most a constant factor of the optimum. Especially, if the topology is a data gathering tree, each node can start up only twice in one scheduling period. The simulation results show that our algorithm outperforms the existing approach in terms of maximum number of state transitions, average number of state transitions, and the number of time slots assigned.

Chapter 4

Enhanced Contiguous Link Scheduling in TWSNs

In this chapter, we study the contiguous link scheduling problem with further improvements. The organization of this chapter is as follows. Firstly, a brief overview is given in Section 4.1. Section 4.2 describes the system model. Section 4.3 and Section 4.4 present the centralized and distributed algorithms algorithms for the contiguous link scheduling problem. Simulation results are discussed in Section 4.5. Finally, Section 4.6 summarizes this chapter.

4.1 Overview

In Chapter 3, we have identified the contiguous link scheduling problem to reduce the frequency of state transitions, and proposed an energy-efficient centralized scheduling algorithm. To deal with the problem, we formulate the contiguous link scheduling as an interval vertex-coloring of the proposed merged conflict graph.

However, the interval vertex-coloring of the merged conflict graph leads to a problem that enlarges the effect of interferences, as the conflict graph and corresponding merged

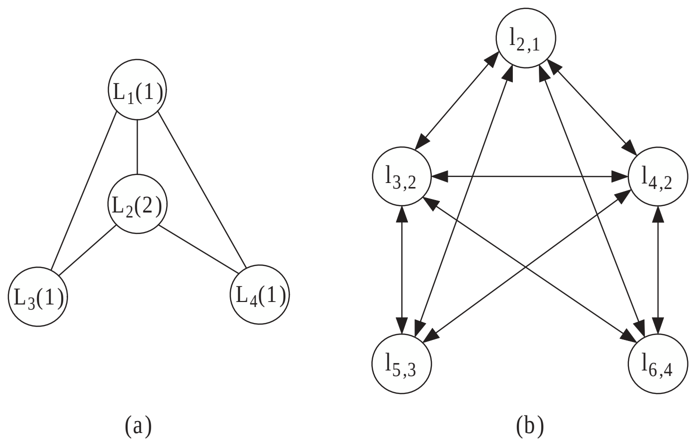


Figure 4.1: Merged conflict graph and corresponding conflict graph: (a) Merged conflict graph, (b) Conflict graph.

conflict graph do not perfectly match. We can construct a merged conflict graph using a given conflict graph, but we cannot recover the conflict graph from a merged conflict graph, as shown in Figure 4.1. Though the conflict graph in Figure 4.1(b) is different from that in Figure 3.5(b), they do have the same merged conflict graph as shown in Figure 4.1(a) and Figure 3.5(c). We can see that the effect of interferences is enlarged by using the merged conflict graph. For example, $l_{4,2}$ and $l_{5,3}$ do not interfere with each other in Figure 3.5(b), but $l_{4,2}$ and $l_{5,3}$ interfere with each other in Figure 4.1(b). Thus, the number of time slots assigned by Algorithm 3.1 is more than necessary, which consequently decreases the network throughput. In order to maximize the network throughput, it is necessary to design more efficient algorithms.

In this chapter, we try to enhance the centralized scheduling (Algorithm 3.1) in the contiguous link scheduling. Considering the drawback of the merged conflict graph, we propose a novel interference matrix, which indicates whether a link can transmit at a particular time slot or not in the presence of interferences. We develop efficient centralized algorithms to achieve a higher efficiency by re-arranging the time slots in the interference matrix, including recursive backtracking and minimum conflicts heuristic. We also develop efficient distributed algorithms, and interestingly, sensor nodes can construct the interference matrices in a distributed manner. Moreover, we study how to efficiently reduce the network delay in the distributed algorithms.

Note that, in Chapter 3, we only consider homogeneous networks, where each sensor has the same transmission range and interference range. In this chapter, we use heterogeneous networks as our network model, where each sensor has a different transmission range and a different interference range. We prove that our proposed algorithms also have performance bounds in the new network model.

The main contributions of this chapter are summarized as follows:

- We address the contiguous link scheduling problem using a new network model.
- We propose centralized algorithms for the contiguous link scheduling with spatial reuse, which also have theoretical performance bounds to the optimum.
- We propose efficient distributed algorithms with performance guarantee.
- We develop simulations to show the efficiency of the proposed algorithms.

4.2 System Model

We assume that a TWSN has n static sensor nodes including a sink node, which are all equipped with single omni-directional antennas. The network is represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and E denotes the set of edges referred to the communication links. In the network, each node v_i has a transmission range r_i. If there is a link $l_{i,j}$ (i.e. the link from v_i to v_j) in E, node v_j is located within the transmission range of node v_i. We also consider two types of network topologies for data collection and aggregation in this chapter, data gathering tree and directed acyclic graph (DAG).

In the network model, we consider heterogeneous networks in this chapter. We suppose the interference range of a node v_i is R_i, and the ratio of the interference range to the transmission range is denoted as $\gamma_i = \frac{R_i}{r_i}$. In practice, $2 \leq \gamma_i \leq 4$. We define $\gamma_{max} = \max\limits_{1\leq i\leq n} \gamma_i$ and $\gamma_{min} = \min\limits_{1\leq i\leq n} \gamma_i$. The maximum transmission range, the minimum transmission range, the maximum interference range and the minimum interference range in a network are denoted as $r_{max} = \max\limits_{1\leq i\leq n} r_i$, $r_{min} = \min\limits_{1\leq i\leq n} r_i$, $R_{max} = \max\limits_{1\leq i\leq n} R_i$ and $R_{min} = \min\limits_{1\leq i\leq n} R_i$ respectively. The ratio of the maximum transmission range to the minimum transmission range is denoted as $\sigma = \frac{r_{max}}{r_{min}}$.

We use the protocol model as the interference model in this chapter, where a transmission from v_i to v_j is considered successful if any node v_k located within a distance R_k from v_j is not transmitting during the same time interval. We also use the energy model in Chapter 3.

4.3 Centralized Algorithms

In this section, we propose the centralized contiguous link scheduling with spatial reuse algorithms, including *recursive backtracking* and *minimum conflicts heuristic.*

The contiguous link scheduling allows the links incident to nodes v_i and v_j to have some time slots overlapped when $L_i(w_i)$ and $L_j(w_j)$ are adjacent in G_{mc}, as long as the same time slots assigned to different links do not cause interferences. A sample is

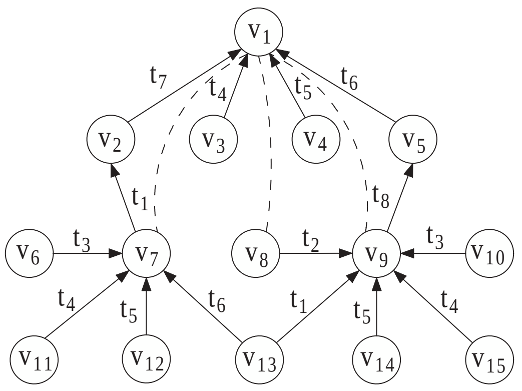


Figure 4.2: Centralized Scheduling with Spatial Reuse (t_1, t_2, $\cdots$, t_8 are the assigned time slots, and dashed lines represent the interference relationship).

illustrated in Figure 4.2. In G_{mc}, $L_1(4)$ is adjacent to $L_7(4)$ as shown in Figure 3.9(b) because $l_{2,1}$ interferes with $l_{6,7}$. But $l_{3,1}$, $l_{4,1}$ and $l_{5,1}$ do not interfere with $l_{6,7}$, $l_{11,7}$, $l_{12,7}$ and $l_{13,7}$, so v_1 can still be assigned time slots from t_4 to t_6 when v_7 are assigned time slots from t_3 to t_6. By this way, we could get another valid contiguous link scheduling using 8 time slots (as shown in Figure 4.2), which is fewer than that in the centralized scheduling (as shown in Figure 3.9(a)). Based on this observation, we propose a *centralized scheduling with spatial reuse* to enhance the centralized scheduling (Algorithm 3.1) by re-arranging the time slots in an *interference matrix* that is defined as follows:

Definition 4.1. An *interference matrix* of node v_i is a $t \times w_i$ matrix $M = (m_{j,k})_{t\times w_i}$ $(1 \leq j \leq t, 1 \leq k \leq w_i)$ that indicates whether a time slot could be assigned to a link l_k incident to v_i without interferences, where

$$m_{j,k} = \begin{cases} - & : \textit{link } l_k \textit{ cannot use time slot } t_j \ (\textit{interference}); \\ 0 & : \textit{link } l_k \textit{ can use time slot } t_j \ (\textit{interference-free}); \\ 1 & : \textit{link } l_k \textit{ selects time slot } t_j \ (\textit{selection}). \end{cases}$$

Here, "−" denotes that assigning the link to this time slot will interfere with the already-scheduled links, "0" denotes that assigning the link to this time slot will be interference-free, and "1" denotes that the link has been assigned this time slot. Note that in the interference matrix, the number of columns w_i is equal to the number of links incident to node v_i, and the number of rows t is the number of time slots already assigned in the scheduling.

Definition 4.2. An *interference submatrix* is a $w_i \times w_i$ matrix $M' = (m_{j,k})_{w_i \times w_i}$, which consists of w_i consecutive rows in the interference matrix M.

Example 4.1. In Figure 4.2, suppose nodes v_7 and v_9 have already been scheduled time slots $\{t_3, t_4, t_5, t_6\}$ and $\{t_1, t_2, t_3, t_4, t_5\}$ respectively. Node v_1 that has 4 incident links $\{l_{2,1}, l_{3,1}, l_{4,1}, l_{5,1}\}$ is to be scheduled. The time slots used by the links that interfere with links $l_{2,1}$, $l_{3,1}$, $l_{4,1}$ and $l_{5,1}$ are $\{t_2, t_3, t_4, t_5, t_6\}$, $\{t_2\}$, $\{t_2\}$ and $\{t_1, t_2, t_3, t_4, t_5\}$ respectively. The interference matrix of node v_1 is shown as M_0 and one interference submatrix of M_0 is shown as M'_0.

$$M_0 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \begin{array}{c} \begin{array}{cccc} l_{2,1} & l_{3,1} & l_{4,1} & l_{5,1} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & - \\ - & - & - & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix} \end{array}, M'_0 = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{cccc} l_{2,1} & l_{3,1} & l_{4,1} & l_{5,1} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & - \\ - & - & - & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \end{pmatrix} \end{array}.$$

An assignment in the interference matrix $M = (m_{j,k})_{t \times w_i}$ of node v_i is said to be *valid* if (i) there is one and only one "1" in each row and each column, and (ii) there are w_i consecutive rows that have "1" in each row in the matrix. The links incident to v_i could be scheduled w_i consecutive time slots by obtaining a valid assignment in the interference matrix.

Algorithm 4.1 describes the centralized scheduling with spatial reuse. We first construct a merged conflict graph G_{mc} based on the communication graph G, and construct an interference matrix for each node v_i. We still assign time slots to nodes in the decreasing order of their weights. Then we assign the smallest w_i available consecutive time slots to node v_i using the *recursive backtracking* (Algorithm 4.2) or *minimum conflict heuristic* (Algorithm 4.3). After the links incident to node v_i are assigned time slots, v_i will broadcast the information to all the nodes whose incident links interfere with the links incident to v_i. Then each node in the network would have the interference matrix updated to indicate the time slots that its incident links could not use.

Algorithm 4.1 Centralized scheduling with spatial reuse

```
Input: A communication graph $G = (V, E)$.
Output: A valid contiguous link scheduling.
1: Construct the merged conflict graph $G_{mc}$, construct an interference matrix for each
   node $v_i$ (initially a zero matrix), and initialize an empty stack $S$.
2: Push the vertices in $G_{mc}$ in the non-decreasing order of weights to the stack $S$.
3: while $S$ is not empty do
4:     Pop a vertex $L_i(w_i)$ from $S$. Assign the smallest $w_i$ consecutive time slots to
       node $v_i$, using recursive backtracking (Algorithm 4.2) or minimum conflict heuristic
       (Algorithm 4.3).
5:     Schedule the $w_i$ time slots sequentially to the $w_i$ links that are incident to node $v_i$
       in $G$.
6:     $v_i$ broadcasts the time slot assignment to the nodes whose incident links interfere
       with the links incident to $v_i$, then the informed nodes update their interference
       matrices.
7: end while
```

4.3.1 Recursive backtracking

In order to reduce the time slots assigned in the scheduling, we propose the recursive backtracking algorithm, as shown in Algorithm 4.2.

Algorithm 4.2 Recursive backtracking

```
Input: An interference matrix $M = (m_{j,k})_{t \times w_i}$.
Output: A valid assignment in $M$.
1: Construct an interference submatrix $M'$ consisting of the smallest $w_i$ consecutive rows
   that have at least one "0" in each row.
2: while a valid assignment is not obtained do
3:     Start the first selection in the first row.
4:     Continue the selection in the next row satisfying the condition that there is only
       one "1" in each column, until a valid assignment is obtained.
5:     if a selection fails to obtain a valid assignment then
6:         Execute the backtracking procedure.
7:     end if
8:     if the recursive backtracking fails then
9:         Update $M'$ by deleting the first row of $M'$ and adding a zero row vector $(0)_{1 \times w_i}$
           in the last row of $M'$.
10:    end if
11: end while
```

Algorithm 4.2 first finds the smallest w_i consecutive rows that have at least one “0” in each row in the interference matrix M, and constructs an interference submatrix M' which consists of the w_i rows. Then it starts the first selection in the first row, and the second selection in the second row without interferences to the selection in the first row. The algorithm continues the selection in the next row until a valid assignment is found. During the process of each selection in a row, there may be several candidates “0”. In these candidates, the selected one is referred to as *predecessor*, and the candidates “0” in the next row are referred to as *successors* of the predecessor. If one successor fails in the selection, it then executes the *backtracking procedure*: the algorithm checks whether the next successor of the predecessor satisfies the condition that there is only one "1" in each column. If the successors are exhausted, the algorithm backtracks to the previous predecessor and tries the next successor of the previous predecessor. If there are no more predecessors, the algorithm adds a new time slot interference-free to all the links incident to v_i, and the corresponding interference matrix adds a zero row vector $(0)_{1\times w_i}$ in the last row.

Example 4.2. We first construct an interference submatrix M'_1 of interference matrix M_0. We then select $m_{1,2}$ and $m_{2,3}$ in the first two rows in M'_1 (marked as "①" in the matrix), but it fails in the third row as shown in M'_2. Then we use the backtracking procedure, and select $m_{1,3}$ and $m_{2,2}$ in the first two rows in M'_1, but it fails again as shown in M'_3. As all the predecessors and successors are exhausted, we delete the first row and add a zero vector $(0)_{1\times w_i}$ as the last row to construct a new interference submatrix M'_4. Finally, we can get a valid assignment shown in M'_5, i.e., links $l_{2,1}$, $l_{3,1}$, $l_{4,1}$ and $l_{5,1}$ are assigned time slots t_7, t_4, t_5 and t_6 respectively, as shown in Figure 4.2.

$$M'_1 = \begin{pmatrix} - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix}, \quad M'_2 = \begin{pmatrix} - & ① & 0 & - \\ - & 0 & ① & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix},$$

$$M'_3 = \begin{pmatrix} - & 0 & ① & - \\ - & ① & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \end{pmatrix}, \quad M'_4 = \begin{pmatrix} - & 0 & 0 & - \\ - & 0 & 0 & - \\ - & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$M'_5 = \begin{pmatrix} - & 1 & 0 & - \\ - & 0 & 1 & - \\ - & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

4.3.2 Minimum Conflicts Heuristic

The recursive backtracking algorithm (Algorithm 4.2) is a brute-force search algorithm, and the time complexity can be $O(w_i!)$ in the worst case. To reduce the complexity, we present a fast algorithm called minimum conflicts heuristic, as shown in Algorithm 4.3. The minimum conflicts heuristic is a local search algorithm in the constraint satisfaction problem [96], whose main idea is to assign each variable a value in the initial state and then change one variable at a time by selecting the value that results in the minimum number of conflicts with other variables.

Algorithm 4.3 Minimum conflicts heuristic

```
Input: An interference matrix $M = (m_{j,k})_{t \times w_i}$.
Output: A valid assignment in $M$.
1: Construct an interference submatrix $M'$ consisting of the smallest $w_i$ consecutive rows
   that have at least one "0" in each row.
2: while a valid assignment is not obtained do
3:     Initialize the matrix $M'$ with a random configuration.
4:     Count the number of conflicts in $M'$, and obtain the conflict matrix $M_C$.
5:     Use a heuristic to reduce the interference until a valid assignment is obtained,
       moving the selection "1" with the largest number of conflicts to the position in the
       same column, where the number of conflicts is minimum.
6:     if the minimum conflicts heuristic fails then
7:         Update $M'$ by deleting the first row of $M'$ and adding a zero row vector $(0)_{1 \times w_i}$
           in the last row of $M'$.
8:     end if
9: end while
```

The minimum conflicts heuristic tries to minimize the number of conflicts in a so-called conflict matrix, which is defined as follows:

Definition 4.3. A *conflict matrix* is a $w_i \times w_i$ matrix $M_C = (c_{j,k})_{w_i \times w_i}$ that describes the number of conflicts in the interference submatrix $M' = (m_{j,k})_{w_i \times w_i}$, and each element $c_{j,k}$ in the matrix is the number of conflicts, which is the sum of the selections "1" in both the row j and column k that have $m_{j,k}$. That is, $c_{j,k} = \sum_{l=1}^{w_i} m_{l,k} + \sum_{l=1}^{w_i} m_{j,l} - m_{j,k}$, if $m_{j,k} \neq -$.

The minimum conflicts heuristic algorithm first constructs an interference submatrix M' which consists of w_i consecutive rows, and then starts with a random initial configuration in M', e.g., with one selection per column. The algorithm then uses a heuristic to determine how to reduce the conflicts by moving the selection "1" with the largest number of conflicts to the position in the same column where the number of conflicts is minimum in the conflict matrix. It continues to reduce the conflicts until there is no conflict or the initial configuration fails. If the initial configuration fails, it will add a new time slot and continue.

Example 4.3. We first construct the interference submatrix M'_1 of interference matrix M_0, and one initial configuration of matrix M'_1 is shown as the matrix M'_6, and the corresponding conflict matrix of matrix M'_6 is shown as the matrix M_{C_1}. For example, $c_{1,3} = m_{1,2} + m_{2,3} + m_{3,3} = 3$, $c_{2,3} = m_{2,3} + m_{3,3} = 2$. As the largest number of conflicts among the selections in M_{C_1} is $m_{2,3}$ and none of the numbers of conflicts in the third column is smaller than $m_{2,3}$, the initial configuration M'_6 fails. Then we add a new time slot, and get the interference submatrix M'_4. One initial configuration of matrix M'_4 is shown as the matrix M'_7, and the corresponding conflict matrix of matrix M'_7 is shown as the matrix M_{C_2}. By moving the selection from $m_{4,4}$ to $m_{3,4}$ in M'_7 according to the minimum conflicts heuristic, we can obtain a valid assignment which is same with M'_5 in the recursive backtracking.

$$M'_6 = \begin{pmatrix} - & 1 & 0 & - \\ - & 0 & \textcircled{1} & - \\ - & 0 & 1 & - \\ - & 0 & 0 & 1 \end{pmatrix}, \quad M_{C_1} = \begin{pmatrix} - & 1 & 3 & - \\ - & 2 & \textcircled{2} & - \\ - & 2 & 2 & - \\ - & 2 & 3 & 1 \end{pmatrix},$$

$$M'_7 = \begin{pmatrix} - & 1 & 0 & - \\ - & 0 & 1 & - \\ - & 0 & 0 & \textcircled{0} \\ 1 & 0 & 0 & \textcircled{1} \end{pmatrix}, \quad M_{C_2} = \begin{pmatrix} - & 1 & 2 & - \\ - & 2 & 1 & - \\ - & 1 & 1 & 1 \\ 2 & 3 & 3 & \textcircled{2} \end{pmatrix}.$$

4.3.3 Performance Analysis

In this section, we prove the theoretical bound of the centralized scheduling with spatial reuse algorithm (recursive backtracking and minimum conflicts heuristic). We rely on the disk coverage concept to prove this. For ease of presentation, we use $D(v_i, x)$ to denote the disk centered at node v_i and with radius x, and $d_{i,j}$ to denote the distance between two nodes v_i and v_j.

Lemma 4.1. *Under the protocol model, the distance between two transmitters in a heterogeneous network is larger than $(\gamma_{min} - 1)r_{min}$ if their concurrent transmissions are interference-free.*

Proof. Suppose that nodes v_i and v_p are two transmitters and their concurrent transmissions $l_{i,j}$ and $l_{p,q}$ are interference-free under the protocol model (as shown in Figure 4.3). Without loss of generality, we assume the transmission range of v_i is no less than that

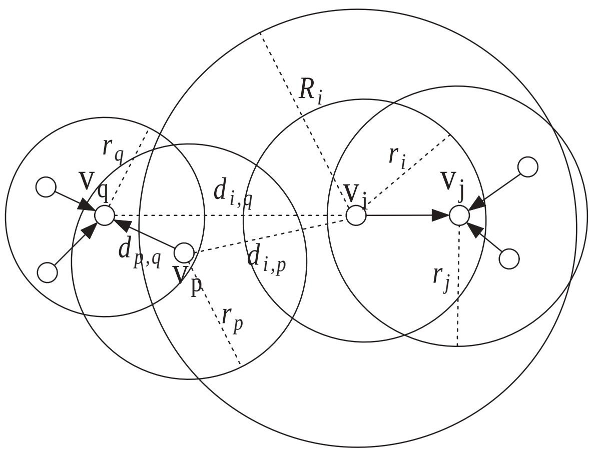


Figure 4.3: Two transmission links that are interference-free.

of v_p, i.e., $r_i \geq r_p$. As v_i's transmission does not interfere v_q's reception, v_q must be outside of $D(v_i, R_i)$, i.e., $d_{i,q} > R_i$. As node v_p can communicate with v_q, v_p is within v_q's transmission range, i.e., $d_{p,q} \leq r_q$. For the distance $d_{i,p}$ between v_i and v_p, using the triangle inequality rule, we can get $d_{i,p} \geq |d_{i,q} - d_{q,p}|$. As $d_{i,q} > R_i$ and $d_{q,p} \leq r_q$, we get $d_{i,p} \geq |d_{i,q} - d_{q,p}| > R_i - r_p \geq R_i - r_i = (\frac{R_i}{r_i} - 1)r_i \geq (\gamma_{min} - 1)r_{min}$. □

Lemma 4.2. *Under the protocol model, for each link e in a heterogeneous network, there are at least $|I(e)|/C''$ time slots needed if all links in $I(e)$ are interference-free, where $C'' = (1 + \frac{2\gamma_{max}\sigma+4\sigma}{\gamma_{min}-1})^2$.*

Proof. Let link $l_{i,j}$ be the link e incident to node v_j. If a link with transmitter v_p belongs to $I(e)$, there are two cases that determine the distance between transmitter v_p and node v_j:

Case 1: A link incident to v_j affects the reception of the link with transmitter v_p. For example, link $l_{i,j}$ incident to v_j affects the reception of link $l_{p,q}$ as shown in Figure 4.4(a). As v_i can communicate with v_j, $d_{i,j} \leq r_j$. As $l_{i,j}$ interferes with $l_{p,q}$, $d_{i,q} \leq R_i$. As v_p can

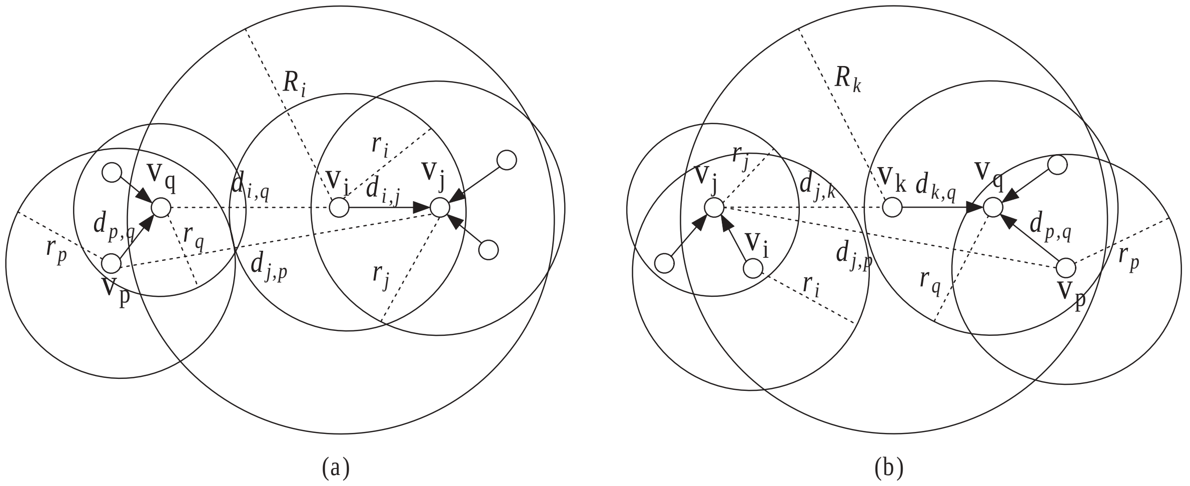


Figure 4.4: (a) Case 1 in Lemma 4.2, (b) Case 2 in Lemma 4.2.

communicate with v_q, $d_{p,q} \leq r_q$. Hence, $d_{j,p} \leq d_{i,j}+d_{i,q}+d_{p,q} \leq r_j+R_i+r_q \leq R_{max}+2r_{max}$.

Case 2: A link adjacent to the link with transmitter v_p affects the reception of v_j. For example, link $l_{k,q}$ adjacent to link $l_{p,q}$ affects the reception of $l_{i,j}$ as shown in Figure 4.4(b). As $l_{k,q}$ interferes with $l_{i,j}$, $d_{j,k} \leq R_k$. As v_k and v_p can communicate with v_q, $d_{k,q} \leq r_q$ and $d_{p,q} \leq r_q$. Hence, $d_{j,p} \leq d_{j,k} + d_{k,q} + d_{p,q} \leq R_k + 2r_q \leq R_{max} + 2r_{max}$.

Therefore, the distance between transmitter v_p and node v_j is at most $R_{max} + 2r_{max}$, as shown in Figure 4.5. That is, for any transmitter belongs to $I(e)$, it must locate in disk $D(v_j, R_{max} + 2r_{max})$. From Lemma 4.1, the distance between two nodes simultaneously transmitting without interferences should be larger than $(\gamma_{min} - 1)r_{min}$. This means that, to be interference-free, each disk with a diameter $(\gamma_{min} - 1)r_{min}$ can have at most one transmitter. In other words, these disks must be non-overlapped. Considering these non-overlapped disks must be placed within disk $D(v_j, R_{max}+2r_{max}+0.5(\gamma_{min}-1)r_{min})$, there are at most $\frac{\pi[R_{max}+2r_{max}+0.5(\gamma_{min}-1)r_{min}]^2}{\pi[0.5(\gamma_{min}-1)r_{min}]^2} = [1+\frac{2R_{max}+4r_{max}}{(\gamma_{min}-1)r_{min}}]^2 \leq (1+\frac{2\gamma_{max}\sigma+4\sigma}{\gamma_{min}-1})^2 = C''$ disks that can be placed at the same time.

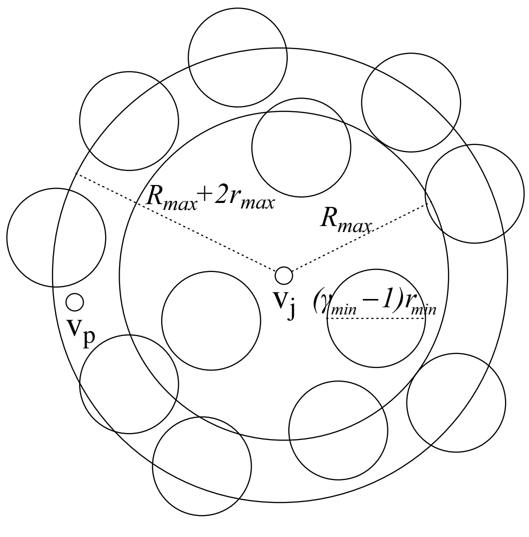


Figure 4.5: Bounding C''.

Thus, there exists a link set with the size at least $|I(e)|/C''$ such that each pair of links in the set interferes with each other. Therefore, at least $|I(e)|/C''$ time slots are needed to schedule all links in $I(e)$. □

Similar to Theorem 3.2, we could get the following theorem:

Theorem 4.1. *The number of time slots assigned by the centralized scheduling with spatial reuse algorithm in heterogeneous networks is at most a constant factor of the optimum, where* $C = 2C''$.

Theorem 4.2. *The number of time slots assigned by the centralized scheduling with spatial reuse algorithm in homogeneous networks is at most a constant factor of the optimum, where* $C = 2C'$.

Proof. In homogeneous networks, as each node has identical transmission range r and interference range R, $\sigma = 1$ and $\gamma_i = \gamma$. From Lemma 4.2, we can get $C' = (1+\frac{2\gamma\sigma+4\sigma}{\gamma-1})^2 = 9(\frac{\gamma+1}{\gamma-1})^2$. □

Similarly, we could prove that the centralized scheduling (Algorithm 3.1) also has a theoretical bound in heterogeneous networks.

