# 國 立 交 通 大 學

資訊學院
資訊工程學系

博 士 論 文

ZigBee 無線感測網路之通訊協定與應用設計

Communication Protocols and Applications for ZigBee-Based
Wireless Sensor Networks

研 究 生：潘孟鉉

指導教授：曾煜棋　教授

中 華 民 國 九 十 七 年 四 月

# ZigBee 無線感測網路之通訊協定與應用設計
# Communication Protocols and Applications for ZigBee-Based Wireless Sensor networks

研　究　生：潘孟鉉　　Student：Meng-Shiuan Pan

指導教授：曾煜棋　　Advisor：Yu-Chee Tseng

國 立 交 通 大 學 資 訊 學 院

資 訊 工 程 學 系

博 士 論 文

A Dissertation

Submitted to Department of Computer Science

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

April 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年四月

# ZigBee 無線感測網路之通訊協定與應用設計

學生：潘孟鉉　　　　　　　　　　指導教授：曾煜棋教授

國立交通大學資訊工程學系(研究所)博士班

## 摘　　要

無線感測網路相關的研究議題得到許多研究單位及學者的關注，近年來ZigBee通訊協定被視為最適用於感測網路的通訊協定，在本篇論文中，我們將提出建構於ZigBee通訊堆疊中之通訊協定與應用。本篇論文包含五個主題，其中前三個主題我們將探討ZigBee網路層通訊協定之設計，內容包含有：ZigBee無線感測網路生成之研究、ZigBee基礎之長鏈狀網路研究以及ZigBee樹狀網路資料傳遞排程。在後兩個主題為兩個可架構於所設計之網路層協定之上的ZigBee無線感測網路應用，分別為：一緊急室內安全監控與逃生系統以及一智慧型燈光控制系統。

在第一個研究主題中，我們研究ZigBee感測網路的生成問題，在ZigBee規範中，一個節點加入到一個網路的條件為：該節點能找到一個父節點能夠給予該節點一ZigBee網路位址，父節點們可利用ZigBee所定義之分散式位址分配法來指定位址給子節點們，這一個位址指定方法相當簡單但是卻限制了一個節點最多可容忍之子節點個數以及整體網路的深度，我們觀察到如果使用ZigBee所定義之網路生成方式，會使得網路位址的使用率偏低，進而造成節點無法連上網路，因此在本論文中，我們提出適用於ZigBe之網路生成方法，我們的目標是為能夠使得網路上的節點能夠自動組態並且形成一個可通訊之網路。

在第二個研究主題中，我們探討一特殊網路-長鏈狀網路，在這一網路中節點們被部署為數個長條狀拓樸，這些長條狀拓樸們連接成一個網路，並且覆蓋整個想要涵蓋之範圍，這一網路看似為一個特例型態網路，但是我們觀察到該網路型態是相當常見於許多無線感測網路之應用中。本論文探討該如何將ZigBee協定用於此一特殊但卻常見之拓樸中。我們從網路層協定來切入討論，發現原有ZigBee

定義之位址指定方式與路由方式並不適用於此長鏈狀網路，因此我們的目標為提出簡單又有效之適用於長鏈狀網路之位址指派方法以及路由通訊協定。

在第三個研究主題中，我們研究ZigBee樹狀網路封包排程，在許多無線感測網路的應用中，網路上的節點被要求回報資料給一個資料收集伺服器，早先所提出之資料回報流排程方法皆著重於要節省網路節點電量消耗與降低回報延遲，但是該些方法皆不能適用於ZigBee網路。在考量ZigBee的特性下，本論文將探討該如何排程節點的信標訊框，並將提出信標排程演算法，目標為針對資料流之特性來排定網路節點之運作時間，以達到省電之目的。

在第四個研究主題中，我們提出一套緊急室內安全監控與逃生系統，該系統可建構於上述所提出之通訊協定方法之上，本系統擁有簡易網路建置方式、可靠拓樸重建以及安全導引人員逃生三項特點。在網路部署完成後，我們的節點將運作於省電模式來監控室內環境狀態，而當網路有緊急事件發生時，節點們則轉換至非省電模式以即時監控網路，同時感測器也會依據急難時感測到的資訊來提供安全的逃生路徑協助室內人員逃離災難現場，保障位處於災難現場人員的生命安全。

最後我們設計一室內智慧型燈光控制系統來提供一更便利以及舒適的生活環境，基於人們在從事各種不同之活動於皆有可能需要不同之照明需求，例如閱讀、看電視等。本系統可依據使用者之需求輔以感測器的回報來自動地調控室內的燈光，調控的目標除了要滿足使用者之需求外，亦強調需達到節能之目的。由本論文所提出之通訊協定方法，我們還可以在其上衍生許多之應用，如老人健康照護、緊急救援，戶外河川水位監測等。

關鍵字：位址分派，回報，IEEE 802.15.4，智慧型建築，燈光控制，長鏈狀網路，孤兒問題，導引，無所不在的運算，排程，無線感測網路，ZigBee。

# Communication Protocols and Applications for ZigBee-Based Wireless Sensor Networks

Student: Meng-Shiuan Pan Advisor: Prof. Yu-Chee Tseng

Department of Computer Science

National Chiao Tung University

## ABSTRACT

ZigBee is a standard which is considered to be suitable for *wireless sensor networks (WSNs)*. In this dissertation, we propose communication protocols and applications based on the ZigBee protocol stack. This dissertation is composed of five works. In the first three works, we put our attention on designing ZigBee-compatible network layer protocols. The first work and second work discuss network formation problems in general ZigBee networks and in a special type of ZigBee network, respectively. Based on the observation that data gathering is a major application of WSNs, in the third work, we design data collection strategies for ZigBee networks. In the last two works, we propose two applications, an emergency guiding and monitoring system and an intelligent light control system, which can operate based on the proposed network layer protocols.

In the first work, we discuss network formation issues in general ZigBee network. According to ZigBee, a device is said to join a network if it can obtain a network address from a parent device. Devices calculate addresses for their child devices by a distributed address assignment scheme. This assignment is easy to implement, but it restricts the number of children of a device and the depth of the network. We observe that if one uses the random formation policy specified in ZigBee, the utilization of the address pool may be very low. Those devices that can not receive network addresses will be isolated from the network and become *orphan* nodes. In this dissertation, we divide the *orphan problem* by two subproblems: the *bounded-degree-and-depth tree formation (BDDTF)* problem and the *end-device maximum matching*

*(EDMM)* problem. We then propose network formation strategies to relieve the orphan problem. The simulation results show that, compared to the ZigBee network formation strategy, the proposed schemes can effectively reduce the number of orphan devices.

Although WSNs have been extensively researched, its deployment is still a big concern. In the second work, we promote a new concept of *long-thin (LT) topology* for WSNs, where a network may have a number of *linear paths* of nodes as backbones connecting to each other. These backbones are to extend the network to the intended coverage areas. At the first glance, a LT WSN only seems to be a special case of numerous WSN topologies. However, we observe, from real deployment experiences, that such a topology is quite general in many applications and deployments. We show that the *address assignment* and thus the *tree routing scheme* defined in the original ZigBee specification may work poorly, if not fail, in a LT topology. We then propose simple, yet efficient, address assignment and routing schemes for a LT WSN. Simulation results are reported.

In most WSN applications, sensors are required to report their sensory data to a sink. This operation is defined as *convergecast*, which means the reverse of broadcast. Existing convergecast solutions have focused on reducing latency and energy consumption. However, a good design should be compliant to standards, in addition to considering these factors. In the third work, we defines a *minimum delay beacon scheduling problem* for quick convergecast in ZigBee tree-based wireless sensor networks and proves that this problem is NP-complete. Our formulation is compliant with the low-power design of IEEE 802.15.4. We then propose optimal solutions for special cases and heuristic algorithms for general cases. Simulation results show that the proposed algorithms can indeed achieve quick convergecast.

In the fourth work, we show a novel indoor emergency guiding and monitoring system by ZigBee WSN. At normal time, the network is responsible for monitoring the environment in low-power mode. When emergency events are detected, all sensors switch to active mode to deal with these events. And the network can adaptively modify its topology to ensure transportation reliability, quickly identify hazardous regions that should be avoided, and find safe navigation paths that can lead people to exits.

In the last work, we introduce an intelligent light control system, which aims to provide a more convenient and comfortable indoor environment for users. Users are considered to have different requirements when doing different activities. The system can automatically decide illuminations for users by sensors' reports and users' demands. The goal is to satisfy all users and to conserve power. Based on the designed network layer protocols, we can further develop more applications, such as elder health-care application, emergency rescue, river level monitoring, and so on.

**Keywords:** address assignment, convergecast, IEEE 802.15.4, intelligent buildings, light control, long-thin network, orphan problem, navigation, pervasive computing, scheduling, wireless sensor network, ZigBee.

# Acknowledgement

Special thanks goes to my advisor Prof. Yu-Chee Tseng for his guidance in my dissertation work. I would also like to thank my dissertation committee members: Prof. Chin-Liang Wang, Prof. Rong-Hong Jan, Prof. Hsiao-kuang Wu, Prof. Ming-Whei Feng, Prof. Robert K. Lai, and Prof. Wen-Chih Peng. They asked me some good questions and gave me useful comments so that I can improve my work in the future.

Let me also say thank to those HSCC members who co-work with me and all guys I meet in NCTU. Because of you, I can have a great time during these years. Finally, I will dedicate this dissertation to my families and my girl friend, Ms. Wu, for their love and support.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The recent progress of wireless communication and embedded micro-sensing MEMS technologies has made wireless sensor networks (WSNs) more attractive. A lot of research works have been dedicated to WSN, including energy-efficient MAC protocols [29][33][71], routing and transport protocols [23][24][33][39][60], self-organizing schemes [41][62][67], sensor deployment and coverage issues [35][49], and localization schemes [16][18][25][50][55]. In the application side, habitat monitoring is explored in [8], the FireBug project aims to monitor wildfires [6], mobile object tracking is addressed in [20][45][63], and navigation applications are explored in [21][40][44][57].

Recently, many WSN platforms have been developed, such as MICA [11] and Dust Network [3]. For interoperability among different systems, standards such as ZigBee/IEEE 802.15.4 [74][37] protocols have been developed. ZigBee/IEEE 802.15.4 specifies a global standard on physical, MAC, and network layers for WSNs requiring high reliability, low cost, low power, scalability, and low data rate.

In this dissertation, we propose communication protocols and applications based on ZigBee protocol stack. This dissertation is composed of five works. In the first three works, we put our attention on designing ZigBee-compatible network layer protocols. The first work and second work discuss network formation problems in general ZigBee networks and in a special type of ZigBee network, respectively. Some network formation strategies are proposed. Considering that data gathering is a major operation of WSNs, in the third work, we design data collection strategies for ZigBee networks. The proposed solution can indeed achieve low-latency and energy-efficient data collection. Then, based on the above designs, in the last two works, we propose an emergency guiding and monitoring system for indoor surveillance and an intelligent light control system considering user activities. The former system can not only monitor the environment, but also can help safely guide people to a building exit when emergencies happen. And the latter system utilizes sensors' reports to automatically control lighting devices to satisfy users and to conserve power.

In the first work, we discuss network formation issues in general ZigBee networks. According to ZigBee, a device is said to join a network successfully if it can obtain a network address from the coordinator or a router. Before forming a network, the coordinator determines the maximum number of children of a router ($Cm$), the maximum number of child routers of a router ($Rm$), and the depth of the network ($Lm$). Note that a child of a router can be a router or an end device, so $Cm \geq Rm$. ZigBee specifies a distributed address assignment using parameters $Cm$, $Rm$, and $Lm$ to calculate nodes' network addresses. While these parameters facilitate address assignment, they also prohibit a node from joining a network. We say that a node becomes an *orphan node* when it can not associate with the network but there are still unused address spaces remaining. We call this the *orphan problem*. For example, in Fig. 1.1, the router-capable device $A$ has two potential parents $B$ and $C$. Router $B$ can not accept $A$ as its child because it has reached its maximum capacity of $Cm = 5$ children. Router $C$ can not accept $A$ either because it has reached the maximum depth of $Lm = 2$. So $A$ will become an orphan node. Given $Cm$, $Rm$, and $Lm$, we model the orphan problem by two subproblems. The first one considers router-capable devices only. We model this subproblem as a *bounded-degree-and-depth tree formation (BDDTF)* problem, which discusses how to include as many routers as possible into a tree with a bounded degree and depth. We show that this subproblem is NP-complete. After connecting routers, end devices need to be connected to routers. We model this as an *end-device maximum matching (EDMM)* problem. To summarize, we design a two-stage network formation policy to relieve the orphan problem. The first stage is to relieve the BDDTF problem so as to connect as many routers as possible.

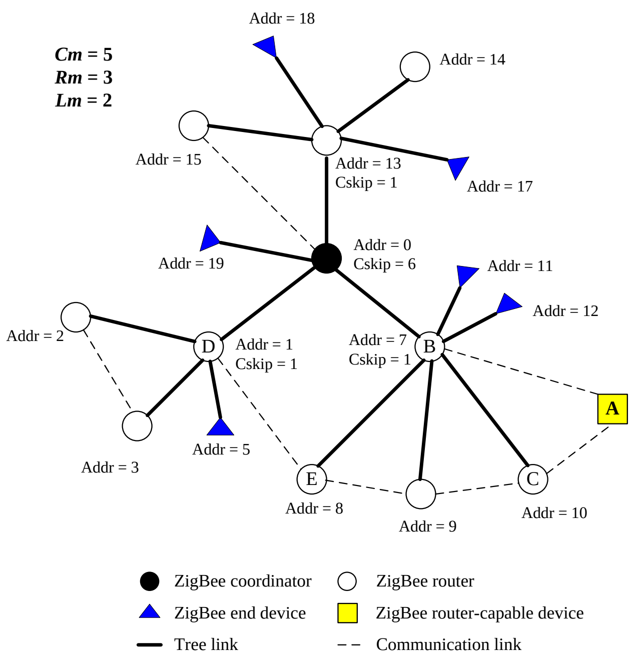


Figure 1.1: An example ZigBee tree network.

And then, based on the result of the first stage, the second stage algorithm, which is designed for the EDMM problem, is used to reduce the number of orphan end devices. For example, the orphan problem in Fig. 1.1 can be relieved if router $E$ is connected to router $D$, so router $B$ has capacity to accept $A$.

Although WSNs have been extensively researched, its deployment is still a big concern. In the second work, we promote a new concept of *long-thin (LT)* topology. The LT architecture is commonly seen in many WSN deployments in many applications, such as gas disclosure detection of fuel pipes, carbon dioxide concentration monitoring in tunnels, and so on. In such a network, nodes may form several long backbones and these backbones are to extend the network to the intended coverage areas. A backbone is a linear path which may contain tens or hundreds of ZigBee routers and may go beyond hundreds or thousands of meters. So the network area can be scaled up easily with limited hardware cost. While the ZigBee address assignment scheme has low complexity, it also prohibits the network from scaling up and thus can not be used in LT networks. In this work, we propose address assignment and routing schemes for ZigBee-based LT WSNs. To assign addresses to nodes, we design rules to divide

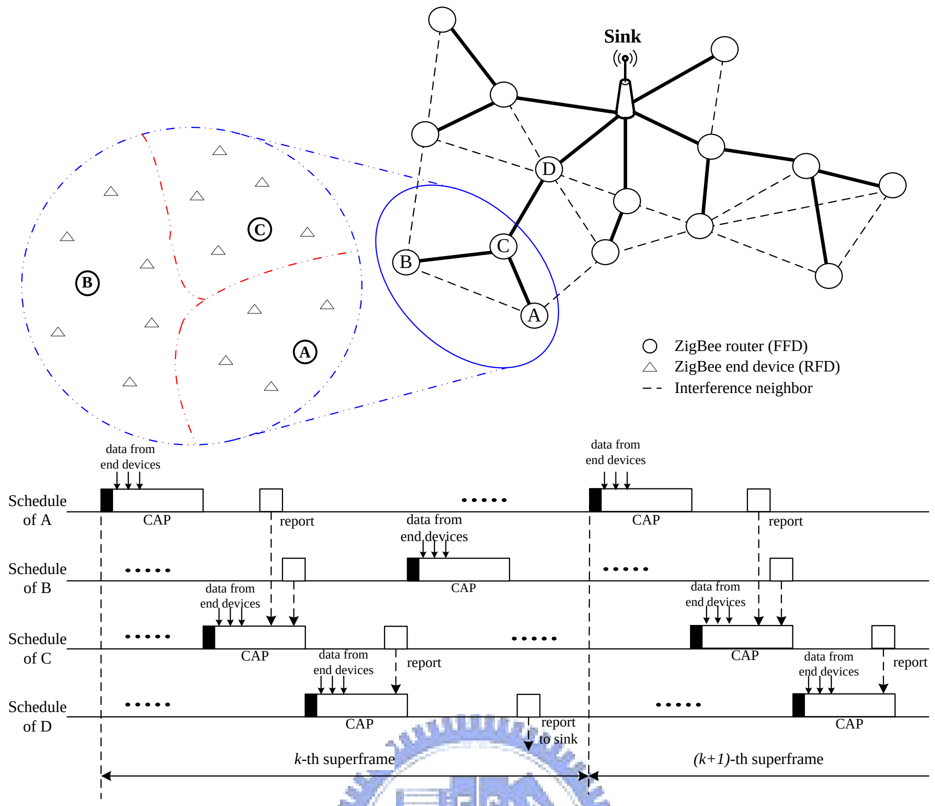


Figure 1.2: An example of convergecast in a ZigBee tree-based network.

nodes into clusters. Each node belongs to one cluster and each cluster has a unique *cluster ID*. All nodes in a cluster have the same cluster ID, but different *node IDs*. The structure of a ZigBee network address is divided into two parts: one is *cluster ID* and the other is *node ID*. Following the same ZigBee design philosophy, the proposed scheme is simple and has low complexity. Existing works [17][52][59][73] have discussed address assignment for WSNs, but they are not designed for ZigBee or LT WSNs. To the best of our knowledge, this is the first work addressing this issue. Moreover, similar to the ZigBee tree routing protocol, the proposed routing protocol can also utilize nodes' network addresses to facilitate routing. In addition, routing can take advantage of shortcuts for better efficiency, so our scheme does not restrict nodes to relay packets only to their parent or child nodes as ZigBee does.

The third work introduces efficient convergecast solutions for WSNs that are compliant with the ZigBee/IEEE 802.15.4 standards. Assuming a tree topology, Fig. 1.2 shows the problem scenario. The network contains one *sink* (ZigBee coordinator), some *full function devices* (ZigBee routers), and some *reduced function devices* (ZigBee end devices). Each ZigBee router is responsible for collecting sensed data from end devices associated with it and relaying incoming data to the sink. According to the ZigBee specification, a ZigBee router can announce a beacon to start a superframe. Each superframe consists of an *active portion* followed by an *inactive portion*. On receiving its parent router's beacon, an end device has also to wake up for an active portion to sense the environment and communicate with its parent device. However, to avoid collision with its neighbors, a router should shift its active portion by a certain amount. Fig. 1.2 shows a possible allocation of active portions for routers A, B, C, and D. The collected sensory data of A in the $k$-th superframe can be sent to C in the $k$-th superframe. However, because the active portion of B in the $k$-th superframe appears after that of C, the collected data of B in the $k$-th superframe can only be relayed to C in the $(k+1)$-th superframe. The delay can be eliminated if the active portion of B in the $k$-th superframe appears before that of C. The delay is not negligible because of the low duty cycle design of IEEE 802.15.4. For example, in 2.4 GHz PHY, with 1.56% duty cycle, a superframe can be up to 251.658 seconds (with an active portion of 3.93 seconds). Clearly, for large-scale WSNs, the convergecast latency could be significant if the problem is not carefully addressed. The quick convergecast problem is to schedule the beacons of routers to minimize the convergecast latency.

In the fourth work, we propose to use a ZigBee WSN in an indoor environment for providing emergency guiding and monitoring services. At normal time, the network is responsible for monitoring the environment. When emergency events are detected, all sensors switch to active mode to deal with these events. And the network can adaptively modify its topology to ensure transportation reliability, quickly identify hazardous regions that should be avoided, and find safe navigation paths that can lead people to exits. Our emergency guiding protocol is distributed, and allows multiple emergency events and multiple exits coexisting in the sensing field. A concept called *hazardous region*, which people should avoid, is introduced. Moreover, we propose a distributed tree reconstruction protocol that can quickly rebuild the

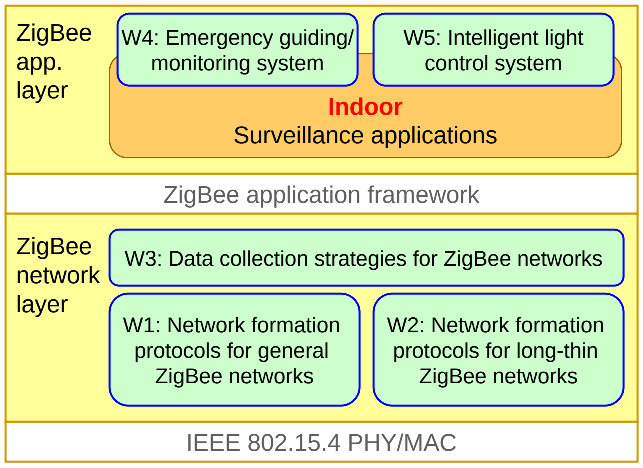

ZigBee app. layer
W4: Emergency guiding/ monitoring system
W5: Intelligent light control system
Indoor
Surveillance applications
ZigBee application framework
ZigBee network layer
W3: Data collection strategies for ZigBee networks
W1: Network formation protocols for general ZigBee networks
W2: Network formation protocols for long-thin ZigBee networks
IEEE 802.15.4 PHY/MAC

Figure 1.3: The relationship between the proposed works and the ZigBee stack.

reporting tree at low communication cost when emergency. Our design emphasizes on local recovery and stability. We will address how to conquer the unstable radio link problem that is frequently seen in short-distance wireless systems, like the ZigBee. Prototyping and simulation results show that our protocols can react to emergencies quickly at low message cost and can find safe paths to exits.

In the last work, we propose an intelligent light control system for indoor environments using ZigBee WSNs. Wireless sensors are responsible for reporting current illuminations to a control host. Two kinds of lighting devices, namely *whole lighting* and *local lighting* devices, are used to provide background and concentrated illuminations, respectively. Users may have various illumination requirements according to their activities. An illumination requirement is as the combination of background and concentrated illumination demands and users' locations. We propose a decision algorithm to determine the proper illuminations of devices to satisfy users. Then a closed-loop device control algorithm is applied to adjust the illumination levels of lighting devices. Prototyping and simulation results verify that our ideas are practical and feasible.

The proposed five works can be compliant to the ZigBee standard. Fig. 1.3 shows the relationship between the proposed works and the ZigBee stack. Based on the designed network layer protocols, we can further develop some outdoor surveillance applications, such as stage measurements in sewers, vibration detection of bridges, and so on.

This dissertation is organized as follows. ZigBee/IEEE 802.15.4 standards are surveyed in Chapter 2. Chapter 3 and Chapter 4 presents network formation problems in general and long-thin ZigBee networks, respectively. In Chapter 5, we discuss the convergecast issues in ZigBee networks. Chapter 6 and Chapter 7 present the proposed emergency guiding and monitoring system and intelligent light control system by ZigBee WSNs, respectively. Finally, we conclude our results and propose some future directions in Chapter 8.

# Chapter 2

# Overview of IEEE 802.15.4 and ZigBee Standards

ZigBee/IEEE 802.15.4 is a global hardware and software standard designed for WSN requiring high reliability, low cost, low power, scalability, and low data rate. Table 2.1 compares ZigBee/IEEE 802.15.4 against several other wireless technologies. The ZigBee alliance [15] is to work on the interoperability issues of ZigBee/IEEE 802.15.4 protocol stacks. The IEEE 802.15 WPAN Task Group 4 [37] specifies physical and data link layer protocols for ZigBee/IEEE 802.15.4. The relationship of ZigBee and IEEE 802.15.4 is shown in Fig. 2.1. In the current development, IEEE 802.15 WPAN working group creates two task groups 15.4a and 15.4b. The former is to specify an alternate physical layer, the ultra wide band (UWB) technologies. The latter is to enhance the IEEE 802.15.4 MAC protocol so that it can tightly couple with the network layer functionalities specified by ZigBee. ZigBee alliance published the version 2.0 standard in Dec. 2006 [74].

Companies such as Chipcon [1], Ember [5], and Freescale [7] provide system-on-chip so-

Table 2.1: Comparison of different wireless technologies [15].

| Standard | ZigBee/IEEE 802.15.4 | Bluetooth | UWB | IEEE 802.11 b/g |
|---|---|---|---|---|
| Working frequency | 868/915 MHz, 2.4GHz | 2.4 GHz | 3.1 - 10.6 GHz | 2.4 GHz |
| Range (m) | 30 – 75+ | 10 – 30 | ~10 | 30 – 100 + |
| Data rate | 20/40/250 kbps | 1 Mbps | 100+ Mbps | 2 – 54 Mbps |
| Devices | 255 – 65k | 8 | | 50 – 200 |
| Power consumption | ~1 mW | ~40 – 100 mW | ~80 – 300 mW | ~160 mW – 600W |
| Cost ($US) | ~2 – 5 | ~4 – 5 | ~5 – 10 | ~20 – 50 |

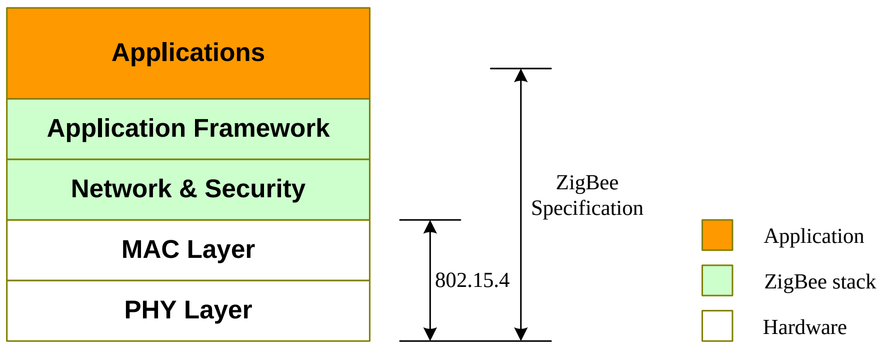


Figure 2.1: The ZigBee/IEEE 802.15.4 protocol stack.

lutions of ZigBee/IEEE 802.15.4. For home networking, ZigBee/IEEE 802.15.4 can be used for light control, heating ventilation air conditioning (HVAC), security monitoring, and emergency event detection. For health case, ZigBee/IEEE 802.15.4 can integrate with sphygmomanometers or electronic thermometers to monitor patients' statuses. For industrial control, ZigBee/IEEE 802.15.4 devices can be used to improve the current manufacturing control systems, detect unstable situations, control production pipelines, and so on.

## 2.1 IEEE 802.15.4 Basics

IEEE 802.15.4 specifies the physical layer and data link layer protocols for low-rate wireless personal area networks (LR-WPAN), which emphasize on simple, low-cost applications. Devices in such networks normally have less communication capabilities and limited power, but are expected to operate for a longer period of time. As a result, energy-saving is a critical design issue. In IEEE 802.15.4, there are two basic types of network topologies, the star topology and the peer-to-peer topology. Devices in a LR-WPAN and can be classified as *full function devices (FFDs)* and *reduced function devices (RFDs)*. One device is designated as the *PAN coordinator*, which is responsible for maintaining the network and managing other devices. A FFD has the capability of becoming a PAN coordinator or associating with an existing PAN coordinator. A RFD can only send or receive data from a PAN coordinator that it associates with. Each device in IEEE 802.15.4 has a unique 64-bit *long address*. After associating to a coordinator, a device will be assigned a 16-bit *short address*. Then packet

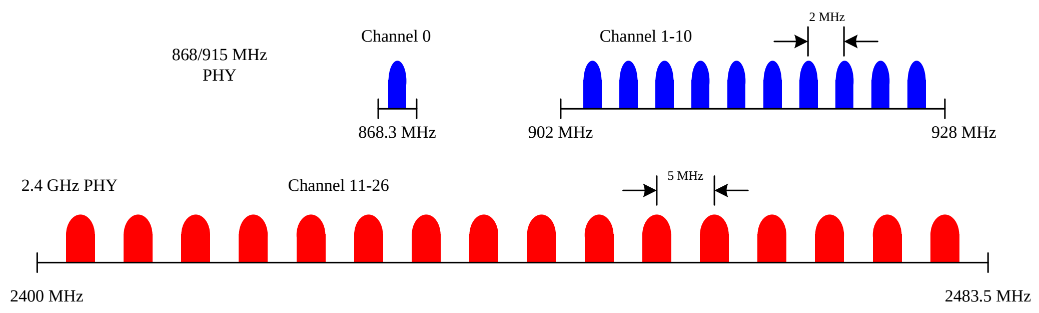


Figure 2.2: Arrangement of channels in IEEE 802.15.4.

exchanges between the coordinator and devices will use short addresses. In the following, the IEEE 802.15.4 physical layer and data link layer protocols are introduced.

### 2.1.1 Physical Layer (PHY)

In IEEE 802.15.4 PHY, there are three operating frequency bands with 27 radio channels. These bands are 868 MHz, 915 MHz, and 2.4 GHz. The channel arrangement is shown in Fig. 2.2 Channel 0 is in the frequency 868.0 to 868.6 MHz, which provides a data rate of 20 kbps. Channels 1 to 10 work in frequency 902.0 to 928.0 MHz and each channel provides a data rate of 40 kbps. Channels 11 to 26 are located in frequency 2.4 to 2.4835 GHz and each channel provides a data rate of 250 kbps.

Channels 0 to 10 use the binary phase shift keying (BPSK) as their modulation scheme, and channels 11 to 26 use the offset quadrature phase shift keying (O-QPSK) as their modulation scheme. The required receiver sensitivity should be larger than -92 dBm for channels 0 to 10, and larger than -85 dBm for channels 11 to 26. The transmit power should be at least -3 dBm (0.5 mW). The transmission radius may range from 10 meters to 75 meters. Targeting at low-rate communication systems, in IEEE 802.15.4, the payload length of a PHY packet is limited to 127 bytes.

### 2.1.2 Data Link Layer

In all IEEE 802 specifications, the data link layer is divided into two sublayers: *logical link control (LLC)* sublayer and *medium access control (MAC)* sublayer. The LLC sublayer in

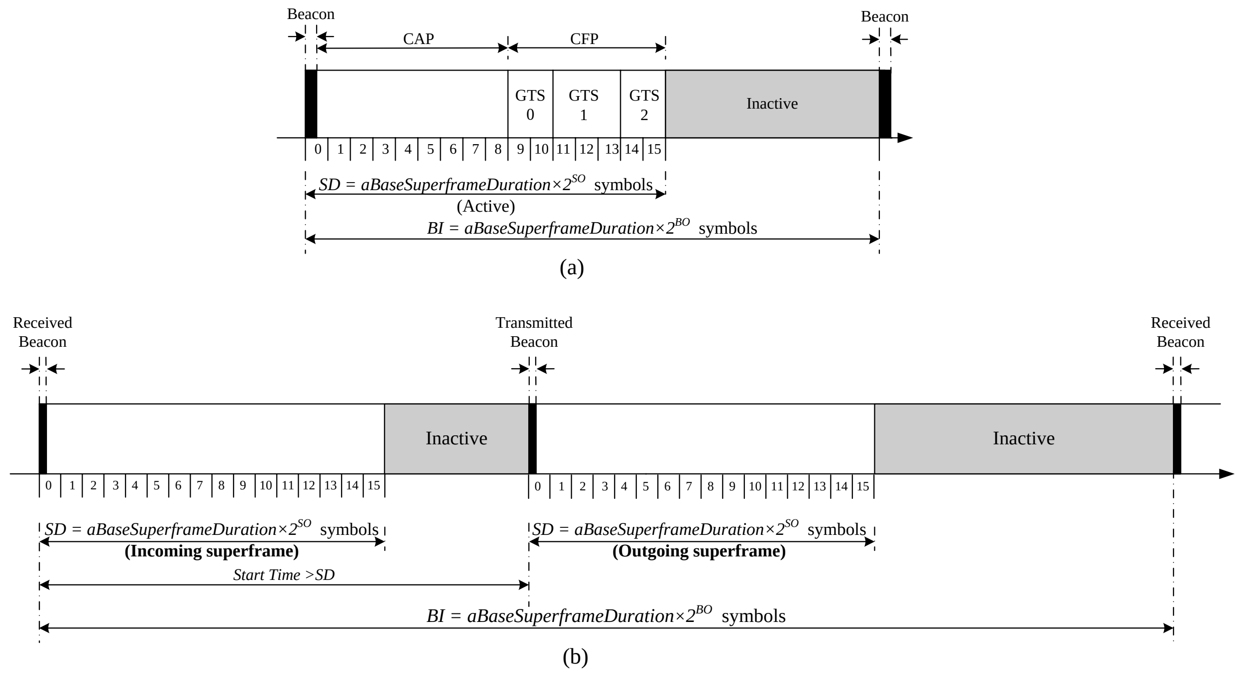


Figure 2.3: IEEE 802.15.4 superframe structure.

IEEE 802.15.4 follows the IEEE 802.2 standard. The MAC sublayer manages superframes, controls channel access, validates frames, and sends acknowledgements. The IEEE 802.15.4 MAC sublayer also supports low power operations and security mechanisms. In the following subsections, we introduce the MAC layer protocols in IEEE 802.15.4.

### Superframe Structure

In IEEE 802.15.4, the superframe structure of a network is defined by its coordinator. The length of a superframe is equal to the time interval of two adjacent beacons sent by a coordinator. A superframe can be divided into an active portion and an inactive portion. An active portion consists of 16 equal-length slots and can be further partitioned into a *contention access period (CAP)* and a *contention free period (CFP)*. The CAP may contain $i$ slots, $i = 1, 2, ..., 16$, and the CFP, which follows the CAP, may contain $16 - i$ slots. The coordinator and network devices can exchange packets during the active portion and go to sleep during the inactive portion. The superframe structure is shown in Fig. 2.3(a).

Beacons are used for starting superframes, synchronizing with other devices, announcing the existence of a PAN, and informing pending data in coordinators. In a beacon-enabled network, devices use the slotted CAMA/CA mechanism to contend for the usage of channels. FFDs which require fixed rates of transmissions can ask for *guarantee time slots (GTS)* from the coordinator. A CFP can include multiple GTSs, and each GTS may contain multiple slots. For example, in Fig. 2.3(a), GTS 0 and GTS 2 use two slots and GTS 1 uses three slots. A coordinator can allocate at most seven GTSs for network devices.

In IEEE 802.15.4, the structure of superframes is controlled by two parameters: *beacon order (BO)* and *superframe order (SO)*, which decide the length of a superframe and its active potion, respectively. For a beacon-enabled network, the setting of BO and SO should satisfy the relationship $0 \leq SO \leq BO \leq 14$. For channels 11 to 26, the length of a superframe can range from 15.36 $ms$ to 215.7 $s$, so can an active potion. Specifically, the length of a superframe is

$$BI = aBaseSuperframeDuration \times 2^{BO}\textit{symbols}$$

, where each symbol is $1/62.5\ ms$ and $aBaseSuperframDuration = 960$ symbols. Note that the length of a symbol is different for channels 0 to 10. The length of each active portion is

$$SD = aBaseSuperframeDuration \times 2^{SO}\textit{symbols}$$

Therefore, each device will be active for $2^{-(BO-SO)}$ portion of the time, and sleep for $1 - 2^{-(BO-SO)}$ portion of the time. Changing the value of $(BO - SO)$ allows us to adjust the on-duty time of devices. However, for a beacon-enabled tree network, routers have to choose different times to start their active portions to avoid collision. Once the value of $(BO-SO)$ is decided, each router can choose from $2^{BO-SO}$ slots as its active portion. In the revised version of IEEE 802.15.4 [38], a router can select one active portion as its *outgoing superframe*, and based on the active portion selected by its parent, the active portion is called its *incoming superframe* (as shown in Fig. 2.3(b)). In an outgoing/incoming superframe, a router is expected

Table 2.2: Relationship of $BO - SO$, duty cycle, and the number of active portions in a superframe.

| $BO - SO$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | $\geq 9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Duty cycle (%) | 100 | 50 | 25 | 12.5 | 6.25 | 3.13 | 1.56 | 0.78 | 0.39 | $\leq 0.195$ |
| Number of active portions (slots) | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | $\geq 512$ |

to transmit/receive a beacon to/from its child routers/parent router. When choosing a slot, neighboring routers' active portions (i.e., outgoing superframes) should be shifted away from each other to avoid interference. Table 2.2 lists possible choices of $(BO - SO)$ combinations.

**CSMA/CA Mechanisms**

There are two channel access mechanisms in IEEE 802.15.4. One is unslotted CSMA/CA and the other is slotted CSMA/CA. The operations of unslotted CSMA/CA are similar to the ones in IEEE 802.11 CSMA/CA. A device that has a data or command frame to send will randomly backoff a period of time. If the medium is idle when the backoff expires, this device can transmit its frame. On the other hand, if the medium is busy, this device will increase its backoff window and waits for another period of time.

The slotted CSMA/CA works differently from unslotted CSMA/CA. In the slotted CSMA/CA mechanism, the superframe structure is needed. A superframe can be further divided into smaller slots called backoff periods, each of length 20 symbols[1]. The start of the first backoff period in a superframe is aligned to the start of beacon transmission. Before transmission, a device first calculates a random number of backoff periods. After timeout, the device should perform *clear channel assessment (CCA)* twice in the upcoming two backoff periods. If the channel is found to be clear in two CCAs, the device can start to transmit a frame to the coordinator. If the channel is found to be busy in any of the two CCAs, the device should double its contention window and perform another random backoff. Fig. 2.4 shows the procedures of the slotted CSMA/CA mechanism in IEEE 802.15.4.

[1]The time required to transmit a symbol varies according to working bands of PHY. For example, in the 2.4 GHz band, the length of a symbol is $16us$; hence, in the 2.4 GHz band, a unit backoff period is $320us$.

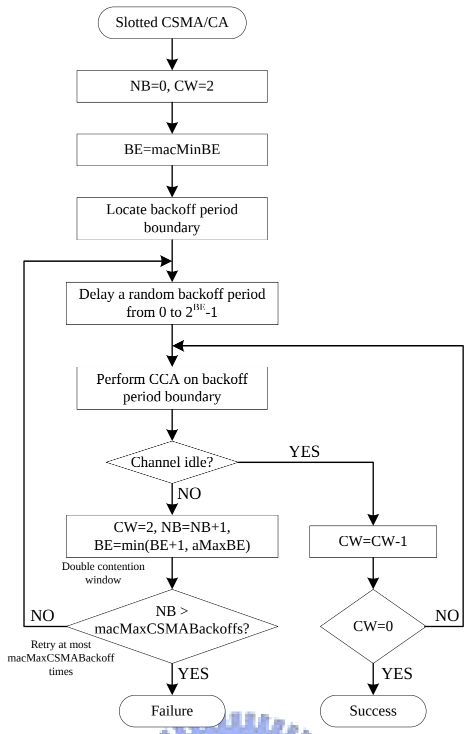

Slotted CSMA/CA
NB=0, CW=2
BE=macMinBE
Locate backoff period boundary
Delay a random backoff period from 0 to 2^BE-1
Perform CCA on backoff period boundary
Channel idle?
YES
NO
CW=2, NB=NB+1, BE=min(BE+1, aMaxBE)
Double contention window
CW=CW-1
NB > macMaxCSMABackoffs?
NO
Retry at most macMaxCSMABackoff times
YES
CW=0
NO
YES
Failure
Success

Figure 2.4: The basic slotted CSMA/CA mechanism in IEEE 802.15.4.

## Association and Disassociation Procedures

A device becomes a member of a PAN by associating with its coordinator. At the beginning, a device should scan channels to find potential coordinators. After choosing a coordinator, the device should locate the coordinator's beacons and transmit an association request command to the coordinator. In a beacon-enabled network, the association request is sent in the CAP of a superframe. In a non-beacon-enabled network, the request is sent by the unslotted CSMA/CA mechanism. On receipt of the association request, the coordinator will reply an ACK. Note that correctly receiving an ACK does not mean that device has successfully associated to the coordinator; the device still has to wait for an association decision from the coordinator. The coordinator will check its resource to determine whether to accept this association request or not. In IEEE 802.15.4, association results are announced in an indirect fashion. A coordinator responds to association requests by appending devices' long addresses in beacon frames to indicate that the association results are available. If a device finds that its address is appended

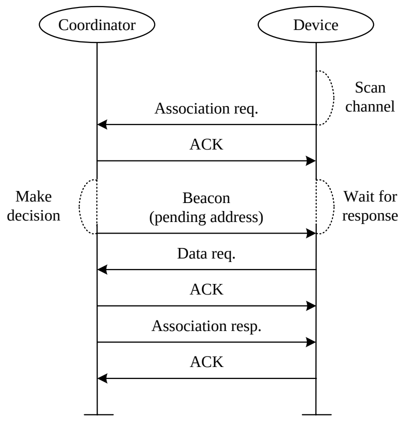


Figure 2.5: The association procedure in IEEE 802.15.4.

in a beacon, it will send a data request to the coordinator to acquire the association result. Then the coordinator can transmit the association result to the device. The association procedure is summarized in Fig. 2.5.

When a coordinator would like an associated device to leave its PAN, it can send a disassociation notification command to the device. After receiving this command, the device will reply an ACK. If the ACK is not correctly received, the coordinator will still consider that the device has been disassociated. When an associated device wants to leave a PAN, it also sends a disassociation notification command to the coordinator. On receipt of the command, the coordinator will reply an ACK and remove the records of the correspond device. Similar to the above case, the device considers itself disassociated even if it does not receive an ACK from the coordinator.

### 2.1.3 Summary of IEEE 802.15.4

IEEE 802.15.4 specifies the physical layer and data link layer protocol for low-rate wireless personal area networks. However, this specification only concerns communications between devices that are within each other's transmission range. For larger sensor networks, the support of network layer protocols is needed. In the next section, we will introduce a developing standard, ZigBee, which supports protocols above the data link layer for connecting IEEE 802.15.4 devices together.

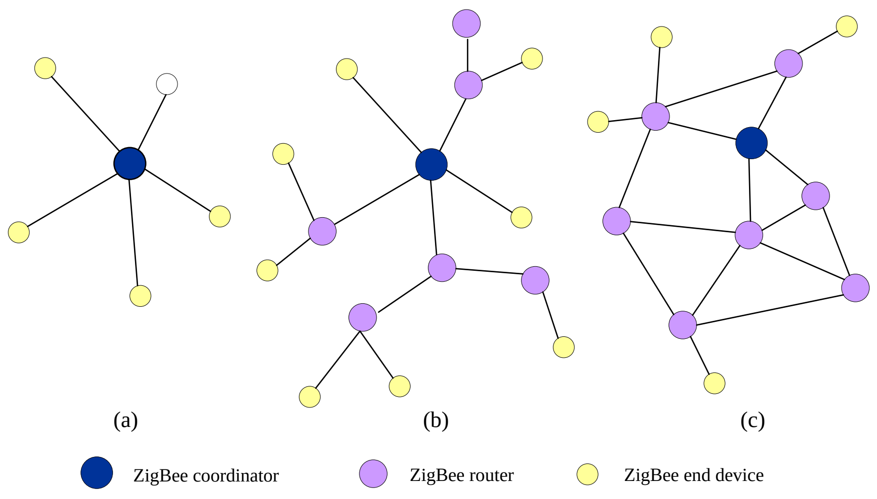


Figure 2.6: Zigbee network topologies: (a) star, (b) tree, and (c) mesh.

## 2.2 ZigBee Network Layer

In ZigBee, the network layer provides reliable and secure transmissions among devices. ZigBee supports three kinds of networks, namely *star*, *tree*, and *mesh* networks. A *ZigBee coordinator* is responsible for initializing, maintaining, and controlling the network. A star network has a coordinator with devices directly connecting to the coordinator. For tree and mesh networks, devices can communicate with each other in a multihop fashion. The network is formed by one ZigBee coordinator and multiple *ZigBee routers*. A device can join a network as an *end device* by the associating with the coordinator or a router. In a tree network, the coordinator and routers can announce beacons. However, in a mesh network, regular beacons are not allowed. Beacons are an important mechanism to support power management. Therefore, the tree topology is preferred, especially when energy saving is a desirable feature. Devices in a mesh network can only communicate with each other by peer-to-peer transmissions specified in IEEE 802.15.4. Some example of ZigBee network topologies are shown in Fig. 2.6.

### 2.2.1 Network Formation

Devices that are coordinator-capable and do not currently join a network can be candidates of ZigBee coordinators. A device that desires to be a coordinator will scan all channels to find a suitable one. After selecting a channel, this device broadcasts a beacon containing a PAN identifier to initialize a PAN. A device that hears beacons of an existing network can join this network by performing the association procedures and specifying its role, as a ZigBee router or as an end device. Note that if there are multiple beacons, the device will choose the sender that is located closer to the sink. When a beacon sender receives a request, it will determine whether to accept the request sender or not by considering its current capacity and its permitted association duration. Then the association response can be carried by its beacons. If a device is successfully associated, the association response will contain a short 16-bit address for the request sender. This short address will be the network address for that device.

### 2.2.2 Address Assignment in a ZigBee Network

In a ZigBee network, network addresses are assigned to devices by a distributed address assignment scheme. After forming a network, the ZigBee coordinator determines the maximum number of children ($Cm$) of a ZigBee router, the maximum number of child routers ($Rm$) of a parent node, and the depth of the network ($Lm$). Note that $Cm \geq Rm$ and a parent can have $(Cm - Rm)$ end devices as its children. In this algorithm, addresses of devices are assigned by their parents. For the coordinator, the whole address space is logically partitioned into $Rm + 1$ blocks. The first $Rm$ blocks are to be assigned to the coordinator's child routers and the last block is reserved for the coordinator's own child end devices. In this scheme, a parent device utilizes $Cm$, $Rm$, and $Lm$ to compute a parameter called $C_{skip}$, which is used to compute the starting addresses of its children's address pools. The $C_{skip}$ for the ZigBee coordinator or a router in depth $d$ is defined as:

$$Cskip(d) = \begin{cases} 1 + Cm \times (Lm - d - 1), & \text{if } Rm = 1. \text{ (a)} \\ \dfrac{1 + Cm - Rm - Cm \cdot Rm^{Lm-d-1}}{1 - Rm}, & \text{otherwise. (b)} \end{cases} \tag{2.1}$$

The coordinator is said to be at depth 0; a node which is a child of another node at depth $d$ is said to be at depth $d + 1$. Consider any node $x$ at depth $d$, and any node $y$ which is a child of $x$. The value of $C_{skip}(d)$ indicates the maximum number of nodes in the subtree rooted at $y$ (including $y$ itself). For example, in Fig. 2.7, since the $C_{skip}$ value of B is 1, the subtree of C will contain no more than 1 node; since the $C_{skip}$ value A is 7, the subtree of B will contain no more than 7 nodes. To understand the formulation, consider again the nodes $x$ and $y$ mentioned above. Node $y$ itself counts for one node. There are at most $Cm$ children of $y$. Among all children of $y$, there are at most $Rm$ routers. So there are at most $Cm \cdot Rm$ grandchildren of $y$. It is not hard to see that there are at most $Cm \cdot Rm^2$ great grandchildren of $y$. So the size of the subtree rooted at $y$ is bounded by

$$C_{skip}(d) = 1 + Cm + CmRm + CmRm^2 + ... + CmRm^{Lm-d-2}, \tag{2.2}$$

since the depth of the subtree is at most $Lm - d - 1$. We can derive that

$$\begin{aligned} Eq.\ (2.2) &= 1 + Cm(1 + Rm + Rm^2 + ... + Rm^{Lm-d-2}) \\ &= 1 + Cm(1 - Rm^{Lm-d-1})/(1 - Rm) = Eq.\ (2.1)(b) \end{aligned} \tag{2.3}$$

Address assignment begins from the ZigBee coordinator by assigning address $0$ to itself and depth $d = 0$. If a parent node at depth d has an address $A_{parent}$, the $n$-th child router is assigned to address $A_{parent} + (n - 1) \times C_{skip}(d) + 1$ and $n$-th child end device is assigned to address $A_{parent} + Rm \times C_{skip}(d) + n$. An example of the ZigBee address assignment is shown in Fig. 2.7. The $C_{skip}$ of the ZigBee coordinator is obtained from Eq. (2.1) by setting $d = 0$, $Cm = 6$, $Rm = 4$, and $Lm = 3$. Then the first, second, and third child routers of the coordinator will be assigned to addresses $0 + (1 - 1) \times 31 + 1 = 1$, $0 + (2 - 1) \times 31 + 1 = 32$, and $0 + (3 - 1) \times 31 + 1 = 63$, respectively. And the two child end devices' addresses are $0 + 4 \times 31 + 1 = 125$ and $0 + 4 \times 31 + 2 = 126$.

## 2.2.3 Routing Protocols

In a ZigBee network, the coordinator and routers can directly transmit packets along the tree without using any route discovery. When a router receives a packet, it first checks if it is the destination or one of its child end devices is the destination. If so, this router will accept the packet or forward this packet to the designated child end device. Otherwise, it will relay packet along the tree. Assume that the depth of this router is $d$ and its address is $A$. This packet

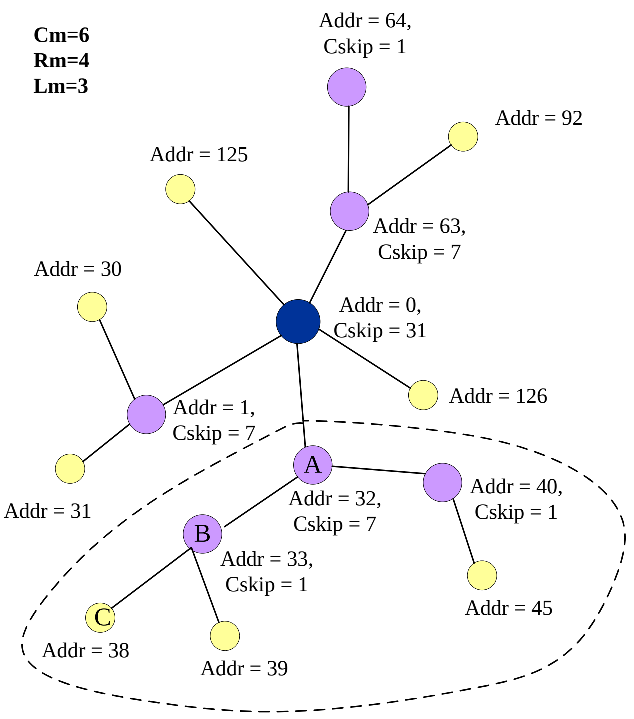


Figure 2.7: An address assignment example in a ZigBee network.

is for one of its descendant devices if the destination address $A_{dest}$ satisfies $A < A_{dest} < A + Cskip(d-1)$, and this packet will be relayed to the child router with address

$$A_r = A + 1 + \left\lfloor \frac{A_{dest} - (A+1)}{Cskip(d)} \right\rfloor \times Cskip(d).$$

If the destination is not a descendant of this device, this packet will be forwarded to its parent.

In a mesh network, ZigBee coordinators and routers are said to have *routing capacity* if they have routing table capacities and route discovery table capacities. Devices that are routing-capable can initiate routing discovery procedures and directly transmit packets to relay nodes. Otherwise, they can only transmit packets through tree links. In the latter case, when receiving a packet, a device will perform the same routing operations as described in tree networks. When a node needs to relay a received packet, it will first check whether it is routing-capable. If it is routing-capable, the packet will be unicast to the next hop. Otherwise, the packet will be relayed along the tree.

A device that has routing capacity will initiate route discovery if there is no proper route entry to the requested destination in its routing table. The route discovery in a ZigBee network is similar to the AODV routing protocol [56] . Links with lower cost will be chosen into the routing path. The cost of link l is defined based on the packet delivery probability on link l. However, how to calculate the packet delivery probability is not explicitly stated in the ZigBee specification.

At the beginning of a route discovery, the source broadcasts a route request packet. A ZigBee router that receives a route request packet first computes the link cost. If this device has routing capacity, it will rebroadcast this request if it does not receive this request before or the link cost recorded in route request plus the cost it just computed is lower than the former received request. Otherwise, it will discard this request. For the case that a ZigBee router that is not routing capable receives a route request, it also determines whether to resend this request based on the same comparison. If this device determines to resend this route request, it will check the destination address and unicast this route request to its parent or to one of its children (in the tree network). An example is shown in Fig. 2.8. In Fig. 2.8, device S broadcasts a route request for destination T and devices A and D receive this packet. Since device A has no routing capacity, it will check the address of destination T and unicast this request to device C. Since device D has routing capacity, it will rebroadcast this request. A device that has resent a route request packet will record the request sender in its route discovery table. This information will be discarded if this device does not receive a route reply within a time interval.

When the destination receives route request packets from multiple paths, it will choose the routing path with the lowest cost and send a route reply packet to the source. The route reply packet will be sent by unicast. An intermediate node that receives the route reply packet checks its route discovery table and sends the route reply to the request sender. After the source node successfully receives the route reply, it can send data packets to the destination node along the discovered route.

The ZigBee network layer also specifies route maintenance mechanisms for mesh and tree networks. In a mesh network, route failure is detected by a failure counter. If the counter of a

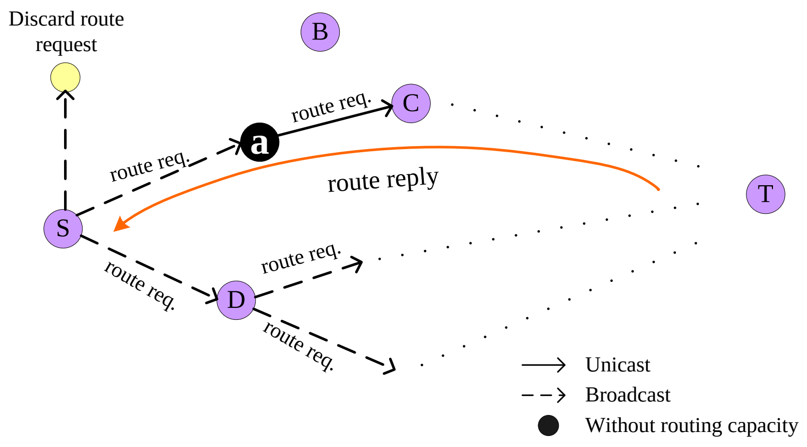


Figure 2.8: An example of route request dissemination in a ZigBee network.

ZigBee router exceeds a threshold, the router can start the route maintenance procedure. For those routers that have routing capacity, they can flood route request packets to find destinations. For routers that do not have routing capacity, they will unicast route request packets to their parents or children according to the destination addresses. However, in a tree network, a router does not broadcast route request packets when it loses its parent. Instead, it disassociates with its parent and tries to re-associate with a new parent. After re-association, it will receive a new short 16-bit network address and can transmit packets to its new parent. Note that a device that re-associates to a new parent will disconnect all its children. Those children that lose their parents will also try to find new parents. On the other hand, when a router cannot send packets to a child, it will directly drop this packet and send a route error message to the packet originator. Then this router will send a disassociation notification command to the child. The disassociated child may reconnect to the same parent or find a new parent depending on its new scan result.

### 2.2.4 Summary of the ZigBee Network Layer

ZigBee is designed to support low-cost network layer. It supports three kinds of network topologies, which are star, tree, and mesh networks. Network developers can choose a suitable network topology for their applications. The pros and cons of these three topologies are

Table 2.3: Pros and cons of different kinds of ZigBee network topologies.

| | Pros | Cons |
|---|---|---|
| Star | 1. Easy to synchronize<br>2. Support low power operation<br>3. Low latency | 1. Small scale |
| Tree | 1. Low routing cost<br>2. Can form superframes to support sleep mode<br>3. Allow multihop communication | 1. Route reconstruction is costly<br>2. Latency may be quite long |
| Mesh | 1. Robust multihop communication<br>2. Network is more flexible<br>3. Lower latency | 1. Cannot form superframes (and thus cannot support sleep mode)<br>2. Route discovery is costly<br>3. Needs storage for routing table |

summarized in Table 2.3.

# Chapter 3

# Network Formation Protocols for General ZigBee Networks

## 3.1 Observations and Motivations

In this work, we propose network formation protocols for general ZigBee networks. By the ZigBee network formation rules, some devices may not be able to join the network even if there are remaining address spaces. Fig. 1.1 is a small-scale example. Here, we present a large-scale simulation result in a circular field of a radius $200\ m$ with a coordinator at the center. There are $800$ router-capable devices randomly deployed in the field. The transmission range of nodes is $35\ m$. We set $Cm = Rm = 3$ and $Lm = 7$, which implies that this network can accommodate up to $3280$ routers. Our simulation result shows that, in average, more than $25\%$ of devices (about $207.45$ devices) will become orphan nodes. Fig. 3.1 shows one simulation scenario, where many devices near the network boundary can not join the network. We see that some devices near the center do not have any child, which means that the address spaces are underutilized. In fact, assuming $Cm = Rm$, a router at depth $d$ serving as a leaf implies a loss of $\frac{1-Rm^{Lm-d+1}}{1-Rm}$ address spaces. Therefore, maintaining sufficient children for nodes near the coordinator is critical.

There could be a misconception that the orphan problem can be trivially solved by enlarging $Cm$, $Rm$, or $Lm$. In practice, devices' capabilities and application demands should be carefully deliberated before doing so. Larger $Cm$ or $Rm$ imposes more memory space

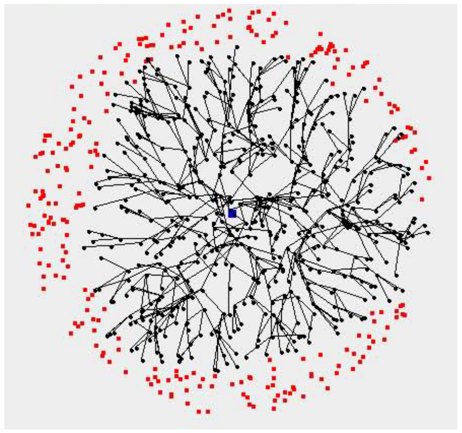


Figure 3.1: A ZigBee network formation example. Isolated dots are orphan nodes.

requirement on routers. A larger $Lm$ may induce longer network delay. Also, enlarging these values incurs longer address space (ZigBee specifies a 16-bit address space). Besides, in theory, it can not be guaranteed that there are no orphan devices with any given $Cm$, $Rm$, and $Lm$ (this will be shown in Section 3.2). Therefore, orphans are an inherent problem given the $Cm$, $Rm$, and $Lm$ constraints. Our simulation results show that proper network formation strategy can effectively reduce the number of orphan devices without enlarging $Cm$, $Rm$, or $Lm$. To the best of our knowledge, this is the first work that discusses the orphan problem in ZigBee-based WSNs.

Several works have investigated the *bounded-degree spanning tree* problem. Reference [28] proposes polynomial-time solutions when additional connectivity and maximum degree of a graph are given. However, the depth constraint is not considered. Reference [43] introduces an approximation algorithm, which can find a spanning tree with a maximum degree of $O(K + log|V|)$, where $K$ is the degree constraint and $V$ is the set of nodes in the graph. The result is not applicable to our case because it does not consider the depth constraint and the number of children of a node is not bounded. In [42], a polynomial time algorithm is proposed to construct a spanning tree with a bounded degree and diameter. However, this algorithm is designed for complete graphs, which is not the case in a ZigBee network.

## 3.2 The Orphan Problem

Given a sensor network, we divide the orphan problem by two subproblems. In the first problem, we consider only router-capable devices and model the network by a graph $G_r = (V_r, E_r)$, where $V_r$ consists of all router-capable devices and the coordinator $t$ and $E_r$ contains all symmetric communication links between nodes in $V_r$. We are given parameters $Cm$, $Rm$, and $Lm$ such that $Cm \geq Rm$. The goal is to assign parent-child relationships to nodes such that as many vertices in $V_r$ can join the network as possible. Below, we translate this problem to a tree formation problem.

**Definition 1** *Given $G_r = (V_r, E_r)$, $Rm$, $Lm$, and an integer $N \leq |V_r|$, the* Bounded-Degree-and-Depth Tree Formation (BDDTF) *problem is to construct a tree $T$ rooted at $t$ from $G_r$ such that $T$ satisfies the ZigBee tree definition and $T$ contains at least $N$ nodes.*

In [32], it is shown that the *Degree-Constrained Spanning Tree (DCST)* as defined below is NP-complete.

**Definition 2** *Given $G = (V, E)$ and a positive integer $K \leq |V|$, the* Degree-Constrained Spanning Tree (DCST) *problem is to find a spanning tree $T$ from $G$ such that no vertex in $T$ has a degree larger than $K$.*

**Theorem 1** *The BDDTF problem is NP-complete.*

**Proof**. 1) Given a tree $T$ in $G_r$, we can check if $T$ satisfies the constraints of $Rm$ and $Lm$ and if $T$ contains more than $N$ nodes in polynomial time. 2) The DCST problem can be reduced to a special case of the BDDTF problem when $Rm = K$, $Lm \rightarrow \infty$, and $N = |V_r|$. $\square$

In the second subproblem, we will connect non-router-capable devices to the tree $T$ constructed earlier following the ZigBee definition such that as many end devices are connected to $T$ as possible. Toward this goal, we model the sensor network by a bipartite graph $G_d = (\{\hat{V}_r \cup V_e\}, E_d)$, where $\hat{V}_r$ consists of the routers, excluding the ones at depth $Lm$, in $T$ formed in the first subproblem, $V_e$ consists of all end devices, and $E_d$ contains all symmetric communication links between $\hat{V}_r$ and $V_e$. Each vertex $v \in \hat{V}_r$ can accept at most $C_v \geq (Cm - Rm)$ end devices. From $G_d$, we construct another bipartite graph $\tilde{G}_d = (\{\tilde{V}_r \cup \tilde{V}_e\}, \tilde{E}_d)$ as follows.

1. From each vertex $v \in \hat{V}_r$, generate $C_v$ vertices $v_1$, $v_2$, ..., $v_{C_v}$ in $\tilde{V}_r$.
2. From each vertex $u \in V_e$, generate a vertex $u$ in $\tilde{V}_e$.
3. From each edge ($v$, $u$) in $E_d$, where $v \in \hat{V}_r$ and $u \in V_e$, connect each of the $C_v$ vertices $v_1$, $v_2$, ..., $v_{C_v}$ generated in rule 1 with the vertex $u$ generated in rule 2. These edges form the set $\tilde{E}_d$.

It is clear that $\tilde{G}_d$ is a bipartite graph with edges connecting vertices in $\tilde{V}_r$ and vertices in $\tilde{V}_e$ only. Intuitively, we duplicate each $v \in \hat{V}_r$ by $C_v$ vertices, and each edge $(v, u) \in E_d$ into $C_v$ edges. Since each vertex in $\tilde{V}_r$ is connected to at most one vertex in $\tilde{V}_e$, this translates the problem to a maximum matching problem as follows.

**Definition 3** *Given a graph* $\tilde{G}_d = (\{\tilde{V}_r \cup \tilde{V}_e\}, \tilde{E}_d)$*, the* End-Device Maximum Matching (EDMM) *problem is to find a maximum matching of* $\tilde{G}_d$*.*

Theorem 1 implies that the first subproblem is intractable. On the contrary, the maximum matching problem in Definition 3 is solvable in polynomial time. Below, we will propose solutions to these problems.

## 3.3 Algorithms for the BDDTF Problem

We propose two algorithms to reduce orphan routers in a ZigBee network. In our algorithms, we will repeatedly generate several BFS trees from $G_r$. For each tree, we may decide to truncate some nodes if the tree is not conformed to the ZigBee definition. The truncation is done based on nodes' *association priorities* in the tree. Below, we show how such priorities are defined.

Given a BFS tree $T$ in $G_r$:

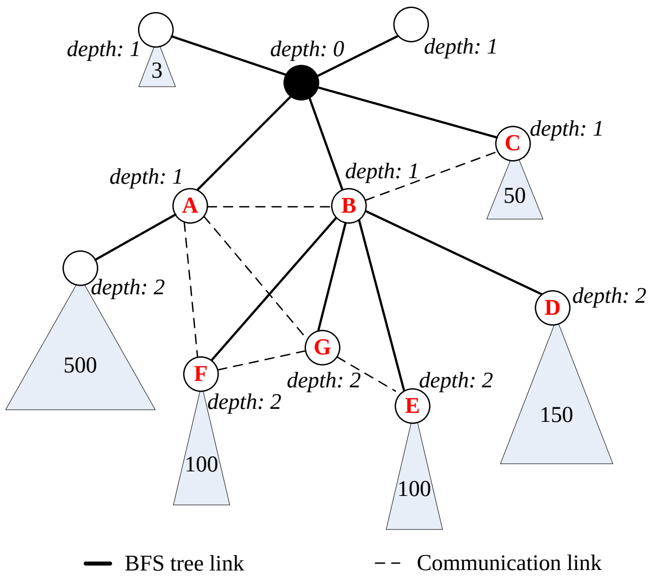


Figure 3.2: Examples of priority assignment in our algorithm.

- A node $x$ has a higher priority than another node $y$ if the subtree rooted at $x$ in $T$ has more nodes than the subtree rooted at $y$.
- If the subtrees rooted at nodes $x$ and $y$ have the same number of nodes, the one with less *potential parents* has a higher priority. A node regards a neighbor as a potential parent if this neighbor has a smaller hop count distance to the root in $T$ than itself.