Corollary 4.1. *The number of time slots assigned by the centralized scheduling algorithm in heterogeneous networks is at most a constant factor of the optimum, where $C = 2C''$.*

Complexity Analysis. Similar to the centralized scheduling algorithm, the centralized scheduling with special reuse uses $O(n^2)$ time to construct the merged conflict graph and $O(nlogn)$ time to sort the vertices in the decreasing order of their weights. Let Δ and Δ' be the maximum number of links incident to a node in G and the maximum degree in G_{mc} respectively. The scheduling period $T < 2(\sum_{l=1}^{L} w_{i_l} + w_i) \leq 2(L+1)\Delta \leq 2(\Delta'+1)\Delta$ because $w_i \leq \Delta$ and $L \leq \Delta'$. To construct the interference matrix for each node takes $O(\Delta T) = O(\Delta'\Delta)$ time, and to construct the interference submatrix for each node takes $O(\Delta^2)$ time. The time required to obtain a valid assignment using the recursive backtracking for scheduling a node v_i is $O(w_i!) = O(\Delta!)$. The time required to update the interference matrices for scheduling a node v_i is $O(\Delta'\Delta^2)$, since each node v_j, whose corresponding vertex $L_j(w_j)$ is adjacent to $L_i(w_i)$ in G_{mc}, needs to update its interference matrix and the worst case is that any link incident to v_j interferes with any link incident to v_i. Therefore, the time complexity of Algorithm 4.1 with recursive backtracking is $O(n^2) + O(nlogn) + O(n\Delta'\Delta) + O(n\Delta^2) + O(n\Delta!) + O(n\Delta'\Delta^2) = O(n^2 + n\Delta! + n\Delta'\Delta^2)$.

In the minimum conflicts heuristic, the initial configuration takes time linear to w_i. Amazingly, if the initial configuration time is not considered, the runtime of the minimum conflicts heuristic is roughly independent of the problem size of the constraint satisfaction problem [96], such as the n-queens problem. For example, it solves even the million-queens problem in an average of 50 steps. After the initial assignment (the time required is $O(w_i)$), minimum conflicts heuristic can find the solution in a limited step. In each step, the time to construct the conflict matrix is $O(w_i^2)$. Hence, the time required to obtain a valid assignment using the minimum conflicts heuristic for scheduling a node v_i is $O(w_i^2) = O(\Delta^2)$. Therefore, the time complexity of Algorithm 4.1 with minimum conflicts heuristic is $O(n^2) + O(nlogn) + O(n\Delta'\Delta) + O(n\Delta^2) + O(n\Delta^2) + O(n\Delta'\Delta^2) = O(n^2 + n\Delta'\Delta^2)$.

4.4 Distributed Algorithms

Wireless sensor networks are self-organized and distributed, centralized algorithms could not be used without a predefined leader. Therefore, it is necessary to design efficient and scalable distributed algorithms. In this section, we propose two distributed algorithms, *distributed scheduling* and *distributed scheduling with efficient delay.*

4.4.1 Distributed Scheduling

In the distributed scheduling, we use a random order rather than a global decreasing order of the weights, and we assume that there is a contention-based MAC (e.g. B-MAC [89]) available for a node to monitor and compete for the channel. The distributed scheduling is simple and efficient so that each sensor node can run the scheduling with less computation. The distributed scheduling is shown in Algorithm 4.4.

In the distributed scheduling algorithm, a node v_i first competes to obtain the channel, then assigns the smallest consecutive time slots for its incident links without interferences using the first-fit heuristic same as the centralized scheduling. After that, v_i and its neighbors should notify the nodes in their interference ranges the time slots they could

Algorithm 4.4 Distributed scheduling (Distributed)

```
Input: A communication graph $G = (V, E)$.
Output: A valid contiguous link scheduling.
1: Construct an interference matrix for each node $v_i$, which is initially a zero matrix.
2: while a valid contiguous link scheduling is not obtained do
3:   $v_i$ that is not yet scheduled monitors and competes for the channel.
4:   if node $v_i$ obtains the channel then
5:     $v_i$ assigns the smallest $w_i$ consecutive time slots sequentially to its incident links
       which do not interfere with the links that have already been scheduled. Suppose
       link $l_{j,i}$ incident to $v_i$ is assigned time slot $t_j$.
6:     $v_i$ broadcasts the assignment information to the nodes that are in the interference
       range of $v_i$, and these nodes could not transmit in the $w_i$ time slots due to
       the interferences. These nodes notified by $v_i$ forward the information to their
       receivers to update the interference matrices.
7:     Each node $v_j$ adjacent to $v_i$ broadcasts the assignment information to the nodes
       that are in the interference range of $v_j$ to update the interference matrices, and
       these nodes could not receive in time slot $t_j$ due to the interferences.
8:   else
9:     $v_i$ waits for a random time.
10:  end if
11: end while
```

not use, and then these nodes can update their interference matrices. For node v_i, it broadcasts the assignment information to the nodes that are in the interference range of v_i, and these nodes could not transmit in these w_i time slots because their transmissions will interfere v_i's packet reception in these time slots. For each link $l_{j,i}$ incident to v_i, time slot t_j is assigned. Node v_j then broadcasts the assignment information to the nodes that are in its interference range, and these nodes could not use time slot t_j to receive packets because their packet receiving will be interfered by v_j's packet transmitting in time slot t_j.

Example 4.4. In Figure 4.2, suppose node v_9 has already been scheduled time slots $\{t_1, t_2, t_3, t_4, t_5\}$. Node v_9 first informs v_5 not to transmit in time slots from t_1 to t_5, as the packet transmission of v_5 will interfere the packet reception of v_9 in these time slots. Then the elements $\{m_{1,4}, m_{2,4}, m_{3,4}, m_{4,4}, m_{5,4}\}$ in the fourth column of the interference matrix are marked as "$-$". Node v_8, the transmitter in link $l_{8,9}$, next informs v_1 not to receive in time slot t_2, as the packet transmission of v_8 will interfere the packet reception of v_1 in time slot t_2. Then the elements $\{m_{2,1}, m_{2,2}, m_{2,3}, m_{2,4}\}$ in the second row of the interference matrix are marked as "$-$". Therefore, we can update the interference matrix of v_1 as follows.

$$M_0'' = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{array} \begin{array}{c} \begin{array}{cccc} l_{2,1} & l_{3,1} & l_{4,1} & l_{5,1} \end{array} \\ \left(\begin{array}{cccc} 0 & 0 & 0 & - \\ - & - & - & - \\ 0 & 0 & 0 & - \\ 0 & 0 & 0 & - \\ 0 & 0 & 0 & - \\ 0 & 0 & 0 & 0 \end{array}\right) \end{array}.$$

Theorem 4.3. *The number of time slots assigned by the distributed scheduling algorithm in heterogeneous networks is at most $\Theta(K)$ times of the optimum, where $K = \frac{w_{max}}{w_{min}}$, $w_{max} = \max_{1 \le l \le L} w_{i_l}$, and $w_{min} = \min_{1 \le l \le L} w_{i_l}$.*

Proof. Suppose that node v_i is scheduled in the latest w_i time slots, and all the other nodes have already been scheduled in some earlier time slots. We denote $K = \frac{w_{max}}{w_{min}}$, where $w_{max} = \max_{1 \le l \le L} w_{i_l}$ and $w_{min} = \min_{1 \le l \le L} w_{i_l}$.

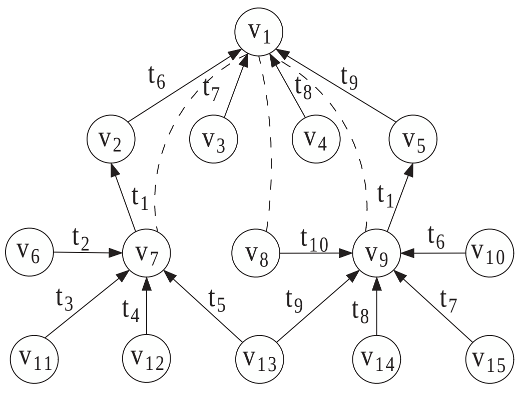


Figure 4.6: Distributed scheduling in the contiguous link scheduling (t_1, t_2, $\cdots$, t_{10} are the assigned time slots, and dashed lines represent the interference relationship).

The number of time slots assigned by the distributed scheduling algorithm is

$$\begin{aligned} T = \sum_{l=1}^{L} g_{i_l} + \sum_{l=1}^{L} w_{i_l} + w_i \qquad &< \sum_{l=1}^{L} w_i + \sum_{l=1}^{L} w_{i_l} + w_i \\ \leq \sum_{l=1}^{L} K \cdot w_{i_l} + \sum_{l=1}^{L} w_{i_l} + w_i &< (K+1)(\sum_{l=1}^{L} w_{i_l} + w_i) \\ \leq (K+1)C'' \cdot T_{opt}. \end{aligned}$$

□

Similarly, we could get the following corollary:

Corollary 4.2. *The number of time slots assigned by the distributed scheduling algorithm in homogeneous networks is at most $\Theta(K)$ times of the optimum.*

Example 4.5. A sample of the distributed scheduling (Algorithm 4.4) is as shown in Figure 4.6. As the scheduling order is randomly selected in the distributed scheduling, we suppose the scheduling order is v_2, v_7, v_5, v_1 and v_9. After the scheduling, v_2 is assigned time slot t_1, v_7 is assigned consecutive time slots from t_2 to t_5, v_5 is assigned time slot t_1, v_1 is consecutive time slots from t_6 to t_9, v_9 is assigned consecutive time slots from t_6

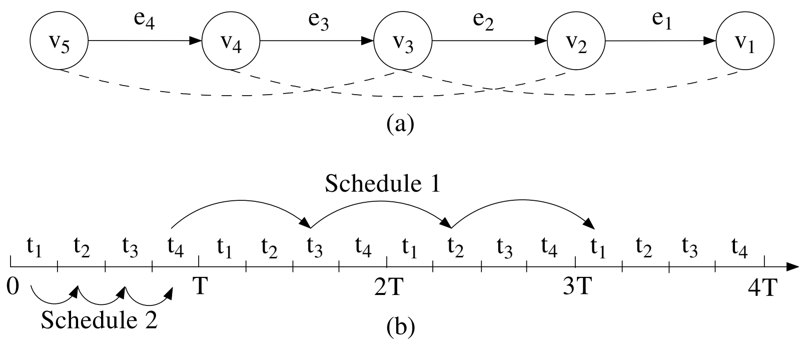


Figure 4.7: Distributed scheduling with efficient delay: (a) Line topology of 5 nodes, (b) Scheduling delay. Dashed lines represent the interference relationship.

to t_{10}. After the scheduling, the total number of time slots assigned is 10. The time slot assigned to each link is indicated in Figure 4.6.

4.4.2 Distributed Scheduling with Efficient Delay

In the TDMA sleep scheduling, a node stays in the sleep state for most time, and periodically starts up to check for activity. As a forwarding node has to wait until its next-hop neighbor starts up and is ready to receive, the message delivery delay will increase. When packets are forwarded from an incoming link to an outgoing link, they could only be forwarded to the outgoing link in the next period T if the incoming link is scheduled to be active after the outgoing link. This kind of delay will accumulate at every hop in the network, which may lead to a long latency. A sample network is illustrated in Figure 4.7(a). As a line topology, the data is transmitted from v_5 to v_1 along the line. If links e_1, e_2, e_3, e_4 are assigned time slots t_1, t_2, t_3, t_4 respectively (Schedule 1), the time delay for a packet transmission from v_5 to v_1 is almost $3T$, as shown in Figure 4.7(b). However, if links e_1, e_2, e_3, e_4 are assigned time slots t_4, t_3, t_2, t_1 respectively (Schedule 2), v_5 could transmit the data to v_1 in one period T. In order to reduce this delay, we

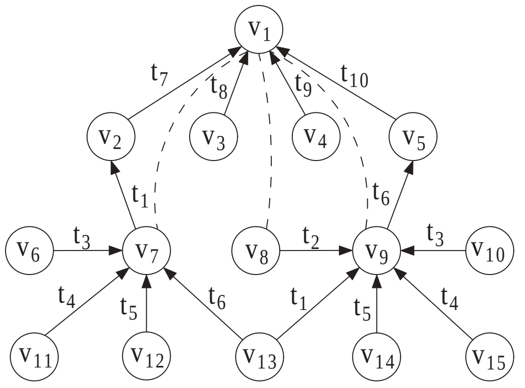


Figure 4.8: Distributed scheduling with efficient delay in the contiguous link scheduling (t_1, t_2, $\cdots$, t_{10} are the assigned time slots, and dashed lines represent the interference relationship).

schedule the links from bottom to top, that is, a node with a higher depth should be scheduled earlier. Hence, a node v_i can only be scheduled until all the children nodes of v_i are already scheduled. The algorithm is described in Algorithm 4.5.

Example 4.6. A sample of the distributed scheduling with efficient delay (Algorithm 4.5) is as shown in Figure 4.8. As the scheduling is from bottom to top, the scheduling order is v_9, v_7, v_2, v_5 and v_1. After the scheduling, v_9 is assigned consecutive time slots from t_1 to t_5, v_7 is assigned consecutive time slots from t_3 to t_6, v_2 is assigned time slot t_1, v_5 is assigned time slot t_6 and v_1 is assigned consecutive time slots from t_7 to t_{10}. After the scheduling, the total number of time slots assigned is 10. The time slot assigned to each link is indicated in Figure 4.8.

As the links are still scheduled in a random order, the algorithm follows a constant bound performance guarantee of the distributed scheduling, that is, the number of time slots assigned by the distributed scheduling with efficient delay is at most a constant factor of the optimum. Similar to Theorem 4.3, we could get the following corollary:

Corollary 4.3. *The number of time slots assigned by the distributed scheduling with*

Algorithm 4.5 Distributed scheduling with efficient delay (Distributed-delay)

Input: A communication graph $G = (V, E)$.

Output: A valid contiguous link scheduling.

```
1: Construct an interference matrix for each node $v_i$, which is initially a zero matrix.
2: while a valid contiguous link scheduling is not obtained do
3:   Each node $v_i$ that is not yet scheduled monitors and competes for the channel if
     all the children nodes of $v_i$ are already scheduled.
4:   if node $v_i$ obtains the channel then
5:     $v_i$ assigns the smallest $w_i$ consecutive time slots sequentially to its incident links
       which do not interfere with the links that have already been scheduled. Suppose
       link $l_{j,i}$ incident to $v_i$ is assigned time slot $t_j$.
6:     $v_i$ broadcasts the assignment information to the nodes that are in the interference
       range of $v_i$, and these nodes could not transmit in the $w_i$ time slots due to
       the interferences. These nodes notified by $v_i$ forward the information to their
       receivers to update the interference matrices.
7:     Each node $v_j$ adjacent to $v_i$ broadcasts the assignment information to the nodes
       that are in the interference range of $v_j$ to update the interference matrices, and
       these nodes could not receive in time slot $t_j$ due to the interferences.
8:   else
9:     $v_i$ waits for a random time.
10:  end if
11: end while
```

efficient delay algorithm in homogeneous and heterogeneous networks is at most $\Theta(K)$ *times of the optimum.*

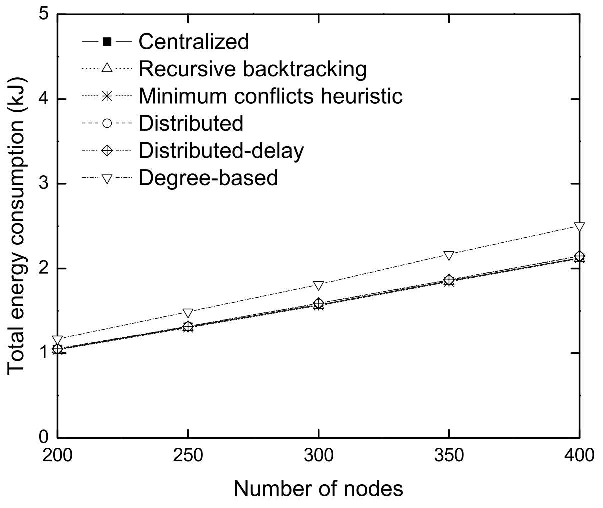
Complexity Analysis. We assume all nodes in the interference range of node v_i are at most k hops away from v_i, i.e., $k = \lceil \gamma_{max} \rceil$. In the distributed algorithms, v_i and its w_i neighbors need to broadcast the assignment information to the nodes in their interference ranges. As a node cannot communicate with all the nodes in its interference range directly, it can broadcast the information to its k-hop neighbors, where $k = \lceil \gamma_{max} \rceil$. In a k-hop broadcast, the worst case is the flooding, where each node in the k-hop region sends a message to its neighbors. Therefore, the message complexities of both Algorithm 4.4 and Algorithm 4.5 are $O(\Delta\rho n)$, where Δ is the maximum number of links incident to a node and ρ is the maximum number of k-hop neighbors.

The time complexity of a distributed algorithm is different from that of a centralized algorithm, as the local computation time is negligible compared to the message transmission time. We assume each message transmission takes one time unit, and then the time complexity of a distributed algorithm is the maximum time required in the worst case. When node v_i is scheduled, v_i and its neighbors need to separately make a k-hop broadcast. The time complexities of both Algorithm 4.4 and Algorithm 4.5 are $O(\Delta n)$, the worst case is that sensors are scheduled sequently and only one sensor is scheduled at a time.

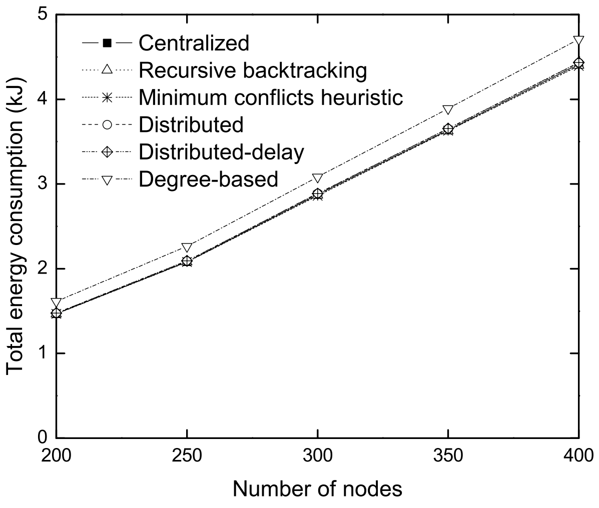
4.5 Simulation Results

Table 4.1: Simulation settings.

Parameter	**Value**
Power consumption in the sleep state	$0.063mW$
Power consumption in the listen state	$59.1mW$
Power consumption in the receive state	$59.1mW$
Power consumption in the transmit state	$52.2mW$
Energy consumption to activate a sensor once	$32.9\mu J$
Data package length	$36bytes$
Data transfer rate	$250kbps$
Time slot size	$4ms$



(a) BFS tree



(b) DAG graph

Figure 4.9: Average total energy consumption in homogeneous networks.

In this section, we evaluate the performance of the proposed algorithms in the contiguous link scheduling in both homogeneous and heterogeneous networks. The algorithms compared in the simulation are the proposed centralized and distributed algorithms (centralized, recursive backtracking, minimum conflicts heuristic, distributed, distributed-delay), and the *degree-based heuristic* [118]. The performance metrics used in the evaluation are total energy consumption, throughput, and time delay.

4.5.1 Homogeneous Networks

In the simulations, nodes with a transmission range of 15 m and an interference range of 30 m are deployed in a square area of 100 $m \times 100$ m. In homogeneous networks, we test the networks when the number of nodes varies from 200 to 400 with a step of 50. For each algorithm, the network operates 10000 scheduling periods. The other simulation settings are listed in Table. 4.1. We construct a breadth first search (BFS) tree and a directed acyclic graph (DAG) rooted at the sink node as the topologies of the network. For each

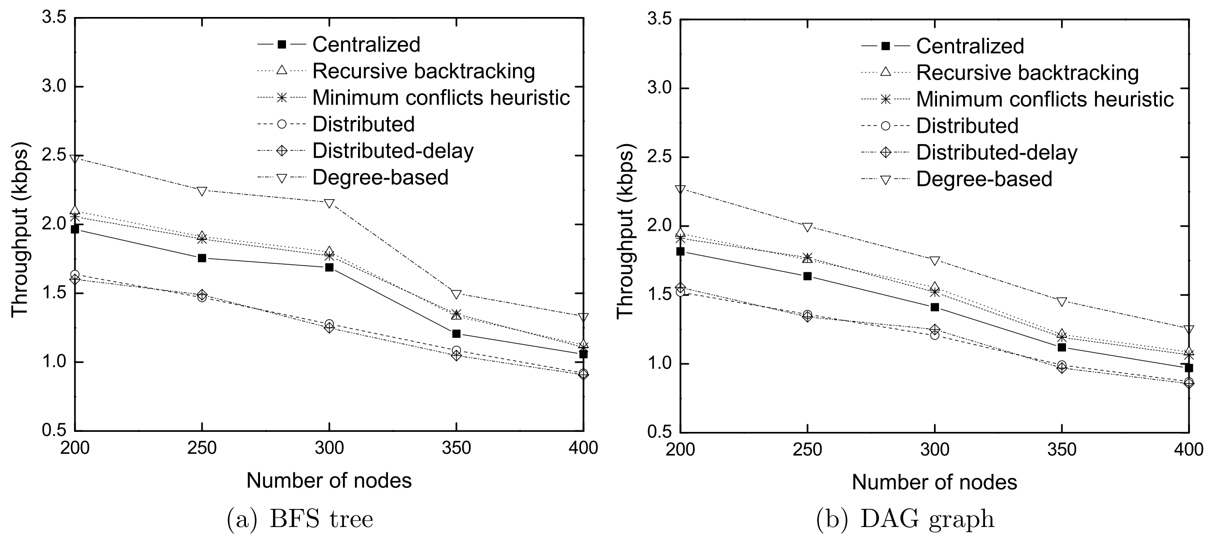
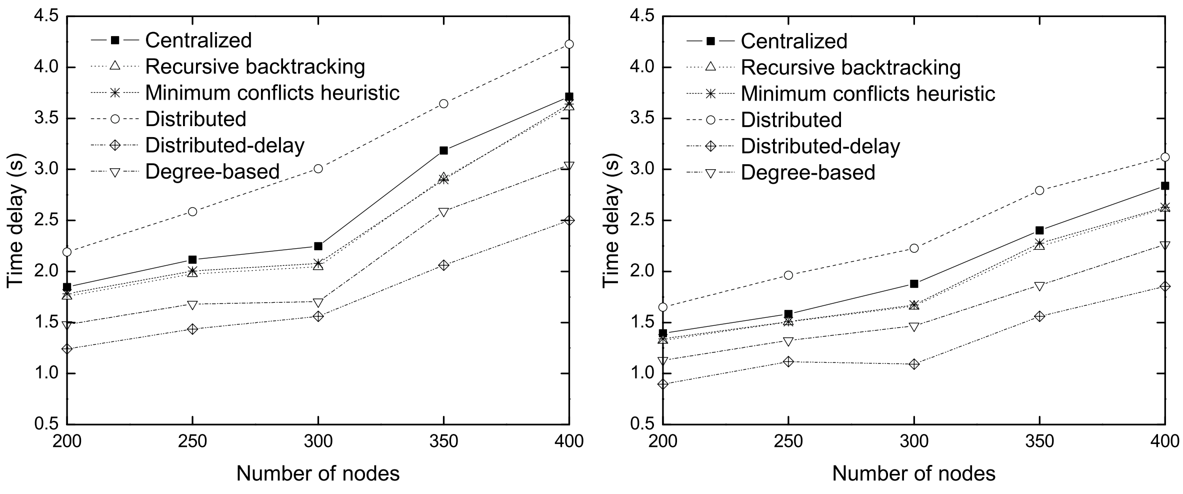


Figure 4.10: Average throughput in homogeneous networks.

case, 50 network topologies are randomly generated where nodes are deployed uniformly, and the average performances over all these randomly generated networks are reported.

Figure 4.9 (a) and (b) show the average total energy consumption in homogeneous networks of the schemes. In both the BFS tree and DAG topologies, the total energy consumption increases as the number of nodes increases. The total energy consumption of the contiguous link scheduling schemes can be reduced about 10% than the degree-based scheme because the contiguous link scheduling schemes can efficiently reduce the frequency of state transitions.

The average throughput in homogeneous networks is shown in Figure 4.10 (a) and (b). In both the BFS tree and DAG topologies, the throughput decreases as the number of nodes increases, for the number of interfering links increases when the number of nodes increases. In the contiguous link scheduling, the links incident to one node are scheduled together to obtain consecutive time slots to avoid the frequent state transitions, and several gaps are formed among the assigned time slots (seen in Figure 3.8), which

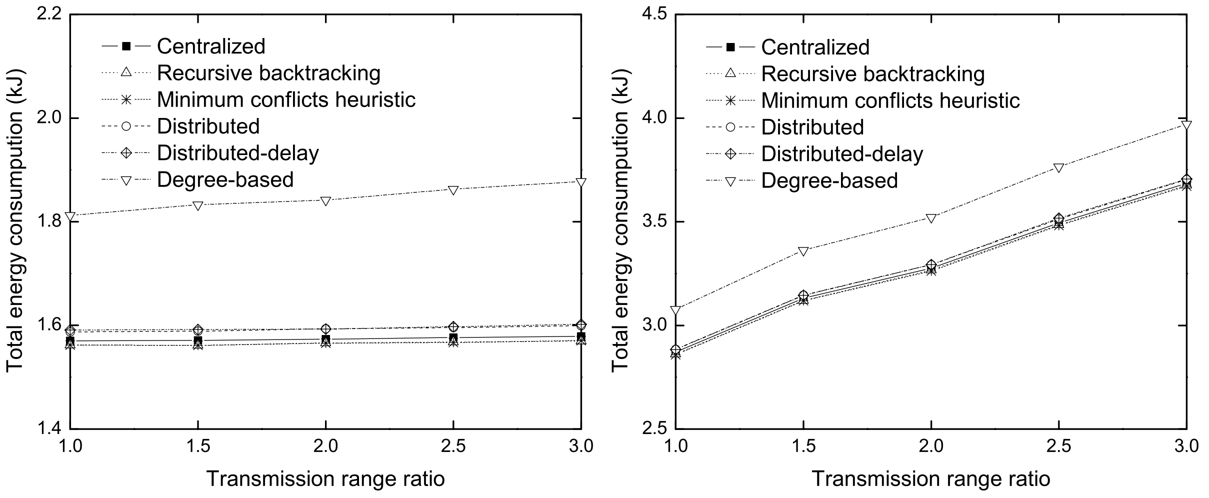


(a) BFS tree (b) DAG graph

Figure 4.11: Average time delay in homogeneous networks.

requires more time slots and hence decreases the throughput. Figure 4.10 (a) and (b) show that the overhead is not high, and the recursive backtracking scheduling scheme has performance comparable to the degree-based scheme. Although the minimum conflicts heuristic may get stuck on a local optimum, it almost has the same performance compared to the recursive backtracking. If the centralized scheduling with spatial reuse is not used, the results would be a little worse, as shown in the centralized scheduling. The two distributed algorithms have the worst performance, due to the fact that they do not have the global information. The throughput in the recursive backtracking scheduling is about 1.3 times the throughput in the distributed algorithms.

Figure 4.11 (a) and (b) show the average time delay in homogeneous networks, and the delay increases as the number of nodes increases in both the BFS tree and DAG topologies. The distributed scheduling with efficient delay scheme has the best performance, for the links are scheduled from the bottom to the top in the scheduling, which is helpful to reduce the network delay.



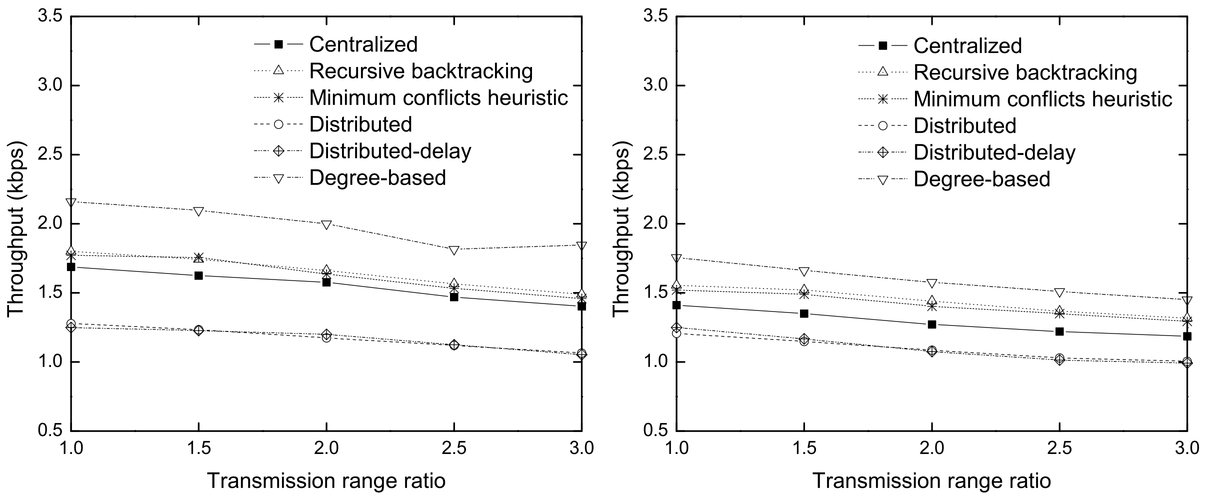
(a) BFS tree (b) DAG graph

Figure 4.12: Average total energy consumption in heterogeneous networks.

4.5.2 Heterogeneous Networks

In heterogeneous networks, we randomly deploy 300 nodes in a square area of 100 $m \times 100$ m, and we vary the transmission range ratio of the maximal transmission range to the minimal transmission range, $\sigma = \frac{r_{max}}{r_{min}}$, from 1 to 3 with a step of 0.5. The transmission range of each node follows a uniform distribution on the interval $[r_{min}, r_{max}]$ with an average of 15 m, and the interference range of each node is twice the transmission range. For each algorithm, the network operates 10000 scheduling periods. For each transmission range ratio, 50 network topologies are generated, and the average performances are reported. The other simulation settings are listed in Table. 4.1.