The above definitions are based on the considerations of address space utilization. The first rule is so defined because node $x$ may have a better utilization. The second rule is so defined because a node with less potential parents may encounter difficulty to attach to the network. For example, in Fig. 3.2, if $Rm = 3$, the coordinator will choose nodes $A$, $B$, and $C$ as its child routers since they have larger subtrees. Similarly, $B$ will choose $D$, $E$, and $F$ as its child routers. However, if $Rm = 2$, the coordinator will choose $A$ and $B$ as its child routers. Further, $B$ will choose $D$ and $E$ as its child routers. Node $F$ is not selected because it has more (two) potential parents and thus has a higher probability to be connected in later stages of the formation.

### 3.3.1 Centralized Span-and-Prune Algorithm

Given a graph $G_r = (V_r, E_r)$, our goal is to find a tree $T = (V_T, E_T)$ from $G_r$ conforming to the ZigBee tree definition. The algorithm consists of a sequence of iterations. Initially, $T$ contains only the coordinator $t$. Then in each iteration, there are two phases: *Span* and *Prune*. In the Span phase, we will pick a node in $T$, say $x$, and span from $x$ a subtree $T'$ to include as many nodes not yet in $T$ as possible. Then we attach $T'$ to $T$ to form a larger tree. However, the new tree may not satisfy the ZigBee definition. So in the Prune phase, some of the newly added nodes in $T'$ may be trimmed. The resulting tree is then passed to the next iteration for another Span and Prune phases. This is repeated until no more nodes can be added. Each node in the network will be spanned at most once. To keep track of the nodes yet to be spanned, a queue $Q$ will be maintained. The algorithm is presented below.

1. Initially, let queue $Q$ contains only one node $t$. Let the depth of $t$ to zero. Also, let the initial tree $T = (\{t\}, \emptyset)$.

2. (*Span Phase*) Check if $Q$ is empty. If so, the algorithm is terminated and $T$ is the final ZigBee tree. Otherwise, let $x = \textit{dequeue}(Q)$ and construct a spanning tree $T'$ from $x$ as follows. Assuming the depth of $x$ in $T$ to be $\textit{depth}(x)$, we try to span a subtree from $x$ with height not exceeding $Lm - \textit{depth}(x)$ in $G_r$ in a breadth-first manner by including as many nodes in $V_r - V_T \cup \{x\}$ as possible. Let the resulting tree be $T'$.

3. (*Prune Phase*) Attach $T'$ to $T$ by joining node $x$. Still, name the new tree $T$. Since some of the nodes in $T'$ may violate the $Rm$ parameter, we traverse nodes in $T'$ from $x$ in a breadth-first manner to trim $T$.

   (a) When visiting a node, say $y$, set $y$ as “traversed” and check the number of children of $y$. If $y$ has more than $Rm$ children, we will compute their priorities based on $T'$ (refer to the definitions of nodes’ priorities in a tree given in the beginning of this section). Only the $Rm$ highest prioritized children will remain in $T$, and the other children will be pruned from $T$.

(b) When each node, say $y'$, that is pruned in step 3(a) or 3(b), let *tree*$(y')$ be the pruned subtree rooted at $y'$. Since *tree*$(y')$ is pruned, we will try to attach $y'$ to another node $n$ in $T'$ if $n$ satisfies the following conditions: 1) $n$ is neighboring to $y'$ but not a descendant of $y'$, 2) $n$ is not traversed yet, and 3) *depth*$(n)+1+$*height*$($*tree*$(y')) \leq Lm$. If so, we will connect the subtree *tree*$(y')$ to node $n$. If there are multiple such candidates, the one with a lower depth is connected first. If no such node $n$ can be found, $y$ prunes all its children. Then for each pruned child, we recursively perform this step 3(b) to try to reconnect it to $T'$. This is repeated until no further reconnection is possible.

4. After the above pruning, call the resulting tree $T$. For nodes that are newly added into $T$ in step 3, insert them into queue $Q$ in such a way that nodes with lower depth values are inserted first (these nodes will go through Span and Prune phases again). Then, go back to step 2.

To summarize, step 3(a) is to prune those nodes violating the $Rm$ constraint. In order to allow more vertices to join the network, step 3(b) tries to recursively reconnect those pruned subtrees to $T'$. Step 4 prepares newly joining nodes in $Q$ for possible spanning in step 2.

Fig. 3.3 illustrates an example. When being traversed, $y$ decides to prune $y'$ and keep $A$, $B$, and $C$ as children. Step 3(b) will try to reconnect $y'$ to $C$ or $D$, which are the neighbors of $y'$ in $T'$ and are not traversed. In this example, only $C$ can be considered because connecting to $D$ violates the depth constraint $Lm$.

The computational complexity of this algorithm is analyzed as follows. The iteration from step 2 to step 4 will be executed at most $|V_r|$ times. In each iteration, the complexity of constructing the tree $T'$ in step 2 is $O(N^2)$, where $N = |V_r| - |V_T|$. Step 3 checks all nodes in $T'$ and will be executed at most $O(N)$ times. For a run in Step 3 (assume visiting node $y$), the cost contains: 1) In step 3(a), $y$ can use a linear search method to find $Rm$ highest prioritized children and the computational cost is $O(D)$, where $D$ is the degree of $G_r$. 2) Since the subtree size of $y$ is at most $O(N)$ and a pruned node checks at most $O(D)$ neighbors

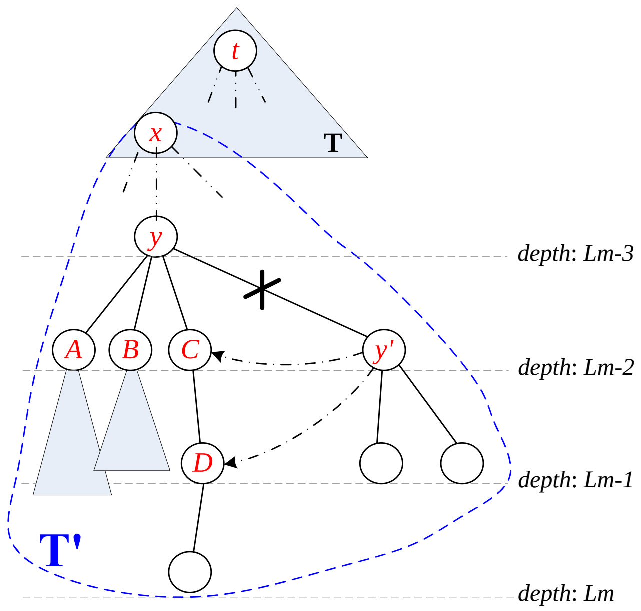


Figure 3.3: An example of the Span-and-Prune algorithm.

to find its new parent, the cost of step 3(b) in a run is $O(ND)$. So, in one iteration, the time complexity of step 3 will be $O(N(D + ND)) = O(N^2D)$. Step 4 sorts new nodes of $T$ according to their depth values, so the time complexity is $O(N^2)$. The complexity in each iteration is $O(N^2 + N^2D + N^2) = O(N^2D) = O(|V_r|^2D)$. Since there are at most $|V_r|$ iterations, the overall time complexity of this algorithm is $|V_r| \times O(|V_r|^2D) = O(|V_r|^3D)$. In practice, the value of $N$ may degrade quickly. So, after several iterations, the time complexity of an iteration will be close to $O(1)$.

### 3.3.2 Distributed Depth-then-Breadth-Search Algorithm

The above Span-and-Prune algorithm is a centralized one. In this section, we present a distributed algorithm, which does a depth-first search followed by a breadth-first-like search. The depth-first search tries to form some long, thin backbones, which are likely to pass through high-node-density areas. Then from these backbones, we span the tree in a breadth-first-like manner. The algorithm is presented below.

1. (Depth Probing) Given a graph $G_r = (V_r, E_r)$, the coordinator $t$ needs to probe the

depth of the tree first. A Probe(*sender_addr*, *current_depth*, $Lm$) packet is used for this purpose. The Probe packets are flooded in a BFS-like manner, until a depth $Lm$ is reached. Note that following the definition of ZigBee, before the final tree is determined, nodes will use their 64-bit MAC addresses to communicate with each other in this stage.

This algorithm begins by the coordinator $t$ flooding a Probe( *Addr*$(t)$, $0$, $Lm$) packet in the network, where *Addr*$(t)$ is $t$'s address. When a node $v$ receives a Probe(*sender_addr*, *current_depth*, $Lm$) packet, it does the following:

(a) If this is the first time $v$ receiving a Probe() packet, $v$ sets its parent *par*$(v)$ $=$ *sender_addr* and its depth *depth*$(v)$ $=$ *current_depth* $+ 1$. If *depth*$(v) < Lm$, $v$ rebroadcasts a Probe(*Addr*$(v)$, *depth*$(v)$, $Lm$) packet.

(b) If this is not the first time $v$ receiving a Probe() packet, it checks if *depth*$(v) >$ *current_depth*$+1$ is true. If so, a shorter path leading to the coordinator is found. So $v$ sets its parent *par*$(v)$ $=$ *sender_addr* and its depth *depth*$(v)$ $=$ *current_depth*$+1$. If *depth*$(v) < Lm$, $v$ rebroadcasts a Probe(*Addr*$(v)$, *depth*$(v)$, $Lm$) packet.

Note that to ensure reliability, a node may periodically rebroadcast its Probe() packet. And each node can know the number of its potential parents by the Probe() packet.

2. (Probe Response) After the above probing, a BFS-like tree is formed. Each node then reports to its parent a Report() packet containing (i) the size of the subtree rooted by itself and (ii) the height of the subtree rooted by itself. In addition, each node $v$ will compute a *tallest_child*$(v)$, which records the child of $v$ whose subtree is the tallest among all child subtrees.

3. (Backbone Formation) After the coordinator $t$ receives all its children's reports, it will choose at most $Rm$ children with the larger subtree sizes as backbone nodes. This is done by sending a Backbone$()$ message to each of the selected children. When a node $v$ receiving a Backbone$()$ message, it further invites its child with the tallest subtree, i.e., node *tallest_child*$(v)$, into the backbone by sending a Backbone$()$ packet to

*tallest_child*$(v)$. After this phase, $t$ has constructed a backbone with up to $Rm$ subtrees, each as a long, thin linear path.

4. (BFS-like Spanning) After the above backbone formation, the coordinator can broadcast beacons to start the network. A node can broadcast beacons only if it has successfully joined the network as a router (according to ZigBee, this is achieved by exchanging Association_Request and Association_Response with its parent). In our rule, a backbone node must associate to its parent on the backbone, and its parent must accept the request.

   For each non-backbone node, it will compete with each other in a distributed manner by its association priority, where the association priority is defined by the size of the subtree rooted by this node in the BFS-like tree formed in step 1. A non-backbone node sends its association requests by specifying its priority. A beacon sender should wait for association requests for a period of time and sorts the received requests by their priorities. Then the beacon sender can accept the higher-priority ones until its capacity ($Rm$) is full.

Compared to the ZigBee protocol, this algorithm requires nodes to broadcast two extra packets (Probe() and Report() packet) to accomplish step 1 and step 2. Also, in step 3, an extra Backbone() packet is needed.

## 3.4 Algorithms for the EDMM Problem

After connecting routers in the BDDTF problem, we can obtain a graph $\tilde{G}_d = (\{\tilde{V}_r \cup \tilde{V}_e, \tilde{E}_d\})$, as defined in Section 3.2. We can apply a bipartite maximum matching algorithm in [27] on $\tilde{G}_d$ to solve the EDMM problem. It is known that an optimal solution exists, so the number of orphan end devices can be minimized under the given $\tilde{G}_d$.

Since the above maximum matching algorithm is a centralized one, we present a distributed algorithm as follows. End devices overhear beacons from routers for opportunity of association. Also, each end device $e$ computes its weight as $(N_r^e)^{-1}$, where $N_r^e$ is the number of $e$'s neighbor routers. Intuitively, an end device has a larger weight if it has less potential

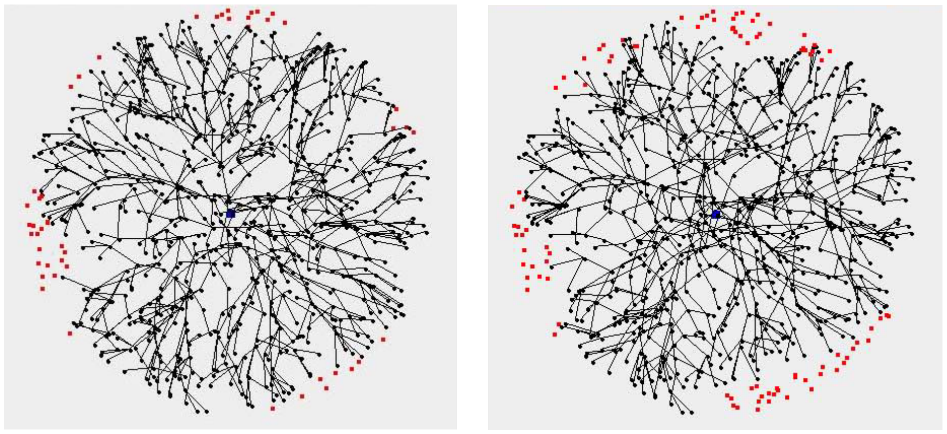

(a) (b)

Figure 3.4: Network formation results by (a) SP and (b) DBS algorithms when applying to the environment in Fig. 3.1.

parents, which also implies that it has less chances to join the network. When performing association procedures, each end device specifies its weight in its association requests. Routers simply accept the end devices that have larger weights.

## 3.5 Simulation Results

In this section, we first compare the proposed Span-and-Prune algorithm (SP) and Depth-then-Breadth-Search algorithm (DBS) for the BDDTF problem against the ZigBee network formation (ZB) algorithm, i.e., we simulate the networks that have only router-capable devices. To visualize how SP and DBS algorithms work, we first re-simulate the environment in Fig. 3.1. As shown in Fig. 3.4, the proposed algorithms can effectively reduce the number of orphan routers.

Next, we test a $90°$-sector environment with radius $200\ m$ and with $400$ randomly deployed router-capable nodes. We set $Cm = Rm = 2$ and $Lm = 8$ and set the transmission range of nodes to $32\ m$. Our simulation result show that the ZB, SP, and DBS algorithms incur $110.2$, $13.7$, and $37.9$ orphan routers, respectively, in average. Fig. 3.5 shows one scenario. In particular, we see that the SP algorithm may leave some nodes near the coordinator unconnected

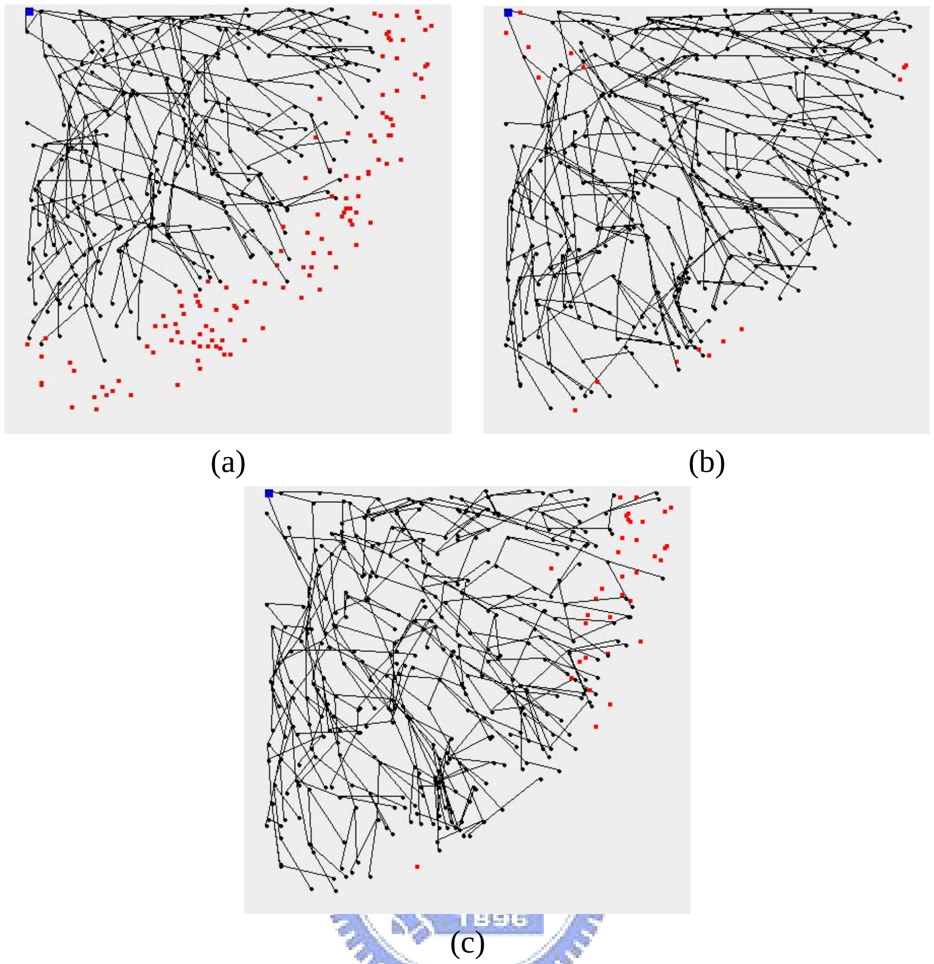
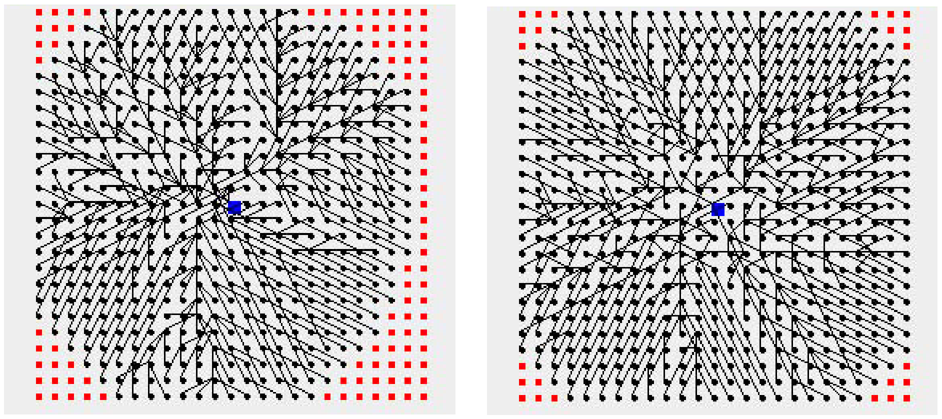


Figure 3.5: Network formation results in a $90^\circ$-sector environment when using (a) ZB, (b) SP, and (c) DBS algorithms.

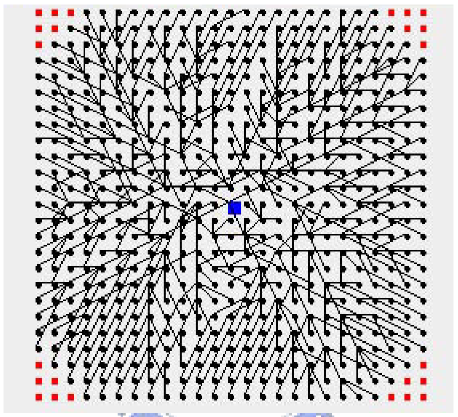
due to the $Rm$ constraint and its greedy nature.

Fig. 3.6 shows a result where routers are placed regularly in a $24 \times 24$ grid. The grid size is $10 \times 10\ m^2$. The transmission ranges of routers are set to $23\ m$. We set $Cm = Rm = 4$ and $Lm = 7$. In this simulation, ZB, SP, and DBS incur $92$, $24$, and $24$ orphan routers, respectively. Again, ZB will result in the most number of orphan routers. The performances of SP and DBS are the same in this case.

In the following simulations, router-capable devices are randomly distributed in a circular region and a coordinator is placed at the center of the network. We set the number of router-



(a) (b)



(c)

Figure 3.6: Network formation results in a $24 \times 24$ grid environment when using (a) ZB, (b) SP, and (c) DBS algorithms.

Table 3.1: Relationship between $Cm$, $Rm$, $Lm$, and network capacity.

| $(Cm = Rm, Lm)$ | (3, 7) | (3, 8) | (3, 9) | (4, 6) | (4, 7) | (5, 6) | (6, 6) |
|---|---|---|---|---|---|---|---|
| Total address spaces | 3280 | 9841 | 29524 | 5461 | 21845 | 19531 | 55987 |
| $(Cm = Rm, Lm)$ | (6, 4) | (7, 4) | (8, 4) | (9, 4) | (10, 4) | (11, 4) | (12, 4) |
| Total address spaces | 1555 | 2801 | 4681 | 7381 | 11111 | 16105 | 22621 |

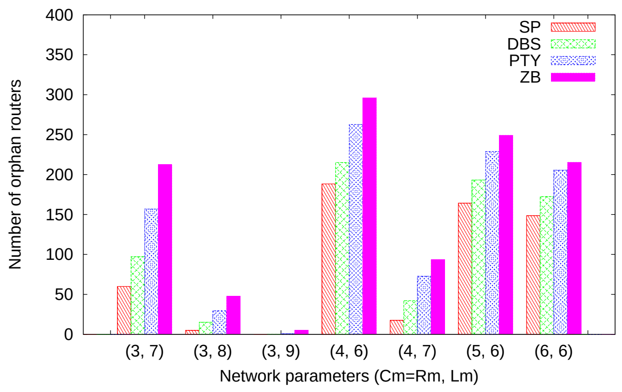


Figure 3.7: Comparison on the number of orphan routers with $N = 800$, $R = 200\ m$, and $TR = 35\ m$.

capable nodes $N = 800$, the network radius $R = 200\ m$, and devices' transmission range $TR = 35\ m$. Here, we restrict $Cm = Rm$ and vary $Rm$ and $Lm$ to observe the number of orphan routers. The settings of $Cm$, $Rm$, and $Lm$ are summarized in Table 3.1, which shows that total address spaces are large enough to contain all 800 nodes.

Fig. 3.7 shows the results. Here, we further compare to a priority-based (PTY) algorithm. PTY works similar to DBS, except that PTY do not form backbones, i.e., the step 3 of DBS is not executed and all nodes are taken as non-backbone nodes in step 4 of DBS. In all cases, SP performs the best, followed by DBS, PTY, and then ZB. DBS can perform better than PTY, which implies that backbones can effectively reduce orphan routers. In fact, our schemes can effectively reduce orphan routers even with smaller $Lm$ values. For example, the number of orphan routers of SP with $Rm = 3$ and $Lm = 7$ (resp., $Lm = 8$) are nearly the same as the number of orphan routers of ZB with $Rm = 3$ and $Lm = 8$ (resp., $Lm = 9$). Fig. 3.8 shows another result with $N = 1600$, $R = 200\ m$, and $TR = 35\ m$. The similar trend is seen.

Next, we set $N = 800$, $R = 200\ m$, $TR = 60\ m$, and $Lm = 4$ and vary $Rm(= Cm)$ to compare the performances of different network formation algorithms. Fig. 3.9 shows our simulation results. In particular, we see that our schemes can allow more devices to join the network even with small $Rm$. For example, the number of orphan routers of SP when $Rm = 6$

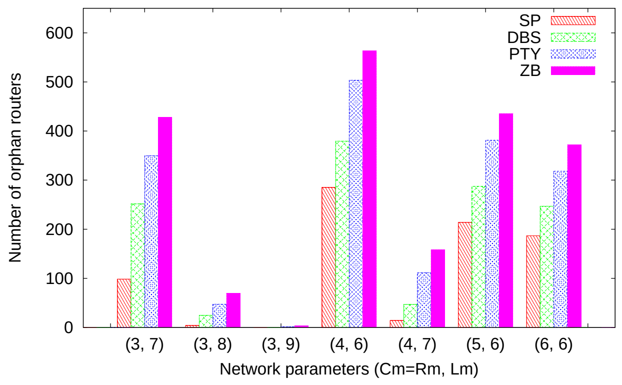


Figure 3.8: Comparison on the number of orphan routers with $N = 1600$, $R = 200\ m$, and $TR = 35\ m$.

are nearly the same as the number of orphan routers of ZB when $Rm = 11$.

Next, we simulate the networks that contain both router-capable devices and end devices. We place $800$ routers in a circular area of radius $200\ m$. The coordinator is located at the center of the network. The transmission range of routers are $35\ m$. We randomly place $8000$ end devices in this network. In this simulation, the transmission distance of end devices set to $15$ to $30\ m$. An end device can only associate to a router located within its transmission range. We set $Cm = 15$, $Rm = 3$, and $Lm = 9$. The proposed SP algorithm is used to connect router-capable devices. Then we compare the proposed centralized algorithm, denoted as OPT, and distributed algorithm, denoted as DIS, against the ZigBee protocol, denoted as ZB. Fig. 3.10 shows the simulation results. We can see that the number of orphan end devices decreases as the transmission range of devices increases. Compare to ZB, the proposed algorithms perform pretty well.

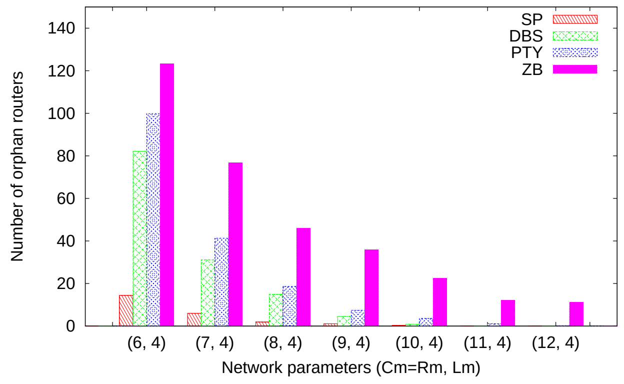


Figure 3.9: Comparison on the number of orphan routers with $N = 800$, $R = 200\ m$, and $TR = 60\ m$.

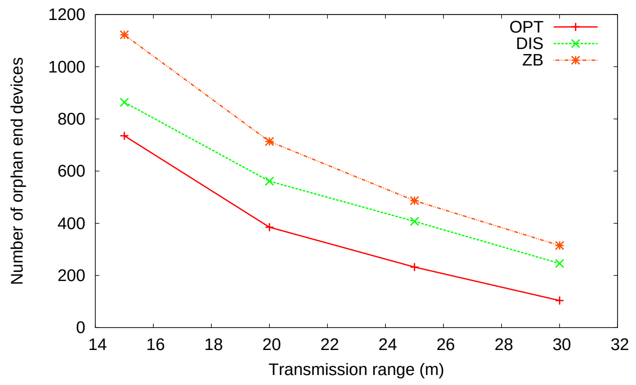


Figure 3.10: Comparison on the number of orphan end devices at various transmission ranges.

# Chapter 4

# Network Formation Protocols for Long-Thin ZigBee Networks

## 4.1 Motivations

In this work, we discuss the *long-thin (LT)* network topology, which seems to have a very specific architecture, but may be commonly seen in many WSN deployments in many applications, such as gas leakage detection of fuel pipes (Fig. 4.1(a)), carbon dioxide concentration monitoring in tunnels (Fig. 4.1(b)), stage measurements in sewers (Fig. 4.1(c)), street lights monitoring in highway systems (Fig. 4.1(d)), flood protection of rivers (Fig. 4.1(e)), and vibration detection of bridges (Fig. 4.1(f)). In such a network, nodes may form several long backbones and these backbones are to extend the network to the intended coverage areas. A backbone is a linear path which may contain tens or hundreds of ZigBee routers and may go beyond hundreds or thousands of meters. So the network area can be scaled up easily with limited hardware cost.

The address assignment and tree routing schemes defined in the original ZigBee specification may work poorly, if not fail, in a LT topology. Since the length of a network address is $16$ bits, the maximum address capacity of a ZigBee network is $2^{16} = 65536$. Obviously, the ZigBee address assignment is much suitable for regular networks, but not for LT WSNs. For example, when setting $Cm = 4$ and $Rm = 2$, the depth of the network can only be $14$. Also, when there are some LT backbones, the address space will not be well utilized. Moreover, in

Figure 4.1: Long-thin networks.

ZigBee tree routing, each node can only choose its parent or child as the next node. Since no shortcut can be taken, this strategy may cause longer delay in LT networks.

## 4.2 Long-Thin Network: Formation, Addressing, and Routing

Our goal is to automatically form a LT WSN, give addresses to nodes, and conduct routing. Fig. 4.2(a) shows an example of a LT WSN. For simplicity, we assume that all nodes are router-capable devices. To form the network, nodes are divided into multiple *clusters*, each as a line segment. For each cluster, we define two special nodes, named *cluster head* and *bridge*. The cluster head (resp., the bridge) is the first (resp., last) in the line segment. As a special case, the coordinator, is also considered as a cluster head. The other nodes are *network nodes* (refer to Fig. 4.2(b)). A cluster $C$ is a *child* cluster of a cluster $C'$ if the cluster head of $C$ is connected to the bridge of $C'$. Reversely, $C'$ is the *parent* cluster of $C$. Note that a cluster must have a linear path as its subgraph. But it may have other extra links beside the linear path. For example, in Fig. 4.2(b), there are two extra radio links ($A$, $A2$) and ($A1$, $A3$) in $A$'s cluster. To be compliant with ZigBee, we divide the ZigBee 16-bit network address into two parts, an $m$-bit *cluster ID* and a $(16-m)$-bit *node ID*. The value of $m$ will be discussed later

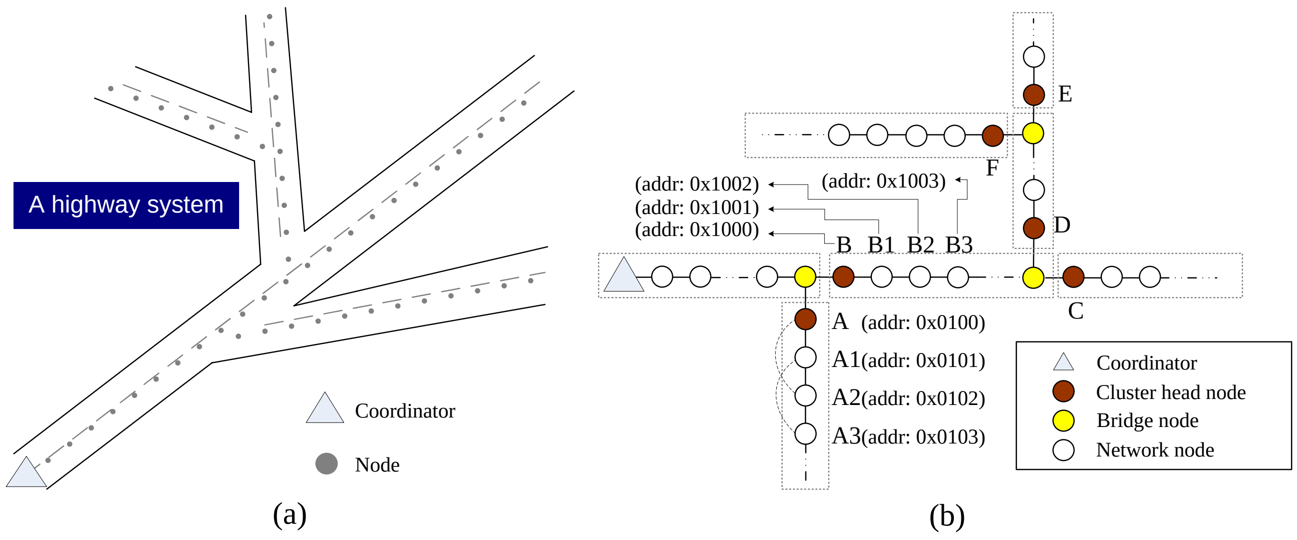


Figure 4.2: (a) A LT WSN. (b) Role assignment.

on. The network address of a node $v$ is thus expressed as ($C_v$, $N_v$), where $C_v$ and $N_v$ are $v$'s cluster ID and node ID, respectively.

### 4.2.1 Node Placement

Before deploying a network, the network manager needs to carefully plan the placement of cluster heads, bridges, and network nodes. There are some basic principles:

1. The network contains a number of linear paths, each called a cluster.
2. For each cluster, the first and the last nodes are pre-assigned (manually) as cluster head and bridge, respectively.
3. A cluster head that is not the coordinator should have a link to the bridge of its parent cluster.
4. Conversely, the bridge of a cluster which has child clusters should have a link to the cluster head of each child cluster.
5. A cluster does not cross other clusters and does not have links with other clusters except those locations nearby the cluster head and bridge areas.

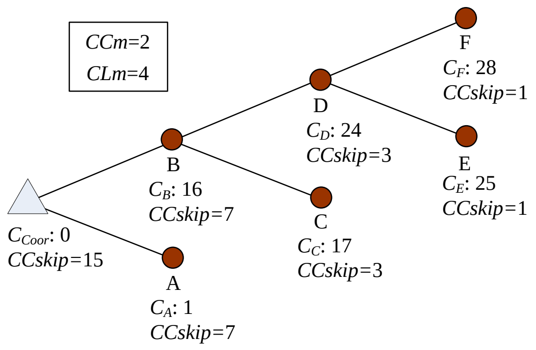


Figure 4.3: The logical network of Fig. 4.2(b).

After planning the network, the network manager can construct a *logical network* $G_L$, in which each cluster is converted into a single node and the parent-child relationships of clusters are converted into edges. For example, Fig. 4.3 is the logical network of Fig. 4.2(b). From $G_L$, we can determine the maximum number of children $CCm$ of a node in $G_L$ and the depth $CLm$ of $G_L$. By $CCm$ and $CLm$, we can know that this network will have at least $CN = \frac{1-CCm^{CLm+1}}{1-CCm}$ clusters. Then the network manager can decide the value of $m$ such that $2^{m-1} < CN \leq 2^m$ is satisfied.

To initialize the network, each node should periodically broadcast HELLO packets including its IEEE 64-bit MAC address, 16-bit network address (initially set to *NULL*), and role. In this work, we consider only symmetric links. A communication link ($u$, $v$) is established only if $u$ receives $v$'s HELLO including $u$ as its neighbor and the HELLO's signal quality is above a threshold. Note that the signal quality should be the average of several packets. Then each node can maintain a neighbor table containing its neighbors' addresses, roles, and ranks. After such HELLO exchanges, the coordinator will start a *node ranking algorithm* to differentiate nodes' distances to it (Section 4.2.2). Then, a *distributed address assignment procedure* will be conducted to assign network addresses to nodes (Section 4.2.3).

## 4.2.2 Node Ranking

We extend the concept of one-dimensional ranking algorithm in [46] to assign a *rank* to each node. In this algorithm, all nodes except the coordinator will perform the same procedure.

Initially, the rank of the coordinator is $0$ and all other nodes have a rank of $K$, where $K$ is a positive constant. At the end of the algorithm, each node will have a stable rank, which will reflect its distance to the sink. Note that here "distance" is not necessarily a hop count. In fact, it reflects its physical distance to the sink following some line segments to the sink.

Except the coordinator, all other nodes will continuously change their ranks. The coordinator will periodically broadcast a *Heartbeat* packet with its rank. On receiving a *Heartbeat*, a node will rebroadcast it by including its current rank. After receiving all its neighbors' *Heartbeat* packets, a node will calculate its new rank by averaging its neighbors' ranks. Since the coordinator's rank is fixed, after receiving several *Heartbeat* packets, nodes that locate closer to the coordinator will have lower ranks.

Now we give the details of the ranking algorithm. The format of *Heartbeat* is *Heartbeat*(*sender's 64-bit address, seq*, *rank*). In the beginning, the coordinator broadcasts a *Heartbeat*(*coordinator*, $0$, $0$). Then it periodically broadcasts *Heartbeat* packets, each time with an incremented *seq*, until *seq* $> h$, where $h$ is the maximum hop count distance from the coordinator to any node. The operations taken by a non-coordinator node $v$ are defined as follows.

1. On receiving a *Heartbeat*($u$, $u$'s *seq*, *$u$'s rank*), $v$ checks if it has broadcast a *Heartbeat* with this sequence number *seq*. If not, $v$ updates its sequence number to this received *seq* and broadcasts a *Heartbeat*($v$, $v$'s *seq*, *$v$'s rank*). Then $v$ keeps a record of the pair ($u$'s *seq*, *$u$'s rank*). If $v$ has received all its neighbors' *Heartbeat* packets with the same *seq* as its own, it updates its rank to the average of its neighbors' ranks (not including its own rank). Otherwise, it sets a timer *WaitHeartbeat*.

2. When timer *WaitHeartbeat* times out, $v$ broadcasts a *NACK*($L$), where $L$ is the list of neighbors whose *Heartbeat*s are still missing. Then it sets another *WaitHeartbeat* timer, until the maximum number of retries is reached.

3. When $v$ receives a *NACK*($L$) such that $v \in L$, it broadcasts a *Heartbeat*($v$, $v$'s *seq*, *$v$'s rank*).

The above step 1 enforces a node to broadcast its rank whenever a new *seq* is received. New *seq*s are issued by the sink. A node can update its rank after receiving ranks of all its neighbors with the same *seq* as its own. Steps 2 and 3 are to improve the procedure proposed in [46] to guarantee reliability due to the fact the broadcast is unreliable in wireless networks. Note that the coordinator needs to broadcast at least $h+1$ *Heartbeat* packets to guarantee that every node can modify its initial rank. The rank of the coordinator will gradually diffuse to the rest of the nodes and thus decrease their ranks. Nodes' ranks will reflect their distances (not Euclidean distances) following the linear paths of the LT WSN to the coordinator. At the end of the algorithm, each node can record its neighbors' final ranks in its neighbor table. We say that a ranking result is *in-order* if for each cluster, (i) the cluster head (resp., bridge) has the smallest (resp., largest) rank value, (ii) the ranks of cluster members correspond to their distances to the cluster head, and (iii) the bridge node's rank value is smaller than the ranks of the cluster's child cluster members.

Reference [46] shows that in a linear path topology, the above ranking method can effectively achieve in-order ranking. However, a LT WSN may have some branches, and thus the ranking result may not always be in-order. Fig. 4.4 shows some results, where the inter-node distance is $20\ m$ and the transmission range is $45\ m$. The ranking result in Fig. 4.4(a) is in-order. In Fig. 4.4(b), the ideal ranking result should satisfy $B < C < D < E < F$. Unfortunately, the result satisfies $B < C < E < D < F$. The ranks of some members of $E$'s cluster are smaller than the ones of some members of $H$'s cluster because some $E$'s members are affected by some members of its parent cluster. We see that $D$ and $E$ have the same number of neighbors but $D$'s rank is affected by some $H$'s cluster members. This makes $D$'s rank higher than $E$'s, causing the final ranking result not in-order. In Fig. 4.4(c), $F$ and $G$ have smaller ranks than $E$ because they are affected by $A$'s and $B$'s ranks. To summarize, we observe that if some members of a cluster have links to the cluster's parent cluster members, the ranking result may not be in-order.

Here we make two remarks. First, if a ranking result is in-order, it will facilitate our address assignment and thus network formation. Second, even if a ranking result is not in-order, in

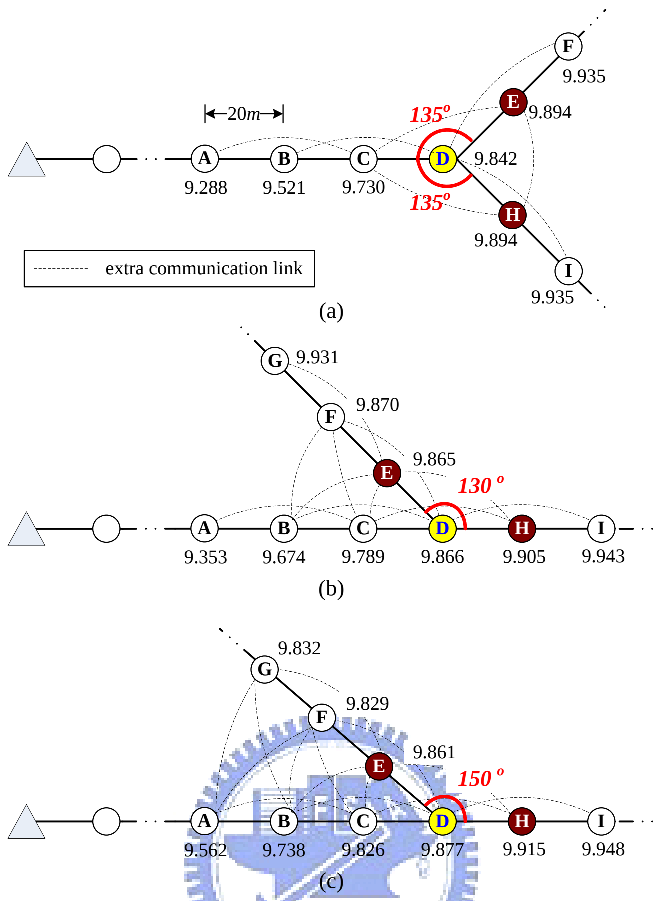


Figure 4.4: Some ranking examples.

some cases we can still assign addresses, and thus route packet, successfully. These will be elaborated further later on.

### 4.2.3 Distributed Address Assignment

The basic idea of our address assignment is as follows. The assignment of cluster IDs depends on the maximum number of branches in the logical network $G_L$. If $CCm = 1$, then the network is a linear path and the address assignment is a trivial job. If $CCm \geq 2$, then we follow the style of ZigBee to assign addresses in a recursive manner. The coordinator has an ID of $0$. For each node at depth $d$ in $G_L$, if its cluster ID is $C$, then its $i$-th child cluster is assigned a cluster ID of $C + (i-1) \times CCskip(d) + 1$, where

$$CCskip(d) = \frac{1 - CCm^{CLm-d}}{1 - CCm}. \tag{4.1}$$

Fig. 4.3 shows the assignment result for the network in Fig. 4.2(b). Since each cluster is a linear path, node IDs of the cluster members can be assigned sequentially. Starting from the cluster head with an address of $0$, the rest of the nodes can gradually increment their node IDs following the former ranking results, until the bridge node is reached. In Fig. 4.2(b), we have shown some assignment results, where each address is expressed in Hex and the first two symbols represent the cluster ID and the last two represent the node ID.

Now we present the detail algorithm. It is started by the coordinator by broadcasting beacons with the predefined $CCm$ and $CLm$. When a node without a network address receives a beacon, it will send an *Association_Request* to the beacon sender. If it receives multiple beacons, the node with the strongest signal strength will be selected. When the beacon sender, say, $v$ at a logical depth $d$, receives the association request(s), it will do the following:

1. If $v$ is *not* a bridge node, it sets a parameter $N = N_v + 1$ (note that when entering this procedure, $v$ already obtains its address ($C_v$, $N_v$)). Then it sorts these request senders according to their ranks in an ascending order into a list $L$. Then $v$ sequentially examines each node $u \in L$. There are two cases:

    - If $u$ is a cluster head node, $v$ skips $u$ and continues to examine the next node in $L$.
    - Otherwise, $v$ assigns address ($C_v$, $N$) to $u$ and increments $N$ by $1$. Then $v$ replies an *Association_Response* to $u$ with this address. In case that $u$ is a bridge node, $v$ stops examining $L$; otherwise $v$ loops back and continues to examine the next node in $L$.

    After finishing the above iteration, let $u$ be the last node in $L$ receiving an address. Then $v$ delegates $u$ as the next beacon sender by sending a command *next_beacon_sender*$(u)$ to $u$.

2. If $v$ is a bridge node, it only accepts requests from cluster heads. At most $CCm$ requests will be accepted, and $v$ will reply to the $i$-th least ranked cluster head, $i \leq CCm$, an *Association_Response* with an address ($C_v + (i-1) \times CCskip(d) + 1$, 0) and a *next_beacon_sender* command. Note that, these cluster heads need to set their logical depths to $d + 1$.

For each node $u$ which receives a *next_beacon_sender*$(u)$ in the above steps, it will use the MLME-START primitive defined in IEEE 802.15.4 to start its beacons. Then the same procedure repeats. Note that we allow a beacon sender to accept multiple children so as to reduce the communication cost of address assignment.

We say that an address assignment result is *as planned* if (i) each pair of cluster head and bridge are assigned to the same cluster ID and (ii) each bridge is correctly connected to its child cluster heads. Below, we make two observations about the address assignment results. First, if the ranking result is in-order and the nodes near-by each cluster head can receive stronger signal from its own cluster head than from others, the address assignment will be as planned. For example, in Fig. 4.4(a), the network will be formed as planned. Second, there are some cases that the formed network is as planned even if the ranking result is not in-order. For example, in Fig. 4.4(b), assuming $B$ as the beacon sender, $B$ will accept nodes $C$ and $D$ with $D$ as the bridge. Although $F$ may send an *Association_Request* to $B$, $B$ will not accept $F$ because the address assignment will stop when a bridge is encountered. However, in Fig. 4.4(c), the assignment may not be as planned. Assuming $A$ as the beacon sender, $A$ will accept $B$, $C$, $F$, and $G$, and then will choose $G$ as the next beacon sender. So $H$ and the descendants of $H$ will not be able to join the network.

### 4.2.4 Routing Rules

Routing in our LT WSN can be purely based on the above address assignment results. Through HELLO packets, a node can collect its neighbors' network addresses. Suppose that a node $v$ at logical depth $d$ receives a packet with a destination address $(C_{dest}, N_{dest})$. If $v$ is the destination, it simply accepts this packet. Otherwise, $v$ performs the following procedures.

1. If the destination is a neighbor of $v$, $v$ sends this packet to the destination directly.

2. If $C_{dest} = C_v$, the destination is within the same cluster. Node $v$ can find an ancestor or a descendant in its neighbor table, say, $u$ such that $C_u = C_{dest}$ and the value of $|N_u - N_{dest}|$ is minimized, and forward this packet to $u$.

3. If $C_{dest}$ is a descendant cluster of $C_v$, i.e., $C_v < C_{dest} \leq C_v + (CCm-1) \times CCskip(d) + 1$, then $v$ checks if it has a neighbor $u$ which satisfies $C_u \leq C_{dest} \leq C_u + (CCm-1) \times CCskip(d+1) + 1$. If such a $u$ exists, then $v$ forwards the packet to $u$. In case that there are multiple candidates, the one with the smallest $|N_u - N_{dest}|$ is selected. Otherwise, $v$ finds a neighbor $u$ which is located in the same cluster and has the maximum $N_u$ and forwards the packet to $u$.

4. For all other cases, $C_{dest}$ must be an ancestor cluster of $C_v$ or not within the same logical subtree. Then $v$ checks if it has a neighbor $u$ which satisfies $C_u < C_v \leq C_u + (CCm - 1) \times CCskip(d-1) + 1$. If such a $u$ exists, $v$ forwards the packet to $u$. Note that the above condition confines that $C_u$ is the parent cluster of $C_v$. Otherwise, $v$ finds a neighbor $u$ which is located in the same cluster and has the minimum $N_u$ and forwards the packet to $u$.

Note that the above design tries to strike a balance between efficiency and simplicity. It basically follows the ZigBee tree-like routing. However, making shortcut along the linear paths of the LT WSN is possible due to the existence of neighbor tables and our design of hierarchical network addresses. Therefore, unlike the original ZigBee tree routing, nodes are not restricted to relay packets only to their parents or children.

## 4.3 Simulation Results

We start by giving three scenarios to demonstrate how our schemes work. The first one is an “imaginary” network planning example on a highway system as shown in Fig. 4.5. After planning, we can obtain $CCm = 3$ and $CLm = 5$. The network can accommodate at most

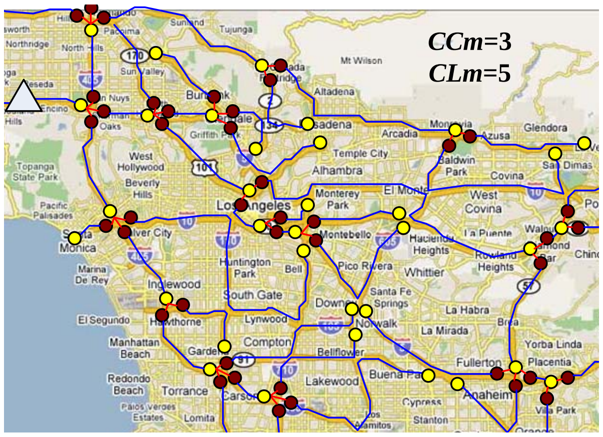


Figure 4.5: A network planning result.

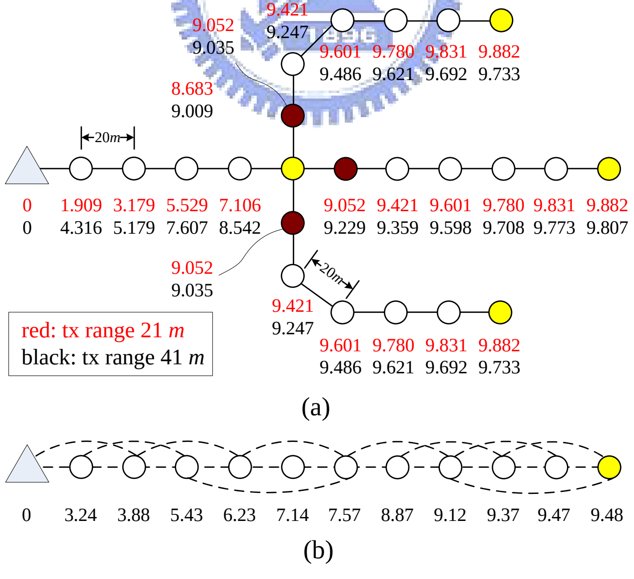


Figure 4.6: Some ranking results.

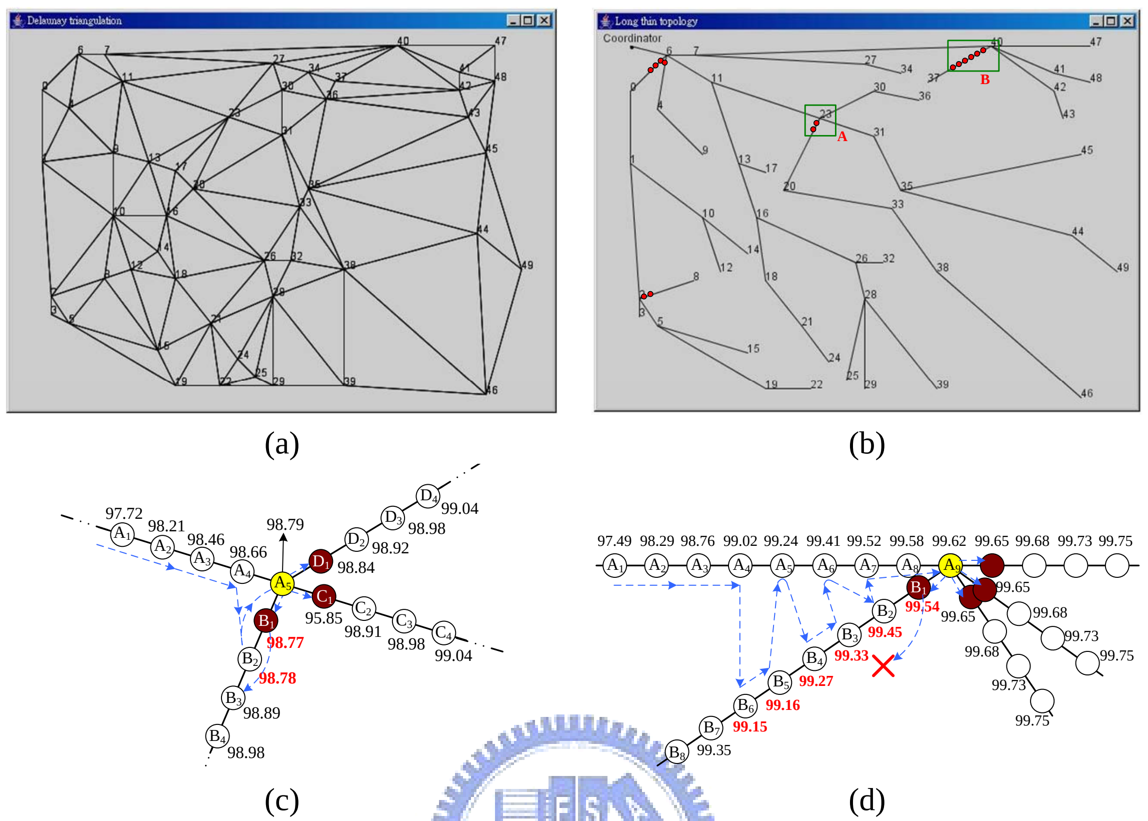


Figure 4.7: (a) A random generated Delaunay triangulation. (b) A LT-WSN generated from the Delaunay triangulation. (c) The ranking result of the region A. (d) The ranking result of the region B.

$(1-3^6)/(1-3) = 364$ clusters, which can be expressed in $9$ bits and each of which can have at most $2^{16-9} = 128$ members. We also simulate the node ranking algorithm in two LT networks as shown in Fig. 4.6, where adjacent nodes are evenly separated by a distance of $20$ $m$. After $20$ *Heartbeat* packets from the coordinator, we see that both networks will have in-order ranking. In particular, note that the linear path in Fig. 4.6(b) has irregular links between nodes.

Next, we simulate some LT-WSNs that are generated by a systematical method as follows. An $n_1 \times n_2$ rectangle region is simulated, on which $k$ nodes are generated randomly to serve as bridge nodes. From these bridges, we conduct Delaunay triangulation. Using the bridge nearest to the upper-left corner of the rectangle as the root, we build a shortest path tree from

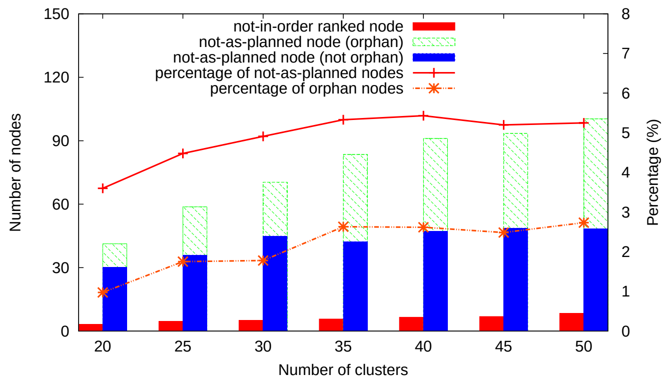


Figure 4.8: Simulation results of the numbers of not-in-order ranked and not-as-planned nodes.

the edges of the Delaunay triangulation to connect to the other $k-1$ bridges. The root is then connected to the coordinator at the left-top corner. Then we traverse the tree from the coordinator and generate nodes at every distance of $d$ on each edge of the shortest path tree. Fig. 4.7(a) shows an example of a random generated Delaunay triangulation. A LT topology based on Fig. 4.7(a) is illustrated in Fig. 4.7(b).

Based on the above model, we generate networks in a $4.8\,km \times 3.2\,km$ field with $d = 20\,m$. As mentioned in Section 4.2.3, the address assignment may not be as planned if the ranking result is not in-ordered. For example, in Fig. 4.7(b) the nodes marked in black small circles are not in-ordered ranked. Fig. 4.7(c) shows the network topology for region A (the dotted lines are the order of address assignment). We can see that the descendant of $B_1$ is not as planned since $B_2$ connects to $B_1$'s parent cluster. Fig. 4.7(d) shows the ranking result and the network topology of region B. In this case, nodes $B_2$, $B_3$ ,..., $B_6$, which are planned to be the descendants of $B_1$, are connected by $B_1$'s parent cluster members. $B_1$ can not find a neighbor to form its cluster, resulting in the descendants of $B_6$ being disconnected from the network. Here, we call these disconnected nodes *orphans*. Fig. 4.8 shows that nodes can still be assigned to the desired address with high probability ($\geq 94\%$) even when there are not-in-order ranked nodes. In average, less than $3\%$ of the nodes will become orphans in our

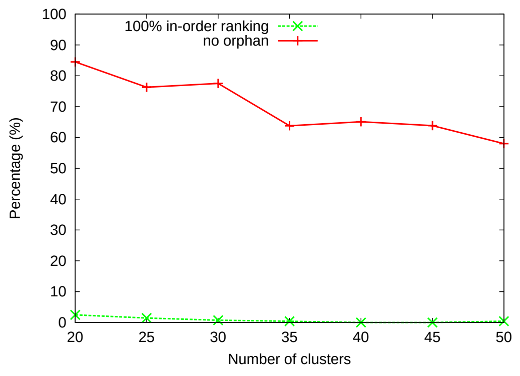


Figure 4.9: The percentages of 100% in-order ranking and no-orphan cases.

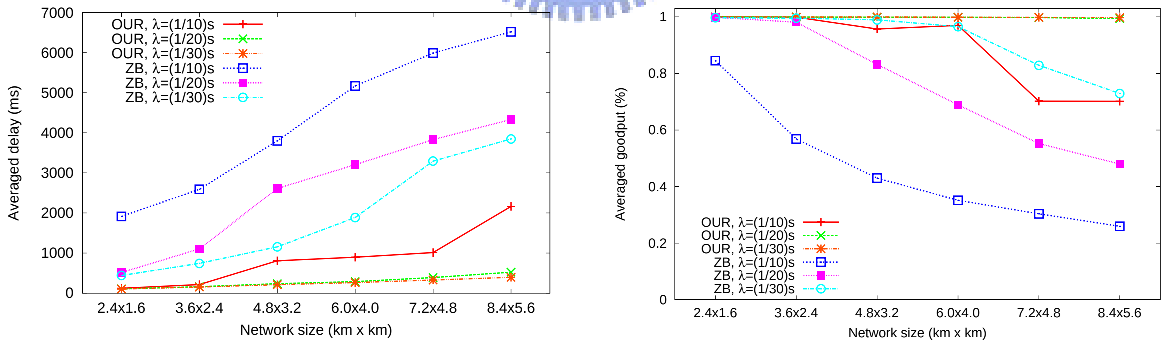
simulations. This result demonstrates that the network formation can connect all nodes with high probability. Fig. 4.9 further shows the percentages of 100% in-order ranking and no orphan. We can see that only few cases can achieve 100% in-ordered ranking. But, in most cases, all nodes can be connected to the network. We observe that to avoid the above orphan problem, the network manager should lower down the density of nodes near by bridges to reduce the numbers of links in such areas.

Next, we evaluate the proposed routing protocol. The results are from networks with $50$ with $d = 20\ m$. IEEE 802.15.4 unslotted CSMA/CA mechanism is implemented. Packets are generated from each node to random destinations with a poisson process at a rate $\lambda$. The buffer size of each node is 6.4 KB. When a node's buffer overflows, no further packets will be accepted. We measure the *goodput* of the network, which is defined as the ratio of packets successfully received by the specified destinations. We compare the proposed routing scheme (denoted as OUR) with the ZigBee scheme (denoted as ZB). When using ZB, the node $v$ that receives a packet will do the following procedures. If $v$ is a normal node, it simply judges to relay the incoming packet to ($C_v$, $N_v + 1$) or ($C_v$, $N_v - 1$). For the case that if $v$ is a cluster head (resp., bridge node), it relays the packet to the bridge node (resp., cluster head) of its parent (resp., the corresponding child) cluster. Some other parameters are list in Table 4.1.

We first set the transmission ranges of nodes to $81\ m$ and vary $\lambda$. Fig. 4.10 shows the result. Note that packets may be delayed or dropped due to buffer constraint. Our scheme

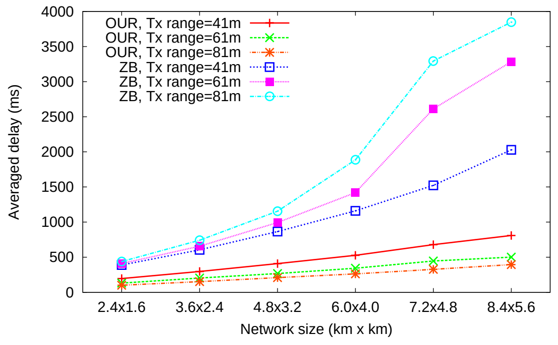
Table 4.1: Simulation parameters (for the proposed LT routing protocol).

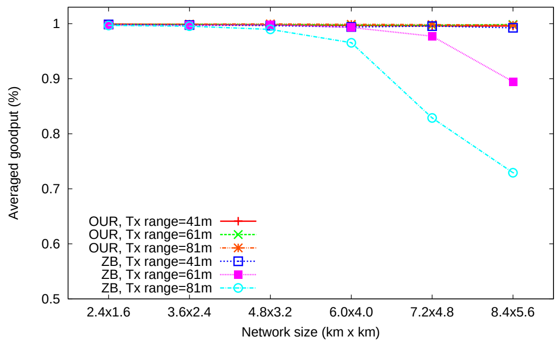
| Parameter | Value |
|---|---|
| length of a frame's header and tail | 18 Bytes |
| length of data payload | 46 Bytes |
| bit rate | 250k bps |
| symbol rate | 62.5k symbols/s |
| aUnitBackoffPeriod | 20 symbols |
| aCCATime | 8 symbols |
| macMinBE | 3 |
| aMaxBE | 5 |
| macMaxCSMABackoffs | 4 |
| maximum number of retransmissions | 3 |



(a) (b)

Figure 4.10: Comparison on (a) delay and (b) goodput at various data rates.





(a) (b)

Figure 4.11: Comparison on (a) delay and (b) goodput at various transmission ranges.

outperforms ZB in averaged delay in all cases. In terms of the goodput, our scheme can guarantee almost $100\%$ packet delivery when $\lambda = (1/20)\ s$ or $(1/30)\ s$, where ZB suffers from lower goodputs as the traffic load increases or the number of clusters increases. Fig. 4.11 shows another result when we vary the transmission ranges of nodes when $\lambda = (1/30)\ s$. It shows that when the transmission range increases, our scheme induces less delay. But this is not the case for ZB because it restricts packets to be transmitted hop-by-hop while ours allows taking shortcuts. The trend is similar when we look at the goodputs.

# Chapter 5

# Data Collection Strategies for ZigBee Networks

## 5.1 Observations and Motivations

This work discuss data collection strategies for ZigBee networks. We aim at designing quick convergecast solutions for ZigBee tree-based, beacon-enabled WSNs. This work is motivated by the following observations. First, we see that most related works are not compliant to the ZigBee standard. Second, we believe that tree-based topology is more suitable if power management is a main concern in WSNs. Third, the specification does not clearly define how to choose the locations of routers' active portions such that the convergecast latency can be reduced.

Convergecast has been investigated in several works [26][30][34][47][66][72]. With the goals of low latency and low energy consumption, reference [66] shows how to connect sensors as a balanced reporting tree and how to assign CDMA codes to sensors to diminish interference among sensors, thus achieving energy efficiency. The work [72] aims to minimize the overall energy consumption under the constraint that sensed data should be reported within specified time. Dynamic programming algorithms are proposed by assuming that sensors can receive multiple packets at the same time. As can be seen, both [66] and [72] are based on quite strong assumptions on communication capability of sensor nodes and they do not fit into the ZigBee specification. In [47], the authors propose an energy-efficient and low-latency

MAC, called *DMAC*. Sensors are connected by a tree and stay in sleep state for most of the time. When sensors change to active state, they are first set to the receive mode and then to the transmit mode. *DMAC* achieves low-latency by staggering wake-up schedules of sensors at the time instant when their children switch to the transmit mode. Similar to [47], reference [34] arranges wake-up schedule of sensors by taking traffic loads into account. Each parent periodically broadcasts an advertisement containing a set of empty slots. Children nodes request empty slots according to their demands. In [30], the authors propose a distributed convergecast scheduling algorithm. The basic concept is to connect nodes by a spanning tree. Then the algorithm reduces the tree to multiple lines. For each line, the algorithm schedules nodes' transmission times in a bottom-up manner. Reference [26] presents a centralized solution to convergecast. The algorithm divides nodes into many segments such that the transmission of a node in a segment does not cause interference to other transmissions in the same segment. The aim is to increase the degree of parallel transmissions to decrease latencies. Although these results [26][30][34][47] are designed for quick convergecast, the solutions are not compliant to the ZigBee standard for the following two reasons. Firstly, in these works, nodes' wake/sleep times are dynamically changed according to their schedules. However, in a ZigBee beacon-enabled tree network, nodes' wake/sleep times must be fixed in the way that each router wakes up twice in each cycle to receive its children's packets and to transmit packets to its parent, respectively. The coordinator (resp., an end device) wakes up once to receive its children's packets (resp., to transmit packets to its parent). Secondly, the scheduling of [26][30][34][47] is transmission-based, while ours are receiving-based. The implication is that the former may cause a router to be active multiple times per cycle. This is incompatible with the ZigBee specification.

## 5.2 The Minimum Delay Beacon Scheduling (MDBS) Problem

This section formally defines the convergecast problem in ZigBee networks. Given a ZigBee network, we model it by a graph $G = (V, E)$, where $V$ contains all routers and the coordinator and $E$ contains all symmetric communication links between nodes in $V$. The coordinator also serves as the sink of the network. End devices can only associate with routers, but are not included in $V$. In our work, we consider two kinds of interference between routers. Two routers have *direct interference* if they can hear each others' beacons. Two routers have *indirect interference* if they have at least one common neighbor. Both interferences should be avoided when choosing routers' active portions. From $G$, we can construct an *interference graph* $G_I = (V, E_I)$, where edge $(v, u) \in E_I$ if there are direct/indirect interferences between $v$ and $u$. There is a duty cycle requirement $\alpha$ for this network. From $\alpha$ and Table 2.2, we can determine the most appropriate value of $BO - SO$. We denote by $k = 2^{BO-SO}$ the number of active portions (or slots) per beacon interval.