Figure 4.12 (a) and (b) show the average total energy consumption in heterogeneous networks. In both the BFS tree and DAG topologies, the total energy consumption in the contiguous link scheduling schemes is much less than that in the degree-based scheme, same as homogeneous networks. The total energy consumption does not vary



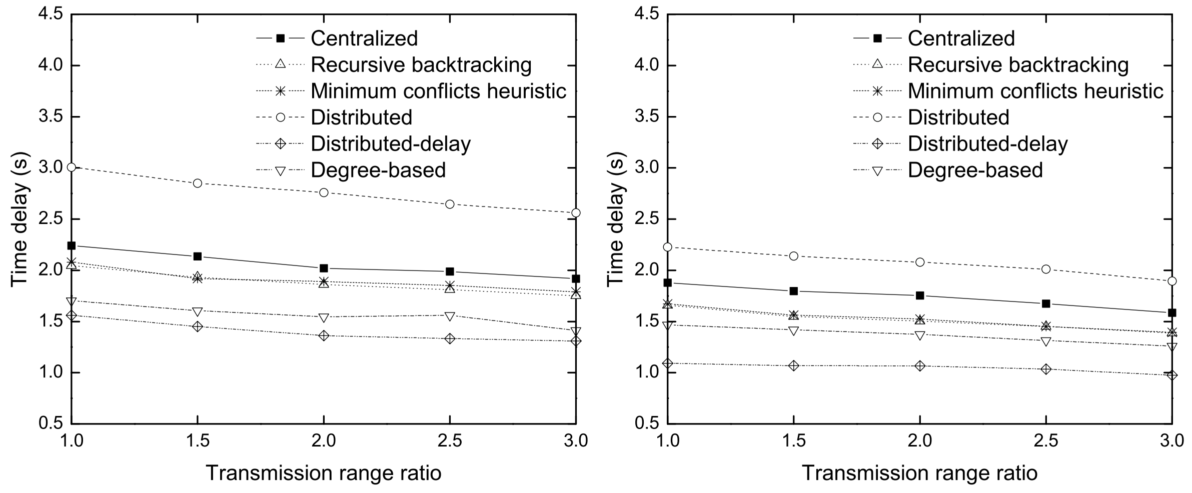
(a) BFS tree (b) DAG graph

Figure 4.13: Average throughput in heterogeneous networks.

significantly as the transmission range ratio σ changes in the BFS tree, while the total energy consumption increases as the transmission range ratio σ increases in the DAG topology.

Figure 4.13 (a) and (b) show the average throughput in heterogeneous networks. In both the BFS tree and DAG topologies, the centralized scheduling with spatial reuse (recursive backtracking and minimum conflicts heuristic) has better performance than the centralized scheduling due to the reduction of the number of time slots assigned. The throughput decreases as the transmission range ratio σ increases, which indicates that the heterogeneous nodes are detrimental to the contiguous link scheduling.

Figure 4.14 (a) and (b) show the average time delay in heterogeneous networks. As the transmission range ratio σ increases, the average time delay decreases in both the BFS tree and DAG topologies because the depth of the network topology reduces as σ increases. In heterogeneous networks, the distributed scheduling with efficient delay scheme still has the best performance.



(a) BFS tree (b) DAG graph

Figure 4.14: Average time delay in heterogeneous networks.

We summarize observations from the simulation results as follows:

- The proposed centralized scheduling with spatial reuse algorithm can reduce the number of time slots assigned, and thus increase network throughput.
- Our proposed distributed algorithms can achieve performance comparable to the centralized algorithms in both homogeneous and heterogeneous networks.
- The distributed scheduling with efficient delay scheme can reduce the network delay.
- In heterogeneous networks, the throughput decreases as the transmission range ratio σ increases, and the average time delay decreases as σ increases.

4.6 Summary

In this chapter, we further investigate the contiguous link scheduling problem to reduce the number of time slots assigned. We propose efficient centralized and distributed algorithms with theoretical performance bounds to the optimum in both homogeneous and heterogeneous networks. Our proposed distributed scheduling with efficient delay algorithm can also reduce the network delay. The simulation results corroborate the theoretical analysis, and show the efficiency of our algorithms in terms of total energy consumption, throughput, and time delay.

Chapter 5

Compact Wakeup Scheduling in TWSNs

In this chapter, we investigate a novel energy-efficient interference-free link scheduling problem called compact wakeup scheduling, in which a node needs to wake up only once to communicate bidirectionally with all its neighbors. The organization of this chapter is as follows. Section 5.1 is the overview of this work. Section 5.2 describes the system model and formulates the compact wakeup scheduling problem. Section 5.3 presents polynomial-time algorithms for trees and grid graphs. Section 5.4 shows the performance evaluation, and finally Section 5.5 summarizes this chapter.

5.1 Overview

In TWSNs, wakeup scheduling [63] is a popular approach to increase the network life, where nodes stay in the sleep state for most of the time, and periodically wake up to check the channel for activity. After the scheduling, each node could operate in a low-duty-cycle mode, and consequently start up several times in a period to communicate with its neighbors.

To reduce the frequency of state transitions, we have identified the contiguous link scheduling problem, and designed energy-efficient centralized and distributed algorithms in Chapters 3 and 4. In the contiguous link scheduling, the incoming links are scheduled together to obtain consecutive time slots. If the network topology is a data gathering tree, each node just needs to start up at most twice in a period: once for receiving data from its children, and once for sending data to its parent. If the network topology is a directed acyclic graph (DAG) where each node v_i has k_i parents, the contiguous link scheduling would require v_i to wake up $k_i + 1$ times as the parent nodes are not scheduled together. In the contiguous link scheduling, the *two-way* (or bidirectional) communication is not taken into consideration. If the two-way communication is considered, this scheduling will require each node to wake up more times in a period. Is it possible to further reduce the frequency of state transitions? An interesting design is to schedule both the incoming links and outgoing links together such that a node could wake up only once and finish all communication tasks with its neighbors consecutively and bidirectionally.

In this chapter, we propose *compact wakeup scheduling*, a novel TDMA approach to the wakeup scheduling problem, to minimize the frequency of state transitions. Compact wakeup scheduling assigns consecutive time slots to all the links incident to a node v_i so that v_i can start up only once to communicate bidirectionally with all its neighbors in one scheduling period T.

Note that we consider the communication links, including both the upstream links and the downstream links, as end-to-end communications. If the downstream links are not end-to-end but broadcast messages, we cannot obtain a compact wakeup scheduling even for a star network. As shown in Figure 5.1(a), links $l_{2,1}, l_{3,1}, l_{4,1}$ are assigned time

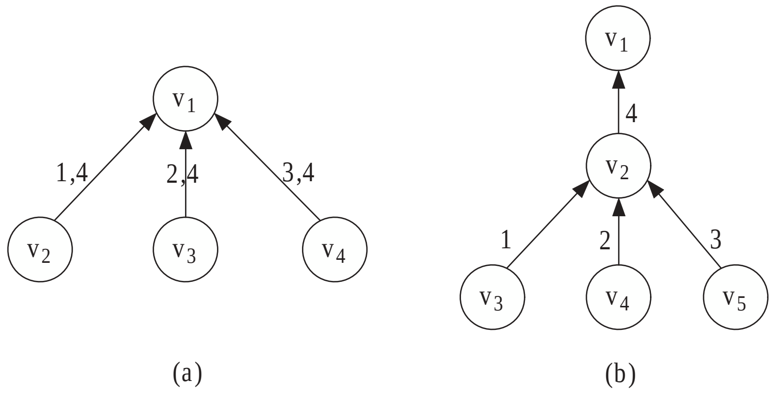


Figure 5.1: (a) Broadcast for the downstream messages, (b) The piggybacking approach

slots 1, 2, 3 respectively, and the broadcast message is assigned time slot 4. Then node v_3 has to start up twice in a period, and thus the scheduling is not a compact wakeup scheduling. Moreover, to piggyback the downstream message to the upstream message using the contiguous link scheduling method does not lead to a compact wakeup scheduling either. As shown in Figure 5.1(b), links $l_{2,1}, l_{3,2}, l_{4,2}, l_{5,2}$ are assigned time slots 4, 1, 2, 3 respectively. Then node v_4 has to start up twice in a period, and thus the scheduling is still not a compact wakeup scheduling.

Apart from reducing the transient time and energy cost in the state transitions, compact wakeup scheduling also has other benefits. The network delay, which is a major concern in the time-critical monitoring systems like that in [71], can be reduced. For instance, a sensor may need to wait until all its neighbors wake up so that it can collect the real time data from these neighbors to make the local computation on these data.

The main contributions of this chapter are summarized as follows.

- We formulate the compact wakeup scheduling problem in TWSNs to minimize the frequency of state transitions, and prove it to be NP-complete.

- We present polynomial-time algorithms using the optimum number of time slots in a period for trees and grid graphs.
- In grid graphs, we point out all the possible coloring patterns and give the lower bound as well as the upper bound of the compact wakeup scheduling.
- We develop simulations to show the efficiency of compact wakeup scheduling.

5.2 System Model and Problem Formulation

In this section, we first present the system model, then formulate the compact wakeup scheduling problem, and finally we present an approach of how to assign a time slot to each transmission link after finding an edge coloring.

5.2.1 System Model

Network Model. We assume that a TWSN has n static sensor nodes equipped with single omni-directional antennas, and all the nodes have the same communication range. The network is represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and $E = \{e_1, e_2, \cdots, e_m\}$ denotes the set of edges referred to all the communication links. If $\{v_i, v_j\} \subseteq V$, the edge $e = (v_i, v_j) \in E$ if and only if v_j is located within the communication range of v_i. We assume that nodes have the ability of data aggregation and can use one time slot to transmit data in one link.

Interference Model. We assume that the interference range is equal to the communication range. Two types of interferences, primary interference and secondary interference [92], exist in the network. The primary interference occurs when a node has more than one communication task in a single time slot. Typical examples are sending and receiving at the same time and receiving from two different transmitters. The secondary interference (or called *the hidden terminal problem* [20]) occurs when a node v_i receives packets from a transmitter v_j and v_i is also within the communication range of another transmitter v_k which is intended for other nodes.

Energy Model. In our energy model, each node operates in three states: active state (transmit, receive and listen), sleep state and transient state (state transition), same as that in Chapter 3.

5.2.2 Problem Formulation

In TDMA wakeup schedulings, each bidirectional communication link l_{ij} is assigned two time slots: one time slot is that v_i is a transmitter and v_j is a receiver, while the other one is that v_j is a transmitter and v_i is a receiver. In the two time slots, nodes v_i and v_j start up, and switch from the sleep state to the active state. After that, nodes v_i and v_j switch to the sleep state again. We can see that node v_i may start up $2w_i$ times to communicate bidirectionally with its neighbors in a scheduling period T in the worst case, where w_i is the number of links incident to v_i. To minimize the frequency of state transitions, we propose a new scheduling approach called compact wakeup scheduling.

Definition 5.1. *Compact wakeup scheduling* is an interference-free wakeup scheduling aiming to assign consecutive time slots to all the links incident to a node v_i, and then v_i needs to start up only once to communicate bidirectionally with all its neighbors.

Compact wakeup scheduling attempts to assign consecutive time slots to all the links incident to a node, but it may fail to find such a scheduling. If it succeeds, the scheduling

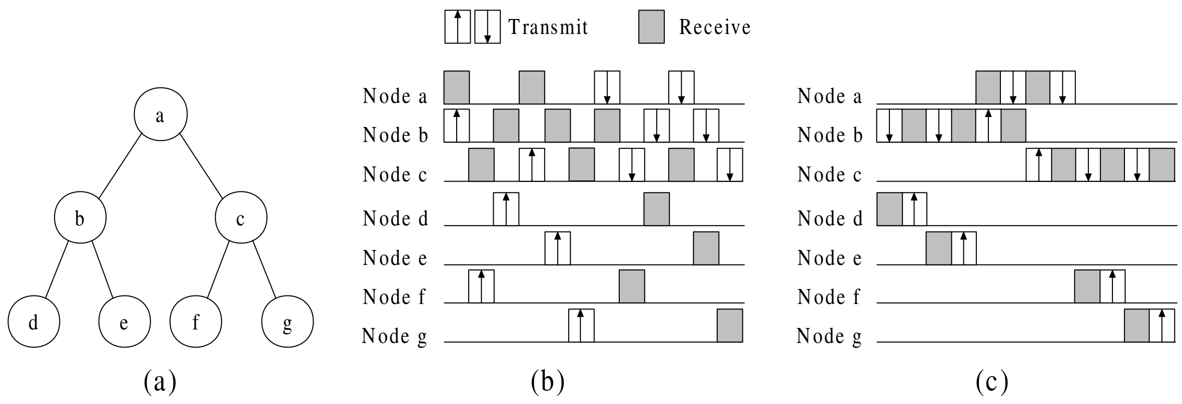


Figure 5.2: Wakeup scheduling and compact wakeup scheduling: (a) Network topology, (b) Wakeup scheduling, (c) Compact wakeup scheduling.

is said to be a *valid* scheduling. If not, the scheduling is said to be not a *valid* scheduling.

In the compact wakeup scheduling, the two time slots assigned to each bidirectional link l_{ij} are adjacent, and node v_i can finish its bidirectional communication with v_j in consecutive time slots. Figure 5.2(a) shows the given network topology. Figure 5.2(b) shows a wakeup scheduling, in which a node starts up numerous times in a period. Figure 5.2(c) shows a compact wakeup scheduling, in which a node could start up only once to communicate bidirectionally with its neighbors. Compact wakeup scheduling can reduce the time for a node to collect the data from its neighbors. As shown in Figure 5.2 (b) and (c), node c needs 5 more time slots to communicate with all its neighbors without the compact wakeup scheduling.

An edge coloring of graph G is called a *valid coloring* if any two adjacent edges of G are assigned different colors. A valid coloring of G is called an *interval (or consecutive) edge-coloring* if, for each vertex v, the colors of edges incident to v form an integer interval.

Theorem 5.1. *The problem of deciding whether a valid compact wakeup scheduling exists for an arbitrary graph G is NP-complete.*

Proof. The compact wakeup scheduling problem is in NP. To verify whether a scheduling is a solution to the compact wakeup scheduling problem, we need to check (i) all the links incident to a node are assigned consecutive time slots; (ii) the scheduling is interference-free. Verifying (i) and (ii) require $O(n)$ and $O(n^2)$ operations respectively, where n is the number of nodes. It is clearly that this verification can be done in polynomial time.

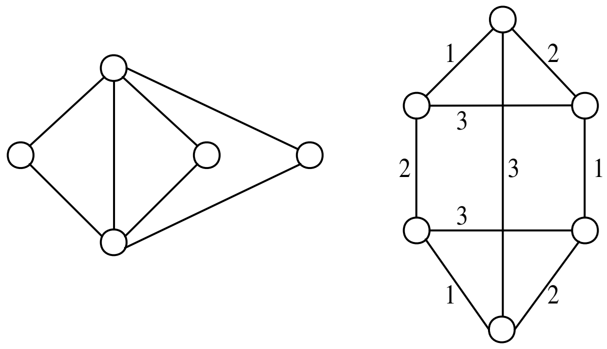
To prove that the compact wakeup scheduling problem is NP-hard, we show a polynomial-time transformation from the problem of deciding the existence of an interval edge-coloring to the compact wakeup scheduling. To decide whether an interval edge-coloring exists for a given graph is proven to be NP-hard in [100].

Suppose $G = (V, E)$ is an undirected graph that has no cycles with $4k + 2$ number of edges, where $k \in N$. For each $e = (u, v) \in E$, construct two directed edges (u, v) and (v, u), and then create a new graph G'. Reference [40] indicates that if a graph has no cycles with $4k + 2$ number of edges, the graph has an interference-free link scheduling by the direction of transmission assignment after an edge coloring. Therefore, there exists an interval edge-coloring for G, if and only if G' has a valid compact wakeup scheduling. Otherwise, there does not exist an interval edge-coloring for G. As the transformation is in polynomial time, the compact wakeup scheduling problem is NP-hard.

Since the compact wakeup scheduling problem is both NP and NP-hard, we conclude the problem is NP-complete. □

Theorem 5.2. *A communication graph G with a valid compact wakeup scheduling has an interval edge-coloring, and belongs to Class 1 graphs.*

Proof. If graph G has a valid compact wakeup scheduling, any node v_i in G can wake up



(a) Class 1 (b) Interval edge-coloring

Figure 5.3: Class 1 graphs without valid compact wakeup schedulings.

once to communicate with all its neighbors. Each two-way communication link can be colored with one color, and then the links incident to one node are assigned consecutive colors. Thus, graph G has an interval edge-coloring. According to [14], graph with an interval edge-coloring belongs to Class 1 graphs *where the edge chromatic number is equal to the maximum degree* Δ *of graph* G. Therefore, graph G is a Class 1 graph. □

Unfortunately, the converse proposition is not true. The graph in Figure 5.3(a) belongs to Class 1 graphs, but has no valid interval edge-colorings, and thus it has no valid compact wakeup schedulings. The Class 1 graphs even with valid interval edge-colorings may not have valid compact wakeup schedulings. For example, the graph in Figure 5.3(b) has an interval edge-coloring, but all valid interval edge-colorings could not avoid the hidden terminal problem. Thus, graphs with valid compact wakeup schedulings are a proper subset of graphs with valid interval edge-colorings, and also a proper subset of Class 1 graphs, as shown in Figure 5.4.

Since not all communication graphs have valid compact wakeup schedulings and the problem of deciding whether a valid scheduling exists for an arbitrary graph is NP-

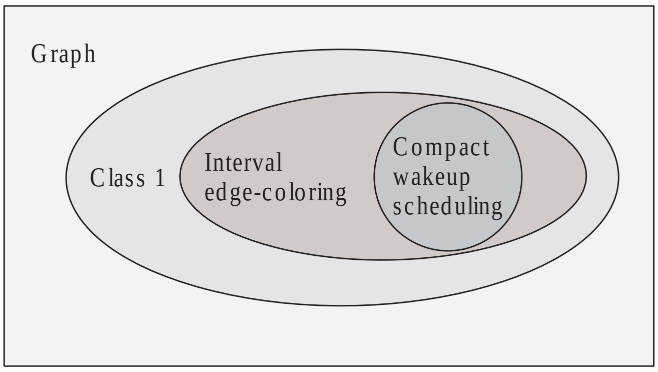


Figure 5.4: The relationship among Class 1 graphs, graphs with valid interval edge-colorings and graphs with valid compact wakeup schedulings.

complete, we will focus on particular graphs, such as tree and grid topologies. Interestingly and surprisingly, we can obtain polynomial-time algorithms using the optimum number of time slots in a period. By minimizing the number of time slots, the overall network throughput can be maximized.

5.2.3 Direction of Transmission Asssignment

In the link scheduling in TWSNs, each edge in the communication graph has two transmission links: one is the upstream link, and the other one is the downstream link. We can easily find an edge coloring of a communication graph using $\Delta + 1$ colors [82], but how can this coloring be used to assign time slots to each transmission link? In [40], each color is mapped to two unique time slots and each transmission link is assigned a time slot according to the direction of transmission assignment (i.e. which end node of edge e will transmit or receive). Using such an assignment, both the hidden terminal problem and the exposed terminal problem can be avoided. When the topologies are acyclic, the overall scheduling requires at most $2(\Delta + 1)$ time slots, where Δ is the maximum degree of a graph. When the topologies have cycles, additional time slots may be needed.

In this chapter, the transmitter is marked with a sign "+" and the receiver is marked with a sign "−". Given a coloring of graph G and a color k, a subgraph $G^k = (V^k, E^k)$ is defined as follows: a) V^k is the set of vertices incident to the edges colored with k. b) E^k is the set of edges with both end vertices in V^k. When a node is assigned a sign "−", the only neighbor assigned a sign "+" in G^k is the neighbor incident to the edge colored with k, and the other neighbors in G^k are individually assigned a sign "−". Then, nodes incident to an edge colored with k always have an opposite sign, and nodes incident to an edge colored with other colors have the same sign. Algorithm 5.1, based on Depth First Search (DFS), can provide a *valid* direction of transmission assignment to each edge in G^k after a valid edge coloring is obtained in acyclic topologies. Such an assignment enables one-way communication. We can reverse the direction of transmission assignment along each edge to support bidirectional communication, and then each edge is assigned two time slots.

Algorithm 5.1 DFS-based sign assignment algorithm [40]

```
Input: A subgraph $G^k = (V^k, E^k)$.
Output: A valid direction of transmission assignment.
 1: Start by visiting any node in $V^k$, and assign a sign "+" to it.
 2: Initiate a Depth First Search (DFS) procedure.
 3: while there are unvisited nodes do
 4:     Let edge $e$ be traversed from a visited node $v_i$ to an unvisited node $v_j$ using the
        DFS procedure.
 5:     if $e$ is colored with $k$ then
 6:         Assign $v_j$ the sign opposite to $v_i$.
 7:     else
 8:         Assign $v_j$ the sign same to $v_i$.
 9:     end if
10: end while
```

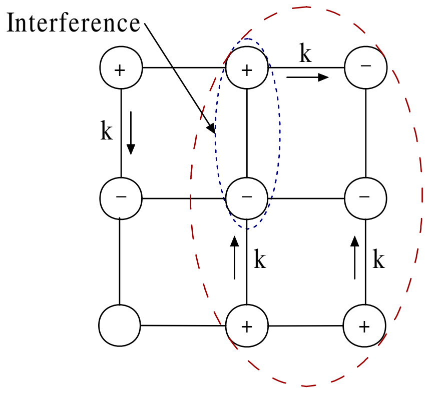


Figure 5.5: A cycle that has odd number of edges with color k cannot be assigned a valid direction of transmission.

Gandham et al. [40] prove that a valid direction of transmission assignment exists in acyclic topologies (e.g. tree graphs). If a valid edge coloring is obtained in the topologies which are not acyclic (e.g. grid graphs), a valid direction of transmission assignment may not exist due to the hidden terminal problem, as shown in Figure 5.5. Interestingly, Gandham et al. [40] also prove that all the nodes in a cycle of G^k can be given a valid sign "+" or "−" if and only if there are an even number of edges with color k in the cycle.

5.3 Compact Wakeup Scheduling Algorithms

In this section, we propose polynomial-time algorithms to produce valid compact wakeup schedulings for tree and grid topologies, which are commonly used in TWSNs [16, 81, 83, 101, 129]. The proposed algorithms can achieve the optimum number of time slots in a period for tree and grid topologies.

5.3.1 Trees

To obtain a valid compact wakeup scheduling of a tree, we first obtain an interval edge-coloring of a tree. After that, we try to assign time slots to each edge, ensuring interference-free by the direction of transmission assignment.

If graph G is a tree of degree Δ, we could get an interval edge-coloring with Δ colors for G using Algorithm 5.2 [45]: We first color any edge with 1, then find an uncolored edge e adjacent to an already colored edge and assign e with a consecutive color until all the edges are colored. In the coloring process, when coloring a new uncolored edge, the consecutiveness of edge-coloring remains invariant, and the edges already colored form a consecutively colored subgraph. After all edges are colored, we could get an interval edge-coloring and the total number of colors assigned is Δ.

Algorithm 5.2 Interval edge-coloring of a tree [45]

```
Input: A tree $G = (V, E)$.
Output: A valid interval edge-coloring with $\Delta$ colors.
1: Color any edge with 1.
2: while there are uncolored edges do
3:     Find an uncolored edge $e$ whose end vertex $v$ is adjacent to an already colored
       edge. Let $\{a, \cdots, b\}$ be the interval of colors assigned to $v$.
4:     if $a > 1$ then
5:         Color edge $e$ with $a - 1$.
6:     else
7:         Color edge $e$ with $b + 1$.
8:     end if
9: end while
```

We now describe how the interval edge-coloring is used to assign time slots to each edge in Algorithm 5.3. The idea is to map color k to two consecutive time slots $\{2k - 1, 2k\}$, and use Algorithm 5.1 to determine a valid direction of transmission assignment for time slot $2k-1$, and then reverse the direction of transmission along each edge to obtain the other assignment for time slot $2k$.

Algorithm 5.3 Compact wakeup scheduling of a tree

```
Input: A tree $G = (V, E)$.
Output: A valid compact wakeup scheduling.
1: Use Algorithm 5.2 to obtain a valid interval edge-coloring with $\Delta$ colors for $G$.
2: for $k = 1$ to $\Delta$ do
3:     Map color $k$ to two consecutive time slots $\{2k-1, 2k\}$.
4:     Use Algorithm 5.1 to determine a valid direction of transmission assignment for
       time slot $2k-1$.
5:     Reverse the direction of transmission along each edge to obtain the other assign-
       ment for time slot $2k$.
6: end for
```

In Figure 5.6, link l_{ab} and l_{ce} are assigned the same color "1" in the interval edge-coloring, while time slot ts_1 and ts_2 are allocated for color "1". If time slot ts_1 is assigned in the directions of transmission as shown in Figure 5.6(a), the hidden terminal problem would happen because the reception at node v_b is garbled due to the collision of transmission from nodes v_a and v_c. Alternatively, if time slot ts_1 is assigned in the directions of transmission as shown in Figure 5.6(b), the hidden terminal problem could be avoided. Similarly, time slot ts_2 is assigned in the reverse directions of transmission as shown in Figure 5.6(c). Inspired by this, we should determine the directions of transmission along each link carefully to avoid the hidden terminal problem, i.e., determine a node when to transmit and when to receive.

A tree does not have any cycles, and thus it is always possible to obtain a valid compact wakeup scheduling. Algorithm 5.1, based on Depth First Search (DFS), can provide a

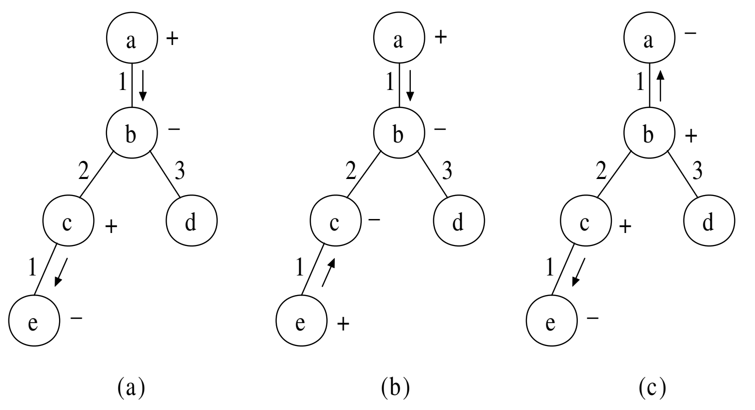


Figure 5.6: Compact wakeup scheduling of a tree: (a) Hidden terminal problem, (b) Avoid the hidden terminal problem, (c) Avoid the exposed terminal problem.

valid direction of transmission assignment to G^k. Note that the time slot assignment also avoids the exposed terminal problem [20], as shown in Figure 5.6(c).

Definition 5.2. The span of a valid compact wakeup scheduling of graph G is the number of colors assigned. The minimum and maximum span over all valid compact wakeup schedulings of G are denoted by $\chi_{cw}(G)$ and $\zeta_{cw}(G)$ respectively.

As any valid coloring in a tree requires at least Δ colors and an interval edge-coloring can be obtained using Δ colors, $\chi_{cw}(G)$ is equal to Δ. Then, the number of time slots assigned in the compact wakeup scheduling is 2Δ, which is the optimum number of time slots. Algorithm 5.3 describes the compact wakeup scheduling of a tree. Both the interval edge-coloring of a tree and the time slot assignment can be obtained using $O(n)$, where n is the number of vertices in a tree. Thus, the time complexity of the compact wakeup scheduling of a tree is $O(n)$.

Theorem 5.3. *The number of time slots assigned by Algorithm 5.3 is* 2Δ, *which is optimum.*

5.3.2 Grid Graphs

A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$) is a square lattice graph composed of $\mathcal{VH}$ vertices. The grid graph has $\mathcal{H}$ vertical paths and $\mathcal{V}$ horizontal paths, where each vertical path consists of $\mathcal{V}$ vertices and each horizontal path consists of $\mathcal{H}$ vertices.

Definition 5.3. In a $\mathcal{V} \times \mathcal{H}$ grid graph, $\overline{V}_{ij}$ $(1 \leq i \leq \mathcal{V} - 1,\ 1 \leq j \leq \mathcal{H})$ denotes the i^{th} vertical edge in the j^{th} vertical path, and $\overline{H}_{ij}$ $(1 \leq i \leq \mathcal{V},\ 1 \leq j \leq \mathcal{H} - 1)$ denotes the j^{th} horizontal edge in the i^{th} horizontal path.

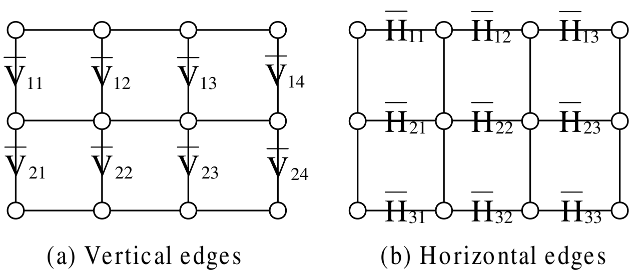


Figure 5.7: Vertical and horizontal edges in grid graphs.

Sample grids with labeled vertical and horizontal edges are illustrated in Figure 5.7 (a) and (b). $\overline{V}_{ij}$ is called *parallel* to $\overline{V}_{mn}$ if $i = m$, $\overline{H}_{ij}$ is called *parallel* to $\overline{H}_{mn}$ if $j = n$. For example, $\overline{H}_{11}$, $\overline{H}_{21}$ and $\overline{H}_{31}$ are parallel in Figure 5.7(b).