The beacon scheduling problem is to find a slot assignment $s(v)$ for each router $v \in V$, where $s(v)$ is an integer and $s(v) \in [0, k-1]$, such that router $v$'s active portion is in slot $s(v)$ and $s(v) \neq s(u)$ if $(v, u) \in E_I$. Here the slot assignment means the position of the outgoing superframe of each router (the position of the incoming superframe, as clarified earlier, is determined by the parent of the router). Motivated by Brook's theorem [69], which proves that $n$ colors are sufficient to color any graph with a maximum degree of $n$, we would assume that $k \geq D_I$, where $D_I$ is the maximum degree of $G_I$.

Given a slot assignment for $G$, the report latency from node $v$ to node $u$, where $(v, u) \in E$, is the number of slots, denoted by $d_{vu}$, that node $v$ has to wait to relay its collected sensory data to node $u$, i.e.,

$$d_{vu} = (s(u) - s(v)) \bmod k. \tag{5.1}$$

Note that the report latency from node $v$ to node $u$ ($d_{vu}$) may not by equal to the report latency from node $u$ to node $v$ ($d_{uv}$). Therefore, we can convert $G$ into a weighted directed graph $G_D = (V, E_D)$ such that each $(v, u) \in E$ is translated into two directed edges $(v, u)$ and $(u, v)$ such that $w((v, u)) = d_{vu}$ and $w((u, v)) = d_{uv}$. The report latency for each $v \in V$ to the sink is the sum of report latencies of the links on the shortest path from $v$ to the sink in $G_D$. The latency of the convergecast, denoted as $L(G)$, is the maximum of all nodes' report latencies.

**Definition 4** *Given $G = (V, E)$, $G$'s interference graph $G_I = (V, E_I)$, and $k$ available slots, the* Minimum Delay Beacon Scheduling (MDBS) problem *is to find an interference-free slot assignment $s(v)$ for each $v \in V$ such that the convergecast latency $L(G)$ is minimized.*

To prove that the MDBS problem is NP-complete, we define a decision problem as follows.

**Definition 5** *Given $G = (V, E)$, $G$'s interference graph $G_I = (V, E_I)$, $k$ available slots, and a delay constraint $d$, the* Bounded Delay Beacon Scheduling (BDBS) problem *is to decide if there exists an interference-free slot assignment $s(v)$ for each $v \in V$ such that the convergecast latency $L(G) \leq d$.*

**Theorem 2** *The BDBS problem is NP-complete.*

**Proof**. First, given slot assignments for nodes in $V$, we can find the report latency of each $v \in V$ by running a shortest path algorithm on $G_D$. We can then check if $L(G) \leq d$. Clearly, this takes polynomial time.

We then prove that the BDBS problem is NP-hard by reducing the 3 conjunctive normal form satisfiability (3-CNF-SAT) problem to a special case of the BDBS problem in polynomial time. Given any 3-CNF formula $C$, we will construct the corresponding $G$ and $G_I$. Then we show that $C$ is satisfiable *if and only if* there is a slot assignment for each $v \in V$ using no more than $k = 3$ slots such that $L(G) \leq 4$ slots.

Let $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where clause $C_j = x_{j,1} \vee x_{j,2} \vee x_{j,3}$, $1 \leq j \leq m$, $x_{j,i} \in \{X_1, X_2, ..., X_n\}$, and $X_i \in \{x_i, \bar{x}_i\}$, where $x_i$ is a binary variable, $1 \leq i \leq n$. We first construct $G$ from $C$ as follows:

1. For each clause $C_j$, $j = 1, 2, ..., m$, add a vertex $C_j$ in $G$.

2. For each literal $X_i$, $i = 1, 2, ..., n$, add four vertices $x_{i1}$, $x_{i2}$, $\bar{x}_{i1}$, and $\bar{x}_{i2}$ in $G$.

3. Add a vertex $t$ as the sink of $G$.

4. Add edges $(t, x_{i2})$ and $(t, \bar{x}_{i2})$ to $G$, for $i = 1, 2, ..., n$.

5. Add edges $(x_{i1}, x_{i2})$ and $(\bar{x}_{i1}, \bar{x}_{i2})$ to $G$, for $i = 1, 2, ..., n$.

6. For each $i = 1, 2, ..., n$ and each $j = 1, 2, ..., m$, add an edge $(C_j, x_{i1})$ (resp., $(C_j, \bar{x}_{i1})$) to $G$ if $x_i$ (resp., $\bar{x}_i$) appears in $C_j$.

Then we construct $G_I$ as follows.

1. Add all vertices and edges in $G$ into $G_I$.

2. Add edges $(x_{i1}, \bar{x}_{i1})$ and $(x_{i2}, \bar{x}_{i2})$ to $G_I$, for $i = 1, 2, ..., n$.

3. Add edges $(C_j, x_{i2})$ and $(C_j, \bar{x}_{i2})$ to $G_I$, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$.

Then we build a one-to-one mapping from each truth assignment of $C$ to a slot assignment of $G$. We establish the following mapping:

1. Set $s(t) = 0$.

2. Set $s(C_j) = 0$, $j = 1, 2, ..., m$.

3. Set $s(x_{i1}) = 1$ and $s(\bar{x}_{i2}) = 1$, $i = 1, 2, ..., n$, if $x_i$ is true; otherwise, set $s(x_{i1}) = 2$ and $s(\bar{x}_{i2}) = 2$.

4. Set $s(x_{i2}) = 1$ and $s(\bar{x}_{i1}) = 1$, $i = 1, 2, ..., n$, if $\bar{x}_i$ is true; otherwise, set $s(x_{i2}) = 2$ and $s(\bar{x}_{i1}) = 2$.

The above reduction can be computed in polynomial time. By the above reduction, vertices $x_{i1}$ or $\bar{x}_{i1}$, $i = 1, 2, ..., n$, that are assigned to slot 1 (resp. slot 2) will have a report latency of 2 (resp. 4) and vertices $x_{i2}$ or $\bar{x}_{i2}$, $i = 1, 2, ..., n$, that are assigned to slot 1 (resp. slot 2) will have a report latency of 2 (resp. 1). Hence, for those vertices $x_{i1}$, $\bar{x}_{i1}$, $x_{i2}$, and $\bar{x}_{i2}$, $i = 1, 2, ..., n$, the longest report latency will be 4.

To prove the *if* part, we need to show that if $C$ is satisfiable, there is a slot assignment such that $k = 3$ and $L(G) \leq 4$. Since $C$ satisfiable, there must exist an assignment such that each clause $C_j$, $j = 1, 2, ..., m$, is true. If a clause $C_j$ is true, at least one variable in

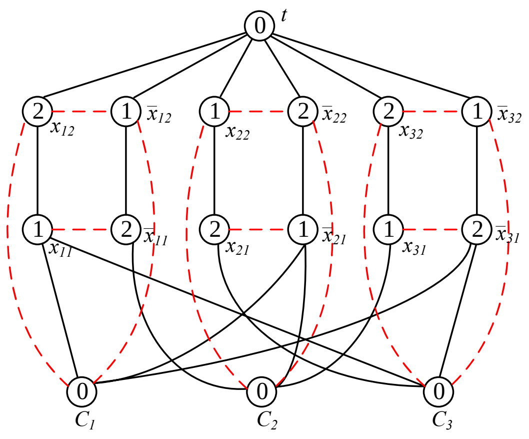


Figure 5.1: An example of reduction from the 3-CNF-SAT to the BDBS problem.

$C_j$ is true. According to the reduction, $C_j$ can always find an edge $(C_j, x_{i1})$ or $(C_j, \bar{x}_{i1})$ with $w((C_j, x_{i1})) = 1$ or $w((C_j, \bar{x}_{i1})) = 1$, where $i = 1, 2, ..., n$. Thus, when $C$ is satisfiable, the reporting latency for each clause is 3. This achieves $L(G) = 4$.

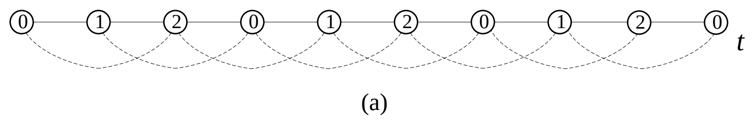
For the *only if* part, if each vertex $C_j$, $j = 1, 2, ..., m$, can find at least an edge with weight 1 to one of $x_{i1}$ and $\bar{x}_{i1}$, for $i = 1, 2, ..., n$, to achieve a report latency of 3, it must be that each clause has at least one variable to be true. So formula $C$ is satisfiable. Otherwise, the report latency of $C_j$, $j = 1, 2, ..., m$, will be 6. $\square$

For example, given $C = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$, Fig. 5.1 shows the corresponding $G$. The truth assignment $(x_1, x_2, x_3) = (T, F, T)$ makes $C$ satisfiable. According to the reduction and the mapping in the above proof, we can obtain the network $G$ and its slot assignment as shown in Fig. 5.1 such that $L(G) = 4$.

## 5.3 Algorithms for the MDBS Problem

### 5.3.1 Optimal Solutions for Special Cases

Optimal solutions can be found for the MDBS problem in polynomial time for regular linear networks and regular ring networks, as illustrated in Fig. 5.2. In such networks, each vertex is connected to one or two adjacent vertices and has an interference relation with each neighbor within $h$ hops from it, where $h \geq 2$. In a regular linear network, we assume that the sink $t$ is



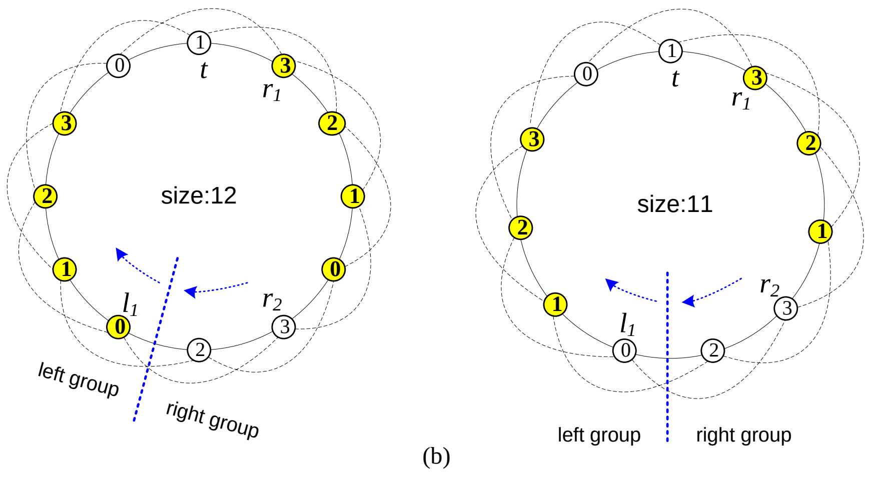


Figure 5.2: Examples of optimal slot assignments for regular linear and ring networks ($h = 2$). Dotted lines mean interference relations.

at one end of the network. Clearly, the maximum degree of $G_I$ is $2h$. We will show that an optimal solution can be found if the number of slots $k \geq h + 1$. The slot assignment can be done in a bottom-up manner. The bottom node is assigned to slot 0. Then, for each vertex $v$, $s(v) = (k' + 1) \bmod k$, where $k'$ is the slot assigned to $v$'s child.

**Theorem 3** *For a regular linear network, if $k \geq h + 1$, the above slot assignment achieves a report latency of $|V| - 1$, which is optimal.*

**Proof**. Clearly, the slot assignment is interference-free. Also the report latency of $|V| - 1$ is clearly the lower bound. □

For a regular ring network, we first partition vertices excluding $t$ into left and right groups as illustrated in Fig. 5.2(b) such that the left group consists of the sink node $t$ and $\lfloor \frac{|V|-1}{2} \rfloor$ other nodes counting counter-clockwise from $t$, and the right group consists of those $\lceil \frac{|V|-1}{2} \rceil$ nodes counting clockwise from $t$. Now we consider the ring as a spanning tree with $t$ as the root and left and right groups as two linear paths. Assuming that $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$ and $k \geq 2h$, the slot assignment works as follows:

1. The bottom node in the left group is assigned to slot 0.

2. All other nodes in the left group are assigned with slots in a bottom-up manner. For each node $v$ in the left group, we let $s(v) = (j + 1) \bmod k$, where $j$ is the slot of $v$'s child.

3. Nodes in the right group are assigned with slots in a top-down manner. For each node $v$ in the right group, we let $s(v) = (j - c) \bmod k$, where $j$ is the slot assigned to $v$'s parent and $c$ is the smallest constant ($1 \leq c \leq k$) that ensures that $s(v)$ is not used by any of its interference neighbors that have been assigned with slots.

It is not hard to prove the slot assignment is interference-free because nodes receives slots sequentially and we have avoided using the same slots among interfering neighbors. Although this is a greedy approach, we show that $c$ is equal to 1 in step 3 in most of the cases except when two special nodes are visited. This gives an asymptotically optimal algorithm, as proved in the following theorem.

**Theorem 4** *For a regular ring network, assuming that $k \geq 2h$ and $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$, the above slot assignment achieves a report latency $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$, which is optimal within a factor of 1.5.*

**Proof**. We first identify three nodes on the ring (refer to Fig. 5.2(b)):

- $l_1$: the bottom node in the left group.
- $r_1$: the first node in the right group.
- $r_2$: the node that is $h$ hops from $l_1$ counting counterclockwise.

The report latency of each node can be analyzed as follows. The parent of node $x$ is denoted by $par(x)$.

**A1.** For each node $v$ in the left group except the sink $t$, the latency from $v$ to $par(v)$ is 1.

**A2.** The latency from $r_1$ to $t$ is $h$.

**A3.** For each node $v$ next to $r_1$ in the right group but before $r_2$ (counting clockwise), the latency from $v$ to $par(v)$ is 1.

**A4.** The latency from $r_2$ to $par(r_2)$ is 1 if the ring size is even; otherwise, the latency is 2.

**A5.** For each node $v$ in the right group that is a descendant of $r_2$, the report latency from $v$ to $par(v)$ is 1.

It is not hard to prove that A1, A2, and A3 are true. To see A4 and A5, we make the following observations. The function $par^i(x)$ is to apply $i$ times the $par()$ function on node $x$. Note that $par^0(x)$ means $x$ itself.

**O1.** When the ring size is even, the equality $s(par^{i-1}(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, ..., \lfloor\frac{|V|-1}{2}\rfloor - h - 1$. More specifically, this means that (i) $l_1$ and $par(r_2)$ will receive the same slot, (ii) $par(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. This can be proved by induction by showing that the $i$-th descendant of $t$ in the right group will be assigned the same slot as the $(h + i - 1)$-th descendant of $t$ in the left group (the induction can go in a top-down manner). This property implies that when assigning a slot to $r_2$ in step 3, $c = 1$ in case that the ring size is even. Further, $r_2$ and its descendants will be sequentially assigned to slots $k - 1$, $k - 2$, ..., $k - h$, which implies that $c = 1$ when doing the assignments in step 3. So properties A4 and A5 hold for the case of an even ring.

**O2.** When the ring size is odd, the equality $s(par^i(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, ...,$ $\lfloor\frac{|V|-1}{2}\rfloor - h$. This means that (i) $par(l_1)$ and $par(r_2)$ will receive the same slot, and (ii) $par^2(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. Again, this can be proved by induction as in O1. This property implies that $c = 2$ when assigning a slot to $r_2$ in step 3, and $c = 1$ when assigning slots to descendants of $r_2$. So properties A4 and A5 hold for the case of an odd ring.

The equality of slot assignments pointed out in O1 and O2 is illustrated in Fig. 5.2(b) by those numbers in gray nodes. In summary, the report latency of the left group is $\lfloor\frac{|V|-1}{2}\rfloor$.

When the ring size is even, the report latency of the right group is the number of nodes in this group, $\frac{|V|}{2}$, plus the extra latency $h-1$ incurred at $r_1$. So $L(G) = \frac{|V|}{2} + h - 1 = \lfloor \frac{|V|-1}{2} \rfloor + h$. When the ring size is odd, the report latency of right group is the number of nodes in this group, $\frac{|V|-1}{2}$, plus the extra latency $h-1$ incurred at $r_1$ and the extra latency 1 incurred at $r_2$. So $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$.

A lower bound on the report latency of this problem is the maximum number of nodes in each group excluding $t$. Applying $\lfloor \frac{|V|-1}{2} \rfloor$ as a lower bound and using the fact that $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$, $L(G)$ will be smaller than $1.5 \times \lfloor \frac{|V|-1}{2} \rfloor$, which implies the algorithm is optimal within a factor of 1.5. Note that the condition $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$ is to guarantee that $t$ will not locate within $h$ hops from $r_2$. Otherwise, the observation O2 will not hold. $\square$

### 5.3.2 A Centralized Tree-Based Assignment Scheme

Given $G = (V, E)$, $G_I = (V, E_I)$, and $k$, we propose a centralized slot assignment heuristic algorithm. Our algorithm is composed of the following three phases:

**phase 1.** From $G$, we first construct a BFS tree $T$ rooted at sink $t$.

**phase 2.** We traverse vertices of $T$ in a bottom-up manner. For these vertices in depth $d$, we first sort them according to their degrees in $G_I$ in a descending order. Then we sequentially traverse these vertices in that order. For each vertex $v$ in depth $d$ visited, we compute a temporary slot number $t(v)$ for $v$ as follows.

1. If $v$ is a leaf node, we set $t(v)$ to the minimal non-negative integer $l$ such that for each vertex $u$ that has been visited and $(u, v) \in E_I$, ($t(u)$ mod $k$) $\neq l$.

2. If $v$ is an in-tree node, let $m$ be the maximum of the numbers that have been assigned to $v$'s children, i.e., $m = \max\{t(child(v))\}$, where $child(v)$ is the set of $v$'s children. We then set $t(v)$ to the minimal non-negative integer $l > m$ such that for each vertex $u$ that has been visited and $(u, v) \in E_I$, ($t(u)$ mod $k$) $\neq$ ($l$ mod $k$).

After every vertex $v$ is visited, we make the assignment $s(v) = t(v)$ mod $k$.

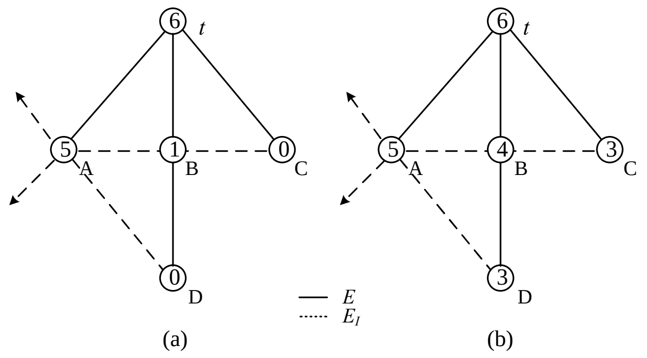


Figure 5.3: (a) Slot assignment after phase 2. (b) Slot compacting by phase 3.

**phase 3.** In this phase, vertices are traversed sequentially from $t$ in a top-down manner. When each vertex $v$ is visited, we try to greedily find a new slot $l$ such that ($s(par(v)) - l$) mod $k <$ ($s(par(v)) - s(v)$) mod $k$, such that $l \neq s(u)$ for each $(u, v) \in E_I$, if possible. Then we reassign $s(v) = l$.

Note that in phase 2, a node with a higher degree means that it has more interference neighbors, implying that it has less slots to use. Therefore, it has to be assigned to a slot earlier. Also note that, the number $t(v)$ is not a modulus number. However, in step 2 of phase 2, we did check that if $t(v)$ is converted to a slot number, no interference will occur. Intuitively, this is a temporary slot assignment that will incur the least latency to $v$'s children. At the end, $t(v)$ is converted to a slot assignment $s(v)$. Phase 3 is a greedy approach to further reduce the report latency of routers. For example, Fig. 5.3(a) shows the slot assignment after phase 2. Fig. 5.3(b) indicates that B, C, and D can find another slots and their report latencies are decreased. This phase can reduce $L(G)$ in some cases.

The computational complexity of this algorithm is analyzed below. In phase 1, the complexity of constructing a BFS tree is $O(|V| + |E|)$. In phase 2, the cost of sorting is at most $O(|V|^2)$ and the computational cost to compute $t(v)$ for each vertex $v$ is bounded by $O(kD_I)$, where $D_I$ is the degree of $G_I$. So the time complexity of phase 2 is $O(|V|^2 + kD_I|V|)$. Phase 3 performs a similar procedure as phase 2, so its time complexity is also $O(kD_I|V|)$. Overall, the time complexity is $O(|V|^2 + kD_I|V|)$.

### 5.3.3 A Distributed Assignment Scheme

In this section, we propose a distributed slot assignment algorithm. Each node has to compute its direct as well as indirect interference neighbors in a distributed manner. To achieve this, we will refer to the *heterogeneity* approach in [70], which adopts power control to achieve this goal. Assuming routers' default transmission range is $r$, interference neighbors must locate within range $2r$. From time-to-time, each router will boost its transmission power to double its default transmission range and send HELLO packets to its neighbor routers. Each HELLO packet further contains sender's 1) depth[1], 2) the location of outgoing superframe (i.e., slot), and 3) number of interference neighbors. Note that all other packets are transmitted by the default power level. When booting up, each router will broadcast HELLO packets claiming that its depth and slot are *NULL*. After joining the network and choosing a slot, the HELLO packets will carry the node's depth and slot information. The algorithm is triggered by the sink $t$ setting $s(t) = k - 1$ and then broadcasting its beacon. A router $v \neq t$ that receives a beacon will decide its slot as follows.

1. Node $v$ sends an association request to the beacon sender.
2. If $v$ fails to associate with the beacon sender, it stops the procedure and waits for other beacons.
3. If $v$ successfully associates with a parent node $par(v)$, it computes the smallest positive integer $l$ such that $(s(par(v)) - l)$ mod $k \neq s(u)$ for all $(u, v) \in E_I$ and $s(u) \neq$ *NULL*. Then $v$ chooses $s(v) = (s(par(v)) - l)$ mod $k$ as its slot.
4. Then, $v$ broadcasts HELLOs including its slot assignment $s(v)$ for a time period $t_{wait}$. If it finds that $s(v) = s(u)$ for any $(u, v) \in E_I$, $v$ has to change to a new slot if one of the following rules is satisfied and goes back to step 3.

   (a) Node $u$ has more interference neighbors than $v$.

[1]The depth of a node is the length of the tree path from the root to the node. The root node is at depth zero.

(b) Node $u$ and $v$ have the same number of interference neighbors but the depth of $u$ is lower than $v$, i.e. $u$ is closer to the sink than $v$.

(c) Node $u$ and $v$ have the same number of interference neighbors and they are at the same depth but the $u$'s ID is smaller than $v$'s.

5. After $t_{wait}$, $v$ can finalize its slot selection and broadcast its beacons.

In this distributed algorithm, slots are assigned to routers, ideally, in a top-down manner. However, due to transmission latency, some routers at lower levels may find slots earlier than those at higher levels. Also note that the time $t_{wait}$ is to avoid possible collision on slot assignments due to packet loss.

## 5.4 Simulation Results

This section presents our simulation results. We first assume that the size of sensory data is negligible and that all routers generate reports at the same time, and compare the performances of different convergecast algorithms. Then we simulate more realistic scenarios where the size of sensory data is not negligible and routers need to generate reports periodically or passively driven by events randomly appearing in certain regions in the sensing field. More specifically, sensors generate reports according to certain application specifications. Devices all run ZigBee and IEEE 802.15.4 protocols to communicate with each other. Routers can aggregate child sensors' reports and report to their parents directly. Each router has a fix-size buffer. When a router's buffer overflows, this router will not accept further incoming frames. We also measure the *goodput* of the network, which is defined as the ratio of sensors' reports successfully received by the sink. Some parameters used in our simulation are listed in Table 5.1.

### 5.4.1 Comparison of Different Convergecast Algorithms

We compare the proposed slot assignment algorithms against a random slot assignment (denoted by RAN) scheme and a greedy slot assignment (denoted by GDY) scheme. In RAN,

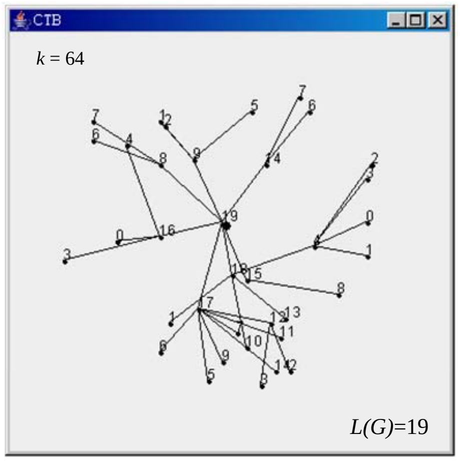
Table 5.1: Simulation parameters (for realistic convergecast scenarios).

| Parameter | Value |
|---|---|
| length of a frame's header and tail | 18 Bytes |
| length of a sensor's report | 16 Bytes |
| beacon length | 18 Bytes |
| maximum length of a frame | 127 Bytes |
| bit rate | 250k bps |
| symbol rate | 62.5k symbols/s |
| aBaseSuperframeDuration | 960 symbols |
| aUnitBackoffPeriod | 20 symbols |
| aCCATime | 8 symbols |
| macMinBE | 3 |
| aMaxBE | 5 |
| macMaxCSMABackoffs | 4 |
| maximum number of retransmissions | 3 |

the slot assignment starts from the sink and each router, after associating with a parent router, simply chooses any slot which has not been used by any of its interference neighbors. In GDY, routers are given a sequence number in a top-down manner. The sink sets its slot to $k-1$. Then the slot assignment continues in sequence. For a node $v$, it will try to find a slot $s(v) = s(u) - l \bmod k$, where $u$ is the predecessor of $v$ and $l$ is the smallest integer letting $s(v)$ is the slot which does not assign to any of $v$'s interference neighbors. In the simulations, routers are randomly distributed in a circular region of a radius $r$ and a sink is placed in the center. Our centralized tree-based scheme and distributed slot assignment scheme are denoted as CTB and DSA, respectively. We compare the report latency $L(G)$ (in terms of slots).

Fig. 5.4 shows some slot assignment results of CTB and DSA when $r = 35\ m$ and $k = 64$. Devices are randomly distributed. The transmission range of routers is set to $20\ m$. In this case, CTB performs better than DSA.

Next, we observe the impact of different $r$, $C_R$ (number of routers), and $T_R$ (transmission distance). Fig. 5.5(a) shows the impact of $r$ when $k = 64$, $T_R = 25\ m$, and $C_R = 3 \times (r/10)^2$. CTB performs the best. DSA performs slightly worse than CTB, but still significantly outperforms RAN and GDY. It can be seen that RAN and GRY could result in very long



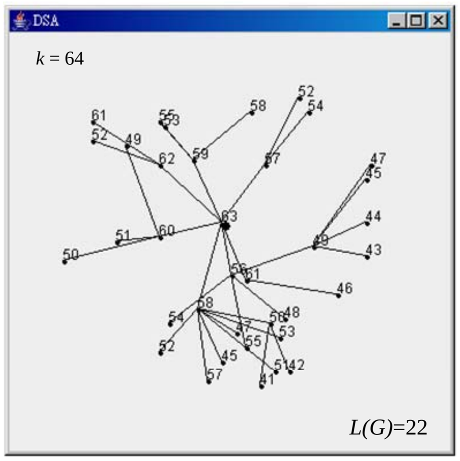
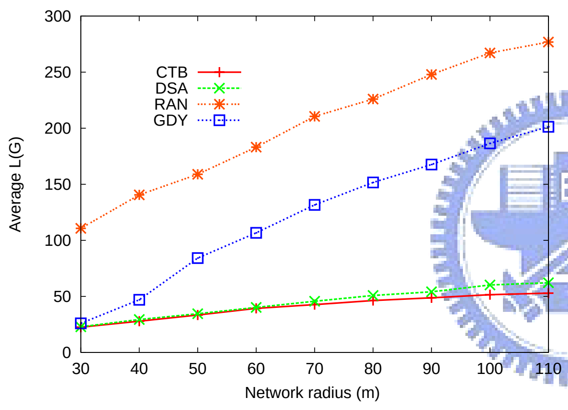
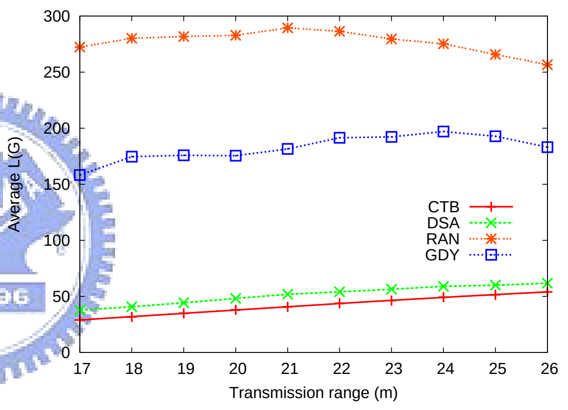


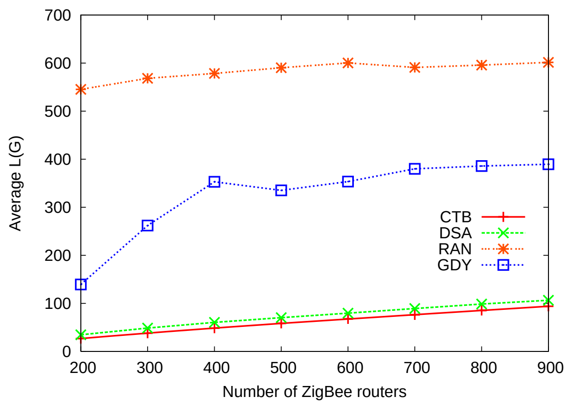
Figure 5.4: Slot assignment examples by CTB and DSA.



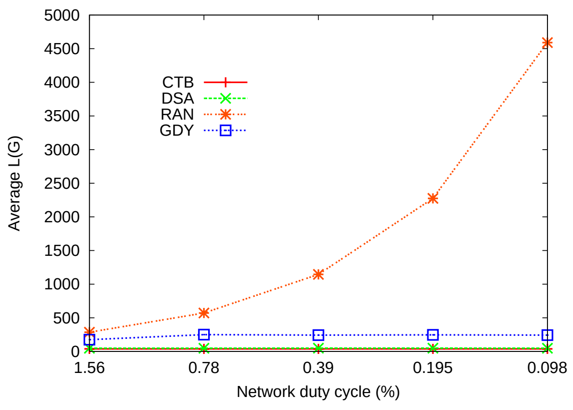
(a)



(b)



(c)



(d)

Figure 5.5: Comparison of report latencies under different configurations.

convergecast latency. Both CTB and DSA are quite insensitive to the network size. But this is not the case for RAN and GDY. Fig. 5.5(b) shows the impact of $T_R$ when $C_R = 300$, $r = 100\ m$, and $k = 64$. Since a larger transmission range implies higher interference among routers, the report latencies of CTB and DSA will increase linearly as $T_R$ increases. The report latency of RAN also increases when $T_R = 17 \sim 21\ m$ because of the increased interference. After $T_R \geq 22\ m$, the latency of RAN decreases because that the network diameter is reduced. Basically, GDY behaves the same as CTB and DSA. But when the transmission range is larger, the report latency slightly becomes small.

Fig. 5.5(c) shows the impact of $C_R$ when $r = 100\ m$, $T_R = 20\ m$, and $k = 128$. As a larger $C_R$ means a higher network density and thus more interference, the report latencies of CTB and DSA increase as $C_R$ increases. Since the network diameter is bounded, the report latency of RAN is also bounded. GDY is sensitive to the number of routers when there are less routers. This is because that each router can own a slot and the report latency increases proportionally to the number of routers. With $r = 100\ m$, $C_R = 300$, and $T_R = 20\ m$, Fig. 5.5(d) shows the impact of routers' duty cycle. Note that a lower duty cycle means a larger number of available slots. Interestingly, we see that the report latencies of CTB, DSA, and GDY are independent of the number of slots. Contrarily, with a random assignment, RAN even incurs a higher report latency as there are more freedom in slot selection.

### 5.4.2 Periodical Reporting Scenarios

Next, we assume that sensors are instructed to report their data in a periodically manner. We set $r = 100\ m$, $T_R = 20\ m$, and $C_R = 300$ with 6000 randomly placed sensors associated to these routers, and we further restrict a router can accept at most 30 sensors. $BO - SO$ is fixed to six, so $k = 2^{BO-SO} = 64$. Since the earlier simulations show that CTB and DSA perform quite close, we will use only CTB to assign routers' slots. Sensors are required to generate a report every 251.66 second (the length of one beacon interval when $BO = 14$). We set the

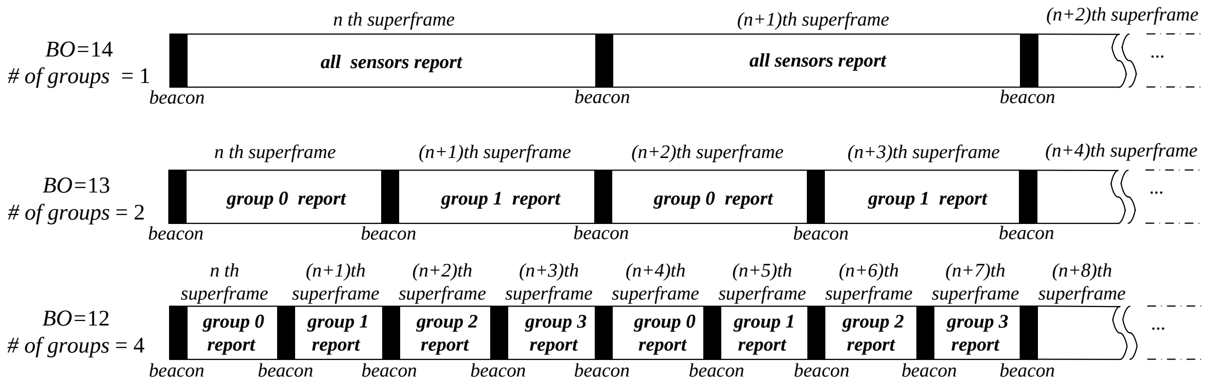


Figure 5.6: An example of report scheduling under different values of BO.