Grid graphs can be consecutively colored with Δ colors, and one interval edge-coloring approach is given below: For a $\mathcal{V} \times \mathcal{H}$ grid graph, let c be a consecutive coloring of each horizontal path with colors 2 and 3. For each $i = 1, 2, \cdots, \mathcal{V}$, we color the edges of i^{th} horizontal path according to c. Let $\{a, \cdots, b\}$ be an interval of colors assigned at each vertex in the corresponding horizontal path, then edge $\overline{V}_{1j}$ is colored with $a-1$, $\overline{V}_{2j}$ with $b+1$, $\overline{V}_{3j}$ with $a-1$, and so forth, where $1 \leq j \leq \mathcal{H}$. By repeating this for all edges, we

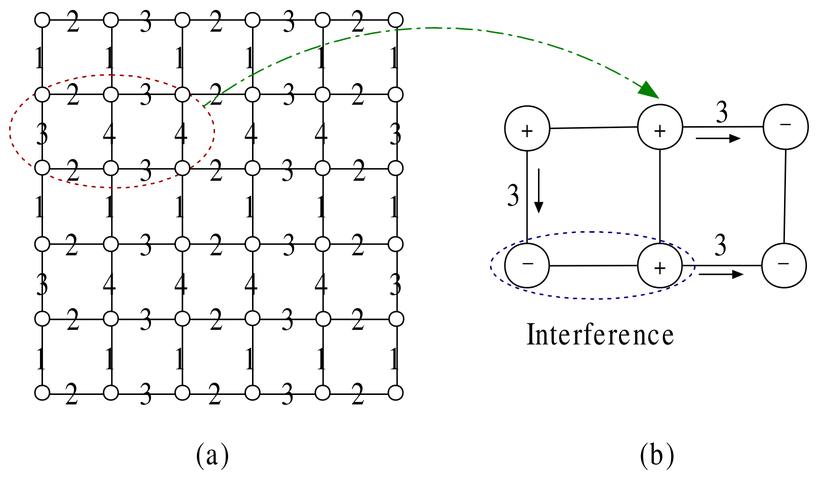


Figure 5.8: An interval edge-coloring of a grid graph: (a) Interval edge-coloring, (b) Hidden terminal problem.

could obtain a interval edge-coloring of G, and a sample of the edge-coloring is shown in Figure 5.8(a).

A valid direction of transmission assignment can be obtained to avoid the hidden terminal problem using Algorithm 5.1 in acyclic subgraphs G^k. But grid graphs contain cycles, a valid assignment does not exist if we use the interval edge-coloring approach above. For example, this edge-coloring cannot avoid the hidden terminal problem as shown in Figure 5.8(b). Interestingly, Gandham et al. [40] prove that all the nodes in a cycle of G^k can be given a valid sign "+" or "−" if and only if there are an even number of edges with color k in the cycle.

If the edges colored with "3" in the cycle of Figure 5.8(b) are assigned with other colors, the consecutiveness of the colors assigned to the edges incident to one node cannot be held. Our solution for a grid graph first considers the property of the hidden terminal problem in the grid graph, and then deals with the consecutiveness of the edge-coloring. Our key results for grid graphs are summarized below:

(1) We obtain an interval edge-coloring according to the parity of $\mathcal{V}$ and $\mathcal{H}$.

- $*$ If both $\mathcal{V}$ and $\mathcal{H}$ are even, $\chi_{cw}(G) = 4$.
- $*$ If one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd, $\chi_{cw}(G) = 5$.
- $*$ If both $\mathcal{V}$ and $\mathcal{H}$ are odd, $\chi_{cw}(G) = 6$.

(2) We point out all the possible coloring patterns.

(3) We give the upper bound of the compact wakeup scheduling.

Definition 5.4. In a grid graph, the maximum degree of vertices is $\Delta = 4$. The vertices of degree 4 are *inner vertices*, the vertices of degree 2 or 3 are *boundary vertices*, the edges incident to at least one inner vertex are *inner edges*, and the edges incident to two boundary vertices are *boundary edges*.

Definition 5.5. In the compact wakeup scheduling of a grid graph, the colors assigned to the inner edges incident to an inner vertex form an interval of 4 integers. When the total number of colors assigned is less than 8, certain color must appear in one of the inner edges and this color is referred to as a *critical color*.

In grid graphs, if the total number of colors assigned is M $(4 \leq M \leq 7)$, the number of critical colors is $8 - M$. For example, if $M = 4$, the set of critical colors is $\{1, 2, 3, 4\}$; if $M = 5$, the set of critical colors is $\{2, 3, 4\}$; if $M = 6$, the set of critical colors is $\{3, 4\}$.

Lemma 5.1. *If K is a critical color assigned to an inner edge e incident to two inner vertices in the compact wakeup scheduling of a grid graph, the inner edges parallel to e are all colored with K.*

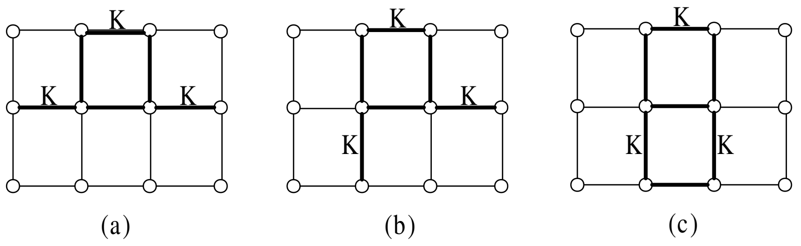


Figure 5.9: Invalid colorings in the compact wakeup scheduling.

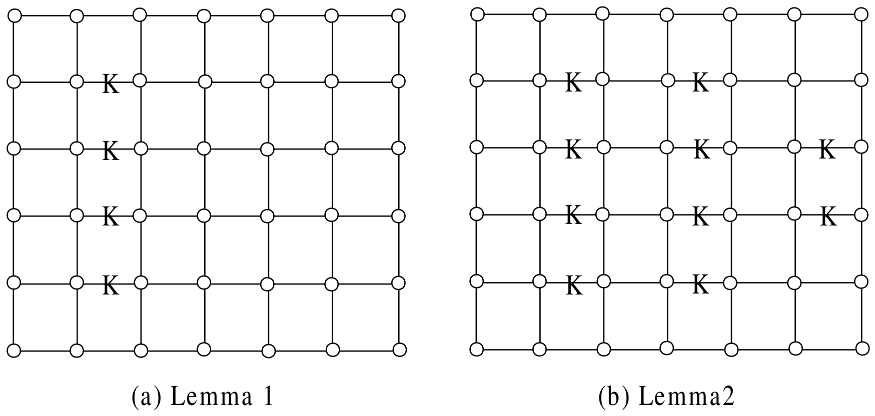
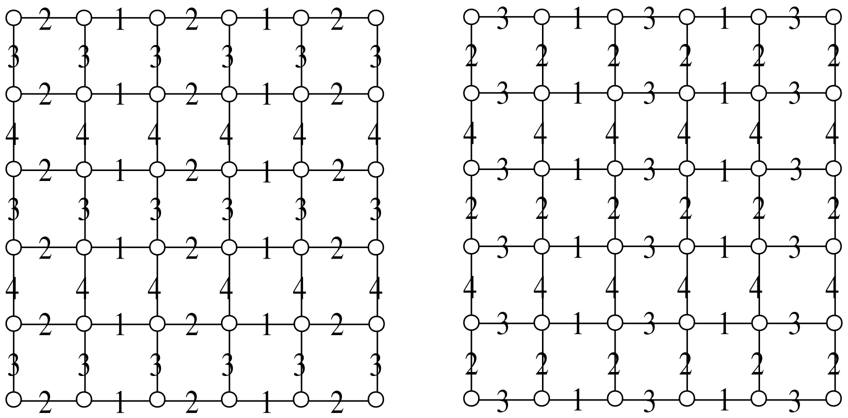


Figure 5.10: Coloring pattern in the compact wakeup scheduling.

Proof. Without loss of generality, we assume that an inner horizontal edge $\overline{H}_{ij}$ $(2 \leq i \leq \mathcal{V}-1,\ 2 \leq j \leq \mathcal{H}-2)$ is colored with K in a $\mathcal{V} \times \mathcal{H}$ grid graph. The cases of colorings shown in Figure 5.9 would lead to odd number of edges with color K in the subgraph G^K (see the thick lines in Figure 5.9), and no feasible direction of transmission can be obtained. Since K is a critical color, $\overline{H}_{(i+1)j}$ $(i+1 \leq \mathcal{V}-1)$ must be colored with K. By applying recursion, the horizontal edges $\overline{H}_{mj}$ $(2 \leq m \leq \mathcal{V}-1)$ are in a parallel pattern, as shown in Figure 5.10(a). □

Lemma 5.2. *If K is a critical color, the inner edges colored with K are in an interlined pattern.*

Proof. Without loss of generality, we assume an inner horizontal edge $\overline{H}_{ij}$ $(2 \leq i \leq \mathcal{V} -$



(a) even×even (pattern 1) (b) even×even (pattern 2)

Figure 5.11: The coloring patterns in the compact wakeup scheduling (Both $\mathcal{V}$ and $\mathcal{H}$ are even).

1, $2 \leq j \leq \mathcal{H}-2$) is colored with K in a $\mathcal{V} \times \mathcal{H}$ grid graph. According to Lemma 5.1, the horizontal edges $\overline{H}_{mj}$ ($2 \leq m \leq \mathcal{V}-1$) are colored with K. Let $k = j+2$ ($k \leq \mathcal{H}-2$), $\overline{H}_{3k}$ must be colored with K, since K is a critical color and $\overline{H}_{3(k-1)}$, $\overline{V}_{2k}$ as well as $\overline{V}_{3k}$ cannot be colored with K. According to Lemma 5.1, the horizontal edges $\overline{H}_{mk}$ ($2 \leq m \leq \mathcal{V}-1$) are colored with K, and the result still holds when $k = j-2$ ($k \geq 2$). By applying recursion, the horizontal edges $\overline{H}_{mk}$ ($k = j \pm 2n$, $n \in N$, $2 \leq m \leq \mathcal{V}-1$, $2 \leq k \leq \mathcal{H}-2$) are colored with K. If $k = 1$ (*or* $\mathcal{H}-1$) and $k = j \pm 2n$, $n \in N$, the horizontal edges $\overline{H}_{mk}$ ($3 \leq m \leq \mathcal{V}-2$) are colored with K, since K is a critical color. Hence, the inner edges colored with a critical color are in an interlined pattern, as shown in Figure 5.10(b). □

Theorem 5.4. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are even) can be consecutively colored with 4 colors in the compact wakeup scheduling, and the possible colorings must be the patterns as shown in Figure 5.11, and $\chi_{cw}(G) = 4$.*

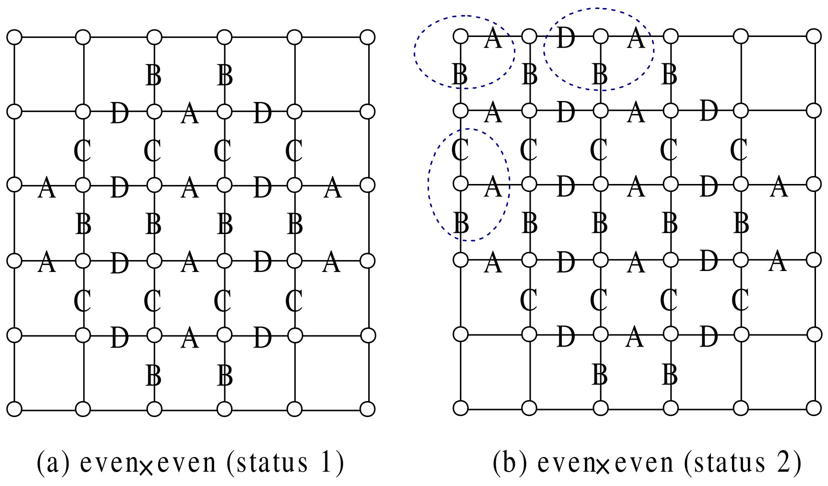
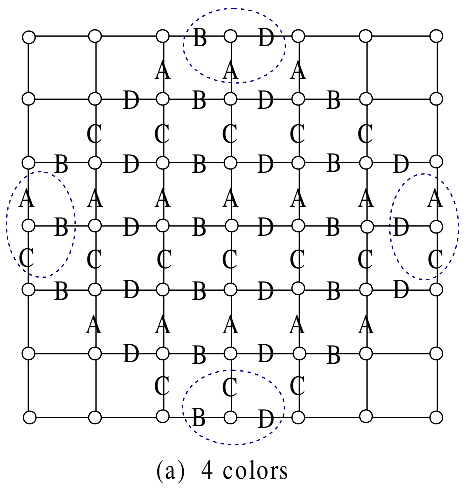


Figure 5.12: The coloring in the compact wakeup scheduling (Both $\mathcal{V}$ and $\mathcal{H}$ are even).

Proof. Figure 5.11 (a) and (b) show two possible colorings in the compact wakeup scheduling, if both $\mathcal{V}$ and $\mathcal{H}$ are even. Since the edges with the same color are in a parallel and interlined pattern, there are an even number of edges with color k ($1 \leq k \leq 4$) in a cycle in the subgraph G^k and then the scheduling could avoid the hidden terminal problem. Therefore, $\chi_{cw}(G) = 4$.

As $\chi_{cw}(G)$ is equal to 4, we assume the four critical colors are A, B, C and D. According to Lemma 5.1 and Lemma 5.2, A, B, C and D are all in a parallel and interlined pattern shown as the status 1 in Figure 5.12(a). To avoid the hidden terminal problem, $\overline{H}_{13}$ cannot be colored with D or C, and can only be colored with A. Then $\overline{H}_{12}$ must be colored with D. Similarly, $\overline{V}_{21}$ and $\overline{V}_{31}$ are colored with C and B respectively. Then $\overline{H}_{11}$, $\overline{V}_{11}$, $\overline{H}_{21}$ and $\overline{V}_{12}$ are colored with A, B, A and B respectively. We can color other edges in a similar way. In the status 2 shown in Figure 5.12(b), we can see the color sets $\{A, B\}$, $\{A, B, D\}$ and $\{A, B, C\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Therefore, $\{A, B, C\}$ and $\{A, B, D\}$ belong to $\{1, 2, 3\}$ and $\{2, 3, 4\}$, $\{A, B\}$



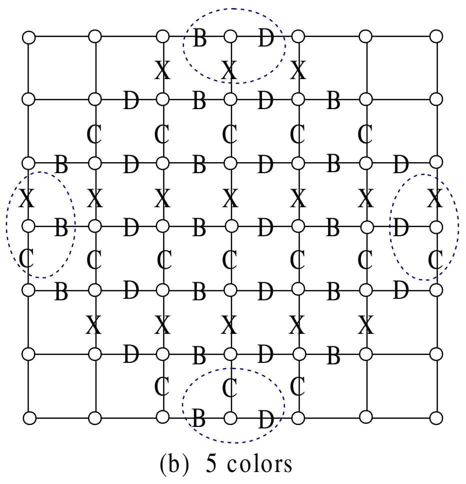
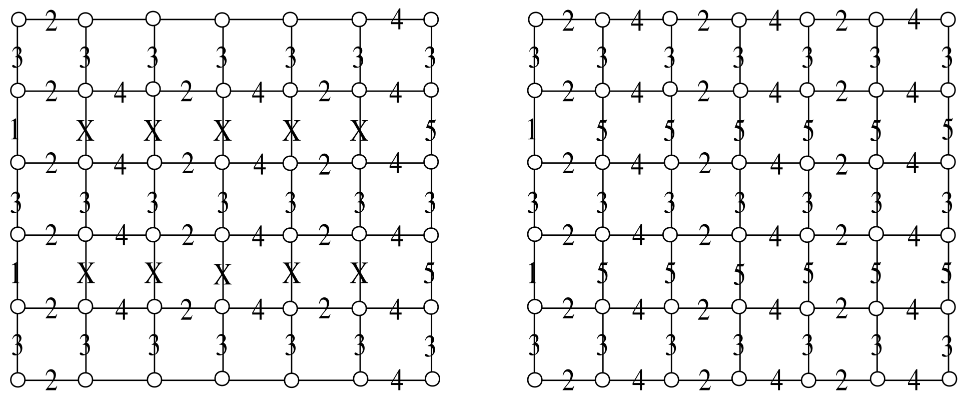


Figure 5.13: The colorings in the compact wakeup scheduling using 4 and 5 colors (Both $\mathcal{V}$ and $\mathcal{H}$ are odd).

belongs to $\{2, 3\}$. For C and D are symmetrical, we can get $C = 4$ and $D = 1$. For the case that $A = 2$ and $B = 3$, the coloring pattern is Figure 5.11(a). For the case that $A = 3$ and $B = 2$, the coloring pattern is Figure 5.11(b). □

Lemma 5.3. *A $\mathcal{V} \times \mathcal{H}$ grid graph($3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are odd) cannot be consecutively colored with 4 or 5 colors in the compact wakeup scheduling.*

Proof. 1) If the grid could be consecutively colored with 4 colors A, B, C and D, the four colors belonging to $\{1, 2, 3, 4\}$ are all critical colors. For an inner vertex has 4 incident inner edges, the inner edges are colored with A, B, C and D respectively. According to Lemma 5.1 and Lemma 5.2, A, B, C and D are all in a parallel and interlined pattern, and the coloring is shown in Figure 5.13(a). As the colors assigned to the edges incident to the vertices in the dashed circles must be consecutive, the color sets $\{A, B, C\}$, $\{B, C, D\}$, $\{A, C, D\}$ and $\{A, B, D\}$ must consist of three consecutive numbers. However, $\{1, 2, 3\}$ and $\{2, 3, 4\}$ are the only two possible cases with three consecutive numbers, which leads



(a) even×odd (general pattern) (b) even×odd (pattern)

Figure 5.14: The general coloring pattern and coloring pattern in the compact wakeup scheduling (One of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd. The uncolored edges depend on the edges colored with X, and $X = 1$ *or* 5).

to a contradiction.

2) If the grid could be consecutively colored with 5 colors, B, C and D belonging to $\{2, 3, 4\}$ are critical colors and the non-critical color 1 or 5 is denoted by X. For an inner vertex has 3 incident critical inner edges, the inner edges are colored with B, C and D respectively. According to Lemma 5.1 and Lemma 5.2, B, C and D are all in a parallel and interlined pattern, and the coloring is shown in Figure 5.13(b). Since the colors assigned to the edges incident to the vertices in the dashed circles must be consecutive, the color sets $\{B, C, X\}$, $\{B, C, D\}$, $\{C, D, X\}$ and $\{B, D, X\}$ must consist of three consecutive numbers. However, $\{1, 2, 3\}$, $\{2, 3, 4\}$ and $\{3, 4, 5\}$ are the only three possible cases with three consecutive numbers, which leads to a contradiction. □

Similarly, we could get the following lemma:

Lemma 5.4. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd) cannot be consecutively colored with 4 colors in the compact wakeup scheduling.*

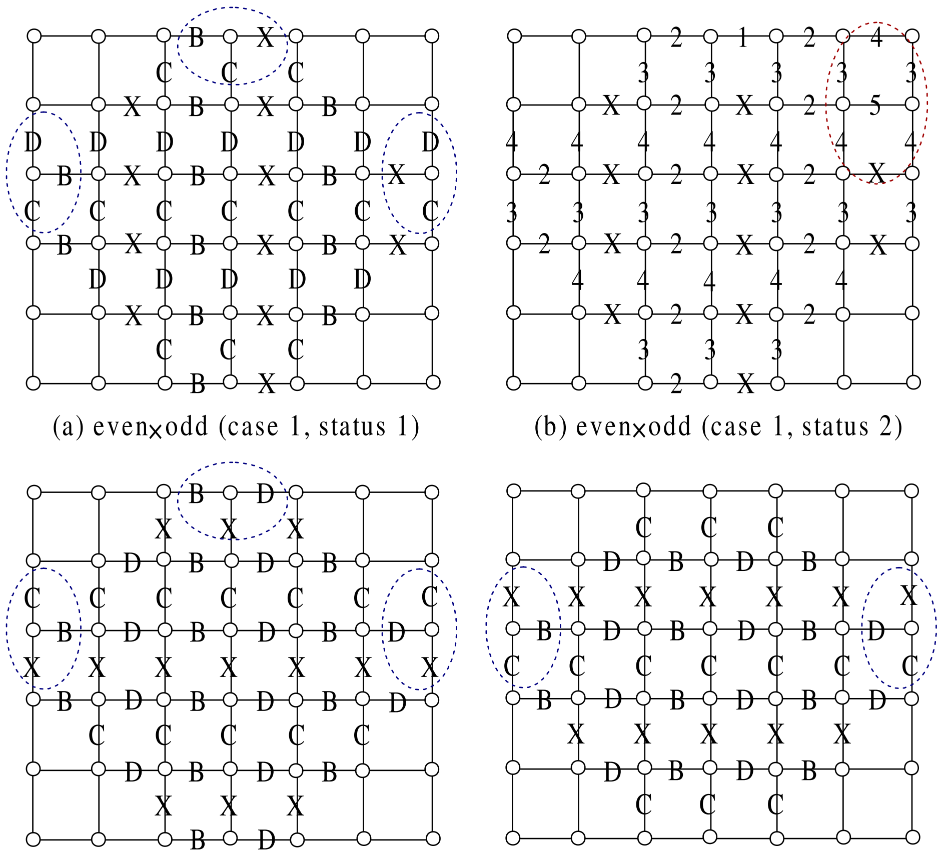

(a) even×odd (case 1, status 1)
(b) even×odd (case 1, status 2)
(c) even×odd (case 2)
(d) even×odd (case 3)

Figure 5.15: The colorings in the compact wakeup scheduling (One of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd).

Theorem 5.5. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd) can be consecutively colored with 5 colors in the compact wakeup scheduling, and the possible coloring must be the pattern as shown in Figure 5.14(a), and $\chi_{cw}(G) = 5$.*

Proof. Figure 5.14(b) shows a possible coloring in the compact wakeup scheduling by determining the colors for the rest uncolored edges in Figure 5.14(a), if one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd. Since the edges with the same color are in a parallel and interlined pattern, there are an even number of edges with color k ($1 \leq k \leq 5$) in a cycle in the subgraph G^k and then the scheduling could avoid the hidden terminal problem. By combining with Lemma 5.4, $\chi_{cw}(G) = 5$.

Since $\chi_{cw}(G)$ is equal to 5, we assume B, C and D belonging to $\{2, 3, 4\}$ are critical colors and the non-critical color 1 or 5 is denoted by X. As an inner vertex has 3 incident critical inner edges, Figure 5.15 (a), (c) and (d) are the possible coloring patterns.

Case 1: In the status 1 shown in Figure 5.15(a), $\overline{H}_{14}$ cannot be colored with B, C or D, and can only be colored with X. Then $\overline{H}_{13}$ must be colored with B. Similarly, $\overline{V}_{21}$, $\overline{V}_{31}$, $\overline{V}_{27}$, $\overline{V}_{37}$ are colored with D, C, D and C respectively. We can see that the color sets $\{B, C, X\}$, $\{B, C, D\}$ and $\{C, D, X\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Then, $\{B, C, X\}$ and $\{C, D, X\}$ belong to $\{1, 2, 3\}$ and $\{3, 4, 5\}$. Then, we can get $C = 3$. If $B = 2$ and $D = 4$, $\overline{H}_{15}$, $\overline{V}_{16}$, $\overline{H}_{26}$ and $\overline{V}_{17}$ are colored with 2, 3, 5 and 3 respectively, shown as the status 2 in Figure 5.15(b). Then $\overline{H}_{16}$ can only be colored with 4, which leads to interferences in the dashed circle. Similarly, if $B = 4$ and $D = 2$, we cannot get an interference-free scheduling either. Hence, the coloring pattern in Figure 5.15(a) is not valid.

Case 2: In Figure 5.15(c), $\overline{H}_{14}$ cannot be colored with B, C or X, and can only be colored with D. Then $\overline{H}_{13}$ must be colored with B. Similarly, $\overline{V}_{21}$, $\overline{V}_{31}$, $\overline{V}_{27}$, $\overline{V}_{37}$ are colored with C, X, C and X respectively. We can see that the color sets $\{B, D, X\}$, $\{B, C, X\}$ and $\{C, D, X\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Moreover, $B, C, D \in \{2, 3, 4\}$ are consecutive. However, $\{1, 2, 3\}$, $\{2, 3, 4\}$ and $\{3, 4, 5\}$ are the only three possible cases with three consecutive numbers. Hence, the coloring pattern in

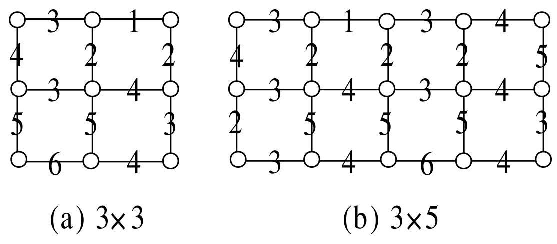


Figure 5.16: The coloring patterns in the compact wakeup scheduling ($\mathcal{V} = 3$ and $\mathcal{H}$ is odd).

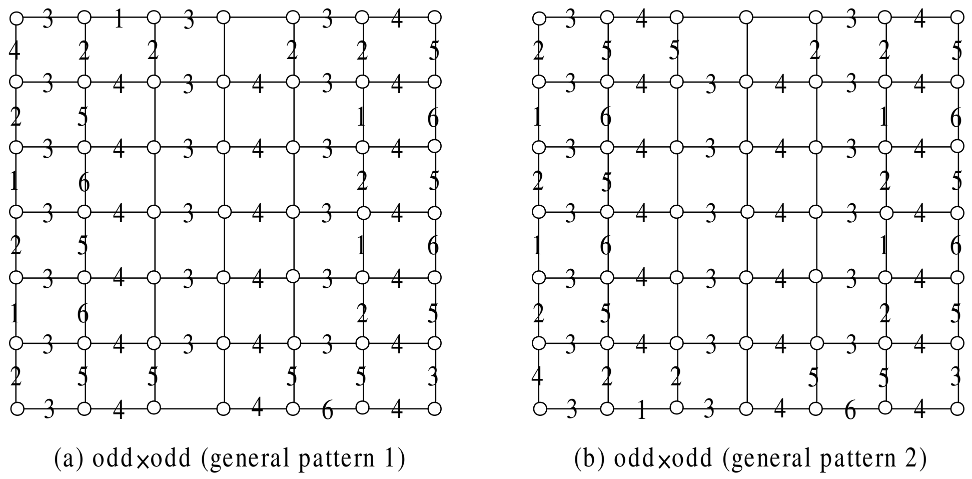
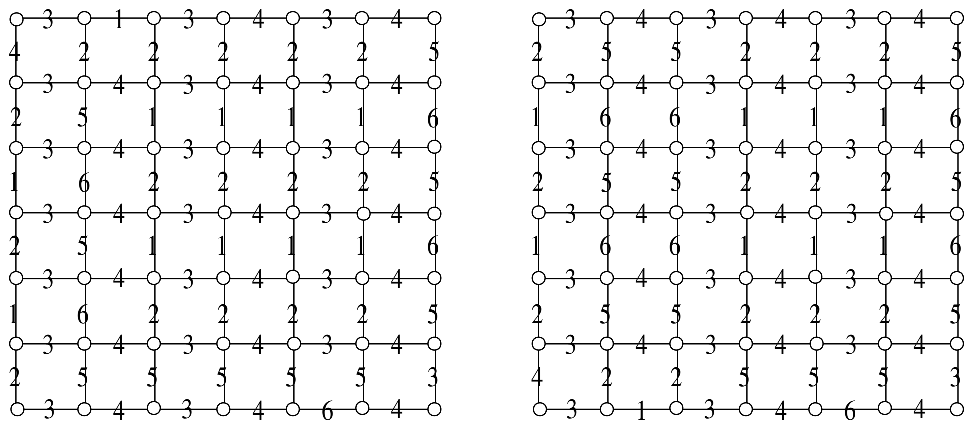

(a) odd×odd (general pattern 1)
(b) odd×odd (general pattern 2)

Figure 5.17: The general coloring patterns in the compact wakeup scheduling (Both $\mathcal{V}$ and $\mathcal{H}$ are odd. The uncolored edges have various alternatives).

Figure 5.15(c) is not valid.

Case 3: In Figure 5.15(d), $\overline{V}_{21}$ cannot be colored with B, C or D, and can only be colored with X. Then $\overline{V}_{31}$ must be colored with C. Similarly, $\overline{V}_{27}$ and $\overline{V}_{37}$ are colored with X and C respectively. We can see that the color sets $\{B, C, X\}$ and $\{C, D, X\}$ must consist of consecutive numbers since these colors assigned to the edges incident to the vertices in the dashed circles must be consecutive. Then $C = 3$. For B and D are symmetrical, we can get $B = 2$ and $D = 4$. By assigning the possible colors in other edges, the coloring pattern in Figure 5.14(a) is obtained. □



(a) odd×odd (pattern 1) (b) odd×odd (pattern 2)

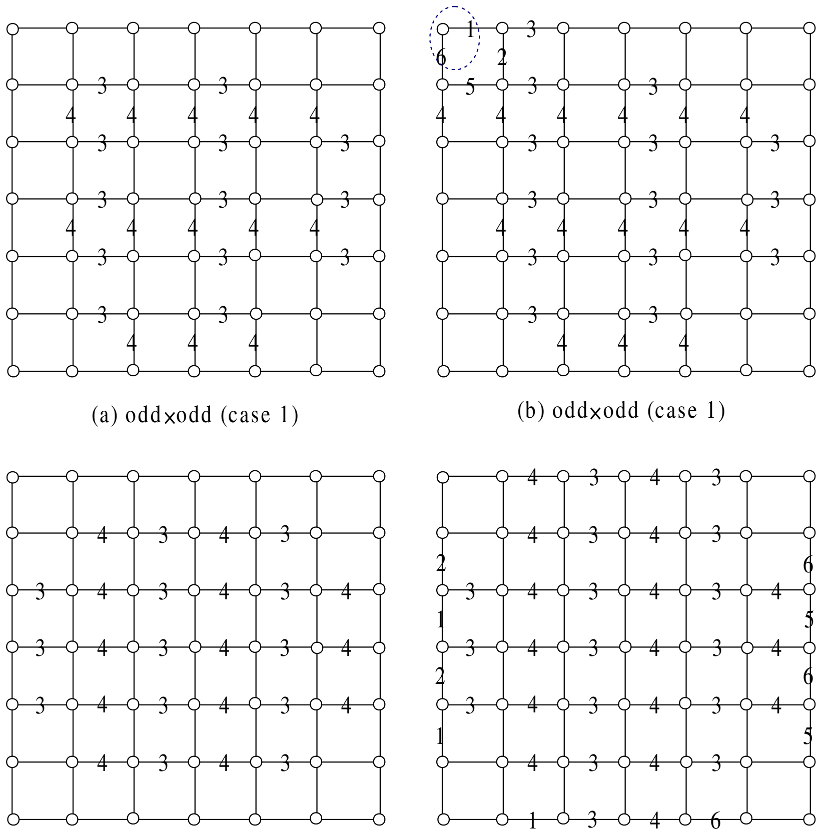
Figure 5.18: The coloring patterns in the compact wakeup scheduling (Both $\mathcal{V}$ and $\mathcal{H}$ are odd).