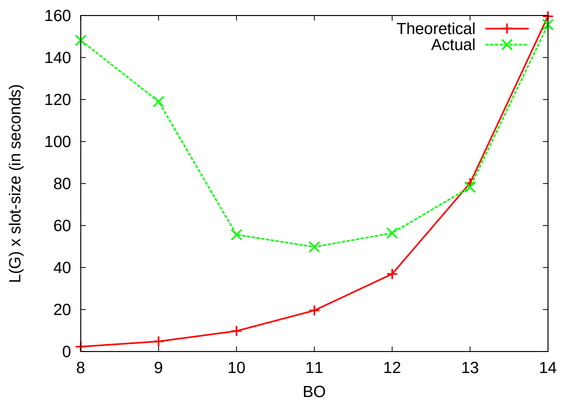
buffer size of each router is 10 KB.[2] We allocate two mini-slots for each child router of the sink as the GTS slot. [3]

Since $(BO - SO)$ is fixed, a small $BO$ implies a smaller slot size (and thus a smaller unit size of $L(G)$). So, a smaller slot size seemingly implies higher contention among sensors if they all intend to report to their parents simultaneously. In fact, a smaller $BO$ does not hurt the overall reporting times of sensors if we can properly divide sensors into groups. For example, in Fig. 5.6, when $BO = 14$, all sensors of a router can report in every superframe. When $BO = 13$, if we divide sensors into two groups, then they can report alternately in odd and even superframes. Similarly, when $BO = 12$, four groups of sensors can report alternately. Since the length of superframes are reduced proportionally, the report intervals of sensors actually remain the same in these cases. In the following experiments, we groups sensors according to their parents' IDs. A sensor belongs to group $m$ if the modulus of its parent's ID is $m$.

Fig. 5.7 shows the theoretical and actual report latencies under different $BO$s. Note that a report may be delayed due to buffer constraint. As can be seen, the actual latency does not always favor a smaller $BO$. Our results show that $BO = 10 \sim 12$ performs better.

[2]Currently, there are some platforms which are equipped with larger RAMs. For example, Jennic JN5121 [9] has a 96KB RAM and CC2420DBK [2] has a 32KB RAM.

[3]There are sixteen mini-slots per active portion (slot).



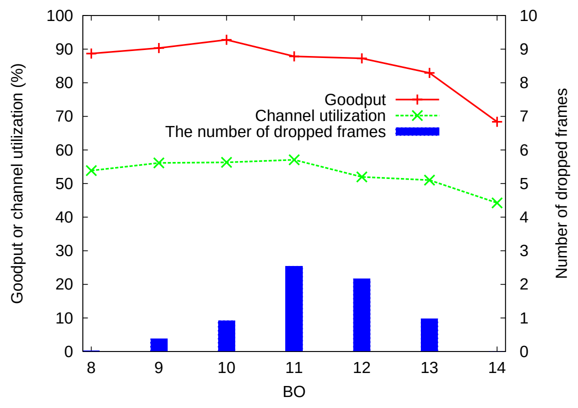


Figure 5.7: Simulations considering buffer limitation and contention effects: (a) theoretical v.s. actual report latencies and (b) goodput, channel utilization, and number of dropped frames.

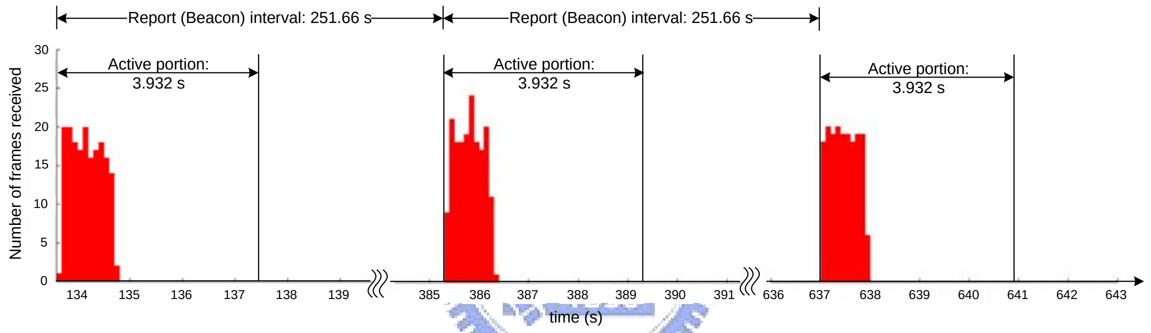
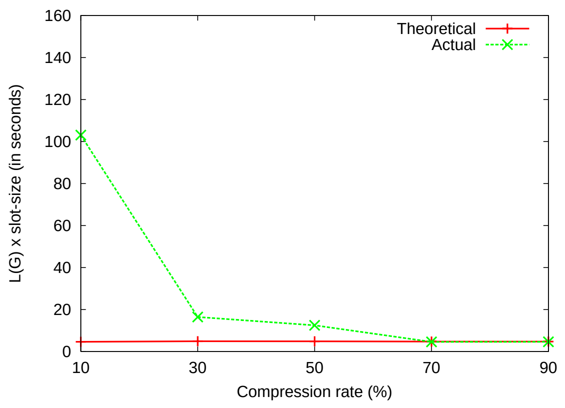


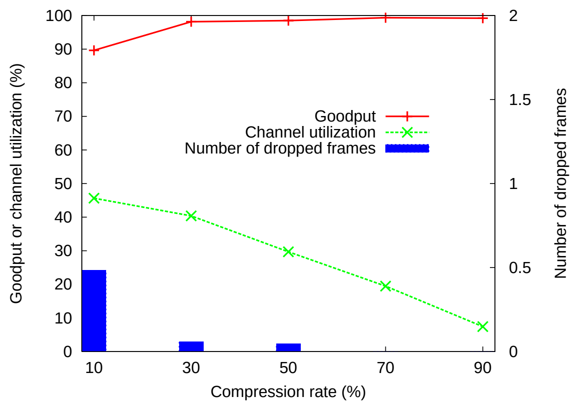
Figure 5.8: A log of the number of frames received by a sink's child router when $BO = 14$.

Fig. 5.7(b) shows the goodput of sensory reports, channel utilization at the sink, and the number of dropped frames at the sink. When $BO = 14$, although there is no frames being dropped at the sink, the goodput is still low. This is because a lot of collisions happen inside the network, causing many sensory reports being dropped at intermediate levels (a frame is dropped after exceeding its retransmission limit). Fig. 5.8 shows a log of the numbers of frames received by a sink's child router when $BO = 14$. We can see that more than half of the active portion is wasted. Overall, $BO = 10$ produces the best goodput and a shorter report latency.

Some previous works can be also integrated in this periodical reporting scenario, such as the adaptive GTS allocation mechanism in [36] and the aggregation algorithms for WSNs in



(a)



(b)

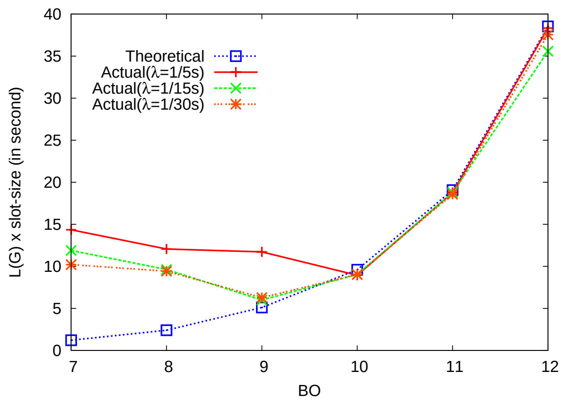
Figure 5.9: Simulations considering data compression: (a) theoretical v.s. actual report latencies and (b) goodput, channel utilization, and number of dropped frames.

[19][31]. Fig. 5.9 shows an experiment that routers can compress reports from sensors with a rate $cr$ when $BO = 10$. If a router receives $n$ reports and each report's size is 16 Bytes (as in Table 5.1), it can compress the size to $16 \times n \times (1 - cr)$. The report latencies decrease when the $cr$ becomes larger. By compressing the report data, the goodput can up to 98% and the report can arrive to the sink more quickly.

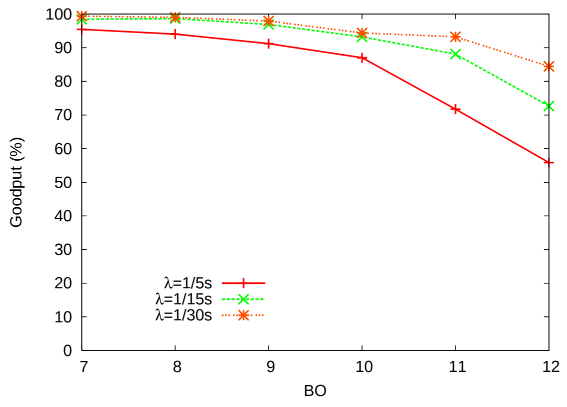
### 5.4.3 Event-Driven Reporting Scenarios

In the following, we assume that sensors' reporting activities are triggered by events occurred at random locations in the network with a rate $\lambda$. The sensing range of each sensors is 3 meters and each event is a disk of a radius of 5 meters. A sensor can detect an event if its sensing range overlaps with the disk of that event. Each router has an 1 KB buffer. When a sensor detects an event, it only tries to report that event once. All other settings are the same as those in Section 5.4.2.

Fig. 5.10 shows the simulation results when $\lambda = 1/5s, 1/15s,$ and $1/30s$. From Fig. 5.10(a), we can observe that when $BO$ is small, the report latency can not achieve to the theoretical value. This is because that an active portion is too small to accommodate all reports from sensors, thus lengthening the report latency. When $BO$ becomes larger, the theoretical and



(a)



(b)

Figure 5.10: Simulation results of event-driven scenarios: (a) theoretical v.s. actual report latencies and (b) goodput.

actual curves would meet. However, the good put will degrade, as shown in Fig. 5.10(b). This is because reports are likely to be dropped due to buffer overflow. How to determine a proper $BO$, which can contain most of the reports and guarantee low latency, is an important design issue for such scenarios.

# Chapter 6

# An Emergency Guiding and Monitoring System by ZigBee WSNs

## 6.1 System Overview

In this work, we design a novel emergency service that aims to guide people to safe places when emergencies happen. At normal time, the network is responsible for monitoring the environment in low-power mode. When emergency events are detected, all sensors switch to active mode to deal with these events. And the network can adaptively modify its topology to ensure transportation reliability, quickly identify hazardous regions that should be avoided, and find safe navigation paths that can lead people to exits. We adopt the idea in TORA [54] to develop our protocol. TORA is a distributed multi-path routing algorithm for mobile ad hoc networks. In TORA, each mobile node is assigned a *temporal-order* sequence number to support multi-path routing from a source to a specific destination node. TORA expresses the sequence number as a quintuple. To handle mobility, TORA adopts a *link reversal* procedure when some hosts lose all their outgoing paths. The concept of multi-path routing in TORA fits well to our needs in emergency navigation services. However, TORA cannot be directly applied to our environment due to several reasons. First, TORA involves a quintuple to express a node's weight. This may be too costly for sensors with weak communication capability. Second, TORA looks for shorter and multi-path routes, while our navigation service looks for safer, but not necessarily shorter, escape paths. Third, there are different considerations for users located at hazardous and non-hazardous regions. Our navigation service is essentially like an all-to-many routing (from all locations to one or multiple exits), and emergency locations will disturb the discovery of safe paths.

In our scheme, each sensor is assigned an *altitude* which can be seen as a degree of danger. Sensors near the exits will be assigned smaller altitudes, and sensors near the emergency locations will be assigned higher altitudes. The escape paths to exits are along sensors with higher attitudes to those with lower attitudes. Initially, each sensor is assigned an altitude according to its distance to the nearest exit. When emergency situations happen, sensors within a certain distance from emergency locations will form hazardous regions by raising their altitudes. After the above step, local-minimum sensors have to re-compute their altitudes to find ways out. The *link reversal* concept in TORA is used to solve this problem. In particular, for quick convergence, we use the variation of neighboring sensors' altitudes to increase a local-minimum sensor's altitude. The algorithm converges as long as all sensors (excluding exit sensors located on exits) finds their ways out. Our simulation and implementation results show that the proposed scheme can achieve the goals of navigation safety and quick convergence.

Reference [44] has a similar goal as our work. In [44], it is assumed that there are multiple emergency points (called *obstacles*) and one exit in the environment. The goal is to find a navigation path from each sensor to the exit without passing any obstacle. The concept of *artificial potential* is used. The exit will generate an *attractive potential*, which pulls sensors to the exit, and each obstacle will generate a *repulsive potential*, which pushes sensors away from it. Each sensor will calculate its potential value and tries to find a navigation path with the least total potential value. Although the algorithm in [44] is shown to be able to find a shorter and safer path from each sensor to the exit, it has the following drawbacks. First, it may incur high message overheads. Since the construction is rippled from the exit to other sensors, a minor change of potential nearby the exit may cause many sensors to recompute their potentials, thus causing a lot of message exchanges and even delays in making the navigation decision. Second, the algorithm has no concept of hazardous regions. With shortest-path routing, this algorithm may determine a path that is very close to the emergency location.

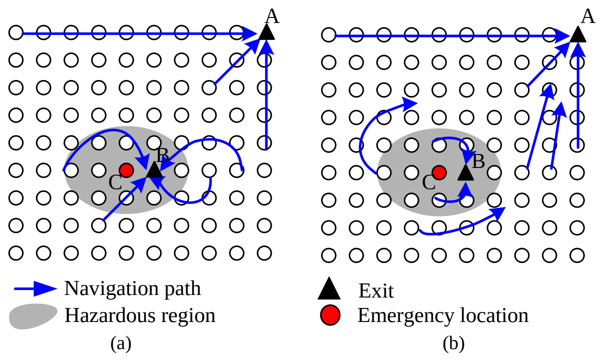


Figure 6.1: Some navigation scenarios when the hazardous region is defined as two hops from the emergency size.

Consider Fig. 6.1(a), where there are two exits, A and B. When an emergency is detected in C, according to [44], some users may be directed to B, which is undesirable because they will pass the hazardous region. Guiding users as in Fig. 6.1(b) will be more desirable because only users inside the hazardous region are directed to exit B.

## 6.2 Network and Guidance Initialization

We are given a set of sensors deployed in a building. Sensors' roles are designated at the deployment stage. Sensors located at the exits of the building are called *exit sensors*, and those located at stairs are called *stair sensors*. Otherwise, they are called *normal sensors*. One sensor is designated as the *sink*, which is connected to the control host.

From the network, we will construct a *communication graph* $G_c = (V, E_c)$ and a *guidance graph* $G_g = (V, E_g)$, where $V$ is the set of sensors. Each edge $(u, v) \in E_c$ represents a communication link between $u$ and $v \in V$, while each edge $(u, v) \in E_g$ represents a walking path between $u$ and $v$. Note that a walking path is a physical route that human can pass. So $E_g$ has to be constructed manually based on the floor plane of the building. Fig. 6.2 shows this concept. In the following, the network and guidance initialization procedures are presented.

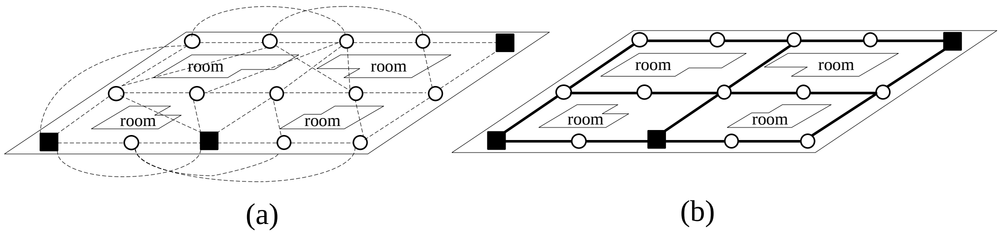


Figure 6.2: (a) Communication graph $G_c$ and (b) guidance graph $G_g$.

## 6.2.1 Network Initialization

The purpose of network initialization is to construct a reliable spanning tree rooted at the sink for reporting purpose. Before establishing the reporting tree, each sensor periodically sends *HELLO* packets including its *ID*, *parent* (initiate to itself), and *hop_count* to the sink (initiate to infinite). A communication link $(u, v)$ is established only if $u$ receives $v$'s *HELLO* including $u$ as its neighbor and the *HELLO*'s signal quality is above a threshold. Each sensor will maintain a neighbor table based on this rule. Note that the signal quality should be the average of several packets. This design is to take the unreliability problem in most short-distance radio systems into consideration. Network initialization is started by the control host flooding an *INIT_N* packet. A node that receives an *INIT_N* selects a set of neighbors with the smallest hop count to the sink and then chooses a neighbor with the best signal quality as its parent. Then this node will rebroadcast the *INIT_N* if it changes its parent. As a result, a minimum spanning tree is formed and dynamically maintained by periodic *HELLO* packets. In our design, sensors also report their neighbor information. The control host can know the $G_c$.

## 6.2.2 Guidance Initialization

The purpose of guidance initialization is to find escape paths leading to exits at normal times on the graph $G_g$.

After planning $G_g$, we will compute for each sensor an altitude according to its hop distance on $G_g$ to the nearest exit. Sensors locating at exits are called *exit sensors*, which will broadcast *INIT_G* packets to start this phase. An *INIT_G* packet is composed of three fields: sender_ID, exit_sensor_ID, and hop_count. In exit sensors' broadcasts, the hop_count field is set to zero. A sensor receiving an *INIT_G* packet from its guidance neighbor should increment the hop_count by one and accept this value as its initial altitude unless it has a smaller altitude. Then it rebroadcasts the *INIT_G* packet with the updated hop_count. After each sensor had its altitude, the initialization phase completes. From time to time, exit sensors have to restart the initialization phase to take care of possible topology changes. Through the above process, each sensor also keeps a guidance neighbor table, in which each entry is of the format <*neighbor_ID, is_exit, altitude*> to keep track of each guidance neighbor's status.

## 6.3 Emergency Guiding and Monitoring Schemes

In this section, we introduce emergency guiding protocol and tree reconstruction protocol for emergency guiding and monitoring.

### 6.3.1 Emergency Guiding Protocol

Our design emphasizes on the correctness in discovering escape paths even if passing hazardous areas is inevitable. When emergencies happen, sensors will update their altitudes in response to these events. Sensors near the emergency locations will raise their altitudes to form hazardous regions. Our protocol will avoid guiding users passing the hazardous regions, if possible. After hazardous regions are formed, some sensors may have local minimum altitudes. The *partial link reversal* concept in [54] will be used to solve this problem. Navigation is done by simply following a sequence of sensors with decreasing altitudes. Below, we first introduce some notations.

- $D$: a constant such that any sensor whose distance to any emergency location is less than or equal to this value is considered within a hazardous region. In this paper, we use hop count to calculate the distance.
- $A_{emg}$: a large constant to be assigned to a sensor that detects an emergency event.

- $A_i$: the altitude of sensor $i$.
- $I_i$: the altitude of sensor $i$ obtained in the initialization phase.
- $e_{i,j}$: the hop count from an emergency sensor $i$ to a sensor $j$.
- EMG packet: the emergency notification packet, which has five fields: (1) event sequence number, (2) ID of the sensor which finds the emergency event, (3) sender's ID, (4) altitude of the sender, and (5) hop count from the sender to the emergency sensor.

Let's assume that a sensor $x$ detects an emergency. It will set its altitude to $A_{emg}$ and immediately broadcast an EMG($seq$, $x$, $x$, $A_{emg}$, 0) packet. The packet will be flooded in $G_g$. The following rules summarize the actions to be taken when a sensor $y$ in $G_g$ receives from a sensor $w$ an EMG($seq$, $x$, $w$, $A_w$, $h$) packet originated from $x$.

1. $y$ judges if this is a new emergency by checking the tuple ($seq$, $x$).

   (a) If this is a new emergency event to $y$, $y$ records this event and sets $e_{x,y}$ to $h+1$.

   (b) Otherwise, $y$ checks if $h+1 < e_{x,y}$. If so, $y$ changes $e_{x,y}$ to $h+1$.

   Then $y$ records $w$'s altitude ($A_w$) in its neighbor table. Moreover, if $w = x$ and $x$ is an exit sensor, $y$ should clear the flag *is_exit* in the entry for $x$ in its neighbor table to avoid guiding users into this emergency location.

2. If $e_{x,y}$ was changed in step 1 and $e_{x,y} \leq D$, $y$ considers itself within the hazardous region formed by sensor $x$. Then $y$ re-calculates its altitude as follows:

$$A_y = \max\{A_y, A_{emg} \times \frac{1}{e_{x,y}^2} + I_y\}. \qquad (6.1)$$

   In our design, the altitude of a sensor inside a hazardous region is increased by an amount inversely proportionate to the square of its distance to the emergency location. The value $I_y$ is included because we intend to reflect $y$'s distance to its nearest exit. The $max$ function is to take into account that $y$ may be located within multiple hazardous regions and thus may receive multiple EMG packets from different sources. In this case, the new altitude of $y$ should reflect its distance to the nearest emergency location.

3. Sensor $y$ has to check if it has a local minimum altitude, unless $y$ is an exit sensor. If $y$ is a local minimum (i.e., its altitude is less than all its neighbors'), it adjusts its altitude as follows:

$$A_y = STA(A_{N_y}) \times \frac{1}{|N_y|} + \min\{A_{N_y}\} + \delta, \tag{6.2}$$

where $N_y$ is the set of all neighbors of $y$, $STA(A_{N_y})$ is the standard deviation of the altitudes of sensors in $N_y$, and $\delta$ is a small constant. The basic idea of using standard deviation is for quick response to emergency situations. When altitudes of sensors in $N_y$ vary significantly, it is likely that $y$ is near a hazardous region. Then it should increase its altitude more quickly to avoid becoming a local minimum again. The fixed constant $\delta$ is to guarantee convergency. Its value should be carefully chosen because a large $\delta$ may easily guide sensors to cross hazardous regions. On the other hand, a small $\delta$ may cost too many message exchanges although it may help find safer paths. The reciprocal of $|N_y|$ is to reflect the number of possible choices that a sensor has to select escape directions. A sensor that has less neighbors will increase its altitude in a faster manner to get away from its local minimum situation. These designs will speed up the convergency time of our algorithm. Also note that each sensor has to keep on going back to this step to check if it becomes a local minimum again.

4. Finally, $y$ broadcasts an EMG($seq$, $x$, $y$, $A_y$, $e_{x,y}$) packet if any of the following conditions is true:

   (a) This is a new emergency packet heard by $y$.

   (b) Sensor $y$ has changed $A_y$ or $e_{x,y}$ in the previous steps.

We remark that the above step 3 adopts the concept of *partial reversal* to adjust local minimum nodes' altitudes. We do not adopt the *full reversal* approach in our design because it may easily guide users in a non-hazardous region to pass through a hazardous region even unnecessarily. Using partial reversal can help guide users to route around a hazardous region. This will be justified by our simulation.

Finally, we discuss how a sensor chooses its escape direction when emergencies happen. When a user is inside a hazardous region and there is an exit sensor nearby, we may guide users either to this exit or to other exits in non-hazardous regions. In our work, we choose a hybrid approach. Sensors inside hazardous regions can choose an exit sensor that is also in hazardous regions if the former is within one-hop from the latter. However, sensors in non-hazardous regions will never choose an exit inside hazardous regions unless it is surrounded by hazardous regions or there are not proper exits in safe areas. If this case, sensors will keep on increasing their altitudes until reaching a level higher than those of sensors in hazardous regions. We summarize the escape rules for any sensor $y$ as follows:

1. If $y$ is in hazardous regions and it sees an exit sensor which is in $N_y$ and which is also in hazardous regions, then $y$ chooses this exit sensor.
2. In all other cases, $y$ directs users to its neighboring sensor which has the lowest altitude.

We claim that as long as there exists at least one exit sensor which is not located in an emergency location, the protocol can find an escape path for each non-exit sensor in a finite number of steps. To prove this, observe that disregarding exit sensors, a sensor has no escape path only if it is a local minimum. Since $\delta$ is a non-zero constant, the protocol has a progress property in the sense that the number of sensors which have no escape paths will reduce. So this protocol will converge.

Finally, we comment on the value of $A_{emg}$, which will affect the navigation results. A value that is too small may result in altitudes at the boundaries of hazardous regions that are smaller than some sensors' initial altitudes. To avoid this problem, assuming that the maximum altitude in initial phase is $MAX_{ini}$, the value of $A_{emg}$ should be at least larger than $MAX_{ini} \times (D+1)^2$.

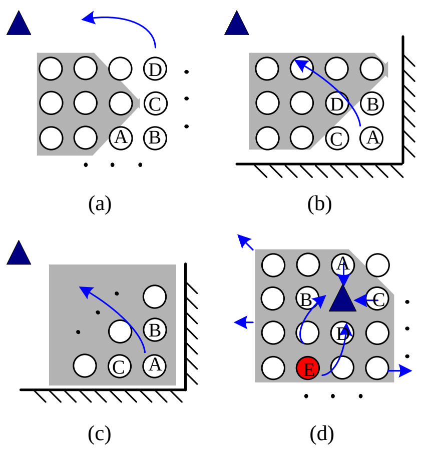


Figure 6.3: Some navigation examples of our algorithm.

### Some Navigation Examples

In the following, we show some typical navigation scenarios in our scheme. Fig. 6.3(a) represents the scenario where hazardous regions do not form a closed region. After forming hazardous regions, sensors A, B, and C may temporarily become a local minimum. However, suppose that sensor D already found an escape path. Later on, A, B and C will eventually find their escape paths via D. In Fig. 6.3(b), sensors A, B, and C are surrounded by a hazardous region. In this scenario, these three sensors should raise their altitudes to a level higher than the altitude of at least one sensor in the hazardous region. Assume that sensor D has the smallest altitude in the hazardous region. With a proper $\delta$, our algorithm will likely to guide users via D in most cases. Fig. 6.3(c) is similar to the previous case except that sensors A, B and C are all inside the hazardous region. So the escape paths for these sensors would be similar. In Fig. 6.3(d), there is an exit sensor in the hazardous region. Sensors A, B, C, and D, which are direct neighbors of the exit sensor, will guide users to that exit. Sensors that are not direct neighbors of the exit will guide users to leave the hazardous region via shortest paths first, and then to other exits outside the hazardous region, unless there are no such exits.

Fig. 6.4 shows how altitudes change in a 7$\times$7 grid network with $D$=2. Three emergency events occur in coordinates (S2,4), (S6,7), and (S5,2), in that order. An exit is located in (S1, 7). Changes of altitudes are shown in both side views and top views. Altitudes are expressed

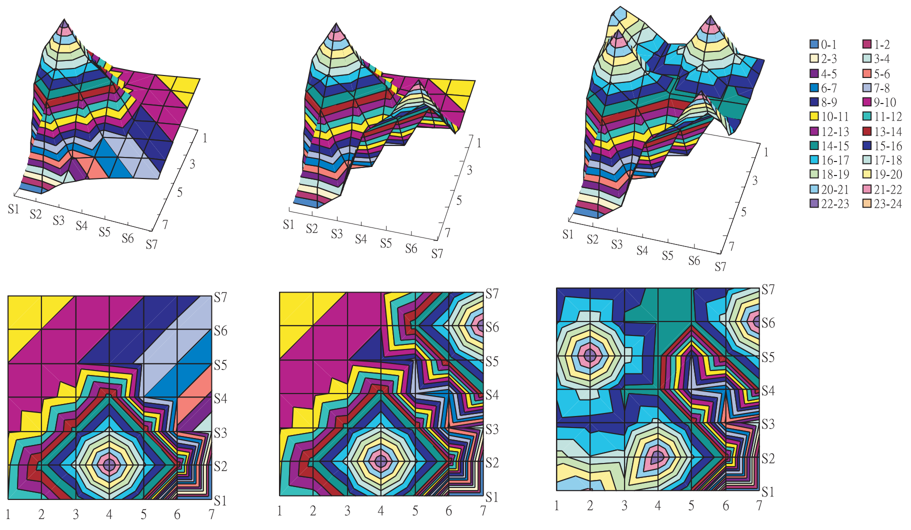


Figure 6.4: Examples of altitude changes when three emergency events occur in coordinates (S2, 4), (S6, 7), and (S5, 2).

in dB. Navigation paths are from sensors with higher altitudes to sensors with lower altitudes.

### 6.3.2 Tree Reconstruction Protocol

In the following, we introduce a tree reconstruction protocol to support emergency monitoring. Emergencies are usually accompanied by damage to communication links, so this protocol is triggered when the reporting tree in $G_c$ is broken.

The protocol works in a distributed manner. When a sensor $x$ loses its parent by receiving a *HELLO* with a larger *hop_count* than its current record or an emergency announcement *EMG* from its parent, $x$ sets *NO_PARENT* = *true* and executes the following steps:

1. Check its neighbor table to find another sensor, say $y$, with a hop count smaller than or equal to that of its original parent.

   (a) If $y$ exists, $x$ sets $y$ as its parent. If multiple candidates exist, the one with the best signal quality is chosen. Then, go to step 2.

   (b) Otherwise, $x$ deletes all its children in its neighbor table. If $x$'s neighbor table is still non-empty, it chooses a neighbor with the smallest hop count as its parent and goes to step 2. Otherwise, it goes to step 3.

2. Broadcast a *HELLO* packet with its new *hop_count* and *parent*, sets *NO_PARENT=false*, and ends the procedure.

3. Set its hop count to infinity and broadcasts a *HELLO* packet with *hop_count* $= \infty$. Then $x$ waits for *HELLO* packets. Any *HELLO* with a finite *hop_count* will cause $x$ to choose the sender as its parent. Then go back to step 2.

The above reconstruction protocol is for quick recovery by avoiding cycles. Step 1(a) is to choose a new parent with at least the same hop count as its original one. Step 1(b) is to find a new parent in other subtrees. Both steps are to guarantee that no internal loops are formed. When there are multiple emergencies, two sensors may set each other as their parents. This can be resolved when an up-to-date *HELLO* is received. Although *HELLO* packets may suffer from loss, up-to-date *HELLO*s will remove temporary cycles.

## 6.4 Simulation Results

This section presents our simulation results. We first consider a 10×10 grid networks. Each sensor has four navigation links to neighboring sensors on its east, west, north, and south. $A_{emg}$ is set to 200 and $\delta$ is set to 0.1. We compare our algorithm with the one in [44]. We use packet count and convergence time as performance metrics. Note that the packet count does not include packets used during the initialization phase. An unslotted CSMA/CA protocol following the IEEE 802.15.4 [37] is simulated with a data rate of 20 kbps. The convergence time is measured in *ms*.