Theorem 5.6. *A $\mathcal{V} \times \mathcal{H}$ grid graph ($3 \leq \mathcal{V} \leq \mathcal{H}$, both $\mathcal{V}$ and $\mathcal{H}$ are odd) can be consecutively colored with 6 colors in the compact wakeup scheduling, and the possible colorings must be the patterns as shown in Figure 5.17, and $\chi_{cw}(G) = 6$.*

Proof. Figure 5.18 (a) and (b) shows two possible colorings in the compact wakeup scheduling by determining the colors for the rest uncolored edges in Figure 5.17 (a) and (b), if both $\mathcal{V}$ and $\mathcal{H}$ are odd. Particularly, if $\mathcal{V} = 3$, the possible colorings are shown in Figure 5.16 (a) and (b). Since there are even number of edges with color k ($1 \leq k \leq 6$) in a circle in the subgraph G^k and then the scheduling could avoid the hidden terminal problem. According to Lemma 5.3, $\chi'(G) = 6$.

Since the grid graph can be consecutively colored with 6 colors, 3 and 4 are critical colors. For an inner vertex has two incident critical inner edges, Figure 5.19 (a) and (c) are the possible coloring patterns.

Case 1: In Figure 5.19(a), $\overline{V}_{21}$, $\overline{V}_{31}$ and $\overline{H}_{31}$ cannot be colored with 3, but can only


(a) odd×odd (case 1) (b) odd×odd (case 1)

(c) odd×odd (case 2) (d) odd×odd (case 2)

Figure 5.19: The colorings in the compact wakeup scheduling (both $\mathcal{V}$ and $\mathcal{H}$ are odd).

be colored with $\{4, 5, 6\}$. For $\overline{V}_{31}$ and $\overline{H}_{31}$ cannot be colored with 4, $\overline{V}_{21}$ must be colored with 4. Similarly, $\overline{H}_{12}$ must be colored with 3. Then $\overline{V}_{11}$, $\overline{V}_{21}$ and $\overline{H}_{21}$ must be colored with $\{4, 5, 6\}$, and $\overline{H}_{11}$, $\overline{H}_{12}$ and $\overline{V}_{12}$ must be colored with $\{1, 2, 3\}$. For $\overline{V}_{12}$, $\overline{V}_{22}$, $\overline{H}_{21}$ and $\overline{H}_{22}$ are consecutively colored, $\overline{V}_{12}$ is colored with 2 and $\overline{H}_{21}$ is colored with 5. Then, $\overline{V}_{11}$ is colored with 6 and $\overline{H}_{11}$ is colored with 1, which leads to an inconsecutive coloring, as shown in Figure 5.19(b). Hence, Figure 5.19(a) is not a possible coloring.

Case 2: In Figure 5.19(c), $\overline{V}_{21}$, $\overline{V}_{23}$ and $\overline{H}_{31}$ cannot be colored with 4, but can only be colored with $\{1, 2, 3\}$; $\overline{V}_{27}$, $\overline{V}_{37}$ and $\overline{H}_{36}$ cannot be colored with 3, but can only be $\{4, 5, 6\}$. According to the symmetrical property, we suppose $\overline{V}_{27}$ and $\overline{V}_{37}$ are colored

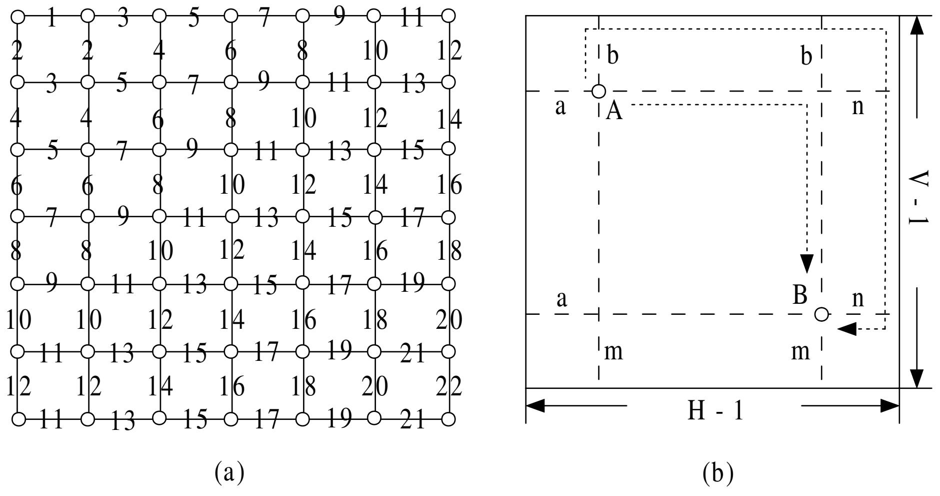


Figure 5.20: $\zeta_{cw}(G)$ in the compact wakeup scheduling: (a) Lower bound of $\zeta_{cw}(G)$, (b) Upper bound of $\zeta_{cw}(G)$.

with 6 and 5 respectively. If $\overline{V}_{21}$ is colored with 2 and $\overline{V}_{31}$ is colored with 1, we can get Figure 5.19(d). By assigning the possible colors in other edges, the coloring pattern in Figure 5.17(a) is obtained. If $\overline{V}_{21}$ is colored with 1 and $\overline{V}_{31}$ is colored with 2, we can get the coloring pattern in Figure 5.17(b).

Hence, the possible colorings must be the patterns as shown in Figure 5.17 (a) and (b), and $\chi'(G) = 6$. $\square$

Theorem 5.7. *In the compact wakeup scheduling of a* $\mathcal{V} \times \mathcal{H}$ *grid graph* ($3 \leq \mathcal{V} \leq \mathcal{H}$), $2\mathcal{V} + 2\mathcal{H} - 6 \leq \zeta_{cw}(G) \leq \frac{1}{6}(13\mathcal{V} + 13\mathcal{H} - 8)$.

Proof. Lower bound: We can get a valid consecutive edge coloring with a valid direction of transmission assignment in the compact wakeup scheduling using $2\mathcal{V} + 2\mathcal{H} - 6$ colors. For example, the number of colors assigned is $22 = 2 \times 7 + 2 \times 7 - 6$ in a 7×7 grid as shown in Figure 5.20(a).

Upper bound: For a consecutive edge coloring in the compact wakeup scheduling of a grid graph G, the difference in colors of edges incident to a node v cannot exceed $deg(v_i)-1$. Suppose that $v_1, v_2, \cdots, v_m$ is the vertex sequence of a path connecting edges with extremal colors, we could get $\zeta_{cw}(G) \leq 1+\sum_{i=1}^{m}(deg(v_i)-1)$. We suppose vertices A and B are on the path connecting edges with minimum and maximum colors respectively, as shown in Figure 5.20(b). We assume vertex A is on the common point of $\overline{H}_{(b+1)(a+1)}$ and $\overline{V}_{(b+1)(a+1)}$, and vertex B is on the common point of $\overline{H}_{(\mathcal{V}-m)(\mathcal{H}-1-n)}$ and $\overline{V}_{(\mathcal{V}-1-m)(\mathcal{H}-n)}$. We can get $\zeta_{cw}(G) \leq 1+3(\mathcal{H}-1-a-n+1)+3(\mathcal{V}-1-b-m) = 3(\mathcal{V}+\mathcal{H}-a-b-m-n)-2$ using route 1. We have also known $\zeta_{cw}(G) \geq 2\mathcal{V}+2\mathcal{H}-6$. $3(\mathcal{V}+\mathcal{H}-a-b-m-n)-2$ should be no less than $2\mathcal{V}+2\mathcal{H}-6$. Otherwise, $2\mathcal{V}+2\mathcal{H}-6$ should also be the upper bound. Then we get, $\mathcal{V}+\mathcal{H}+4 \geq 3(a+b+m+n)$. Without loss of generality, we assume $a+m \geq b+n$. Then, $b+n \leq \frac{1}{6}(\mathcal{V}+\mathcal{H}+4)$. We can also get $\zeta_{cw}(G) \leq 1+3b+2(\mathcal{H}-a-1)+1+2(\mathcal{V}-m-1)+3n = 2(\mathcal{V}+\mathcal{H}-a-m)+3b+3n-2$ using route 2. Then, $\zeta_{cw}(G) = 2(\mathcal{V}+\mathcal{H})+3(b+n)-2(a+m)-2 \leq 2(\mathcal{V}+\mathcal{H})+b+n-2 \leq \frac{1}{6}(13\mathcal{V}+13\mathcal{H}-8)$.

Thus, $\zeta_{cw}(G)$ is bounded by $2\mathcal{V}+2\mathcal{H}-6$ and $\frac{1}{6}(13\mathcal{V}+13\mathcal{H}-8)$. □

According to Theorem 5.3, 5.4 and 5.5, the number of time slots assigned is optimum. If both $\mathcal{V}$ and $\mathcal{H}$ are even, the number of time slots assigned in a period is $4 \times 2 = 8$ in a $\mathcal{V} \times \mathcal{H}$ grid graph. If one of $\mathcal{V}$ and $\mathcal{H}$ is even and the other is odd, the number of time slots assigned is $5 \times 2 = 10$. If both $\mathcal{V}$ and $\mathcal{H}$ are odd, the number of time slots assigned is $6 \times 2 = 12$. Algorithm 5.4 and Algorithm 5.5 describe the interval edge-coloring and compact wakeup scheduling of a grid graph respectively. The time complexity of the compact wakeup scheduling of a grid graph is $O(n)$.

Theorem 5.8. *The number of time slots assigned by Algorithm 5.5 is optimum.*

Algorithm 5.4 Interval edge-coloring of a grid graph

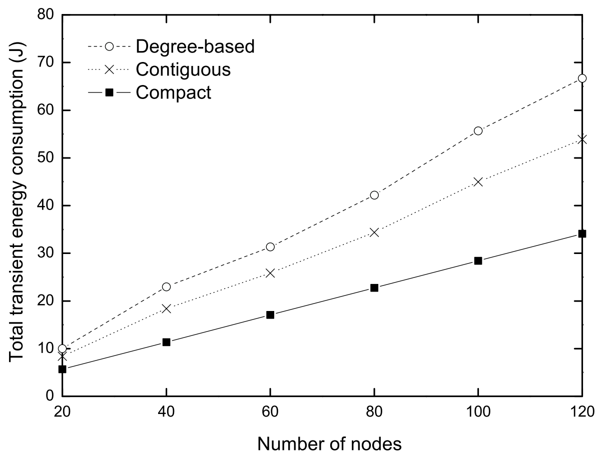
```
Input: A $\mathcal{V} \times \mathcal{H}$ grid graph $G$ ($3 \leq \mathcal{V} \leq \mathcal{H}$).
Output: A valid interval edge-coloring with $\chi_{cw}(G)$ colors.
1: Decide the parity of $\mathcal{V}$ and $\mathcal{H}$ (even or odd).
2: if both $\mathcal{V}$ and $\mathcal{H}$ are even then
3:     Color $G$ using the pattern in Figure 5.11.
4: else if one of $\mathcal{V}$ and $\mathcal{H}$ is even and one is odd then
5:     Color $G$ using the pattern in Figure 5.14(b).
6: else
7:     Color $G$ using the pattern in Figure 5.18.
8: end if
```

Algorithm 5.5 Compact wakeup scheduling of a grid graph

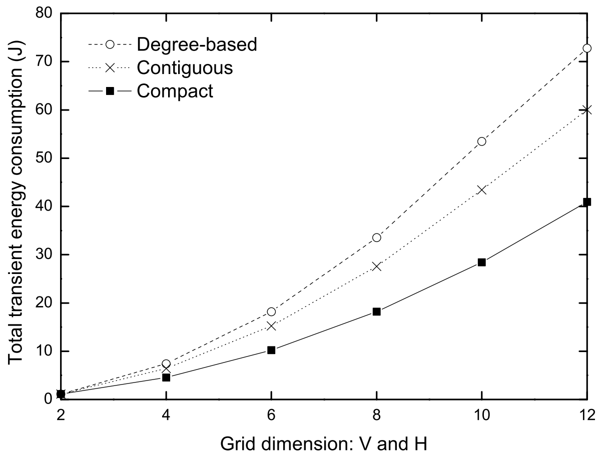
```
Input: A $\mathcal{V} \times \mathcal{H}$ grid graph $G$ ($3 \leq \mathcal{V} \leq \mathcal{H}$).
Output: A valid compact wakeup scheduling.
1: Use Algorithm 5.4 to obtain a valid interval edge-coloring with $\chi_{cw}(G)$ colors for $G$.
2: for $k = 1$ to $\chi_{cw}(G)$ do
3:     Map color $k$ to two consecutive time slots $\{2k-1, 2k\}$.
4:     Use Algorithm 5.1 to determine a valid direction of transmission assignment for
       time slot $2k-1$.
5:     Reverse the direction of transmission along each edge to obtain the other assign-
       ment for time slot $2k$.
6: end for
```

5.4 Simulation Results

In this section, we study the performance of the compact wakeup scheduling of trees and grid graphs, and we also compare our algorithms with the *degree-based heuristic* in [118] and the *contiguous link scheduling* (Algorithm 3.1). The performance metrics used in the evaluation are total transient energy consumption and waiting period. The total energy consumption is an important metric in WSNs, but the energy consumption except the transient energy consumption is the same among the three schemes under identical traffic



(a) Tree



(b) Grid Graph

Figure 5.21: Average total transient energy consumption.

conditions, so we will only focus on the total transient energy consumption. The waiting period is defined as the total time a node stays in the waiting status from the first neighbor waking up to the last neighbor waking up as the node waits for gathering the information from all its neighbors. The waiting period reflects the extra delay caused by the node if it stays in the sleep state for the wakeup of its neighbors.

We adopt the following parameters in our simulation: the transient energy to activate a sensor is 32.9 μJ [99], a time slot is 0.1 second, a scheduling period T is 10 seconds (=100 time slots), and the network operating time is 1 day. In the tree construction of n nodes, the number of children nodes of each sensor is randomly set from 1 to 4. The root node first determines its children nodes, and then each child node determines its children nodes, and so on and so forth until the total number of nodes in the tree reaches n. In the tree construction, we vary n from 20 to 120 with a step of 20, 10 trees are generated and the average performances over all these trees are reported. For the grid graph, we use square grid graphs, where $\mathcal{V} = \mathcal{H}$. In the grid graph construction, we vary $\mathcal{V}$ from 2

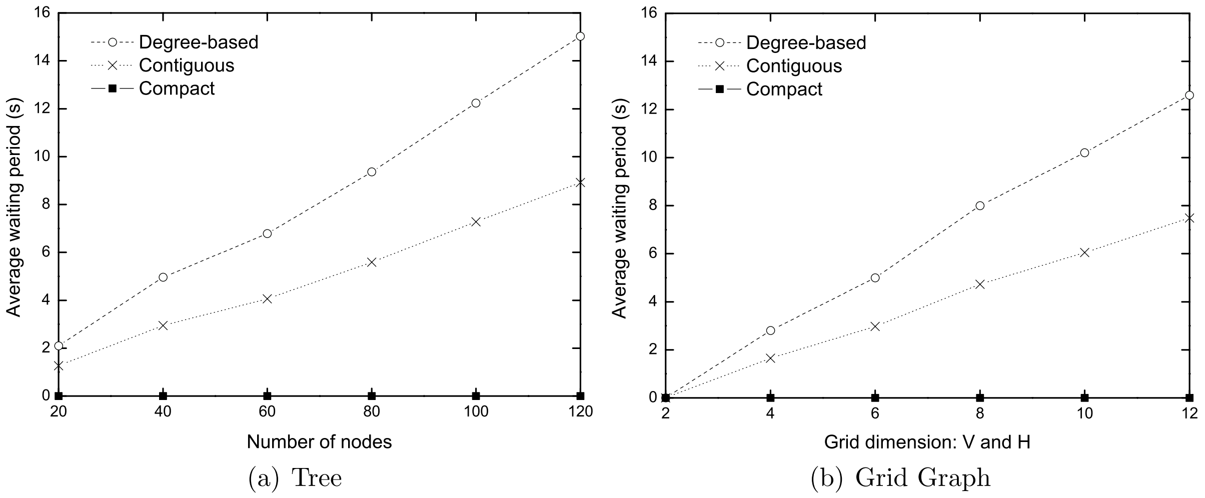


Figure 5.22: Average waiting period.

to 12 with a step of 2.

Figure 5.21 shows the total transient energy consumption of the following schemes: degree-based heuristic (degree-based), contiguous link scheduling (contiguous) and compact wakeup scheduling (compact). In both the tree and grid topologies, the total transient energy consumption increases as the number of nodes increases. The energy consumption in the compact wakeup scheduling is the smallest among the three schemes, for the frequency of state transitions is minimized in the scheduling. As shown in Figure 5.21(a), compact wakeup scheduling reduces the transient energy consumption significantly by approximately 50% as compared to that in the degree-based heuristic, and about 35% as compared to that in the contiguous link scheduling.

Figure 5.22 shows the average waiting period increases as the number of nodes increases in the degree-based heuristic and contiguous link scheduling, while the waiting period is zero in the compact wakeup scheduling. With a smaller waiting period, it would be faster for nodes to gather the information from their neighbors, thus reducing network delay.

We summarize observations from the simulation results as follows:

- The waiting period of trees and grid graphs with valid compact wakeup scheduling is zero.
- Compact wakeup scheduling can significantly reduce network delay and energy consumption.

5.5 Summary

In this chapter, we address a new interference-free TDMA wakeup scheduling problem in TWSNs, called compact wakeup scheduling. In the scheduling, a node needs to wake up only once to communicate bidirectionally with all its neighbors, thus reducing the time overhead and energy cost in the state transitions. We propose polynomial-time algorithms to achieve the optimum number of time slots assigned in a period for trees and grid graphs. In grid graphs, we point out all the possible coloring patterns and give the lower bound as well as the upper bound of the compact wakeup scheduling. In the process of time slot assignments, both the hidden terminal and exposed terminal problems can be avoided. The simulation results corroborate the theoretical analysis and show the efficiency of compact wakeup scheduling.

Chapter 6

Spatio-Temporal Link Scheduling in UWSNs

In this chapter, we investigate the spatio-temporal link scheduling problem in UWSNs. We present efficient centralized and distributed scheduling algorithms that have theoretical performance bounds for both unified and weighted traffic loads. The organization of this chapter is as follows. Section 6.1 is the overview of this work. Section 6.2 describes the system model and then formulates the spatio-temporal link scheduling problem for UWSNs. Section 6.3 presents the centralized and distributed algorithms in the unified traffic load scenario. Section 6.4 considers the link scheduling in the weighted traffic load scenario where each link can transmit its packets in multiple consecutive or inconsecutive time slots. Section 6.5 shows the performance evaluation of the proposed scheduling algorithms. Finally, Section 6.6 summarizes this chapter.

6.1 Overview

Due to the long propagation delay of acoustic signals [134], current terrestrial MAC approaches are not suitable for UWSNs. In traditional MACs, the arrival of a packet is

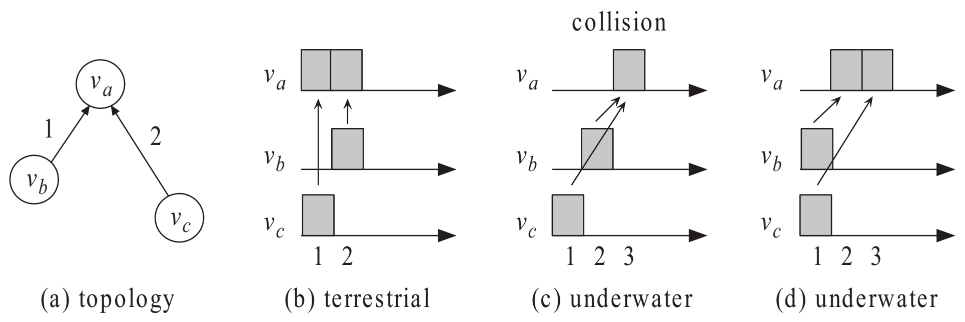


Figure 6.1: Propagation delay in UWSNs.

generally uncertain, and this uncertainty only considers the transmission time uncertainty. As the propagation delay is neglected, the reception time of a packet is assumed to be the same as the transmission time, i.e., the reception time uncertainty is removed. In UWSNs, however, the reception time uncertainty depends on both the transmission time and relative propagation delay to the receiver. This phenomenon is called "spatio-temporal uncertainty" in [107].

Figure 6.1 illustrates the effect of the propagation delay on the time slot assignment in UWSNs. Figure 6.1(a) shows the network topology. In the underwater scenario, the propagation delays from nodes v_b and v_c to node v_a are 1 and 2 time units respectively. In the terrestrial scenario, the collision at v_a can be avoided when nodes v_b and v_c transmit at time 2 and 1 respectively as shown in Figure 6.1(b). But the same assignment for the topology in the underwater scenario results in a collision at v_a as shown in Figure 6.1(c). On the contrary, if nodes v_b and v_c transmit at the same time as shown in Figure 6.1(d), the collision at v_a is eliminated, which is significantly different from the terrestrial scenario.

In this chapter, we adopt the TDMA MAC protocols for UWSNs where the time is slotted and node's transmission time is synchronized. We study the link scheduling problem in UWSNs which aims to eliminate collisions and guarantee fairness for UWSNs. In this chapter, we focus on dealing with the problem of spatio-temporal uncertainty, and propose several spatio-temporal link scheduling algorithms for UWSNs.

The main contributions of this chapter are summarized as follows:

- We identify the spatio-temporal link scheduling problem in UWSNs, which is significantly different from terrestrial wireless networks and also NP-hard.
- We propose a novel slotted spatio-temporal conflict graph which is constructed based on the network topology, conflict relationship, propagation delay and link transmission delay.
- We present both centralized and distributed scheduling algorithms that have theoretical performance bounds under both unified and weighted traffic load scenarios. In the weighted traffic load scenario, we consider the scheduling with and without the consecutive constraint.
- We develop simulations to show the efficiency of the proposed algorithms.

6.2 System Model and Problem Formulation

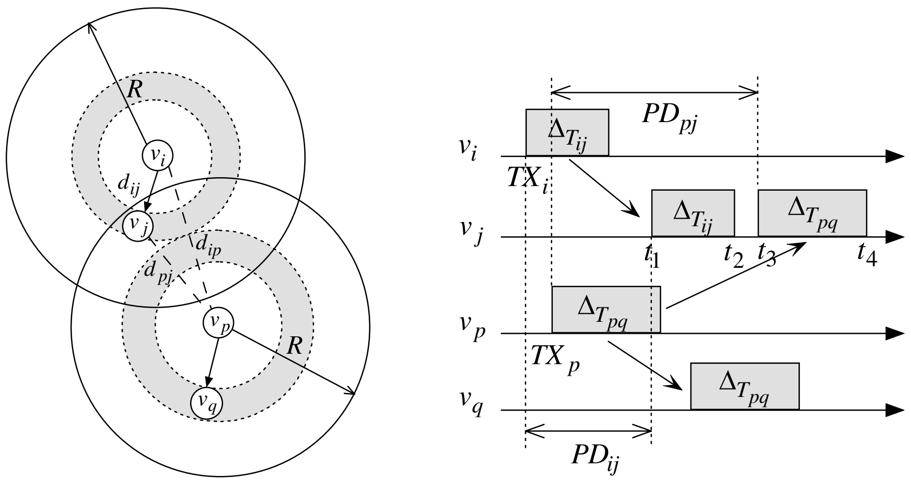
In this section, we present the system model consisting of a network model and an interference model, then we formulate the spatio-temporal link scheduling problem for UWSNs.

6.2.1 System Model

We assume that an UWSN has n static sensor nodes, which are all equipped with single half-duplex acoustic modems. For the acoustic signal propagation in the underwater environment, we adopt a cylindrical propagation model for the two-dimensional scenario and a spherical propagation model for the three-dimensional scenario [110]. Therefore, the network can be represented as a communication graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes the set of nodes, and E denotes the set of directed edges referring to all the communication links. If $\{v_i, v_j\} \subseteq V$, the edge $l_{ij} = (v_i, v_j) \in E$ when v_j is located within the transmission range of v_i. For link l_{ij}, node v_i is the transmitter and v_j is the receiver. The traffic load of link l_{ij} is D_{ij}, the data rate of underwater sensors is B, and then the transmission delay of link l_{ij} is $\Delta_{T_{ij}} = \frac{D_{ij}}{B}$. The propagation speed of acoustic signals in underwater environments is c.

In acoustic networks, the packets transmitted by a node can be received by multiple nodes within its transmission range, and thus interferences may occur when two nodes transmit packets simultaneously. Similar to wireless networks, there are two types of interferences in acoustic networks: primary interference and secondary interference [92]. The primary interference occurs when a node has more than one communication task in a single time slot. Typical examples are RX-RX interference (receiving from two different transmitters simultaneously), TX-TX interference (transmitting to two different receivers simultaneously), and RX-TX interference (receiving and transmitting simultaneously). The secondary interference occurs when a node tuned to a particular transmitter is also within the transmission range of another transmission intended for other nodes. Both primary interference and secondary interference are considered in this chapter.

The interference between two links in the network depends on the interference model, and we use the protocol model [48]. In the protocol model, each node v_i has a transmission range r and an interference range R, where $R > r$. We denote the ratio between the



(a) Link transmission in UWSNs (b) Time diagram

Figure 6.2: Interference analysis in UWSNs.

interference range and the transmission range as $\gamma = \frac{R}{r}$. The transmission time of node v_i is denoted by TX_i and the propagation delay between nodes v_i and v_j is denoted by PD_{ij}.

Lemma 6.1. *In UWSNs, a transmission from v_i to v_j fails due to the secondary interference if the transmission time TX_p of node v_p, which is located within a distance R from v_j meets the following formula:*

$$TX_i + PD_{ij} - PD_{pj} - \Delta_{T_{pq}} \leq TX_p \leq TX_i + PD_{ij} - PD_{pj} + \Delta_{T_{ij}}, \tag{6.1}$$

where $\Delta_{T_{ij}}$ and $\Delta_{T_{pq}}$ are the link transmission delays of link l_{ij} and link l_{pq} respectively.

Proof. As shown in Figure 6.2, node v_j starts to receive the data from node v_i at time $t_1 = TX_i + PD_{ij}$ due to the propagation delay. Node v_j can finish the reception at time $t_2 = TX_i + PD_{ij} + \Delta_{T_{ij}}$ considering the link transmission delay. As another transmitter v_p is located within a distance R from v_j, v_p starts to interfere the data reception of v_j at time $t_3 = TX_p + PD_{pj}$ and ends the interference at time $t_4 = TX_p + PD_{pj} + \Delta_{T_{pq}}$.

If the transmission of link l_{pq} interferes the reception of link l_{ij}, intervals $[t_1, t_2]$ and $[t_3, t_4]$ must have an overlap. We can see that these two intervals $[t_1, t_2]$ and $[t_3, t_4]$ are overlapped if and only if $t_3 \leq t_2$ and $t_1 \leq t_4$. If $t_3 \leq t_2$, i.e., $TX_p + PD_{pj} \leq TX_i + PD_{ij} + \Delta_{T_{ij}}$, we can get $TX_p \leq TX_i + PD_{ij} - PD_{pj} + \Delta_{T_{ij}}$. If $t_1 \leq t_4$, i.e., $TX_i + PD_{ij} \leq TX_p + PD_{pj} + \Delta_{T_{pq}}$, we can get $TX_P \geq TX_i + PD_{ij} - PD_{pj} - \Delta_{T_{pq}}$. □

Note that the primary interference can be treated as a special case of the secondary interference, we can extend Lemma 6.1 to the primary interference where two links share a common node.

Lemma 6.2. *In UWSNs, a transmission from v_i to v_j fails due to the primary interference if the transmission time of another link meets the following formulas:*

(1) For the RX-RX interference that two links l_{ij} and l_{pj} have the same receiver v_j, $TX_i + PD_{ij} - PD_{pj} - \Delta_{T_{pj}} \leq TX_p \leq TX_i + PD_{ij} - PD_{pj} + \Delta_{T_{ij}}$.

(2) For the TX-TX interference that two links l_{ij} and l_{iq} have the same transmitter v_i, $TX_i - \Delta_{T_{iq}} \leq TX'_i \leq TX_i + \Delta_{T_{ij}}$, where TX'_i is the transmission time from v_i to v_q.

(3) For the RX-TX interference that the receiver v_j in link l_{ij} is also the transmitter in link l_{jq}, $TX_i + PD_{ij} - \Delta_{T_{jq}} \leq TX_j \leq TX_i + PD_{ij} + \Delta_{T_{ij}}$.

Proof. Since the primary interference is a special case of the secondary interference, we extend Lemma 6.1 to prove Lemma 6.2 as following:

(1) For the RX-RX interference, link l_{pj} plays the role of link l_{pq} in Lemma 6.1, then we can replace the subscript q with j in Eq. (1), and get $TX_i + PD_{ij} - PD_{pj} - \Delta_{T_{pj}} \leq$

$TX_p \leq TX_i + PD_{ij} - PD_{pj} + \Delta_{T_{ij}}$.

(2) For the TX-TX interference, link l_{iq} plays the role of the link l_{pq} in Lemma 6.1, then we can replace the subscript p with i in Eq. (1), and get $TX_i + PD_{ij} - PD_{ij} - \Delta_{T_{iq}} \leq TX'_i \leq TX_i + PD_{ij} - PD_{ij} + \Delta_{T_{ij}}$, i.e., $TX_i - \Delta_{T_{iq}} \leq TX'_i \leq TX_i + \Delta_{T_{ij}}$.

(3) For the RX-TX interference, link l_{jq} plays the role of the link l_{pq} in Lemma 6.1, then we can replace the subscript p with j in Eq. (1), and get $TX_i + PD_{ij} - PD_{jj} - \Delta_{T_{jq}} \leq TX_j \leq TX_i + PD_{ij} - PD_{jj} + \Delta_{T_{ij}}$, i.e., $TX_i + PD_{ij} - \Delta_{T_{jq}} \leq TX_j \leq TX_i + PD_{ij} + \Delta_{T_{ij}}$.

$\square$

6.2.2 Problem Formulation

Given an interference model, the interference of the links in the communication graph $G = (V, E)$ can be represented as a *conflict graph* [56]. To deal with the propagation delay in UWSNs, Hsu et al. [53] proposed the spatial-temporal conflict (STC) graph $G'(V', E')$, where $V' = E$ and E' is the set of conflict relationships between transmission links. There is a conflict relationship $Conflict(u \rightarrow v)$ ($u, v \in V'$), if the transmission of link l_{pq} (denoted by u) interferes the reception of link l_{ij} (denoted by v). The conflict delay $c_{u,v}$ is used to describe the spatio-temporal uncertainty for each edge $e(u, v) \in E'$. Link u would conflict with link v if the transmission time TX_p of u is just $c_{u,v}$ time units after the transmission time TX_i of v, i.e., $TX_p = TX_i + c_{u,v}$. If two links u and v have the same destination, we can easily get $c_{u,v} = -c_{v,u}$.

Figure 6.3 shows a sample UWSN and its corresponding conflict graph. Figure 6.3(a)

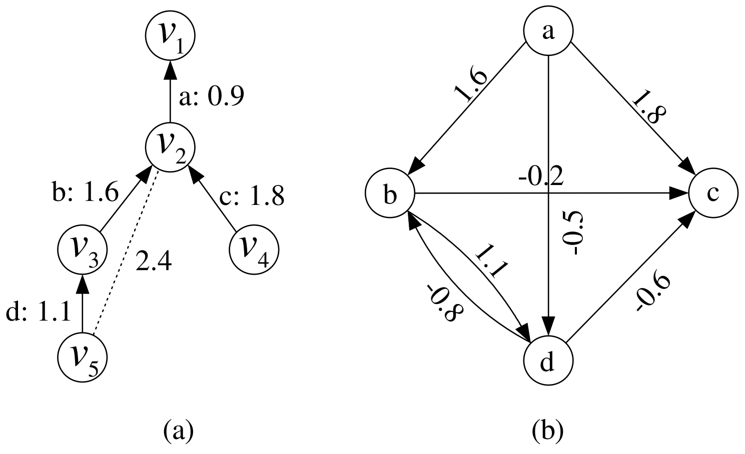


Figure 6.3: Communication graph and corresponding conflict graph: (a) Communication graph, (b) Spatial-temporal conflict graph. The dashed line represents the interference relationship, and the link transmission delay is assumed to be 0.9 time unit.

shows the communication graph, where the dashed line denotes that node v_2 is in the interference range of node v_5 and the numbers on each edge denote the propagation delays of data transmissions. Figure 6.3(b) shows the corresponding STC graph. For example, links b and d (i.e., vertices b and d in Figure 6.3(b)) interfere with each other, so both $Conflict(b \rightarrow d)$ and $Conflict(d \rightarrow b)$ exist, and the conflict delays are $c_{b,d} = 1.1$ and $c_{d,b} = -0.8$. For links b and c that have the same destination node v_2, $Conflict(b \rightarrow c)$ and $Conflict(c \rightarrow b)$ both exist, and $c_{b,c} = -0.2$ and $c_{c,b} = 0.2$. Due to the symmetrical conflict relationships between links b and c, their conflict relationship is represented in only one direction in the STC graph.

In the STC graph, since the link transmission delay is not considered, the conflict delay is only a time value. Thus, it underestimates the effect of interferences in UWSNs. Suppose that link u interferes the reception of link v. When link u is transmitting a data packet, the transmission process needs a time duration to complete. Thus, the time slots that link v cannot be assigned actually form a time interval. Therefore, the conflict delay

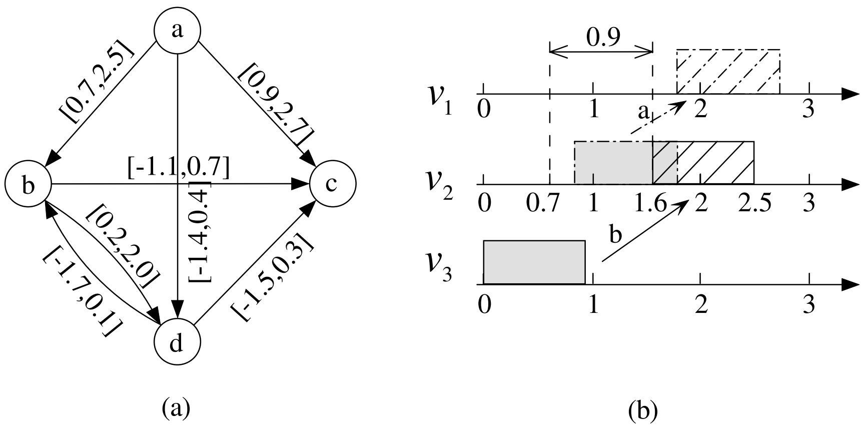


Figure 6.4: Considering the link transmission delay: (a) Improved spatial-temporal conflict graph, (b) Time diagram.

$c_{u,v}$ is not a time value, but a time interval. As $c_{u,v} = TX_p - TX_i$, according to Lemma 6.1, $c_{u,v}$ should be an interval $[PD_{ij} - PD_{pj} - \Delta_{T_{pq}}, PD_{ij} - PD_{pj} + \Delta_{T_{ij}}]$ for the secondary interference. According to Lemma 6.2, for the RX-RX, TX-TX and RX-TX interferences in the primary interference, $c_{u,v}$ should be intervals $[PD_{ij} - PD_{pj} - \Delta_{T_{pj}}, PD_{ij} - PD_{pj} + \Delta_{T_{ij}}]$, $[-\Delta_{T_{iq}}, \Delta_{T_{ij}}]$ and $[PD_{ij} - \Delta_{T_{jq}}, PD_{ij} + \Delta_{T_{ij}}]$ respectively.

To further consider the link transmission delay, we revise the STC graph to be a new graph, as shown in Figure 6.4(a). In the given sample network, assume the link transmission delay for a packet is $\Delta_T = 0.9$ time unit, where one time unit is equal to the length of one time slot. The conflict delay for each link is an interval with $2 \times \Delta_T = 1.8$ time units. For example, the conflict delay between link a and link b belongs to $[1.6 - 0.9, 1.6 + 0.9] = [0.7, 2.5]$. This means that link a would interfere with link b's reception if link a starts to transmit a packet $[0.7, 2.5]$ time units after link b. As shown in Figure 6.4(b), if link b is assigned at time slot 0 (i.e. link b starts to transmit at time 0), link a cannot start to transmit during the time interval $[0.7, 2.5]$. As each link starts to transmit its package from the beginning of the assigned time slot, link a

cannot be assigned the time slots which begin during the interval, i.e., time slots 1 and 2. Interestingly, although the time interval occupies part of time slot 0 (i.e., from 0.7 to 1.0), link a can start to transmit at time 0, for links a and b can transmit simultaneously.

To eliminate the conflict delay in the conflict graph, we add a time dimension in the conflict graph, and transform the improved spatial-temporal conflict graph to a novel three-dimensional conflict graph called *slotted spatio-temporal conflict graph* (S-STC graph). In the S-STC graph $G_{sstc} = (V_{sstc}, E_{sstc})$, $V_{sstc} = \{(u, t_i)|u \in E, t_i \in T\}$, $E_{sstc} = \{e((u, t_i), (v, t_j))|u, v \in E, t_i, t_j \in T\}$, where $e((u, t_i), (v, t_j))$ denotes the transmission of link u at time t_i conflicts the transmission of link v at time t_j. Figure 6.5 shows the S-STC graph corresponding to the communication graph in Figure 6.3(a). For example, $(b, 0)$, $(a, 1)$, $(a, 2)$ denote link b at time slot 0, link a at time slot 1 and link a at time slot 2 respectively. From the improved spatial-temporal conflict graph, link a cannot transmit 1 or 2 time units later than link b. Therefore, there is an edge between $(b, 0)$ and $(a, 1)$ as well as an edge between $(b, 0)$ and $(a, 2)$ in the S-STC graph.

In TDMA, we assume that time is divided into slots with slot size t_s. To overcome the spatio-temporal uncertainty in UWSNs, we identify the spatio-temporal link scheduling problem, in which each link u is assigned a set of time slots to transmit. We assume that the scheduling is periodical and its period T is composed of $|T|$ consecutive time slots. Let $X_{u,t} \in \{0, 1\}$ be an indicator variable, $X_{u,t} = 1$ if link u transmits at time t and $X_{u,t} = 0$ if link u does not transmit at time t. We can get $X_{u,t} = X_{u,t} + i \cdot T$ for any integer i. For a link u, let $I(u, t)$ be the set of links v that interfere with u if v transmits at time t', where $t' = t + c_{u,v}$. Therefore, a spatio-temporal link scheduling is interference-free if $X_{u,t} + X_{v,t'} \leq 1$ for any $v \in I(u, t)$. The objective of the spatio-temporal link

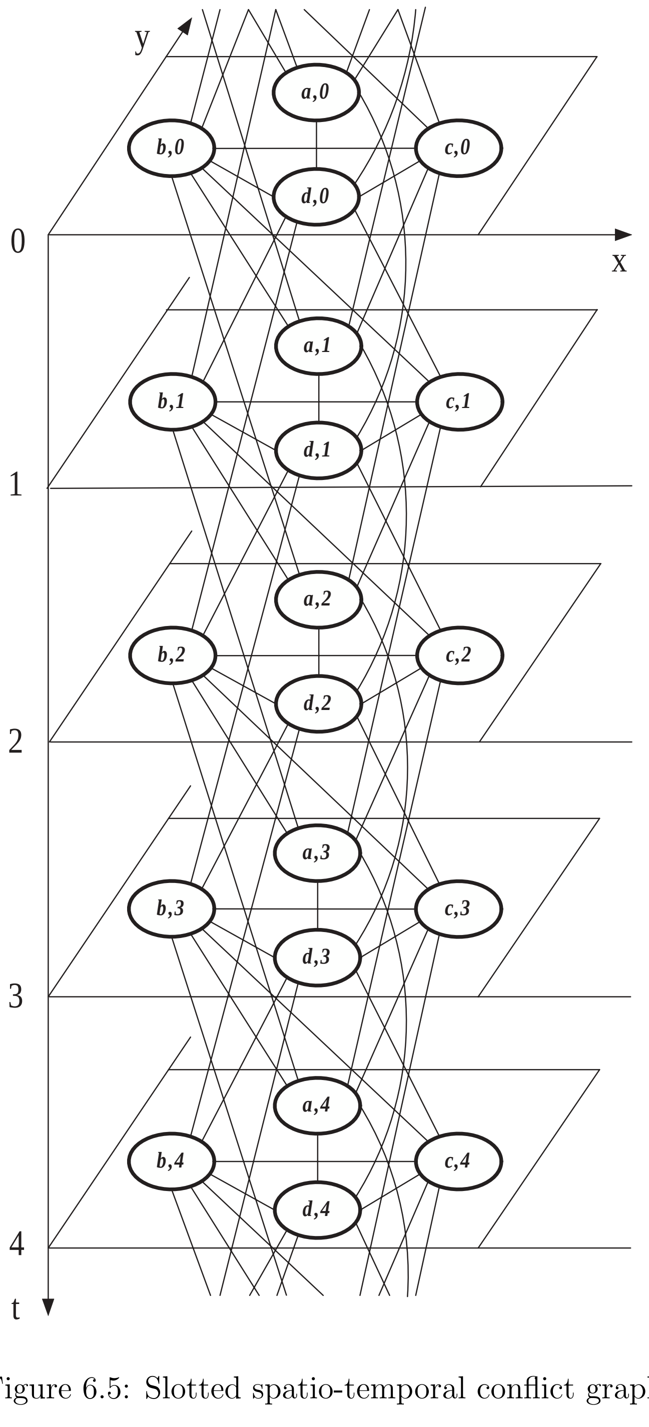


Figure 6.5: Slotted spatio-temporal conflict graph.

scheduling is to find a scheduling with the minimal period T, i.e., the number of time slots assigned is minimum. As the throughput of the network is inverse proportional to T, the network throughput can be maximized by obtaining the minimal period. In [54], it is proven that the link scheduling in a single broadcast domain UWSN is NP-hard. As the single broadcast domain UWSN is a special topology of UWSNs, the spatio-temporal link scheduling problem is also NP-hard. Therefore, it is necessary to design efficient heuristic algorithms with theoretical performance bounds.

In this chapter, we consider both unified and weighted traffic load scenarios. In the unified traffic load scenario, as the traffic load of each link l_{ij} is unified to be D, the link transmission delay of each link is $\Delta_T = \frac{D}{B}$. We assume that all links can transmit their packets using one time slot, i.e., $\Delta_T \leq t_s$. In the weighted traffic load scenario, we assume that each link l_{ij} has a weight $w_{ij} = \lceil \frac{D_{ij}}{B \cdot t_s} \rceil$, which is the number of time slots it requires to complete the transmission.

6.3 Spatio-Temporal Link Scheduling with Unified Traffic Load

In this section, we investigate the spatio-temporal link scheduling with unified traffic load in UWSNs. We propose both centralized and distributed scheduling algorithms that have theoretical performance bounds to the optimum solutions.

6.3.1 Centralized Scheduling

In this subsection, we propose a centralized approximation algorithm called *Centralized Spatio-Temporal link scheduling with Unified traffic load* (CIST-U), which is shown in

Algorithm 6.1.

Algorithm 6.1 Centralized Spatio-Temporal link scheduling with Unified Traffic Load (CIST-U)

```
1: Construct the S-STC graph $G_{sstc}$ based on $G$ with $t = 4\delta^*(G_{stc}) + \lceil \frac{c\Delta_T + R}{c \cdot t_s} \rceil + 1$, and
   let graph $G^* = G_{sstc}$. Initialize stack $S = \varnothing$.
2: while $G^* \neq \varnothing$ do
3:     Select a link $u \in E$ with the smallest degree in $G^*$ and remove all the vertices $(u, t_i)$
       from $G^*$ and all their incident edges. Push $u$ into stack $S$.
4: end while
5: while $S \neq \varnothing$ do
6:     Pop link $u$ from $S$. Assign $u$ with the smallest available time slot $t_u$. For any
       neighbor $(v, t_v)$ of $(u, t_u)$ in $G_{sstc}$, time slot $t_v$ turns unavailable for link $v$.
7: end while
```

We first describe how to utilize the S-STC graph in the spatio-temporal link scheduling. Different from the traditional vertex coloring, the S-STC graph is three-dimensional. In the S-STC graph, we need to select one and only one vertex (u, t_u) in all (u, t_i) where $i = 0, 1, 2, ...$ as it indicates that link u transmits at time slot t_u. After (u, t_u) is assigned, the vertices adjacent to (u, t_u) cannot be selected. This procedure continues until every vertex is selected, that is, each link is assigned its own transmission time. Take Figure 6.5 as an example, we can select vertices $(a, 0)$, $(b, 0)$, $(c, 2)$ and $(d, 3)$, that is, links a, b, c and d transmit at time slots 0, 0, 2 and 3 respectively. As there exists the propagation delay in UWSNs, the transmission and reception of a link are not at the same time, and they should be scheduled separately. In the assignment, the selected time slot represents the transmission time, and the reception time can be calculated by adding the propagation delay of the link to the transmission time. The scheduling period is $T = \max\limits_{u \in E}\{t_u + \lceil \frac{\Delta_T + PD_u}{t_s} \rceil\} \leq \max\limits_{u \in E}\{t_u + \lceil \frac{c\Delta_T + R}{c \cdot t_s} \rceil\}$, where PD_u is the propagation delay of link u. The objective of the link scheduling is to minimize the scheduling period T.

As the parameters c, Δ_T, R and t_s are determined by the network environment, we can try to minimize $\max\limits_{u \in E}\{t_u\}$.

To solve the spatio-temporal link scheduling problem in the S-STC graph, we use the smallest-degree-last ordering method [80]. The basic idea is to firstly sort the links using the smallest-degree-last ordering which is shown as follows: Every time we select a link with the smallest degree from the remaining graph, and then remove the link from the graph. Repeat this until the remaining graph becomes empty. We reverse the order of the selected links, and assign time slots to these links in sequence using the first-fit heuristic, that is, the smallest available time slots that are not used by the interfering links will be assigned. Take the sample network in Figure 6.3(a) as an example, the CIST-U algorithm first constructs the S-STC graph as shown in Figure 6.5, and then finds a scheduling order using the smallest-degree-last method as $< b, d, c, a >$. Then each link is assigned the smallest available time slot without interferences one by one. After the assignment, links a, b, c and d are assigned time slots 3, 0, 3 and 1 respectively.

In the S-STC graph, the degree $deg(u_i)$ of a vertex (u, t_i) is the number of vertices adjacent to it. Then $deg(u_1), deg(u_2), deg(u_3), \cdots$ are all equal to a value $deg(u)$, which is called the degree of link u in the S-STC graph. This degree is different from the degree of link u in the STC graph, denoted as $deg'(u)$, which is defined as the number of vertices adjacent to link u in the STC graph. For example, for link b in the communication graph shown as Figure 6.3(b), its degree in the S-STC graph shown as Figure 6.5 is $deg(b) = 7$, and its degree in the STC graph shown as Figure 6.3(b) is $deg'(b) = 3$. It is easy to see that $deg'(u)$ is different from $deg(u)$. However, they are closely related, which is given by the following lemma.

Lemma 6.3. *Given a communication graph and its corresponding STC and S-STC graphs, for any link u, the degree of u in the S-STC graph is at most four times the degree of u in the STC graph.*

Proof. Suppose that link v is adjacent to link u in the STC graph. In the worst case, link u may interfere the reception of link v with conflict delay $c_{u,v}$, and link v may interfere the reception of link u with conflict delay $c_{v,u}$. Then, the vertices adjacent to a vertex (u, t_i) in the S-STC graph are (v, t) and (v, t'), where $t = t_i - c_{u,v}$ and $t' = t_i + c_{v,u}$. As both $c_{u,v}$ and $c_{v,u}$ are time intervals of $2\Delta_T$, both t and t' are also time intervals of $2\Delta_T$. If $0 \leq \Delta_T < 0.5t_s$, the number of vertices in all (v, t_j) adjacent (u, t_i) is not larger than 2. If $0.5t_s \leq \Delta_T < t_s$, the number of vertices in all (v, t_j) adjacent to (u, t_i) is not larger than 4. Hence, we can get that for each vertex adjacent to link u in the STC graph, there are at most four vertices adjacent to (u, t_i) in the S-STC graph. Therefore, $deg(u) \leq 4deg'(u)$. □

We denote the minimum degree of all vertices by $\delta(G)$ in $G = (V, E)$, and the inductivity of G by $\delta^*(G) = max_{U \subseteq V} \delta(G[U])$, where $G[U]$ is the subgraph of G induced by $U \subseteq V$. A vertex coloring of G is an assignment of colors to nodes in V such that adjacent vertices are assigned different colors. It is well known that we can produce a proper vertex coloring in G using at most $\delta^*(G) + 1$ colors by applying a smallest-degree-last ordering of vertices in $O(|V| + |E|)$ time [80]. Although the scheduling in the S-STC graph is different from the vertex coloring problem, the number of transmitting time slots assigned by the CIST-U algorithm is at most $\delta^*(G_{sstc}) + 1$. From Lemma 6.3, we have $\delta^*(G_{sstc}) \leq 4\delta^*(G_{stc})$. Therefore, the scheduling period T of the spatio-temporal link scheduling is at most $4\delta^*(G_{stc}) + \lceil \frac{c\Delta_T + R}{c \cdot t_s} \rceil + 1$.

Theorem 6.1. *The upper bound of the number of transmitting time slots assigned by the CIST-U algorithm is $\delta^*(G_{sstc}) + 1$.*

Proof. In the CIST-U algorithm, we suppose that the scheduling order is $< u_1, u_2, \cdots, u_n >$. Let G_i be the S-STC graph induced by links $u_1, u_2, \cdots, u_i$, and $deg_i(u)$ be the degree of u in G_i. Thus, $deg_n(u)$ is the degree of u in G_{sstc}.

Suppose k_i is the number of transmitting time slots assigned after links $u_1, u_2, \cdots, u_i$ are scheduled by the CIST-U algorithm. When considering link u_{i+1}, if an assigned time slot can also be assigned to u_{i+1}, we have $k_{i+1} = k_i$; otherwise, a new time slot has to be assigned to it, then we have $k_{i+1} = k_i + 1$ and $deg_{i+1}(u_{i+1}) \geq k_i$. By mathematical induction on i, we deduce $k_i \leq 1 + max\{deg_i(u_i) | i = 1, 2, \cdots, n\}$. Since $deg_i(u_i)$ is clearly bounded by both $deg_n(u_i)$ and $i - 1$ (i.e., the total number of vertices in G_i excluding u_i), $k_n \leq 1 + max\{\min_{1 \leqslant i \leqslant n} \{deg_n(u_i), i - 1\}\}$.

Notice that, we use the smallest-degree-last ordering method, then we have $deg_i(u_i) = min\{deg_i(u_j) | u_j \in G_i\}$. According to the definition, we have $k_n \leq \delta^*(G_{sstc}) + 1$, that is, the number of transmitting time slots assigned by the CIST-U algorithm is at most $\delta^*(G_{sstc}) + 1$. □

Lemma 6.4. *The distance between two nodes that can simultaneously transmit without interferences should be at least $c\Delta_T$, where c denotes the acoustic signal's propagation speed.*

Proof. Suppose that receiver v_j in link l_{ij} is in the interference range of another transmitter v_p in link l_{pq}, as shown in Figure 6.2(a). We use d_{xy} to denote the distance between nodes x and y. Due to the propagation delay, we have $d_{ij} = c \cdot PD_{ij}$ and $d_{pj} = c \cdot PD_{pj}$.

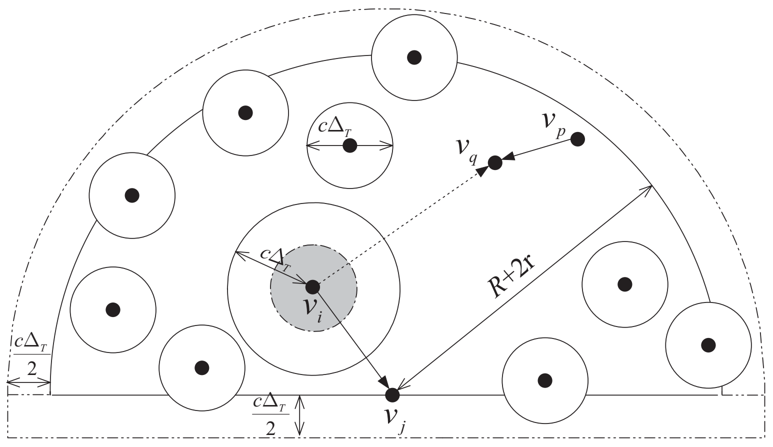


Figure 6.6: Bounding minimum degree in the CIST-U algorithm.

From Lemma 6.1, we get $|TX_p + PD_{pj} - TX_i - PD_{ij}| > \Delta_T$ because l_{pq} does not interfere with l_{ij}. As nodes v_p and v_i transmit simultaneously, $TX_p = TX_i$, then $|PD_{pj} - PD_{ij}| > \Delta_T$. Thus, the distance between v_i and v_p is $d_{ip} \geq |d_{pj} - d_{ij}| = c|PD_{pj} - PD_{ij}| > c\Delta_T$. □

Theorem 6.2. *The lower bound of the number of transmitting time slots assigned by the CIST-U algorithm is* $\frac{\delta^*(G_{sstc})}{4C}$ *in two-dimensional networks, where* $C = \frac{1}{2}(2H + \frac{4H}{\gamma} + 1)^2 + \frac{4}{\pi}(H + \frac{2H}{\gamma} + \frac{1}{2}) - 1$, $H = \frac{R}{c\Delta_T}$ *and* $\gamma = \frac{R}{r}$.

Proof. Let G^* be a subgraph of G_{sstc} such that every vertex in G^* has a degree of at least $\delta^*(G_{sstc})$. Let $v_j \in G^*$ be the bottom-most node in this subgraph and v_i is the transmitter of link l_{ij}. In G^*, the transmitters of the links that interfere with link l_{ij} lie in a semi-disk with radius $R + 2r$, as shown in Figure 6.6. There are two cases:

Case1: A link incident to v_j interferes the reception of the link with transmitter v_p. For example, l_{ij} interferes the reception of link l_{pq} as shown in Figure 6.6. As v_p can communicate with v_q, $d_{pq} \leq r$. As l_{ij} interferes with l_{pq}, $d_{qi} \leq R$. As v_i can communicate with v_j, $d_{ij} \leq r$. Hence, $d_{pj} \leq d_{pq} + d_{qi} + d_{ij} + \leq R + 2r$.

Case 2: A link with transmitter v_p interferes the reception of v_j, and $d_{pj} \leq R$.

From Lemma 6.4, the distance between two nodes simultaneously transmitting without interferences should be at least $c\Delta_T$. That is, in a two-dimensional network, there is at most one transmitter within a disk with diameter $c\Delta_T$ that can simultaneously transmit with v_i. Therefore, the number of non-overlapped disks in the semi-disk with radius $R + 2r$, excluding the disk centered at v_i, is upper-bounded by

$$C = \frac{\frac{1}{2}\pi(R + 2r + \frac{c\Delta_T}{2})^2 + 2(R + 2r + \frac{c\Delta_T}{2})(\frac{c\Delta_T}{2})}{\pi(\frac{c\Delta_T}{2})^2} - 1. \tag{6.2}$$

With $H = \frac{R}{c\Delta_T}$ and $\gamma = \frac{R}{r} > 1$, Eq. (6.2) can be presented as

$$C = \frac{1}{2}(2H + \frac{4H}{\gamma} + 1)^2 + \frac{4}{\pi}(H + \frac{2H}{\gamma} + \frac{1}{2}) - 1.$$

From Lemma 6.3, the number of links that interfere with the link incident to v_j is at least $\frac{\delta^*(G_{sstc})}{4}$. So the number of transmitting time slots assigned is at least $\frac{\delta^*(G_{sstc})}{4C}$. □

We are now ready to obtain the approximation ratio of the CIST-U algorithm in two-dimensional networks.

Theorem 6.3. *The number of transmitting time slots assigned by the CIST-U algorithm is at most $4C$ times of the optimum in two-dimensional networks.*

Proof. According to Theorems 6.1 and 6.2, the approximation ratio of the CIST-U algorithm in two-dimensional networks is $\frac{\delta^*(G_{sstc})+1}{\frac{\delta^*(G_{sstc})}{4C}} \simeq 4C$. □

Theorem 6.4. *The number of transmitting time slots assigned by the CIST-U algorithm in three-dimensional networks is at most $4C'$ times of the optimum, where $C' = \frac{1}{2}(2H + \frac{4H}{\gamma} + 1)^3 + 3(H + \frac{2H}{\gamma} + \frac{1}{2})^2 - 1$.*

Proof. The approximation ratio in three-dimensional networks is determined similarly to that in two-dimensional networks. The transmitters of the links that interfere the reception of a receiver v_j must lie in a semi-sphere with radius $R+2r$ and center v_j. Therefore, the number of non-intersecting spheres with diameter $c\Delta_T$ in a three-dimensional network is upper-bounded by:

$$\begin{aligned} C' &= \frac{\frac{1}{2}\cdot\frac{4}{3}\pi(R+2r+\frac{c\Delta_T}{2})^3+\pi(R+2r+\frac{c\Delta_T}{2})^2\cdot\frac{c\Delta_T}{2}}{\frac{4}{3}\pi(\frac{c\Delta_T}{2})^3}-1 \\ &= \frac{1}{2}(2H+\frac{4H}{\gamma}+1)^3+3(H+\frac{2H}{\gamma}+\frac{1}{2})^2-1. \end{aligned}$$

According to Lemma 6.3, the lower bound of the number of transmitting time slots assigned is $\frac{\delta^*(G_{sstc})}{4C'}$. As the upper bound of the number of transmitting time slots assigned is still $\delta^*(G_{sstc})+1$, we can get the approximation ratio of the CIST-U algorithm in three-dimensional networks is $\frac{\delta^*(G_{sstc})+1}{\frac{\delta^*(G_{sstc})}{4C'}} \simeq 4C'$. □

Complexity Analysis: The number of vertices of the STC graph (i.e. the number of links in the communication graph) is denoted as m, the maximum degree of the STC graph is denoted as Δ, and $\delta^*(G_{stc}) \leq \Delta$. The S-STC graph is three-dimensional with $t = 4\delta^*(G_{stc}) + \lceil\frac{c\Delta_T+R}{c\cdot t_s}\rceil + 1$, and to construct the graph takes $O(t\Delta m) = O(\delta^*(G_{stc})\Delta m) = O(\Delta^2 m)$ time. According to [80], to make the smallest-degree-last ordering and use the first-fit heuristic takes $O(tm + t\Delta m) = O(\Delta^2 m)$ time. Therefore, the time complexity of the CIST-U algorithm is $O(\Delta^2 m)$.