Fig. 6.5 shows our simulation results. In case 1, the sensor located in the middle of the network detects an emergency event. However, this emergency event does not change the relative altitudes of neighboring sensors. So our algorithm only spends very few messages and quickly converges. In case 2, the placement of sensors is the same as the first case, except

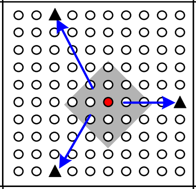
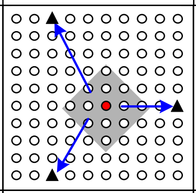
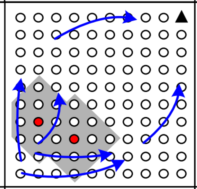
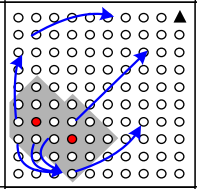
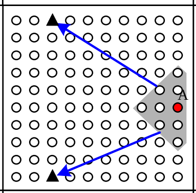
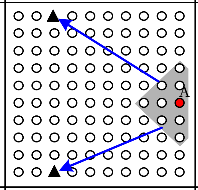
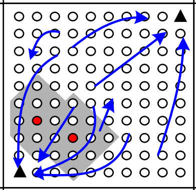
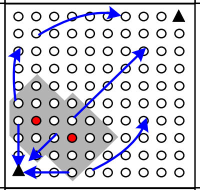
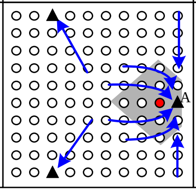
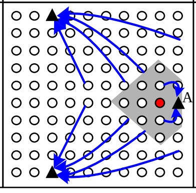
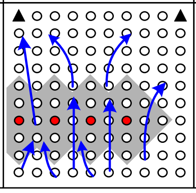
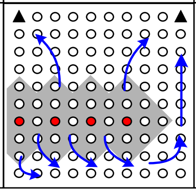
| No | Method of Li et al. | | Our method (*D*=2) | | No | Method of Li et al. | | Our method (*D*=2) | |
|---|---|---|---|---|---|---|---|---|---|
| | Path | pkt. count/ cnvg. time | Path | pkt. count/ cnvg. time | | Path | pkt. count/ cnvg. time | Path | pkt. count/ cnvg. time |
| 1 |  | 660 / 350.62 |  | 100 / 59.95 | 4 |  | 979 / 468.83 |  | 252 / 78.99 |
| 2 |  | 1215 / 530.81 |  | 130 / 87.5 | 5 |  | 731 / 372.0 |  | 264 / 107.15 |
| 3 |  | 742 / 442.52 |  | 137 / 87.89 | 6 |  | 1254 / 350.1 |  | 408 / 67.29 |

Figure 6.5: Comparison of packet count and convergence time (in $ms$) in a 10×10 grid network.

that an emergency is detected by the exit sensor A. Although both algorithms will compute the same navigation paths, the algorithm in [44] will incur more messages, because sensors near A will first be attracted to, and then repelled from, A. In case 3, an emergency event is detected near exit A. As shown in the figure, without the concept of hazardous region, [44] will guide some users to pass through the hazardous region, which is undesirable. Our algorithm can effectively avoid guiding users through the hazardous region. In case 4, some sensors are bounded by hazardous regions. Although guiding users through hazardous regions is inevitable, our scheme will choose paths that are as farther away from emergency locations as possible. In case 5, we add one more exit in the lower-left corner. The algorithm in [44] will direct some sensors to pass the hazardous regions to reach that exit, but the problem can be avoided in our algorithm. Case 6 shows a scenario that the network is almost partitioned by emergencies. Again, we see that the navigation paths discovered by our algorithm are safer than what are discovered by [44].

Fig. 6.6 illustrates our navigation results in networks with various forms. As can be seen,

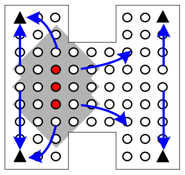
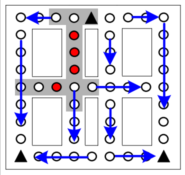
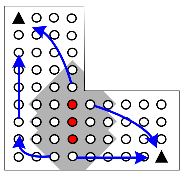
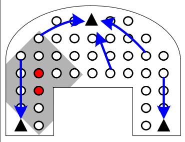
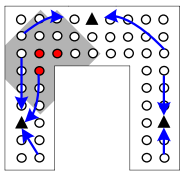
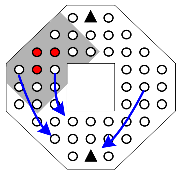
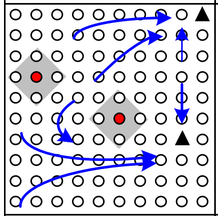
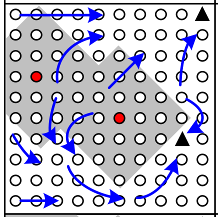
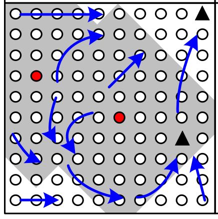
| No | Path | pkt. count/ cnvg. time | No | Path | pkt. count/ cnvg. time |
|---|---|---|---|---|---|
| 1 |  | 205 / 79.6 | 4 |  | 309 / 286.6 |
| 2 |  | 164 / 107.9 | 5 |  | 83 / 72.2 |
| 3 |  | 148 / 93.6 | 6 |  | 94 / 78.9 |

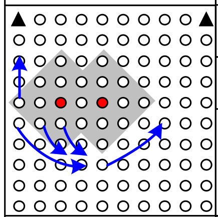
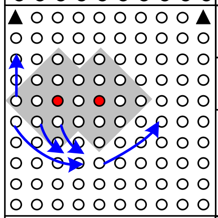
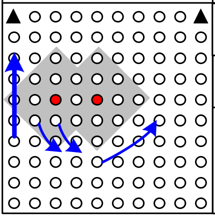
Figure 6.6: Navigation results in various forms of networks.

our scheme can effectively lead people to exits and avoid hazardous regions. These results also imply that our protocol can be applied to variable forms of buildings. We have also simulated our algorithm in a large-scale sensor network with 2500 sensor nodes. There are 1% to 5% of random sensors being selected as exits, and 1% of random sensors as emergency points. The parameter $D$ is set to 5. The convergence times of 1%, 2%, 3%, 4%, and 5% exit sensors are 21.6$s$, 10$s$, 9.1$s$, 2.9$s$, and 2.0$s$, respectively. This result demonstrates that our algorithm is quite scalable when applying to large networks.

*D* is an important parameter in our algorithm, which is used to form hazardous regions. While the value of $D$ is to reflect the dangerous range affected by an emergency event, its value may also affect the navigation results and system performance. For a small network, a *D* that is too large is meaningless, because a few emergency events may result in a network which is all covered by hazardous regions. Fig. 6.7(a) shows different settings of *D* in a 10$\times$10 grid network. A small $D$ may result in users being guided via paths close to emergency sources. On the contrary, a large *D* can help find safer paths, but the message overhead also increases. Fig. 6.7(b) shows the effects of $\delta$ and $A_{emg}$ on the quality of escape paths and

| Network( 10x10 ) | $D$ | Pkt. count |
|---|---|---|
|  | 1 | 204 |
|  | 3 | 211 |
|  | 5 | 398 |

(a)

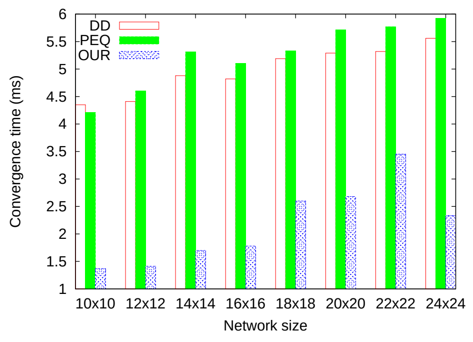
| Network( 10x10 ) | $A_{emg}$= 150 | $A_{emg}$= 300 |
|---|---|---|
|  | $\delta$=0.3 | $\delta$=0.3 |
| | Pkt. count | Pkt. count |
| | 292 | 274 |
|  | $\delta$=1.1 | $\delta$=1.1 |
| | Pkt. count | Pkt. count |
| | 256 | 257 |
|  | $\delta$=2.0 | $\delta$=5.0 |
| | Pkt. count | Pkt. count |
| | 244 | 249 |

(b)

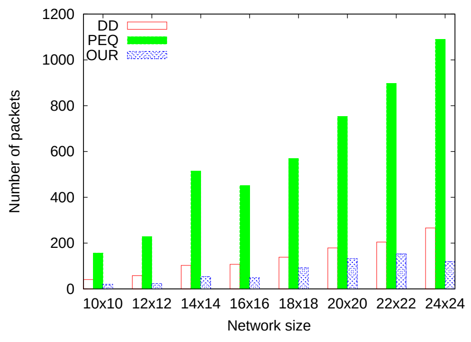
Figure 6.7: (a) The effect of $D$ on the quality of escaping paths and message overheads. (b) The effects of $\delta$ and $A_{emg}$ on the quality of escape paths and message overheads.

message overheads. We observe that using a smaller $A_{emg}$ with a larger $\delta$ may make it easier to guide users to cross hazardous regions (as the third case in Fig. 6.7(b)). This is because a larger $\delta$ may quickly increase the altitudes of sensors with local minimum to values larger than those a sensors in hazardous regions. We thus recommend to use a relatively larger $A_{emg}$ with a relatively smaller $\delta$. In our current design, these parameters, $D$, $A_{emg}$ and $\delta$, are configured at the deployment stage, and can be determined via simulations by manual involvement.

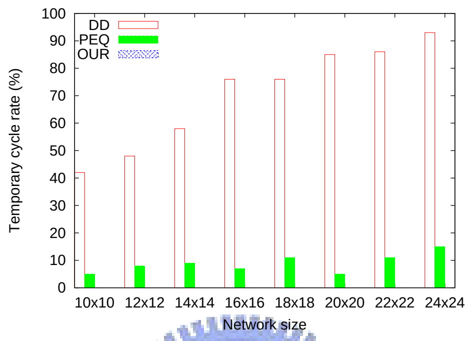
Next, we compare our emergency monitoring scheme against DD [39] and PEQ [23]. We consider a grid sensing field ranging from $10m \times 10m$ to $24m \times 24m$. In each $1m \times 1m$ grid, we deploy a sensor at a random location. Sensors' transmission distance ranges from 2 to 3 $m$. A sink is placed at the upper left corner of the network. We randomly generate 20% sensors as emergency nodes so as to trigger our tree reconstruction protocol, and observe the convergence time, number of packet exchanges, and number of temporary cycles. Each result is the average of 100 simulations. Assuming perfect channels, Fig. 6.8(a) compares the



(a)



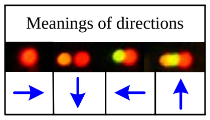
(b)



(c)

Figure 6.8: Simulation results of (a) convergence time, (b) communication cost, and (c) temporary cycle rate under perfect channels.

convergence time of different schemes. In all cases, our scheme performs better than DD and PEQ. In PEQ, it takes long for a child of a failed node to broadcast SEARCH and to wait for responses to determine a new path. In DD, its negative reinforcements may go several hops to arrive at the children of failed nodes, thus causing long latency. Fig. 6.8(b) compares that the communication costs required to reach convergence. Fig. 6.8(c) shows the rates that temporary cycles are generated during our simulations under perfect channels. PEQ may cause cycles when sensors simultaneously change routes and DD may easily cause cycles when sensors select new parents from their interest data caches regardless of their hop counts to the sink. Our scheme causes no temporary cycles. Note that when *HELLO* packets may be lost, our scheme may cause temporary cycles.



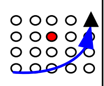
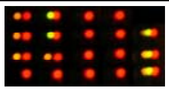
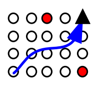
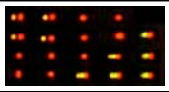
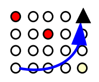
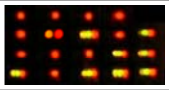
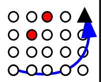
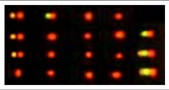
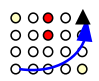
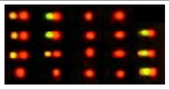
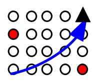
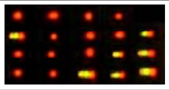
| No | Simulation | Experiment | Pkt. Count Sim./Exp. | No | Simulation | Experiment | Pkt. Count Sim./Exp. |
|---|---|---|---|---|---|---|---|
| 1 |  |  | 25/26 | 4 |  |  | 39/47 |
| 2 |  |  | 36/40 | 5 |  |  | 38/44 |
| 3 |  |  | 40/44 | 6 |  |  | 36/40 |

Figure 6.9: Simulation results vs. experimental results.

## 6.5 Prototyping Results

We have developed a prototyping system by MICAz motes [11]. We use photo sensors, which can read light degrees, to simulate and trigger emergency events. A sensor which senses a light degree above a threshold is considered detecting an emergency event and will broadcast EMG packets. Since broadcast communications are not reliable [64], packets may be lost. So we enforce sensors to periodically rebroadcast EMG packets to improve reliability.

A $4 \times 5$ grid network is tested, as shown in Fig. 6.9. Fig. 6.9 also shows both our simulation and real experimental message costs and navigation paths. Due to packet loss, slightly higher message overheads can be seen in experimental results. Navigation directions are reflected by LEDs of motes. Since the network scale is small, the navigation paths in both simulations and experiments are exactly the same.

# Chapter 7

# An Intelligent Light Control System by ZigBee WSNs

## 7.1 System Overview

In this work, we propose an intelligent light control system which considers users' activities in indoor environments. Fig. 7.1 shows the network scenario. The network field is divided into regular grids. Each grid has a fixed sensor. Together, these sensors form a multi-hop ad hoc network. One of the nodes is designated as the *sink* of the network and is connected to a *control host*. The control host can issue light control commands. In our system, there are two kinds of lighting devices, called *whole lighting* and *local lighting devices*. A whole lighting device is one such as a fluorescent light, which can provide illuminations for multiple grids. For example, in Fig. 7.1, the light in $G_{13}$ is a whole lighting device, which covers grids $G_7$, $G_8$, $G_9$, $G_{12}$, $G_{13}$, $G_{14}$, $G_{17}$, $G_{18}$, and $G_{19}$. A local lighting device is one such as a table lamp, which can only provide concentrated illumination.

In our system, we assume that the location of each user is known and each user carries a wireless sensor, which can detect its local light intensity. Users are considered to have various illumination requirements according to their activities. For example, in Fig. 7.1, user $A$ is watching television in $G_{25}$ and user $B$ is reading in $G_{16}$. Both $A$ and $B$ require sufficient background illuminations in their surroundings, and $B$ needs concentrated illumination for reading. In this work, we model an illumination requirement as the combination of back-

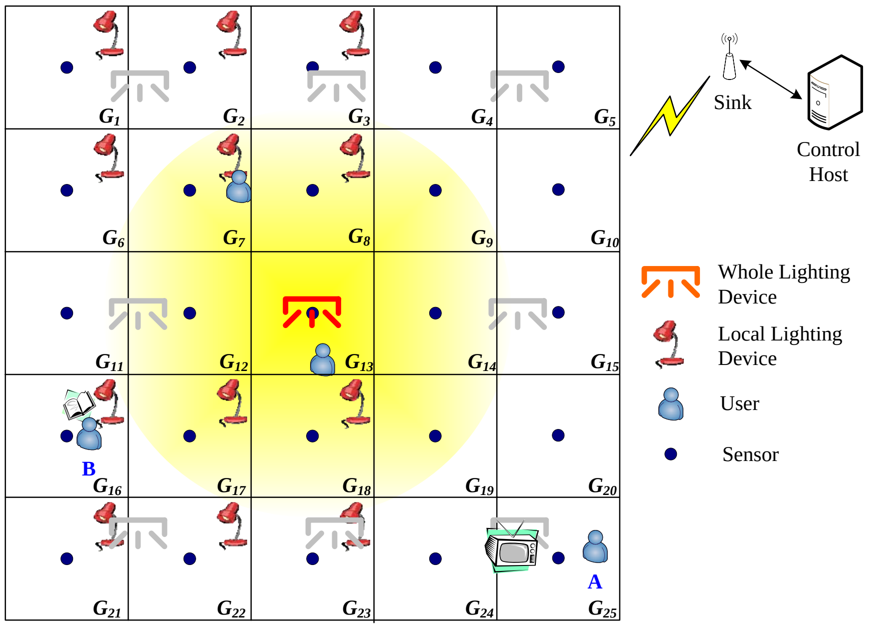


Figure 7.1: The network scenario of our system.

ground and concentrated lighting according to the user's current activity. An illumination requirement consists of an *illumination interval* and a *coverage range*. A user is said to be *satisfied* if the provided light intensity is in the specified interval for all grids in the coverage range. We design an illumination decision algorithm trying to satisfy all users such that the total power consumption is minimized. However, it may not be possible to satisfy all users simultaneously. In this case, we will gradually relax users' illumination intervals until all users are satisfied. Then the outputs are sent to a closed-loop device control algorithm to adjust the illuminations of lighting devices. Our prototyping results and system demonstrations verify that our ideas are practical and feasible.

Several works [51][53][61][68] have investigated using WSNs in light control for energy conservation. References [51] and [68] introduce light control using wireless sensors to save energy for commercial buildings. Lighting devices are adjusted according to daylight intensity. Reference [53] defines several kinds of user requirements and their corresponding cost functions. The goal is to adjust lights to minimize the total cost. However, the result is mainly for media production. The work [61] models the light control problem as a trade-off between energy conservation and user requirements. Each user is assigned a utility function with respect to light intensity. The goal is to maximize the total utility. However, it does not consider the fact that people need different illuminations under different activities. Also, some users may suffer from very low utilities, while others enjoy high utilities. In [53][61], it is necessary to measure all combinations of dimmer settings of all devices and the resulting light intensities at all locations. If there are $k$ interested locations, $d$ dimmer levels, and $m$ lighting devices, the complexity is $O(kdm)$. Moreover, the above works only consider one type of lighting devices. In real life, lighting devices can be classified as whole lighting and local lighting ones.

## 7.2 System Models

In this system, there are $k$ grids, $n$ users, $m$ whole lighting devices, and $m'$ local lighting devices. All lighting devices are adjustable. The $k$ grids represent the network area and are labeled as $G_1$, $G_2$, ..., and $G_k$. In each grid $G_i$, $i = 1..k$, there is a fixed sensor $f_i$, and each user $u_j$, $j = 1..n$, also carries a portable wireless sensor $p_j$. Users can specify their current activities to the control host via their portable devices. We also assume that via a localization scheme (such as [20]), users' current grid locations are known to the control host.

The whole lighting devices are named $D_1$, $D_2$, ..., $D_m$, and the local lighting devices are named $d_1$, $d_2$, ..., $d_{m'}$. The fixed sensor that is closest to $D_i$, $i = 1..m$, is denoted as $f_{c(D_i)}$. However, since users are mobile, we use a function $bound(u_j)$, $j = 1..n$, to denote the association between users and local lighting devices. This function restricts a local lighting device to serve at most one user at one time. If there is no local lighting device near user $u_j$, $bound(u_j) = \emptyset$; otherwise, $bound(u_j)$ is the ID of the nearest local lighting device. Light intensities sensed by $f_i$, $i = 1..k$, and $p_j$, $j = 1..n$, are denoted by $s(f_i)$ and $s(p_j)$, respectively. Since the value of $s(f_i)$ may be contributed by multiple sources, we denote by $l(D_i)$, $i = 1..m$, the portion of light intensity contributed by $D_i$ to the fixed sensor closest to $D_i$, i.e., $f_{c(D_i)}$. Note that $l(D_i) \leq s(f_{c(D_i)})$ because $s(f_{c(D_i)})$ may be affected by other whole lighting devices and sunlight. Similarly, we denote by $l(d_i)$, $i = 1..m'$, the portion of light intensity contributed by $d_i$ to portable sensor $p_j$ if user $u_j$ satisfies $bound(u_j) = i$. If there exists no $u_j$ such that

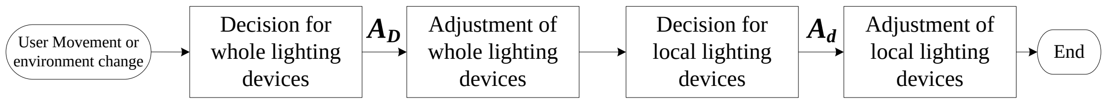

User Movement or environment change
Decision for whole lighting devices
$A_D$
Adjustment of whole lighting devices
Decision for local lighting devices
$A_d$
Adjustment of local lighting devices
End

Figure 7.2: The system architecture of our light control system.

$bound(u_j) = i$, we let $l(d_i) = 0$. Note that in reality, the values of $l(D_i)$ and $l(d_i)$ can not be directly known, unless there are no other light sources. We will address this issue in Section 7.2.

In the system, sensors periodically report their readings to the sink. For simplicity, we define the following column vectors:

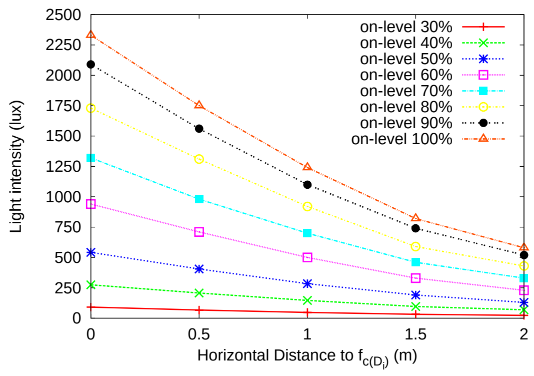
$$\begin{aligned} S_f &= \left[s(f_1), \quad s(f_2), \quad \ldots, \quad s(f_k)\right]^T, \\ S_p &= \left[s(p_1), \quad s(p_2), \quad \ldots, \quad s(p_n)\right]^T, \\ L_D &= \left[l(D_1), \quad l(D_2), \quad \ldots, \quad l(D_m)\right]^T, \\ L_d &= \left[l(d_1), \quad l(d_2), \quad \ldots, \quad l(d_{m'})\right]^T. \end{aligned}$$

Note that in practice, each $D_i$ has its limitation. So we let $l^{max}(D_i)$ be the upper bound of $l(D_i)$ and let

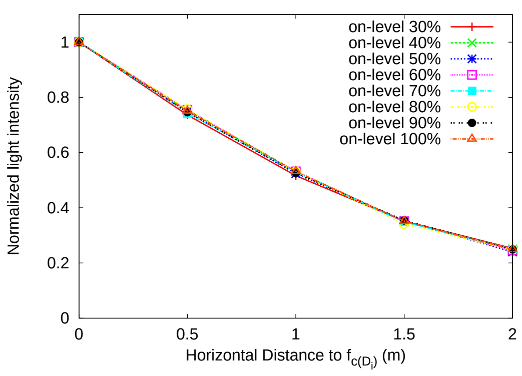
$$L_D^{max} = \left[l^{max}(D_1), \quad l^{max}(D_2), \quad \ldots, \quad l^{max}(D_m)\right]^T.$$

We make some assumptions about lighting devices. First, we assume that a local lighting device can always satisfy a user's need when the user is underneath this device. Second, we assume that there is no obstacle between whole lighting devices and fixed sensors. Third, the illumination provided by a local lighting device does not affect the measured light intensity of fixed sensors.

Fig. 7.2 shows our system architecture. Light adjustments are triggered by users' movements or environment changes. First, the illuminations of whole lighting devices are determined, followed by those of the local lighting devices. Feedbacks from sensors are then sent to the sink to decide further adjustment of lighting devices so as to satisfy users' demands.



(a)



(b)

Figure 7.3: An experiment for characterizing the degradation of light signals.

## Computing $L_D$ and $L_d$

Earlier, we mentioned that the values of $L_D$ and $L_d$ can not be known directly. Below, we first use an experimental method to derive $L_D$. Assuming no other light source existing, Fig. 7.3(a) shows the measured intensities of a whole lighting device $D_i$ by $f_{c(D_i)}$ and other fixed sensors at different distances from $f_{c(D_i)}$, under different on-levels of $D_i$. We see that the measured intensity degrades following a similar trend. In fact, if we further normalize the value to the intensity measured by $f_{c(D_i)}$, we see that the degrading trends are almost the same, as shown in Fig. 7.3(b). Therefore, assuming the impact factor of $D_i$ on $f_{c(D_i)}$ to be $w^i_{c(D_i)} = 1$, the impact factor of $D_i$ on any other $f_j$ can be written as a weighted factor $w^i_j$, where $0 \leq w^i_j \leq 1$. Putting all impact factors together, we define a weight matrix

$$W = \begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^m \\ w_2^1 & w_2^2 & \cdots & w_i^m \\ \vdots & \vdots & \cdots & \vdots \\ w_k^1 & w_k^2 & \cdots & w_k^m \end{bmatrix}.$$

Since light intensities are additive [61], the light intensity measured by $f_{c(D_i)}$ is the sum of intensities from sunlight, $D_i$, and neighboring devices. The intensities of the sunlight to all fixed sensors are written as a $k \times 1$ column vector $S_{sun}$. So we have

$$S_f = W \cdot L_D + S_{sun}. \tag{7.1}$$

In Eq. (7.1), there are $m$ unknowns in $L_D$ and $k$ equations, where $m \leq k$. Any typical $k$-means algorithm [48] can solve Eq. (7.1) by inducing the least mean square error. Here, we simply construct a new $m \times m$ matrix $\hat{W}$ by keeping all $c(D_i)$-th rows, $i = 1..m$, in $W$ and removing the other $k - m$ rows. So, Eq. (7.1) can be rewritten as

$$S_f - S_{sun} = \hat{W} \cdot L_D \Rightarrow L_D = \hat{W}^{-1} \cdot (S_f - S_{sun}). \tag{7.2}$$

The weight matrix $W$ can be measured at the deployment stage, vector $S_{sun}$ can be measured on-line when all lights are off, and vector $S_f$ can be obtained on-line. So the calibration complexity is $O(km)$. This is lower than those of [53][61].

The calculation of $L_d$ is quite straightforward. Due to the property of our approach, before a user arrives at a $d_i$, no measurement can be obtained for $l(d_i)$. At this time, $l(d_i) = 0$. When a portable sensor, say, $p_k$ is getting close to and bounded with $d_i$, the local lighting device $d_i$ may be triggered. Here, we simply use the reading of the fixed sensor, say, $f_j$ located at the same grid as $d_i$ as the background light intensity. We let the light intensity provided by $d_i$ to $p_k$ be

$$l(d_i) = s(p_k) - s(f_j).$$

## 7.3 Illumination Decision Algorithm

Each user profile consists of a number of activity-requirement pairs. Given an activity, the system should try to satisfy the corresponding requirement. Each requirement of a user $u_i$ has three parts:

1. Expected illumination interval of whole lighting: $[B_D^l(u_i), B_D^u(u_i)]$ (in lux), where $B_D^l(u_i)$ and $B_D^u(u_i)$ are the lower and the upper bounds, respectively.

2. Expected illumination interval of local lighting: $[B_d^l(u_i), B_d^u(u_i)]$, where $B_d^l(u_i)$ and $B_d^u(u_i)$ are the lower and the upper bounds, respectively.

3. Coverage range of whole lighting: $R_i = [r_i(G_1), r_i(G_2), \ldots, r_i(G_k)]^T$, where for each $j = 1..k$, $r_i(G_j) = 1$ if grid $G_j$ is expected to receive a light intensity within $[B_D^l(u_i), B_D^u(u_i)]$

for user $u_i$; otherwise, $r_i(G_j) = 0$. This array defines the range of grids which should meet the whole lighting requirement.

For example, a possible requirement of a reading user B in Fig. 7.1 can be $[B_D^l(u_B), B_D^u(u_B)] = [200, 600]$, $[B_d^l(u_B), B_d^u(u_B)] = [500, 1000]$, and $R_B = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0]^T$.

Let $L_D$ and $L_d$ be the current intensity vectors provided by whole and local lighting devices, respectively. To facilitate the presentation, let $X_K = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$ be a $1 \times K$ row vector, and $\bar{R}_i$ a $k \times k$ matrix such that

$$\bar{R}_i = \begin{bmatrix} r_i(G_1) & 0 & \cdots & 0 \\ 0 & r_i(G_2) & \cdots & 0 \\ 0 & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & r_i(G_k) \end{bmatrix}.$$

We formulate our problem $P$ as a linear programming problem with inputs $S_f$, $S_p$, $L_D$, $L_d$, $W$, and user requirements. Our goal is to find the adjustment vectors

$$A_D = \begin{bmatrix} a(D_1), & a(D_2), & \ldots, & a(D_m) \end{bmatrix}^T$$
$$A_d = \begin{bmatrix} a(d_1), & a(d_2), & \ldots, & a(d_{m'}) \end{bmatrix}^T$$

for whole and local lighting devices, respectively, where $a(D_i)$, $i = 1..m$, and $a(d_j)$, $j = 1..m'$, are the amounts of adjustment required for $D_i$ and $d_j$, respectively, such that the following two objectives are satisfied:

$$\min \quad X_m(A_D + L_D) \tag{7.3}$$

$$\min \quad X_{m'}(A_d + L_d) \tag{7.4}$$

subject to:

$$B_D^l(u_i)R_i \le \bar{R}_i(S_f + WA_D) \le B_D^u(u_i)R_i, \ \forall i \in [1, n] \tag{7.5}$$

$$O \le A_D + L_D \le L_D^{max} \tag{7.6}$$

$$B_d^l(u_i) \le a(d_j) + s(p_i) \le B_d^u(u_i), \ \text{ if } bound(u_i) = j, \ \forall i \in [1, n]. \tag{7.7}$$

Eq. (7.3) and Eq. (7.4) mean that the total power consumptions of both whole and local lighting devices after the adjustment should be minimized. Eq. (7.5) imposes the whole lighting requirement, where $S_f + WA_D$ is the light intensity vector after adjustment and matrix $\bar{R}_i$ is to filter out those grids not in the coverage range of whole lighting. Eq. (7.6) is to confine the adjustment result within the maximum and the minimum capacities of devices, where $O$ is a zero vector. Eq. (7.7) is to impose the requirement of each local lighting if a user is bounded to it. Here we assume that local lighting can always provide extra illuminations to satisfy users' requirements. So we do not specify upper bounds as that in Eq. (7.6).

Since we assume that the illuminations of local lighting devices do not affect the measured light intensity of fixed sensors, the decision of whole lighting levels can be made independently of the decision of local lighting levels. (However, the reverse is not true because the decision of whole lighting levels does affect the decision of local lighting levels.) This allows us to solve problem $P$ in two stages as formulated below.

**P1**: Given $S_f$, $L_D$, $W$, and user requirements, solve $A_D$ for Eq. (7.3), Eq. (7.5), and Eq. (7.6).

**P2**: Given $S_p$, $L_d$, and user requirements, solve $A_d$ for Eq. (7.4) and Eq. (7.7).

**Theorem 5** *Problem $P$ is equivalent to the joint problems $P1$ and $P2$.*

Problem $P1$ is a linear programming problem, which can be solved by the Simplex method [27], unless the problem itself is infeasible, which may happen when two users have conflicting requirements on the same grid. When no feasible solution can be found, our system will try to eliminate some constraints to make $P1$ feasible. Reference [58] already shows that finding a feasible subsystem of a linear system by eliminating the fewest constraints is NP-hard. Hence, we propose a heuristic below.

The idea is to gradually relax some requirements until a feasible solution appears. We first define some notations. Given the current values of $S_f$, $L_D$, and $L_D^{max}$, it is easy to compute the minimum and maximum possible illuminations of grids by $S_f^{min} = S_f - WL_D$ and $S_f^{max} = S_f + W(L_D^{max} - L_D)$. Also, consider $c$ intervals on $\mathbb{R}$ (the set of reals) which define $c$ users' requirements on whole lighting. We say that an interval $[a, b] \in \mathbb{R}$ has an *overlapping degree* of $d$ if for each point $p \in [a, b]$, $p$ falls in at least $d$ of the above $c$ intervals. An interval $[a, b]$ is said to be a *max-interval* if there exists no other interval $[a', b']$ which has a higher overlapping degree than $[a, b]$ and $[a', b']$ is a superset of $[a, b]$. It is not hard to see that given any $c$ intervals, there must exist a max-interval. Also it is easy to design a polynomial-time linear search algorithm to find a max-interval (we omit the details here). Our algorithm works as follows.

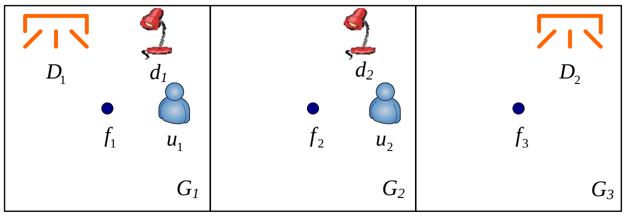
1. For each grid $G_i$, $i = 1..k$, find the set of users $U_i$ whose coverage ranges contain $G_i$, i.e., $U_i = \{u_j | r_j(G_i) = 1, \forall j \in [1, n]\}$. For each user $u_j \in U_i$, check if $[B_D^l(u_j), B_D^u(u_j)] \cap [S_f^{min}[i], S_f^{max}[i]] = \emptyset$. If so, the requirement cannot be satisfied. So we set $r_j(G_i) = 0$ and update $\bar{R}_j$.

2. Again, for each grid $G_i$, $i = 1..k$, consider the set $U_i$. Check if there is a common overlapping interval for the requirements of all users in $U_i$. If not, find a max-interval, say, $[a, b]$ for the requirements of all users in $U_i$. For each user $u_j \in U_i$, check if $[B_D^l(u_j), B_D^u(u_j)] \cap [a, b] = \emptyset$. If so, we will give up the requirement of $u_j$. So we set $r_j(G_i) = 0$ and update $\bar{R}_j$.

3. Try to solve problem $P1$. If there exists no feasible solution $A_D$, relax the whole lighting requirement of each user $u_i$, $i = 1..n$, to $[B_D^l(u_i) - \alpha, B_D^u(u_i) + \alpha]$, where $\alpha$ is a predefined constant. Then repeat this step again.

4. After deciding $A_D$, solve problem $P2$ as follows. For each $d_j$, $j = 1..m'$, check if there is a user $u_i$ such that $bound(u_i) = j$. If so, set $a(d_j) = B_d^l(u_i) - s(p_i)$; otherwise, we can inform the system to turn $d_j$ off.