6.3.2 Distributed Scheduling

In UWSNs, it is necessary to design efficient distributed algorithms, as the centralized algorithms could not be used without a predefined leader. For the distributed scheduling, we use the smallest-ID-first ordering rather than the smallest-degree-last ordering, i.e., an unscheduled node with the smallest ID among its unscheduled k-hop neighbors is scheduled first. The distributed algorithm, called *Distributed Spatio-Temporal link scheduling with Unified traffic load* (DIST-U), is shown in Algorithm 6.2.

Algorithm 6.2 Distributed Spatio-Temporal link scheduling with Unified Traffic Load (DIST-U)

```
1: Each node $v_j$, with a unique ID, knows its own location. Mark each node $v_j$ initially
   as unscheduled.
2: Each node $v_j$ broadcasts its ID and location information to its $k$-hop neighbors using
   a simple flooding with TTL = $k$.
3: if node $v_j$ is unscheduled and its ID is smaller than any other unscheduled $k$-hop
   neighbor then
4:    $v_j$, as a receiver, collects the information when its transmitters cannot transmit
      due to the interferences.
5:    $v_j$ assigns the smallest available time slots to its incident links which do not interfere
      with the links that have already been scheduled.
6:    $v_j$ broadcasts the information to its $k$-hop neighbors, and these nodes cannot trans-
      mit or receive in the corresponding time slots considering the propagation delay
      due to the interferences.
7:    $v_j$ marks itself as scheduled.
8: end if
```

By virtue of the smallest-ID-first ordering, the receivers of the links interfering with each other are scheduled separately, and then invalid schedulings can be avoided. The receivers of the links that interfere with a link incident to v_j are at most $R + r$ away from v_j. We assume that all nodes within v_j's interference range are at most $\lceil \gamma \rceil$ hops away from v_j, and then the receivers of the links interfering with a link incident to v_j are at most $k = \lceil \gamma \rceil + 1$ hops away from v_j.

Each node v_j, with a unique identity (ID), first collects the ID and location information

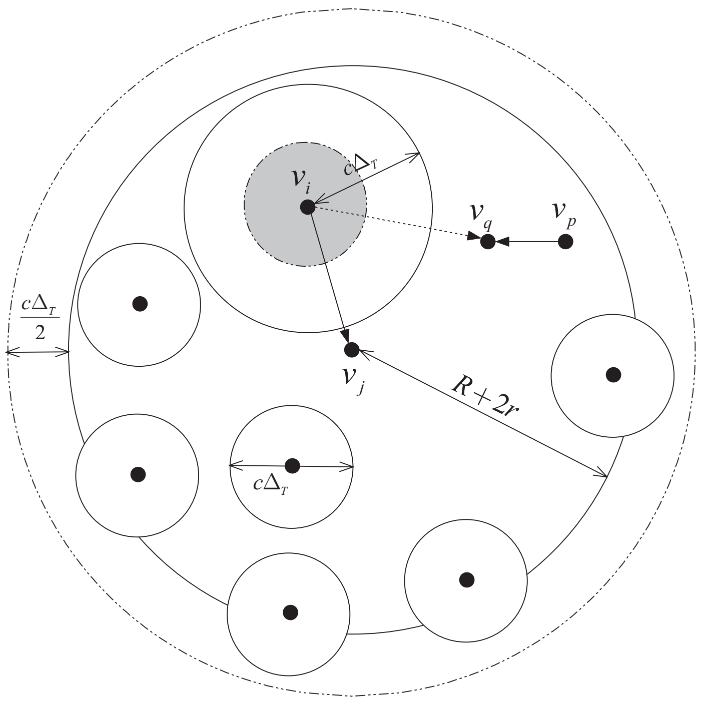


Figure 6.7: Bounding minimum degree in the DIST-U algorithm.

of the nodes that are within its k-hop range, which can be done by using a simple flooding with a time-to-live (TTL) value equal to k. In the time slot assignment, we still use the first-fit heuristic to assign links with the smallest available time slots that are not used by the interfering links. After that, v_j will notify the k-hop nodes the time slots they cannot use due to the interferences, and these nodes will not transmit or receive in the corresponding time slots considering the propagation delay, which can be calculated using the location information. Take Figure 6.2(a) as an example, if link l_{ij} incident to v_j is assigned time slot t_j for the transmission, the transmitter v_p of link l_{pq} cannot transmit at time interval $[t_j + PD_{ij} - PD_{pj} - \Delta_T, t_j + PD_{ij} - PD_{pj} + \Delta_T]$.

Theorem 6.5. *The number of transmitting time slots assigned by the DIST-U algorithm is at most $4C_1$ times of the optimum in two-dimensional networks, where $C_1 = (2H + \frac{4H}{\gamma} + 1)^2 - 1$.*

Proof. The maximum degree of a vertex in the slotted spatio-temporal conflict graph G_{sstc} is denoted as $\Delta(G_{sstc})$. Suppose v_j is the receiver of link l_{ij} which has the maximum degree $\Delta(G_{sstc})$. Similar to Theorem 6.2, the transmitters of the links that interfere with link l_{ij} lie in a disk with radius $R + 2r$, as shown in Figure 6.7.

From Lemma 6.4, the distance between two nodes simultaneously transmitting without interferences should be at least $c\Delta_T$. That is, for a disk with diameter $c\Delta_T$, there is at most one transmitter which can simultaneously transmit with transmitter v_i. Therefore, the number of non-overlapped disks in the disk with radius $R + 2r$, excluding the disk centered at v_i, is upper bounded by

$$C_1 = \frac{\pi(R + 2r + \frac{c\Delta_T}{2})^2}{\pi(\frac{c\Delta_T}{2})^2} - 1 = (2H + \frac{4H}{\gamma} + 1)^2 - 1.$$

From Lemma 6.3, the number of links that interfere with the link incident to v_j is at least $\frac{\Delta(G_{sstc})}{4}$. So the lower bound of the number of transmitting time slots assigned by the DIST-U algorithm is $\frac{\Delta(G_{sstc})}{4C_1}$. As the first-fit heuristic is used in the algorithm, the upper bound of the number of transmitting time slots assigned is $\Delta(G_{sstc}) + 1$. Hence, the approximation ratio in two-dimensional networks is $\frac{\Delta(G_{sstc})+1}{\frac{\Delta(G_{sstc})}{4C_1}} \simeq 4C_1$. □

Theorem 6.6. *The number of transmitting time slots assigned by the DIST-U algorithm in three-dimensional networks is at most $4C_1'$ times of the optimum, where $C_1' = (2H + \frac{4H}{\gamma} + 1)^3 - 1$.*

Proof. The approximation ratio in three-dimensional networks is determined similarly to that in two-dimensional networks. The transmitters of the links that interfere the reception of a receiver v_j lie in a sphere with radius $R + 2r$ and center v_j. Thus, we can get that the number of non-intersecting spheres with diameter $c\Delta_T$ is upper bounded by:

$$C_1' = \frac{\frac{4}{3}\pi(R + 2r + \frac{c\Delta_T}{2})^3}{\frac{4}{3}\pi(\frac{c\Delta_T}{2})^3} - 1 = (2H + \frac{4H}{\gamma} + 1)^3 - 1.$$

According to Lemma 6.3, the lower bound of the number of transmitting time slots assigned is $\frac{\Delta(G_{sstc})}{4C_1'}$. As the upper bound of the number of transmitting time slots assigned is still $\Delta(G_{sstc})+1$, we can get the approximation ratio of the DIST-U algorithm in three-dimensional networks is $\frac{\Delta(G_{sstc})+1}{\frac{\Delta(G_{sstc})}{4C_1'}} \simeq 4C_1'$. $\square$

Complexity Analysis: For each node v_j, it first conducts a k-hop broadcast to send its information to the nodes within its k-hop range. After v_j is scheduled, it needs to conduct another k-hop broadcast to send the assignment information. Therefore, the message complexity of Algorithm 6.2 is $O(\rho n)$, where ρ is the maximum number of k-hop neighbors and n is the number of nodes in the network. The time complexity of a distributed algorithm is the number of communication rounds until the last node terminates the algorithm. The time complexity of Algorithm 6.2 is $O(n)$, and the worst case is that sensors are deployed in a line with node ID in an increasing order and consequently sensors are scheduled in sequence.

6.4 Spatio-Temporal Link Scheduling with Weighted Traffic Load

In this section, we investigate the spatio-temporal link scheduling with weighted traffic load in UWSNs. In general, the traffic load of each link is different. For the link scheduling with weighted traffic load, we assume that link u has weight $w_u = \lceil \frac{D_u}{B \cdot t_s} \rceil$ upon the load requirement, and the traffic load D_u is determined by a certain routing. Then in the link scheduling, link u is assigned w_u time slots.

We consider the spatio-temporal link scheduling with weighted traffic load under two scenarios: One scenario is that each link does not have the constraint of consecutive time slots, that is, each link can transmit its packets in multiple time slots arbitrarily in each scheduling period such as [118]. The other one is that each link has the consecutive constraint, that is, each link must use consecutive time slots to transmit its packets and can only transmit once in each scheduling period such as [38, 77].

6.4.1 Spatio-Temporal Link Scheduling with Weighted Traffic Load

The *Centralized Spatio-Temporal link scheduling with Weighted traffic load* (CIST-W) algorithm is shown in Algorithm 6.3, which does not consider the consecutive constraint. The basic idea is to create a clique with size w_u for each vertex (u, t_i) in the S-STC graph, and then there is a set of virtual links u_1, u_2, $\cdots$, u_{w_u} for each link u, and all these virtual links are assigned time slots using the CIST-U algorithm. Finally, link u is assigned the w_u time slots which are assigned to its virtual links. In the scheduling, the w_u time slots assigned to link u can be non-consecutive.

Theorem 6.7. *The number of transmitting time slots assigned by the CIST-W algorithm is at most $4C$ times of the optimum in two-dimensional networks.*

Proof. The minimum number of transmitting time slots assigned by the CIST-U algorithm in G'_{sstc} and the minimum number of transmitting time slots assigned by the CIST-W algorithm in G_{sstc} are denoted by $\chi(G'_{sstc})$ and $\chi(G_{sstc})$ respectively. Notice

Algorithm 6.3 Centralized Spatio-Temporal link scheduling with Weighted Traffic Load (CIST-W)

```
Input: A graph $G = (V, E)$ with weights on each link.
Output: A valid CIST-W scheduling.
1: Construct the S-STC graph $G_{sstc}$ with $t = 4W\delta^*(G_{stc}) + \lceil \frac{c\Delta_T + R}{c \cdot t_s} \rceil + 1$, and assign a
   weight $w_u$ to each vertex $(u, t_i)$.
2: Construct a new S-STC graph $G'_{sstc}$ as follows: For each vertex $(u, t_i)$ with weight
   $w_u$, we create $w_u$ virtual vertices, $(u_1, t_i), (u_2, t_i), \cdots, (u_{w_u}, t_i)$, and add them to $G'_{sstc}$.
   Add to graph $G'_{sstc}$ the edges connecting $(u_j, t_i)$ and $(u_k, t_i)$ for all $1 \leq j < k \leq w_u$.
   Add to graph $G'_{sstc}$ an edge between $(u_j, t_i)$ and $(v_k, t_l)$ if and only if there is an edge
   between $(u, t_i)$ and $(v, t_l)$ in $G_{sstc}$.
3: Run the CIST-U algorithm on $G'_{sstc}$.
4: Assign link $u$ all the time slots which are assigned to $u_j$ for $1 \leq j \leq w_u$ in $G'_{sstc}$.
```

that for any valid CIST-W scheduling for G_{sstc}, link u is assigned at least w_u time slots. By assigning each virtual link u_j in G'_{sstc} a distinct time slot from the w_u time slots which are assigned to link u, we obtain a valid CIST-U scheduling for G'_{sstc}. Thus, $\chi(G'_{sstc}) \leq \chi(G_{sstc})$. Since the CIST-U algorithm will return a scheduling with at most $4C \cdot \chi(G'_{sstc})$ transmitting time slots, the CIST-W algorithm produces a scheduling with at most $4C \cdot \chi(G'_{sstc}) \leq 4C \cdot \chi(G_{sstc})$ transmitting time slots. □

Theorem 6.8. *The number of transmitting time slots assigned by the CIST-W algorithm in three-dimensional networks is $4C'$ times of the optimum.*

Proof. Similar to Theorem 6.7, the CIST-W algorithm in three-dimensional networks has the same approximation ratio as the CIST-U algorithm in three-dimensional networks, which is $4C'$. □

The *Distributed Spatio-Temporal link scheduling with Weighted traffic load* (DIST-W) is shown in Algorithm 6.4. The basic idea is to use the smallest-ID-first ordering rather than the smallest-degree-last ordering. Each node v_j needs to calculate the weight information of its incident links, and v_j uses the first-fit heuristic to assign time slots when it is scheduled.

Algorithm 6.4 Distributed Spatio-Temporal link scheduling with Weighted Traffic Load (DIST-W)

```
1: Each node $v_j$, with a unique ID, knows its own location. Mark each node $v_j$ initially
   as unscheduled.
2: Calculate $w_u$ of each link $u$.
3: Each node $v_j$ broadcasts its ID and location information to its $k$-hop neighbors using
   a simple flooding with TTL = $k$.
4: if node $v_j$ is unscheduled and its ID is smaller than any other unscheduled $k$-hop
   neighbor then
5:     $v_j$, as a receiver, collects the information when its transmitters cannot transmit
       due to the interferences.
6:     For $v_j$'s each incident link $u$ with weight $w_u$, assign the smallest $w_u$ available time
       slots to link $u$ which do not interfere with the links that have already been sched-
       uled.
7:     $v_j$ broadcasts the information to its $k$-hop neighbors, and these nodes cannot trans-
       mit or receive in the corresponding time slots considering the propagation delay
       due to the interferences.
8:     $v_j$ marks itself as scheduled.
9: end if
```

Theorem 6.9. *The number of transmitting time slots assigned by the DIST-W algorithm is at most $4C_1$ times of the optimum in two-dimensional networks, and $4C_1'$ times of the optimum in three-dimensional networks.*

Proof. Similar to Theorem 6.7, the DIST-W algorithm in two-dimensional networks and three-dimensional networks has the same approximation ratios as the DIST-U algorithm in two-dimensional networks and three-dimensional networks, which are $4C_1$ and $4C_1'$ respectively. □

Complexity Analysis: In the new S-STC graph G'_{sstc}, each vertex in G_{sstc} is replaced with w_u nodes, and $w_u \le W$ where W is the maximum weight of all the links in the network, i.e., $W = \max\limits_{1 \leqslant i \leqslant m} \{w_i\}$. As both to construct the new S-STC graph and to invoke the CIST-U algorithm in the CIST-W algorithm take $O(t \cdot W\Delta \cdot Wm) = O(W^3 \delta^*(G_{stc})\Delta m) = O(\Delta^2 m)$ time, the time complexity of the CIST-W algorithm is $O(\Delta^2 m)$. The message complexity and time complexity of the DIST-W algorithm are same to those of the DIST-U algorithm, which are $O(\rho n)$ and $O(n)$ respectively.

6.4.2 Spatio-Temporal Link Scheduling with Weighted Traffic Load under the Consecutive Constraint

The *Centralized Spatio-Temporal link scheduling with Weighted traffic load under the Consecutive constraint* (CIST-WC) algorithm is shown in Algorithm 6.5. The basic idea is that a link with a heavier traffic load will be scheduled earlier. Different from the CIST-W algorithm, gaps exist in the time slot assignment (i.e., the assigned time slots for different links may not be adjacent), because each link u has to be assigned w_u consecutive time slots. For example, suppose the weights of links a, b, c and d in Figure 6.5 are 3, 4, 5, 6 respectively. Based on the CIST-WC algorithm, the scheduling order is d, c, b and a. After the scheduling, we can easily get that link d is assigned time slots from 0 to 5, link c is assigned time slots from 7 to 11, and link b is assigned time slots from 12 to 15. Then link a cannot be assigned time slots from 0 to 5, from 8 to 13 and from 13 to 17 due to the interferences with links d, c and b respectively. Because of the consecutive constraint, time slots 6 and 7 cannot be assigned, and these time slots are referred to as gaps in the scheduling. By using the first-fit heuristic, link a is assigned time slots from 18 to 20.

Algorithm 6.5 Centralized Spatio-Temporal link scheduling with Weighted Traffic Load under the Consecutive Constraint (CIST-WC)

```
Input: A graph $G = (V, E)$ with weights on each link.
Output: A valid CIST-WC scheduling.
1: Construct the S-STC graph $G_{sstc}$ with $t = 4W\delta^*(G_{stc}) + \lceil \frac{c\Delta_T + R}{c \cdot t_s} \rceil + 1$, and assign a
   weight $w_u$ to each vertex $(u, t_i)$.
2: Sort the vertices according to the weight information.
3: Schedule the link with higher weight earlier, and assign each link $u$ the smallest $w_u$
   available consecutive time slots.
```

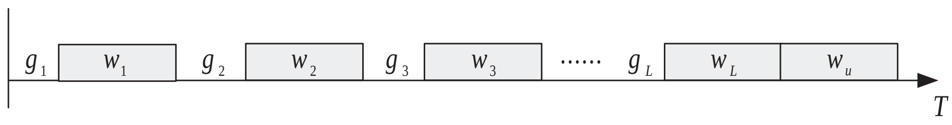


Figure 6.8: The time slots that cannot be assigned to link u in G_{sstc} when u is scheduled.

Theorem 6.10. *The number of transmitting time slots assigned by the CIST-WC algorithm is at most $8C_1$ times of the optimum in two-dimensional networks.*

Proof. Suppose that all the links have been already scheduled using the CIST-WC algorithm, and link u is assigned the last w_u time slots, as shown in Figure 6.8. In the figure, w_i $(1 \leqslant i \leqslant L)$ is the number of consecutive time slots occupied by the links that interfere with link u in the S-STC graph G_{sstc}, and $g_1, g_2, \cdots, g_L$ represent the gaps which are not large enough for scheduling link u. Note that the links scheduled before link u may reuse some time slots. Since the links are scheduled in the non-increasing order of weights, w_i $(1 \leqslant i \leqslant L)$ is not smaller than w_u. As link u is assigned the smallest w_u consecutive time slots using the first-fit heuristic, g_i $(1 \leqslant i \leqslant L)$ is smaller than w_u. The upper bound of the number of transmitting time slots assigned by the CIST-WC algorithm is

$$\sum_{i=1}^{L} g_i + \sum_{i=1}^{L} w_i + w_u < L \cdot w_u + \sum_{i=1}^{L} w_i + w_u < 2(\sum_{i=1}^{L} w_i + w_u).$$

In the CIST-WC algorithm, links with heavier traffic loads are scheduled earlier. Similar to Theorem 6.5, the lower bound of the number of transmitting time slots assigned by the CIST-WC algorithm is $\frac{\sum_{i=1}^{L} w_i + w_u}{4C_1}$. Therefore, we get the approximation ratio of the CIST-WC algorithm in two-dimensional networks is $\frac{2(\sum_{i=1}^{L} w_i + w_u)}{\frac{\sum_{i=1}^{L} w_i + w_u}{4C_1}} = 8C_1$. □

Theorem 6.11. *The number of transmitting time slots assigned by the CIST-WC algorithm is at most* $8C_1'$ *times of the optimum in three-dimensional networks.*

Proof. Similar to Theorem 6.10, the upper bound and the lower bound of the number of transmitting time slots assigned by the CIST-WC algorithm are $2(\sum_{i=1}^{L} w_i + w_u)$ and $\frac{\sum_{i=1}^{L} w_i + w_u}{4C_1'}$ respectively. Therefore, we get the approximation ratio of the CIST-WC algorithm in three-dimensional networks is $\frac{2(\sum_{i=1}^{L} w_i + w_u)}{\frac{\sum_{i=1}^{L} w_i + w_u}{4C_1'}} = 8C_1'$. □

The *Distributed Spatio-Temporal link scheduling with Weighted Traffic Load under the Consecutive Constraint* (DIST-WC) is shown in Algorithm 6.6. The basic idea is to use the smallest-ID-first ordering rather than the weight ordering. In the time slot assignment, each link u is assigned the smallest w_u available consecutive time slots that are not used by the interfering links after obtaining the channel.

Theorem 6.12. *The number of transmitting time slots assigned by the DIST-WC algorithm is at most* $4(K+1)C_1$ *times of the optimum in two-dimensional networks, where* $K = \frac{W}{w}$*,* $W = \max\limits_{1 \leqslant i \leqslant m} \{w_i\}$ *and* $w = \min\limits_{1 \leqslant i \leqslant m} \{w_i\}$.

Proof. Suppose that link u is scheduled in the last w_u time slots, and all the other links have already been scheduled. w_i $(1 \leqslant i \leqslant L)$ is the number of consecutive time slots occupied by the links that interfere with link u in the S-STC graph G_{sstc}, and $g_1, g_2, \cdots, g_L$

Algorithm 6.6 Distributed Spatio-Temporal link scheduling with Weighted Traffic Load under the Consecutive Constraint (DIST-WC)

```
1: Each node $v_j$, with a unique ID, knows its own location. Mark each node $v_j$ initially
   as unscheduled.
2: Calculate $w_u$ of each link $u$.
3: Each node $v_j$ broadcasts its ID and location information to its $k$-hop neighbors using
   a simple flooding with TTL = $k$.
4: if node $v_j$ is unscheduled and its ID is smaller than any other unscheduled $k$-hop
   neighbor then
5:     $v_j$, as a receiver, collects the information when its transmitters cannot transmit
       due to the interferences.
6:     For each incident link $u$ with weight $w_u$ of $v_j$, assign the smallest $w$ available
       consecutive time slots to link $u$ which do not interfere with the links that have
       already been scheduled.
7:     $v_j$ broadcasts the information to its $k$-hop neighbors, and these nodes cannot trans-
       mit or receive in the corresponding time slots considering the propagation delay
       due to the interferences.
8:     $v_j$ marks itself as scheduled.
9: end if
```

represent the gaps which are not large enough for scheduling link u. The upper bound of the number of transmitting time slots assigned by the DIST-WC algorithm is

$$\begin{aligned} T = \sum_{i=1}^{L} g_i + \sum_{i=1}^{L} w_i + w_u \qquad &< \sum_{i=1}^{L} w_u + \sum_{i=1}^{L} w_i + w_u \\ \leq \sum_{i=1}^{L} K \cdot w_i + \sum_{i=1}^{L} w_i + w_u &< (K+1)(\sum_{i=1}^{L} w_i + w_u). \end{aligned}$$

Similar to Theorem 6.5, the lower bound of the number of transmitting time slots assigned by the DIST-WC algorithm is $\frac{\sum_{i=1}^{L} w_i + w_u}{4C_1}$. Therefore, we get the approximation ratio of the DIST-WC algorithm in two-dimensional networks is $\frac{(K+1)(\sum_{i=1}^{L} w_i + w_u)}{\frac{\sum_{i=1}^{L} w_i + w_u}{4C_1}} = 4(K+1)C_1$. □

Theorem 6.13. *The number of transmitting time slots assigned by the DIST-WC algo-*

rithm is at most $4(K+1)C_1'$ times of the optimum in three-dimensional networks.

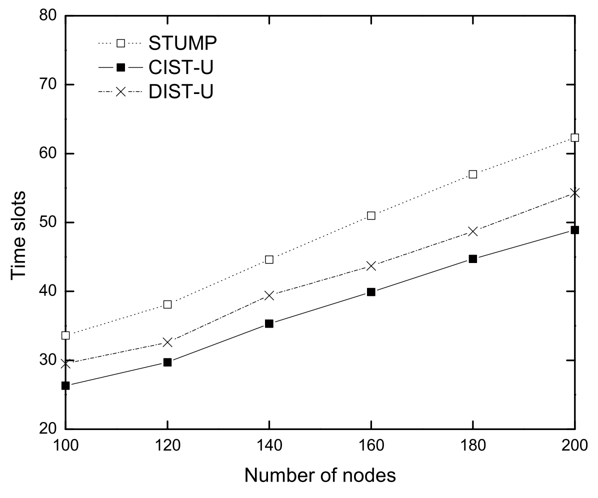
Proof. Similar to Theorem 6.12, the upper bound and the lower bound of the number of transmitting time slots assigned by the DIST-WC algorithm are $(K+1)(\sum_{i=1}^{L} w_i + w_u)$ and $\frac{\sum_{i=1}^{L} w_i+w_u}{4C_1'}$ respectively. Therefore, we get the approximation ratio of the DIST-WC algorithm in three-dimensional networks is $\frac{(K+1)(\sum_{i=1}^{L} w_i+w_u)}{\frac{\sum_{i=1}^{L} w_i+w_u}{4C_1'}} = 4(K+1)C_1'$. □

Complexity Analysis: In the CIST-WC algorithm, the time required to construct the S-STC graph is $O(t\Delta m) = (\Delta^2 m)$, and the time to sort the vertices and use the first-fit heuristic is $O(mlogm)$. Hence, the time complexity of the CIST-WC algorithm is $O(\Delta^2 m + mlogm)$. The message complexity and time complexity of the DIST-WC algorithm are same to those of the DIST-U algorithm, which are $O(\rho n)$ and $O(n)$ respectively.

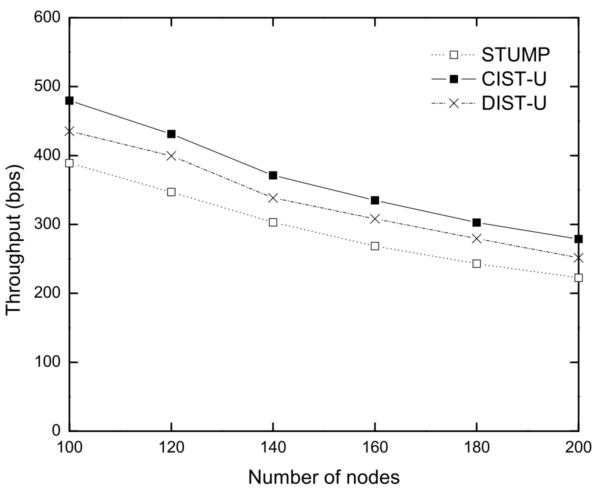
6.5 Simulation Results

In this section, we study the performance of the spatio-temporal link scheduling in UWSNs, and we also compare our approaches with the STUMP algorithm [64]. We evaluate the performance in terms of two metrics, the number of transmitting time slots assigned and throughput. Though energy consumption is an important metric in UWSNs, we do not show its effect on the performance in this section since all the approaches consume the same energy under identical conditions due to the utilization of TDMA.

We adopt the following simulation parameters: the propagation speed of the acoustic signal is 1.5 km/s, the data packet length is 300 bytes, the radio bandwidth is 15 kbps, and the link transmission delay of one data packet is 0.16 s. The length of the time slot (i.e., one time unit) is set to be 0.2 s. In the deployment, nodes with a transmission



(a) Average number of transmitting time slots assigned



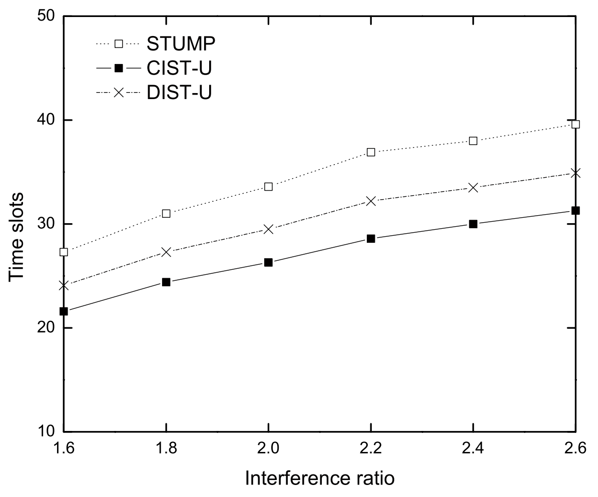
(b) Average throughput

Figure 6.9: The impact of network size under unified traffic load.

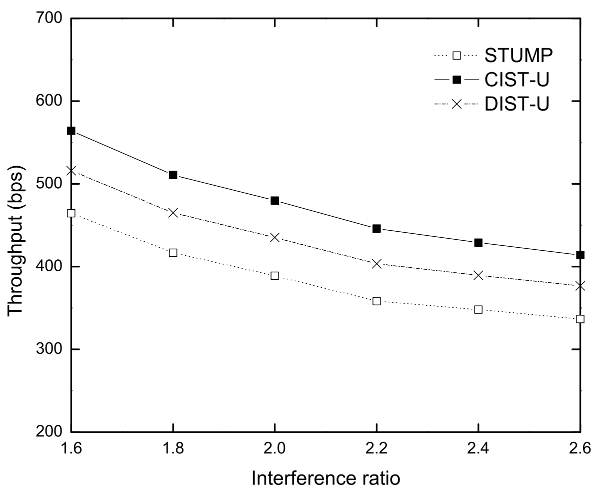
range of 1.5 km and an interference range of 3 km are deployed in a three-dimensional space of 10 $km \times 10$ $km \times 1$ km. We assume there is a sink on the center of the top surface of the 3-D space, and all traffics are towards it. We construct a shortest path tree rooted at the sink node as the topology of the network, and this topology determines the routing of each source to the sink. For a given number of nodes, 50 network topologies are randomly generated, and the average performances over all these networks are reported. In the simulations, we test the link scheduling algorithms under both unified and weighted traffic load scenarios.

6.5.1 Under Unified Traffic Load Scenario

In this subsection, we evaluate the performance of our algorithms under the unified traffic load scenario. In the unified traffic load scenario, we assume that nodes have the ability of data aggregation and can use one time slot to transmit all data in one link, then each link is assigned one time slot to transmit its package.



(a) Average number of transmitting time slots assigned

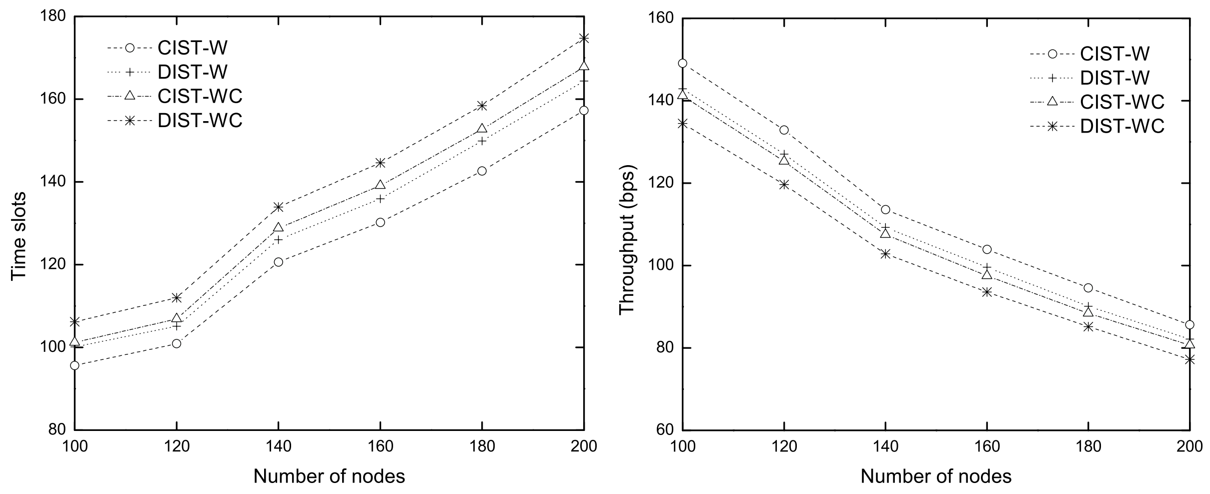


(b) Average throughput

Figure 6.10: The impact of interference ratio under unified traffic load.

To test the impact of network size, the network varies from 100 nodes to 200 nodes with a step of 20 nodes, and the interference ratio is set to be 2, i.e., the interference range of each node is 3 km. To test the impact of interference ratio, the interference ratio varies from 1.6 to 2.6 with a step of 0.2, and the network size is set to be 100.

We first evaluate the average number of transmitting time slots assigned and average throughput of our algorithms (CIST, DIST) and STUMP when the network size varies. As the number of nodes increases, the average number of transmitting time slots assigned in all the three algorithms increases, as shown in Figure 6.9(a). Consequently, the average throughput decreases as the number of nodes increases as shown in Figure 6.9(b). The STUMP algorithm performs worse than the CIST-U algorithm as it overestimates the effect of interferences. The performance of the DIST-U algorithm is comparable to that of the centralized scheduling, e.g., the throughput in the DIST-U algorithm is about 10% less than that in the CIST-U algorithm.



(a) Average number of transmitting time slots assigned

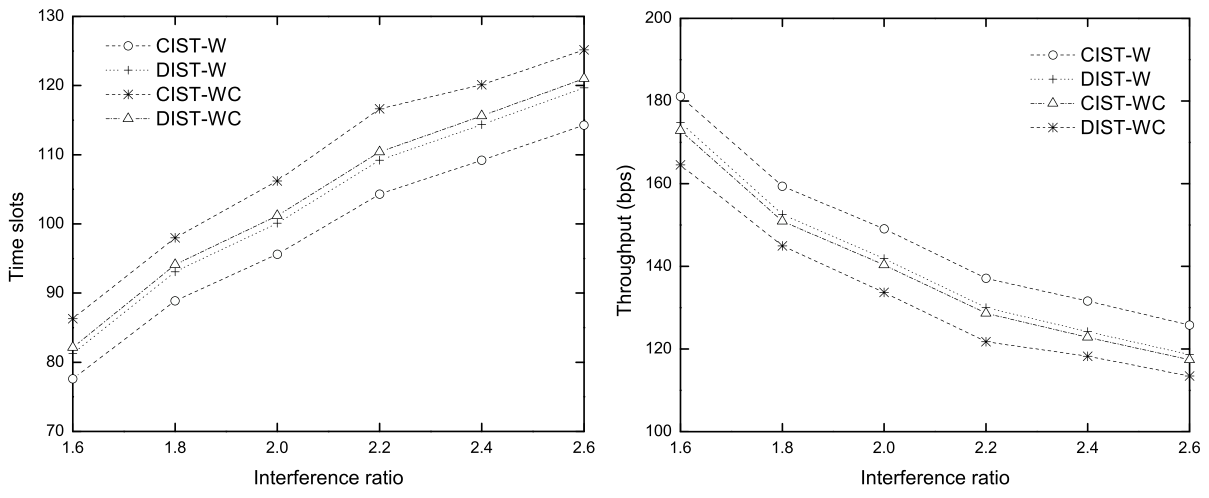
(b) Average throughput

Figure 6.11: The impact of network size under weighted traffic load.

We then evaluate the average number of transmitting time slots assigned and average throughput of our algorithms (CIST, DIST) and STUMP when the interference ratio varies. For all the three algorithms, as the interference ratio increases, the average number of transmitting time slots assigned increases and the average throughput decreases as shown in Figure 6.10. This indicates that the interference is detrimental to the link scheduling in UWSNs. Among the three algorithms, the CIST-U algorithm still has the best performance.

6.5.2 Under Weighted Traffic Load Scenario

In this subsection, we evaluate the performance of our algorithms under the weighted traffic load scenario. In the weighted traffic load scenario, the traffic load of each link is calculated by the total amount of traffics that need to be transmitted, and then each link u has its weight information w_u.



(a) Average number of transmitting time slots assigned

(b) Average throughput

Figure 6.12: The impact of interference ratio under weighted traffic load.

Figure 6.11 shows the average number of transmitting time slots assigned and average throughput of our algorithms (CIST-W, DIST-W, CIST-WC, DIST-WC) when the network size varies. For all the four algorithms, as the number of nodes increases, the average number of transmitting time slots assigned increases and the average throughput decreases. We can see that the CIST-WC algorithm has less throughput than the CIST-W algorithm due to the consecutive constraint. The performances of the distributed algorithms, i.e. DIST-W and DIST-WC, are worse than those of the corresponding centralized algorithms, but still comparable.

Figure 6.12 shows the average number of transmitting time slots assigned and average throughput of our algorithms (CIST-W, DIST-W, CIST-WC, DIST-WC) when the interference ratio varies. We can also see that the CIST-WC algorithm has a worse performance than the CIST-W algorithm. The performances of the distributed algorithms are comparable to those of the corresponding centralized algorithms.

We summarize observations from the simulation results as follows:

- Our proposed algorithms have better performance than STUMP.
- Our proposed distributed algorithms can achieve performance comparable to the centralized algorithms.

6.6 Summary

In this chapter, we investigate the spatio-temporal link scheduling in UWSNs. To overcome the spatio-temporal uncertainty, we propose a novel slotted spatio-temporal conflict graph which considers both the packet propagation delay and link transmission delay. We present efficient scheduling algorithms that have theoretical performance bounds for both unified and weighted traffic loads. Finally, we evaluate the proposed algorithms via simulations, which show the efficiency of the algorithms in terms of the number of transmitting time slots assigned and throughput.

Chapter 7

Conclusions and Suggestions for Future Research

In this chapter, we conclude this thesis by summarizing our original contributions in Section 7.1, and outline the directions for future research in Section 7.2.

7.1 Conclusions

In this thesis, we investigate the energy-efficient interference-free link schedulings in WSNs, where where sensors can be deployed in terrestrial or underwater areas.

In TWSNs, we first identify the contiguous link scheduling problem to reduce the frequency of state transitions, which is proven to be NP-complete. We formulate the contiguous link scheduling as an interval vertex-coloring of a so-called merged conflict graph, and we propose a centralized scheduling algorithm with a theoretical performance bound to the optimum. Especially, if the topology is a tree, each node can start up at most twice in a scheduling period.

Considering the drawback of the merged conflict graph, we improve the centralized scheduling algorithm using an interference matrix, including recursive backtracking and minimum conflicts heuristic. We also develop efficient distributed algorithms, and the distributed scheduling with efficient delay algorithm can even reduce the network delay. We prove that our proposed centralized and distributed algorithms have theoretical performance bounds to the optimum in both homogeneous and heterogeneous networks. We conduct extensive simulations to show the effectiveness of the proposed centralized and distributed algorithms.

To minimize the frequency of state transitions, we identify the compact wakeup scheduling problem. However, not all communication graphs have valid compact wakeup schedulings, and the problem of deciding whether a valid compact wakeup scheduling exists for an arbitrary graph is NP-complete. Hence, we focus on particular network topologies, such as trees and grid graphs in the compact wakeup schedulings. We propose polynomial-time algorithms with the optimum solutions for trees and grid graphs. In grid graphs, we present all the possible coloring patterns, and analyze both upper and lower bounds of the compact wakeup scheduling. Simulations demonstrate the effectiveness of our proposed algorithms.

In UWSNs, we address the spatio-temporal link scheduling problem, which is also NP-hard. To deal with the problem, we construct a three-dimensional slotted spatio-temporal conflict graph. We propose centralized and distributed scheduling algorithms with theoretical performance bounds to the optimum for both unified and weighted traffic loads. In the weighted traffic load scenario, we consider the scheduling with and without the constraint of consecutive time slots. The simulation results illustrate that our proposed algorithms outperform the existing approach.

7.2 Suggestions for Future Research

In this section, we provide some suggestions for future research. The research work that has been completed so far can be extended in the directions as follows.

- **Contiguous link scheduling under the physical interference model:** The contiguous link scheduling problem has been investigated under the protocol interference model in this thesis. The problem is still open under the physical interference model, which is closer to realistic networks. In the physical interference model, the transmission of a link is successful if the SINR at the receiver is above a certain threshold. Recently, the link scheduling under the physical interference model has been investigated in [113, 132], and could serve as a guideline.

- **Bidirectional communication in contiguous link scheduling:** In the contiguous link scheduling, we mainly consider the upstream links for data aggregation and schedule the incoming links together to reduce the frequency of state transitions. It is interesting to consider the bidirectional communication, and one possible solution is to schedule all the incoming links together and all the outing links together.

- **Contiguous link scheduling in UWSNs:** The contiguous link scheduling can be extended to UWSNs, where the long propagation delay needs to be considered. After such a scheduling in UWSNs, a node is assigned consecutive time slots to receive data from its neighbors. Note that the transmission time of a link is different from the reception time in underwater networks, each neighbor can determine its transmission time by considering the propagation delay. Due to the unique feature of UWSNs, a new conflict graph can be constructed to describe the interference relationship.

- **Investigation of other topologies with valid compact wakeup schedulings:** The compact wakeup scheduling mainly focuses on trees and grid graphs, as not all communication graphs have valid compact wakeup schedulings. We can still obtain efficient algorithms for other kinds of network topologies with valid compact wakeup schedulings, such as hexagon grid graphs.

- **Compact wakeup scheduling for general graphs:** For a given topology where a valid compact wakeup scheduling does not exist, one approach is to find an efficient scheduling with the minimum waiting period. Interestingly, Giaro et al. [46] investigate the graphs without valid interval edge-colorings, and raise an issue called consecutive coloring deficiency of a graph G, which is the minimum number of pendant edges attached to G such that G can be consecutively colorable. The solutions to deal with the consecutive coloring deficiency problem can be helpful in the compact wakeup scheduling for general graphs.

- **Spatio-temporal link scheduling in heterogeneous networks:** The spatio-temporal link scheduling has been addressed in homogeneous UWSNs in this thesis. We can further extend our approaches to heterogeneous networks, where each node has a different transmission range and a different interference range. In heterogeneous networks, a new theoretical performance bound analysis is needed.

- **Broadcast scheduling in UWSNs:** Another interesting topic in UWSNs is to design an efficient broadcast scheduling to deal with the broadcast storm problem [85], in which a simple flooding leads to serious redundancy and collisions. Similar to the spatio-temporal link scheduling in UWSNs, the spatio-temporal uncertainty due to the long propagation delay makes the broadcast scheduling a challenging issue.

References

[1] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. Occupancy-driven energy management for smart building automation. In *Proc. of the ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building (BuildSys)*, 2010.

[2] G. S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo. Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[3] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257–279, 2005.

[4] I. F. AKyildiz, D. Pompili, and T. Melodia. State-of-the-art in protocol research for underwater acoustic sensor networks. *ACM Mobile Computing and Communication Review*, 11:11–22, 2007.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[6] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4):445–487, 2005.

[7] M. Alicherry, R. Bhatia, and L. E. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[8] Z. Alliance. *ZigBee Specification, version 1.0*, 2005.

[9] T. Alsulaiman, S. K. Prasad, and A. Zelikovsky. Distributed algorithms for TDMA link scheduling in sensor networks. *International Journal of Networking and Computing*, 3(1):55–74, 2013.

[10] M. A. Ameen, S. M. R. Islam, and K. Kwak. Energy saving mechanisms for MAC protocols in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2010.

[11] E. Arikan. Some complexity results about packet radio networks. *IEEE Transactions on Information Theory*, 30(4):681–685, 1984.

[12] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[13] A. S. Asratian and R. R. Kamalian. Interval coloring of the edges of a graph (in russian). *Applied Math*, 5:25–34, 1987.

[14] A. S. Asratian and R. R. Kamalian. Investigation on interval edge-colorings of graphs. *Journal of Combinatorial Theory, Series B*, 62(1):34–43, 1994.

[15] C. R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. D. Minassians, G. Dervisoglu, L. Gutnik, M. B. Haick, C. Ho, M. Koplow, J. Mangold,

S. Robinson, M. Rosa, M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E. S. Leland, T. Pering, and P. K. Wright. Wireless sensor networks for home health care. In *Proc. of the IEEE International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, 2007.

[16] S. Bapat, V. Kulathumani, and A. Arora. Analyzing the yield of exscal, a large-scale wireless sensor network experiment. In *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2005.

[17] I. Bekmezci and F. Alagoz. Energy efficient, delay sensitive, fault tolerant wireless sensor network for military monitoring. *International Journal of Distributed Sensor Networks*, 5(6):729–747, 2009.

[18] C. Berge. *Graphs and Hyper Graphs.* North-Holland, 1973.

[19] R. E. Best. *Phase-locked Loops: Design, Simulation and Applications.* McGraw-Hill, 2003.

[20] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A media access protocol for wireless LAN's. In *Proc. of the annual conference of the Special Interest Group on Data Communication (SIGCOMM)*, 1994.

[21] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, 2004.

[22] E. Cayirci, H. Tezcan, Y. Dogan, and V. Coskun. Wireless sensor networks for underwater survelliance systems. *Ad Hoc Networks*, 4(4):431–446, 2006.

[23] Y. D. Chen, C. Y. Lien, S. W. Chuang, and K. P. Shih. DSSS: A TDMA-based MAC protocol with dynamic slot scheduling strategy for underwater acoustic sensor networks. In *Proc. of the IEEE OCEANS*, 2011.

[24] O. Chipara, C. Lu, and J. Stankovic. Dynamic conflict-free query scheduling for wireless sensor networks. In *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2006.

[25] N. Chirdchoo, W. S. Soh, and K. C. Chua. Aloha-based MAC protocols with collision avoidance for underwater acoustic networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[26] N. Chirdchoo, W. S. Soh, and K. C. Chua. MU-Sync: A time synchronization protocol for underwater mobile networks. In *Proc. of the ACM International Workshop on UnderWater Networks (WUWNet)*, 2008.

[27] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990.

[28] S. Croce, F. Marcelloni, and M. Vecchio. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *The Computer Journal*, 51(2):227–239, 2008.

[29] Crossbow Technology Inc. *Mica2 Data Sheet [Online]*. Available: http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[30] B. P. Crow, I. Widjaja, L. G. Kim, and P. T. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–126, 1997.

[31] J. H. Cui, J. Kong, M. Gerla, and S. Zhou. The challenges of building mobile underwater wireless networks for aquatic applications. *IEEE Network*, 20(3):12–18, 2006.

[32] S. Cui, A. J. Goldsmith, and A. Bahai. Energy-constrained modulation optimization. *IEEE Transactions on Wireless Communications*, 4(5):2349–2360, 2005.

[33] D. de Werra and A. Hertz. Consecutive colorings of graphs. *Mathematical Methods of Operations Research*, 32(1):1432–2994, 1988.

[34] P. Djukic and S. Valaee. Link scheduling for minimum delay in spatial re-use TDMA. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[35] A. Ephremedis and T. V. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, 38(4):456–460, 1990.

[36] S. C. Ergen and P. Varaiya. TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks*, 16(4):985–997, 2010.

[37] L. Evers, P. J. M. Havinga, J. Kuper, M. E. M. Lijding, and N. Meratnia. Sensorscheme: Supply chain management automation using wireless sensor networks. In *Proc. of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2007.

[38] L. Fu, S. C. Liew, and J. Huang. Power controlled scheduling with consecutive transmission constraints: Complexity analysis and algorithm design. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[39] V. Gabale, K. Chebrolu, B. Raman, and S. Bijwe. PIP: A multichannel, TDMA-based MAC for efficient and scalable bulk transfer in sensor networks. *ACM Transactions on Sensor Networks*, 8(4):28:1–28:34, 2012.

[40] S. Gandham, M. Dawande, and R. Prakash. Link scheduling in sensor networks: Distributed edge coloring revisited. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[41] R. Gandhi, S. Parthasarathy, and A. Mishra. Minimizing broadcast latency and redundancy in ad hoc networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[42] T. Gao, D. Greenspan, M. Welsh, R. R. Juang, and A. Alm. Vital signs monitoring and patient tracking over a wireless network. In *Proc. of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, 2005.

[43] K. Giaro. *Compact task scheduling on dedicated processors with no waiting periods (in Polish)*. PhD thesis, Technical University of Gdansk, Poland, 1999.

[44] K. Giaro and M. Kubale. Consecutive edge-colorings of complete and incomplete cartesian products of graphs. *Congressus Numerantium*, 128:143–149, 1997.

[45] K. Giaro and M. Kubale. Compact scheduling of zero-one time operations in multi-stage systems. *Discrete Applied Mathematics*, 145(1):95–103, 2004.

[46] K. Giaro, M. Kubale, and M. Malafiejski. On the deficiency of bipartite graphs. *Discrete Applied Mathematics*, 94(1):193–203, 1999.

[47] V. C. Gungor, B. Lu, and G. P. Hancke. Opportunities and challenges of wireless sensor networks in smart grid. *IEEE Transactions on Industrial Electronics*, 57(10):3557–3564, 2010.

[48] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[49] J. A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile. IEEE 802.15. 4: A developing standard for low-power low-cost wireless personal area networks. *IEEE Network*, 15(5):12–19, 2001.

[50] Z. J. Haas, J. Deng, B. Liang, P. Papadimitratos, and S. Sajama. *Wireless ad hoc Networks*. John Wiley & Sons, Inc., 2002.

[51] S. Han, Y. Nohy, U. Leez, and M. Gerla. M-FAMA: A multi-session MAC protocol for reliable underwater acoustic streams. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[52] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li. Research challenges and applications for underwater sensor networking. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2006.

[53] C. C. Hsu, K. F. Lai, C. F. Chou, and K. C. J. Lin. ST-MAC: Spatial-temporal MAC scheduling for underwater sensor networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[54] P. H. Huang, Y. Chen, B. Krishnamachari, and A. Kumar. Link scheduling in a single broadcast domain underwater networks. In *Proc. of the Third IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 2010.

[55] S. Hussain, S. Schaffner, and D. Moseychuck. Applications of wireless sensor networks and RFID in a smart home environment. In *Proc. of the Annual Conference on Communication Networks and Services Research (CNSR)*, 2009.

[56] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2003.

[57] G. Jolly and M. Younis. An energy-efficient, scalable and collision-free MAC layer protocol for wireless sensor networks. *Wireless Communications and Mobile Computing*, 5(3):285–304, 2005.

[58] L. T. Jung and A. Abdullah. Wireless sensor networks: Data packet size optimization. In N. Zaman, K. Ragab, and A. B. Abdullah, editors, *Wireless Sensor Networks and Energy Efficiency: Protocols, Routing and Management*, pages 305–328. IGI Global, 2012.

[59] K. Kredo II and P. Mohapatra. A hybrid medium access control protocol for underwater wireless networks. In *Proc. of the ACM International Workshop on UnderWater Networks (WUWNet)*, 2007.

[60] K. Kredo II and P. Mohapatra. Medium access control in wireless sensor networks. *Computer Networks*, 51:961–994, 2007.

[61] K. Kalpakis, K. Dasgupta, and P.Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, 2003.

[62] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[63] A. Keshavarzian, H. Lee, and L. Venkatraman. Wakeup scheduling in wireless sensor networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.

[64] K. Kredo, P. Djukic, and P. Mohapatra. STUMP: Exploiting position diversity in the staggered TDMA underwater MAC protocol. In *Proc. of IEEE INFOCOM Mini-Conference*, 2009.

[65] B. Krishnamachari. *Networking Wireless Sensors.* Cambridge University Press, 2005.

[66] S. O. Krumke, M. V. Marathe, and S. S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks*, 7(6):575–584, 2001.

[67] M. Kubale. Interval vertex-coloring of a graph with forbidden colors. *Discrete Mathematics*, 74(1-2):125–136, 1989.

[68] M. Kubale. Interval edge coloring of a graph with forbidden colors. *Discrete Mathematics*, 121(1-3):135–143, 1993.

[69] Q. Lampin, D. Barthel, and F. Valois. Efficient route redundancy in DAG-based wireless sensor networks. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2010.

[70] A. Ledeczi, A. Nadas, P. Volgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dora, K. Molnar, M. Maroti, and G. Simon. Countersniper system for urban warfare. *ACM Transactions on Sensor Networks*, 1(2):153–177, 2005.

[71] M. Li and Y. Liu. Underground structure monitoring with wireless sensor networks. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[72] Q. Li and D. Rus. Global clock synchronization in sensor networks. *IEEE Transactions on Computers*, 55(2):214–226, 2006.

[73] Z. Li, Z. Guo, H. Qu, F. Hong, P. Chen, and M. Yang. UD-TDMA: A distributed TDMA protocol for underwater acoustic sensor network. In *Proc. of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2009.

[74] L. Liu, S. Zhou, and J. H. Cui. Prospects and problems of wireless communication for underwater sensor networks. *Wireless Communications and Mobile Computing*, 8(8):977–994, 2008.

[75] M. Lopez-Nores, J. J. Pazos-Arias, J. Garcia-Duque, and Y. Blanco-Fernandez. Monitoring medicine intake in the networked home: The iCabiNET solution. In *Proc. of the International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, 2008.

[76] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Proc. of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[77] J. Ma, W. Lou, Y. Wu, X. Y. Li, and G. Chen. Energy efficient TDMA sleep scheduling in wireless sensor networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[78] R. Mahjourian, F. Chen, R. Tiwari, M. Thai, H. Zhai, and Y. Fang. An approximation algorithm for conflict-aware broadcast scheduling in wireless ad hoc networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2008.

[79] J. Mao, Z. Wu, and X. Wu. A TDMA scheduling scheme for many-to-one communications in wireless sensor networks. *Computer Communications*, 30(4):863–872, 2007.

[80] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 30(3):417–427, 1983.

[81] V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff. Design of surveillance sensor grids with a lifetime constraint. In *Proc. of the European Conference on Wireless Sensor Networks (EWSN)*, 2004.

[82] J. Misra and D. Gries. A constructive proof of vizings theorem. *Information Processing Letters*, 41(3), 1992.

[83] D. Musiani, K. Lin, and T. S. Rosing. An active sensing platform for wireless structural health monitoring. In *Proc. of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[84] C. Y. Ngo and V. O. K. Li. Centralized broadcast scheduling in packet radio networks via genetic-fix algorithms. *IEEE Transactions on Communications*, 51(9):1439–1441, 2003.

[85] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[86] Y. Noh, P. Wang, U. Lee, D. Torres, and M. Gerla. DOTS: A propagation delay-aware opportunistic MAC protocol for underwater sensor networks. In *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2010.

[87] J. Partan, J. Kurose, and B. N. Levine. A survey of practical issues in underwater networks. In *Proc. of the ACM International Workshop on UnderWater Networks (WUWNet)*, 2006.

[88] Y. Peng, Z. Li, D. Qiao, and W. Zhang. Delay-bounded MAC with minimal idle listening for sensor networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[89] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[90] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[91] S. Ramanathan. A unified framework and algorithms for (T/F/C)DMA chanel assignment in wireless networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 1997.

[92] S. Ramanathan and E. L. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking*, 1(2):166–177, 1993.

[93] R. Ramaswami and K. K. Parhi. Distributed scheduling of broadcasts in a radio network. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 1989.

[94] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-MAC: A hybrid MAC for wireless sensor networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[95] I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.

[96] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 2nd Edtion, 2003.

[97] S. Santini, B. Ostermaier, and A. Vitaletti. First experiences using wireless sensor networks for noise pollution monitoring. In *Proc. of the workshop on Real-world wireless sensor networks (REALWSN)*, 2008.

[98] W. K. G. Seah, Y. K. Tan, and A. T. S. Chan. Research in energy harvesting wireless sensor networks and the challenges ahead. In D. Filippini, editor, *Autonomous*

Sensor Networks, Springer Series on Chemical Sensors and Biosensors, pages 73–93. Springer Berlin Heidelberg, 2013.

[99] Sentilla Corporation. *Tmote Sky Datasheet [Online].* Available: http://www.sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf.

[100] S. V. Sevastjanov. On interval colorability of a bipartite graph (in russian). *Metody Diskretnogo Analiza*, 50:61–72, 1990.

[101] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2003.

[102] M. Skubic, G. Alexander, M. Popescu, M. Rantz, and J. Keller. A smart home application to eldercare: Current status and lessons learned. *Technology and Health Care*, 17(3):183–201, 2009.

[103] W. Z. Song, R. Huang, B. Shirazi, and R. LaHusen. TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks. In *Proc. of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2009.

[104] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson. DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2008.

[105] A. Syed and J. Heidemann. Time synchronization for high latency acoustic networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[106] A. A. Syed, W. Ye, and J. Heidemann. T-lohi: A new class of MAC protocols for underwater acoustic sensor networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[107] A. A. Syed, W. Ye, B. Krishnamachari, and J. Heidemann. Understanding spatio-temporal uncertainty in medium acess with ALOHA protocols. In *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2007.

[108] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson. PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[109] N. Thepvilojanapong, Y. Tobe, and K. Sezaki. On the construction of efficient data gathering tree in wireless sensor networks. In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2005.

[110] H. Urban. *Handbook of Underwater Acoustic Engineering.* STN Atlas Elekronik, 2002.

[111] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[112] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[113] P. J. Wan, O. Frieder, X. Jia, F. Yao, X. Xu, and S. Tang. Wireless link scheduling under physical interference model. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[114] P. J. Wan, C. Ma, Z. Wang, B. Xu, M. Li, and X. Jia. Weighted wireless link scheduling without information of positions and interference/communication radii. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[115] A. Wang, S. Cho, C. Sodini, and A. Chandrakasan. Energy efficient modulation and MAC for asymmetric RF microsensor systems. In *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2001.

[116] L. Wang, T. Gu, H. Chen, X. Tao, and J. Lu. Real-time activity recognition in wireless body sensor networks: From simple gestures to complex activities. In *Proc. of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2010.

[117] Q. Wang, M. Hempstead, and W. Yang. A realistic power consumption model for wireless sensor network devices. In *Proc. of the Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2006.

[118] W. Wang, Y. Wang, X. Y. Li, W. Z. Song, and O. Frieder. Efficient interference-aware TDMA link scheduling for static wireless networks. In *Proc. of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2006.

[119] Y. Wu, X. Y. Li, Y. Liu, and W. Lou. Energy-efficient wake-up scheduling for data collection and aggregation. *IEEE Transactions on Parallel and Distributed Systems*, 21(2):275–287, 2010.

[120] Y. C. Wu, Q. Chaudhari, and E. Serpedin. Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine*, 28(1):124–138, 2011.

[121] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[122] X. Xu, X. Y. Li, P. J. Wan, and M. Song. Efficient aggregation scheduling in multihop wireless sensor networks with SINR constraints. *IEEE Transactions on Mobile Computing*, 2012.

[123] J. Yackoski and C. C. Shen. UW-FLASHR: Achieving high channel utilization in a time-based acoustic MAC protocol. In *Proc. of the ACM International Workshop on UnderWater Networks (WUWNet)*, 2008.

[124] H. Yang and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *Proc. of the International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[125] W. Ye and J. Heidemann. Medium access control in wireless sensor networks. In C. S. Raghavendra, K. M. Sivalingam, and T. Znati, editors, *Wireless Sensor Networks*, pages 73–91. Kluwer Academic Publishers, 2004.

[126] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2002.

[127] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[128] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *Proc. of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2005.

[129] C. Zhang, J. Kurose, Y. Liu, D. Towsley, and M. Zink. A distributed algorithm for joint sensing and routing in wireless networks with non-steerable directional antennas. In *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2006.

[130] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In *Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[131] B. Zhou, J. Cao, X. Zeng, and H. Wu. Adaptive traffic light control in wireless sensor network-based intelligent transportation system. In *Proc. of the IEEE Vehicular Technology Conference (VTC)*, 2010.

[132] Y. Zhou, X. Y. Li, M. Liu, Z. Li, S. Tang, X. Mao, and Q. Huang. Distributed link scheduling for throughput maximization under physical interference model. In *Proc. of IEEE INFOCOM Mini-conference*, 2012.

[133] Z. Zhou, Z. Peng, J. H. Cui, and Z. Jiang. Handling triple hidden terminal problems for multichannel MAC in long-delay underwater sensor networks. *IEEE Transactions on Mobile Computing*, 11(1):139–154, 2012.

[134] Y. Zhu, Z. Jiang, Z. Peng, M. Zuba, J. H. Cui, and H. Chen. Toward practical MAC design for underwater acoustic networks. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, 2013.