**Example 1**: Fig. 7.4 shows a scenario with three grids, two users, two whole lighting devices, and two local lighting devices. User $u_1$'s requirements are $[B_D^l(u_1), B_D^u(u_1)] = [200, 400]$, $[B_d^l(u_1), B_d^u(u_1)] = [700, 900]$, and $R_1 = [1, 0, 0]^T$. User $u_2$'s requirements are $[B_D^l(u_2), B_D^u(u_2)] = [300, 500]$, $[B_d^l(u_2), B_d^u(u_2)] = [800, 1000]$, and $R_2 = [0, 1, 0]^T$. Problem $P1$ has the objective:



$$S_f = [100 \quad 100 \quad 100]^T \qquad W = \begin{bmatrix} 1 & 0 \\ 0.6 & 0.6 \\ 0 & 1 \end{bmatrix}$$

$$L_D = [0 \quad 0]^T \qquad L_D^{max} = [1000 \quad 1000]^T$$

$$bound(u_1) = 1 \qquad bound(u_2) = 2$$

Figure 7.4: An example of illumination decision.

$$\begin{aligned} & \min \quad \begin{bmatrix} 1 & 1 \end{bmatrix} \left( \begin{bmatrix} a(D_1) \\ a(D_2) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) \\ \equiv & \quad \min \quad (a(D_1) + a(D_2)) \end{aligned}$$

subject to:

$$200 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left( \begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0.6 & 0.6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a(D_1) \\ a(D_2) \end{bmatrix} \right) \leq 400 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\equiv \begin{bmatrix} 200 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 100 + a(D_1) \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 400 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 300 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 100 + 0.6a(D_1) + 0.6a(D_2) \\ 0 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 500 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} a(D_1) \\ a(D_2) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 1000 \\ 1000 \end{bmatrix}.$$

Since $P1$ is feasible, the results are $a(D_1) = 184$ and $a(D_2) = 150$.

After adjusting whole lighting devices, $S_p = [s(p_1), s(p_2)]^T = [284, 300]^T$ and $L_d = [l(d_1), l(d_2)]^T = [0, 0]^T$. So problem $P2$ has the objective:

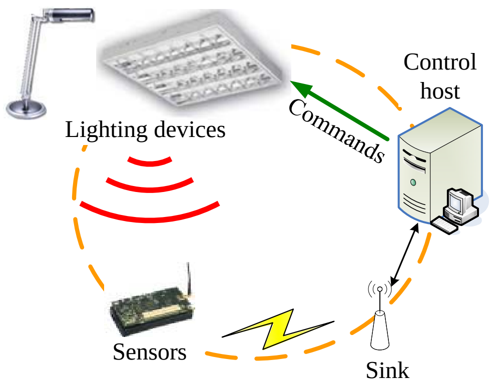


Figure 7.5: The closed-loop device control procedure.

$$\begin{aligned} & \min \quad \begin{bmatrix} 1 & 1 \end{bmatrix} \left( \begin{bmatrix} a(d_1) \\ a(d_2) \end{bmatrix} + \begin{bmatrix} 284 \\ 300 \end{bmatrix} \right) \\ \equiv & \min \quad (a(d_1) + a(d_2) + 584) \end{aligned}$$

subject to:

$$700 \leq a(d_1) + 284 \leq 900$$
$$800 \leq a(d_2) + 300 \leq 1000.$$

The adjustments of local lighting devices are as $a(d_1) = B_d^l(u_1) - s(p_1) = 416$ and $a(d_2) = B_d^l(u_2) - s(p_2) = 500$.

## 7.4 Device Control Algorithm

Given the light intensities contributed by devices to sensors, i.e., $L_D$ and $L_d$, the algorithm in Section 7.3 will determine the target adjustment amounts, i.e., $A_D$ and $A_d$. However, since what reported by sensors are accumulated values, we have to convert these values to the actual adjustment amounts. If the actual amounts do not match the target amounts, we will adopt a binary search technique to gradually approach these amounts.

Below, let $L_D^{(1)}$ and $L_d^{(1)}$ be the current contributed intensities of whole and local lighting devices, respectively, and $L_D^{(*)} = L_D^{(1)} + A_D$ and $L_d^{(*)} = L_d^{(1)} + A_d$ be the target ones. Our algorithm contains multiple iterations. In the $i$-th iteration, $i \geq 1$, based on $L_D^{(i)}$ and $L_d^{(i)}$, we will adjust devices leading to new intensities $L_D^{(i+1)}$ and $L_D^{(i+1)}$. This will be repeated until the target values are reached or no further improvement is possible. Such a closed loop control is illustrated in Fig. 7.5. The binary search procedure can be explained by the following example. Suppose that device $D_i$'s current on-level is $40\%$ with contribution $l^{(0)}(D_i) = 300$ lux to sensor $f_{c(D_i)}$ and $l^{(*)}(D_i) = 200$ lux. The control host will first adjust the on-level of $D_i$ to $(0 + 40)/2 = 20\%$. After first iteration, the control host will collect sensors' reports to compute $L_D^{(1)}$ and thus $l^{(1)}(D_i)$. With $l^{(1)}(D_i)$, the next guess will be an on-level of $10\%$ or $30\%$. The similar trial will be done for all whole and local lighting devices.

In practice, the on-levels of dimmers are discrete and have finite levels. The termination conditions of the above binary search can be controlled by a threshold, say, $\beta$ when $|l^{(i+1)}(D_j) - l^{(i)}(D_j)| \leq \beta$. To accelerate the decision, the control host can even record the relationship between the contributed light intensities and on-levels of devices (we omit the details here).

## 7.5 Prototyping Results

This section presents our implementation of the intelligent light control system. Fig. 7.6 shows the system architecture and the related protocol components. The system can be divided into three parts: wireless sensor network, actuators, and control host. In the following, we describe each part in details.

### Wireless Sensor Network

Our sensor nodes are developed using Jennic JN5121 [9] as the radio module and Si photodiode IC [12] as the photo sensor (Fig. 7.7). Users can indicate their current activities to the system by clicking the buttons on the sensor board. Fixed sensors are used to form the backbone of the network. A portable sensor will associate with the nearest fixed sensor. Fixed and portable sensors periodically report aggregated light intensity values to the sink. The sink forwards sensing data to the control host via an RS232 interface. Note that when a sensor finds

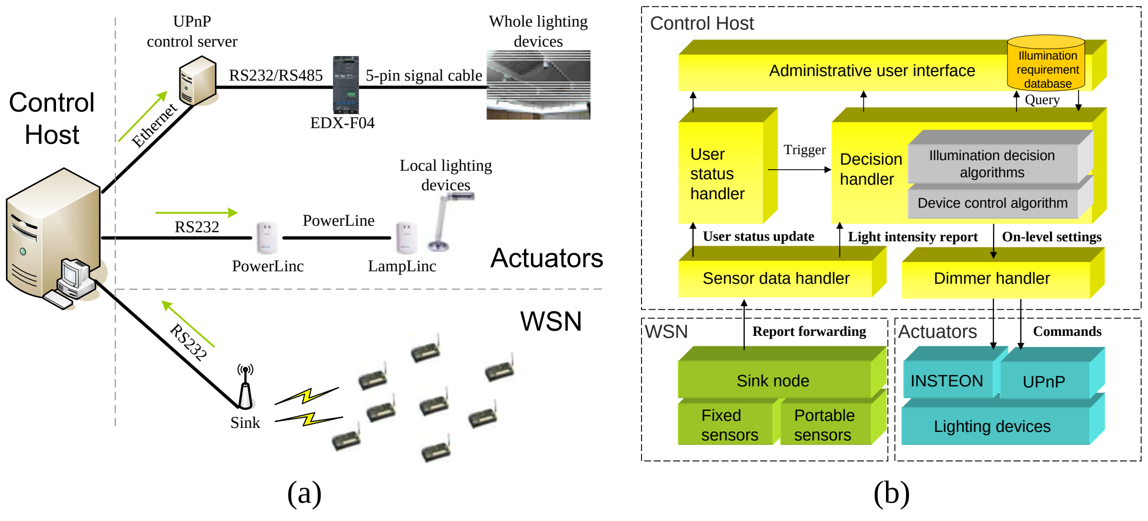


Figure 7.6: (a) System architecture and (b) components of our intelligent light control system.

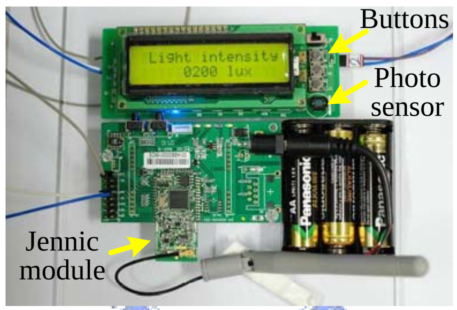


Figure 7.7: The implemented sensor board.

that its surrounding light intensity changes rapidly, it will also report. This happens when the control host is adjusting lighting devices. Moreover, we implement a reduced version of the localization scheme in [20] to trace users' locations. Once a portable sensor decides its owner's location, it issues a location update to the control host.

## Actuators

In our current implementation, whole and local lighting devices are controlled by different ways. We implement the UPnP Lighting Controls V1.0 standard [14] to control whole lighting devices. The control host issues UPnP device control commands to the UPnP control server through the Internet. Then the UPnP control server controls some dimmer EDX-F04 dimmers

[4], which are connected to whole lighting devices. On the other hand, we use the INSTEON LampLinc dimmer and PowerLinc controller manufactured by SmartHome [13] to control local lighting devices. Each local lighting device is plugged in a LampLinc dimmer. The PowerLinc controller is connected to the control host. When receiving control commands from the control host, the PowerLinc controller can control dimmers through the power-line network.

## Control Host

The control host is implemented by Java. It consists of five components.

- *Sensor data handler*: Its main task is to classify the report data from the sink into two types: *user status update* and *light intensity report*. Then it relays these data to the corresponding components.

- *User status handler*: This component tracks the latest locations and activities of users. When detecting any change of users' locations or activities, it triggers the decision handler component to compute new illumination requirements.

- *Decision handler*: This component implements the algorithms in Section 7.3 and Section 7.4. It is triggered by the user status handler component or by any change in the environment. We use Matlab to implement our algorithms in Section 7.3. The Matlab program is translated to a Java program by the Matlab builder for Java [10]. After making device control decisions, it sends on-level settings to the dimmer handler.

- *Dimmer handler*: This component serves as the interface between the control host and the actuators and issues commands to the UPnP control server and the INSTEON PowerLinc controller.

- *Administrative user interface*: We implement a graphical user interface (GUI), which contains three panels: 1) The *monitor panel* shows the locations of users, fixed sensors, and lighting devices. 2) The *configuration panel* is for the system manager to plan

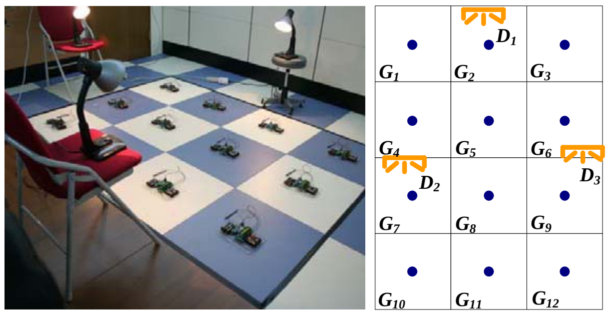


Figure 7.8: The scenario to verify the measured $L_D$.

the network and set system parameters. 3) The *information panel* shows the reported sensory readings, the connection statuses of sensor nodes, and so on.

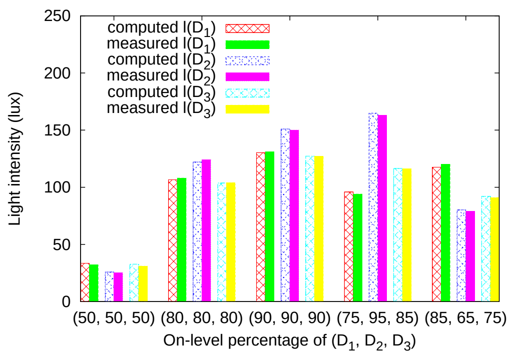
We build the light control system in a room of size $5\ m \times 5\ m$, which is divided into $3 \times 3$ grids. More details and demo videos can be found in http://wsn-research.blogspot.com/.

## 7.6 Performance Evaluations

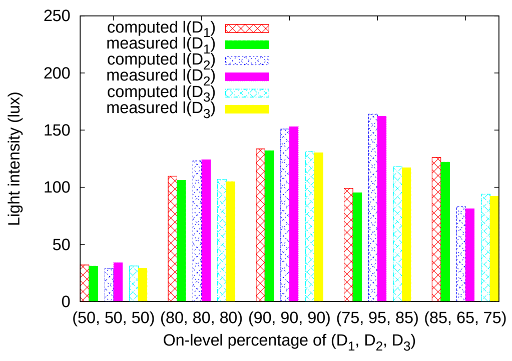
We use some experiments and simulations to verify our results.

*A) Verification of the estimation of $L_D$*: In Section 7.2, we show how to evaluate $L_D$. Here we use the network scenario in Fig. 7.8 with $12$ grids and three whole lighting devices to verify the result. Here, we simply use lamps as whole lighting devices. With different on-levels for lamps, we compute $L_D$ and compare it against the actual measured value. Fig. 7.9 shows the comparison without and with sunlight effect. We can see that the computed and the measured values are quite close.

*B) Verification of the illumination decision algorithm (IDA)*: We set up two scenarios, $S1$ and $S2$. Scenario $S1$ has $5 \times 5$ grids with $9$ whole lighting devices as in Fig. 7.1. Scenario $S2$ has $9 \times 9$ grids with $25$ whole lighting devices. In both scenarios, each whole lighting device can cover its nearby $9$ grids. The weighted factors of each whole lighting device $D_i$ on nearby fixed sensors are set as follows. (1) The weighted factor of $D_i$ on the fixed sensor at $G_{c(D_i)}$ is $1$. (2) For fixed sensors in left, right, up, and down grids of $G_{c(D_i)}$, the weights are set to $0.5$.

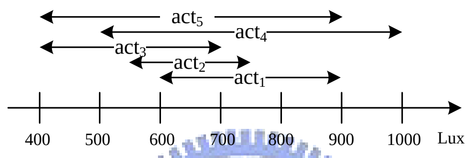


(a)

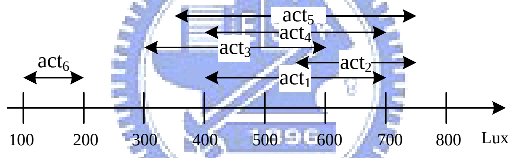


(b)

Figure 7.9: Experiments on computed and measured $L_D$ when the environment is (a) without and (b) with sunlight effect.



(a)



(b)

Figure 7.10: Activity-requirement pools: (a) $AR1$ and (b) $AR2$.

(3) For fixed sensors in upper-left, lower-left, upper-right, and lower-right grids of $G_{c(D_i)}$, the weights are $0.25$. (4) For all other fixed sensors, the weights are $0$. Local lighting devices are not simulated since they have no impact on performance. All lighting devices are initially set to be turned off.

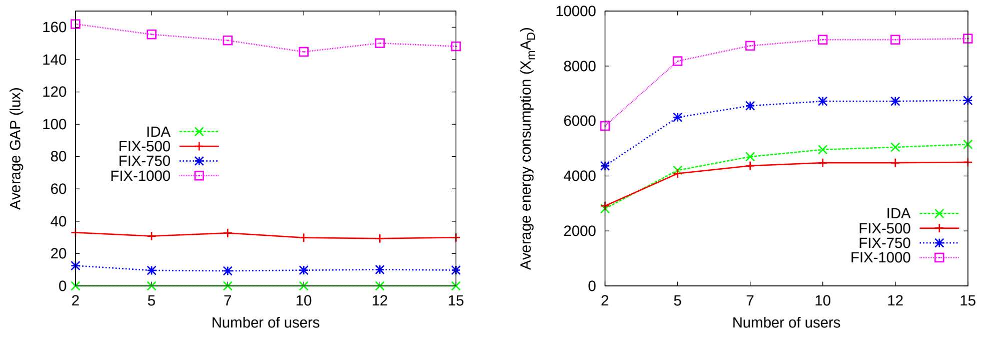
We define two activity-requirement pools, called $AR1$ and $AR2$, as shown in Fig. 7.10. Each $act_i$ in Fig. 7.10 represents an expected illumination interval of whole lighting. In our simulations, users randomly select their activities from a pool. The coverage range of a user's requirement is the five nearest grids. We compare our algorithm against a fixed adjustment scheme (denoted by FIX), where lighting devices are set to fixed levels. If a user's requirement coverage range overlaps a lighting device's coverage range, this device is turned to that level. Below, we use FIX-$n$ to indicate that each device can provide at most $n$ lux.

We consider two performance indices. First, considering that our algorithm may enlarge users' illumination requirements when conflicts occur, we define a metric *GAP* to represent the difference between the provided light intensity and the original requirement of a user. For user $u_i$ with coverage range $R_i$, if grid $G_j$ satisfies $r_i(G_j) = 1$, we compute a gap value as

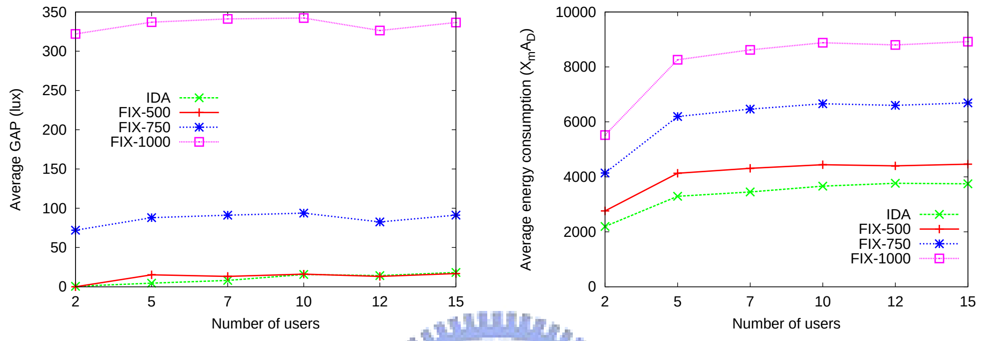
$$gap(u_i, G_j) = \begin{cases} 0 & \text{if } B^l_D(u_i) \leq s(f_j) \leq B^u_D(u_i) \\ \min(|B^l_D(u_i) - s(f_j)|, |B^u_D(u_i) - s(f_j)|) & \text{o.w.,} \end{cases}$$

where $s(f_j)$ is the final sensory value of $f_j$. Then we define GAP of $u_i$ as the average of $gap(u_i, G_j)$ for all $G_j$ such that $r_i(G_j) = 1$. The second index is $X_m A_D$, which represents the energy consumption of one control decision.

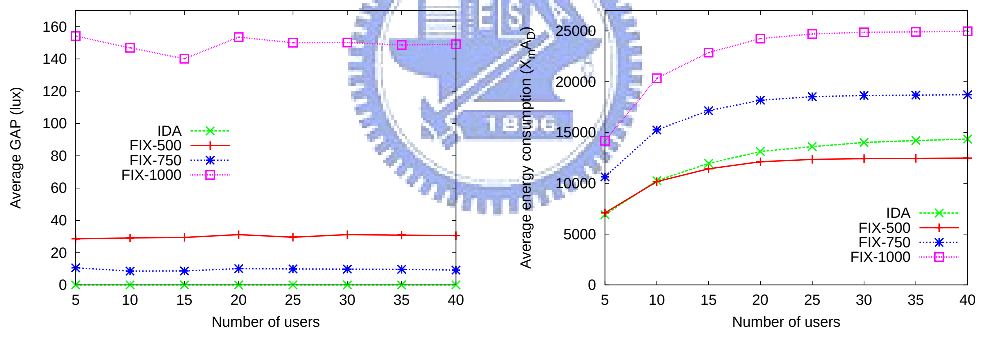
Fig. 7.11(a), Fig. 7.11(b), Fig. 7.11(c), and Fig. 7.11(d) show our simulation results under different combinations of $S1$/$S2$ and $AR1$/$AR2$. In the left figure of Fig. 7.11(a), we see that the average GAP of users is almost zero for IDA. This is because the illumination intervals in $AR1$ have common overlapping, which allows our algorithm to satisfy all users in most cases. The right figure of Fig. 7.11(a) compares the energy consumption of different schemes. FIX-500 has a slightly lower value than ours because some users' requirements are violated. Fig. 7.11(b) adopts $AR2$. Since some requirements are violated, we see that our scheme also induces some gaps (note that $act_6$ has no overlapping with others). In terms of energy cost, IDA outperforms the other schemes. Fig. 7.11(c) and Fig. 7.11(d) adopt $S2$ and the trends are similar. This demonstrates that our scheme is quite scalable to network size.



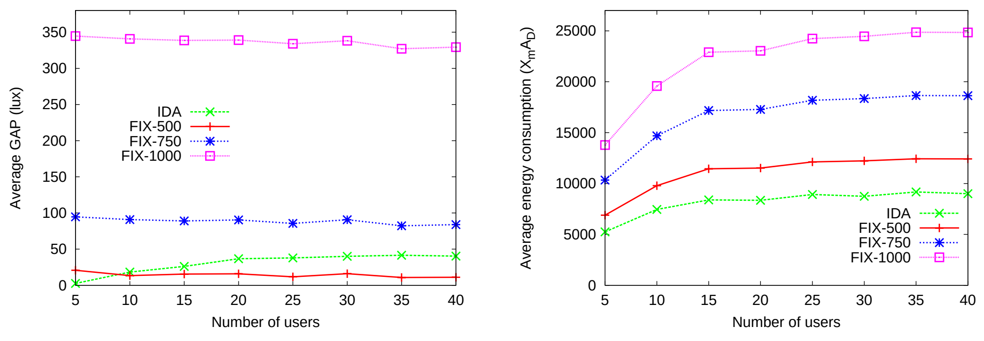
(a)



(b)



(c)



(d)

Figure 7.11: Comparison of the proposed IDA and the FIX schemes when the network scenario and user-activity are (a) $S1$ and $AR1$, (b) $S1$ and $AR2$, (c) $S2$ and $AR1$, and (d) $S2$ and $AR2$, respectively.

# Chapter 8

# Conclusions and Future Directions

This dissertation contain five works. In the first three works, we discuss communication protocols in ZigBee network layer. In the last two works, we introduce two applications, which can operate based on the designed network layer protocols. In the following, we summarize this dissertation.

In Chapter 3, we have identified a new orphan problem in ZigBee-based wireless sensor networks. We show that the problem is non-trivial because a device is not guaranteed to join a network even if there are remaining address spaces. We model this orphan problem in two subproblems, namely the BDDTF problem and the EDMM problem. We prove the BDDTF problem is NP-complete and propose a two-stage network formation policy, which can greatly relieve the orphan problem. Compared to the network formation scheme defined in ZigBee, our algorithms can effectively reduce the number of orphan devices.

In Chapter 4, we have proposed hierarchical address assignment and routing schemes for ZigBee-based LT WSNs. The proposed address assignment scheme divides nodes into several clusters and then assigns each node a cluster ID and a node ID as its network address. With such a hierarchical structure, routing can be easily done based on addresses of nodes and the spaces required for the network addresses can be significantly reduced. We also show how to allow nodes to utilize shortcuts. With our design, not only network addresses can be efficiently utilized, but also the network scale can be enlarged to cover wider areas without suffering from address shortage. We verify our schemes by simulation programs.

In Chapter 5, we have defined a new minimum delay beacon scheduling (MDBS) problem for convergecast with the restrictions that the beacon scheduling must be compliant to the ZigBee standard. We prove the MDBS problem is NP-complete and propose optimal solutions for special cases and two heuristic algorithms for general cases. Simulation results indicate the performance of our heuristic algorithms decrease only when the number of interference neighbors is increased. Compared to the random slot assignment and greedy slot assignment scheme, our heuristic algorithms can effectively schedule the ZigBee routers' beacon times to achieve quick convergecast.

In Chapter 6, we have proposed an emergency guiding and an emergency monitoring services for indoor environments. The proposed emergency guidance scheme can quickly converge and find safe guidance paths to exits when emergencies occur. The tree reconstruction protocol reduces the occurrence of temporary cycles and further shortens the convergence time. We verify both our schemes by real implementation and simulation programs.

In Chapter 7, we have presented an intelligent light control system considering user activities. In this system, there are two types of lighting devices. We use wireless sensors to collect light intensities in the environment. Considering users' activities, we model the illumination requirements of users. An illumination decision algorithm and a device control algorithm are presented to meet user requirements and to conserve energy. The proposed schemes are verified by real implementation in an indoor environment.

Based on the results presented above, several issues worth further investigation are summarized as follows.

- According to the result of our first work, we can know that the orphan problem is hard to solve. In the future, we can further discuss how to set $Cm$, $Rm$, and $Lm$, which can induce less than $p$ % of orphan devices if some parameters (ex. node density, network size, node's transmission range, and so on) are provided.

- It deserves to further discuss address assignment and routing schemes for more complicated topologies such as meshes that are connected by "long-thin" links.

- According to ZigBee standard, regular beacons are not allow in mesh networks. It deserves to consider an asynchronous sleep scheduling method to support energy-efficient convergecast in ZigBee mesh networks.

- In our current guiding system, the hazardous region is defined by the numbers of hops in $G_g$ and the altitudes of nodes in hazardous regions are adjusted by a static function. In fact, the definition of hazardous regions and altitude adjustments can be application- or scenario-dependent. For example, if the temperature is larger than $100°C$, sensors will trigger a fire emergency. In this case, sensors detecting a temperature larger than $70°C$ can consider themselves as in a hazardous region. The altitude adjustment function can be designed according to the sensed temperature. No matter how sensors in hazardous regions adjust their altitudes, the proposed local minimum adjustment rules can be applied.

- In the light control application, the current user requirement is defined as a binary model, i.e., a user who is satisfied returns a satisfaction value of one. We can further model users' satisfaction values as a function, which return value is decided by users' surrounding light intensities. Moreover, we can also enhance the user interfaces at the portable sensor nodes.

# Bibliography

[1] Chipcon corporation. http://www.chipcon.com/.

[2] Chipcon CC2420DBK. http://www.chipcon.com/.

[3] Dust network Inc. http://dust-inc.com/flash-index.shtml.

[4] Edx-f04. http://www.liteputer.com.cn/china/liteputer-tw/product.asp.

[5] Ember - wireless semiconductor. http://www.ember.com/.

[6] Design and construction of a wildfire instrumentation system using networked sensors. http://firebug.sourceforge.net/.

[7] Freescale semiconductor. http://www.freescale.com/.

[8] Habitat monitoring on great duck island. http://www.greatduckisland.net/technology.php.

[9] Jennic JN5121. http://www.jennic.com/.

[10] Matlab Builder for Java. http://www.mathworks.com/products/ javabuilder/.

[11] Motes, smart dust sensors, wireless sensor networks. http://www.xbow.com/.

[12] Si photodiode s1133. http://jp.hamamatsu.com/en/index.html.

[13] SmartHome Inc. http://www.smarthome.com.

[14] UPnP forum. http://www.upnp.org.

[15] Zigbee Alliance. http://www.zigbee.org/.

[16] A. A. Ahmed, H. Shi, and Y. Shang. SHARP: A new approach to relative localization in wireless sensor networks. In *Proc. of Int'l Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2005.

[17] M. Ali and Z. A. Uzmi. An energy-efficient node address naming scheme for wireless sensor networks. In *Proc. of IEEE Int'l Networking and Communications Conference (INCC)*, 2004.

[18] J. Bachrach, R. Nagpal, M. Salib, and H. Shrobe. Experimental results and theoretical analysis of a self-organizing global coordinate system for ad hoc sensor networks. *Telecommunications Systems Journal*, 26(2-4):213–234, 2004.

[19] S.-J. Baek, G. de Veciana, and X. Su. Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation. *IEEE Journal on Selected Areas in Communications*, 22(6):1130–1140, 2004.

[20] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proc. of IEEE INFOCOM*, 2000.

[21] M. A. Batalin, G. S. Sukhatme, and M. Hattig. Mobile robot navigation using a sensor network. In *Proc. of IEEE Int'l Conference on Robotics and Automation (ICRA)*, 2004.

[22] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 45(1):1–51, 1995.

[23] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proc. of ACM/IEEE Int'l Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2004.

[24] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[25] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proc. of Hawaii Int'l Conference on Systems Science (HICSS)*, 2001.

[26] H. Choi, J. Wang, and E. A. Hughes. Scheduling for information gathering on sensor network. *ACM/Kluwer Wireless Networks*, 2007 (Published online).

[27] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[28] A. Czumaj and W.-B. Strothmann. Bounded degree spanning trees. In *Proc. of European Symposium on Algorithms (ESA)*, 1997.

[29] T. Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[30] S. Gandham, Y. Zhang, and Q. Huang. Distributed minimal time convergecast scheduling in wireless sensor networks. In *Proc. of IEEE Int'l Conference on Distributed Computing Systems (ICDCS)*, 2006.

[31] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multiresolution storage for sensor networks. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[32] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[33] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proc. of Hawaii Int'l Conference on Systems Science (HICSS)*, 2000.

[34] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Proc. of ACM/IEEE Int'l Conference on Information Processing in Sensor Networks (IPSN)*, 2004.

[35] C.-F. Huang, Y.-C. Tseng, and L.-C. Lo. The coverage problem in three-dimensional wireless sensor networks. *Journal of Interconnection Networks*, 8(3):209–227, 2007.

[36] Y.-K. Huang, A.-C. Pang, and T.-W. Kuo. AGA: Adaptive GTS allocation with low latency and fairness considerations for IEEE 802.15.4. In *Proc. of IEEE Int'l Conference on Communications (ICC)*, 2006.

[37] IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), 2003.

[38] IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)(revision of IEEE Std 802.15.4-2003), 2006.

[39] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Networking*, 11(1):2–16, 2003.

[40] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer. Distributed search and rescue with robot and sensor teams. In *Proc. of Int'l Conference on Field and Service Robotics (FSR)*, 2003.

[41] M. Kochhal, L. Schwiebert, and S. Gupta. Role-based hierarchical self organization for wireless ad hoc sensor networks. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003.

[42] J. Konemann, A. Levin, and A. Sinha. Approximating the degree-bounded minimum diameter spanning tree problem. *Algorithmica*, 41(2):117–129, 2004.

[43] J. Konemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proc. of ACM Symposium on Theory of Computing (STOC)*, 2000.

[44] Q. Li, M. DeRosa, and D. Rus. Distributed algorithm for guiding navigation across a sensor network. In *Proc. of ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[45] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. Mobile Computing*, 5(8):1044–1056, 2006.

[46] Z. Lotker, M. M. de Albeniz, and S. Perennes. Range-free ranking in sensors networks and its application to localization. In *Proc. of Int'l Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW)*, 2004.

[47] 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 IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[48] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[49] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. of IEEE INFOCOM*, 2001.

[50] D. Niculescu and B. Nath. DV based positioning in ad hoc networks. *Telecommunications Systems Journal*, 22(1-4):267–280, 2003.

[51] F. O'Reilly and J. Buckley. Use of wireless sensor networks for fluorescent lighting control with daylight substitution. In *Proc. of Workshop on Real-World Wireless Sensor Networks (REANWSN)*, 2005.

[52] E. Ould-Ahmed-Vall, D. M. Blough, B. S. Heck, and G. F. Riley. Distributed unique global ID assignment for sensor networks. In *Proc. of IEEE Mobile Adhoc and Sensor Systems Conference (MASS)*, 2005.

[53] H. Park, M. B. Srivastava, and J. Burke. Design and implementation of a wireless sensor network for intelligent light control. In *Proc. of ACM/IEEE Int'l Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[54] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. of IEEE INFOCOM*, 1997.

[55] N. Patwari, I. Alfred O. Hero, M. Perkins, N. S. Correal, and R. J. O' Dea. Relative location estimation in wireless sensor networks. *IEEE Trans. Signal Processing*, 51(8):2137–2148, 2003.

[56] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing protocol. In *IETF RFC (3561)*, 2003.

[57] R. Peterson and D. Rus. Interacting with a sensor network. In *Proc. of Australasian Conference on Robotics and Automation (ICRA)*, 2002.

[58] J. Sankaran. A note on resolving infeasibility in linear programs by constraint relaxation. *Operations Research Letters*, 13(1):19–20, 1993.

[59] C. Schurgers, G. Kulkarni, and M. B. Srivastava. Distributed on-demand address assignment in wireless sensor networks. *IEEE Trans. Parallel and Distributed System*, 13(10):1056–1065, 2002.

[60] C. Schurgers and M. B. Srivastava. Energy efficient routing in wireless sensor networks. In *Proc. of Military Communications Conference(MILCOM)*, pages 357–361, 2001.

[61] V. Singhvi, A. Krause, C. Guestrin, J. H. Garrett, and H. S. Matthews. Intelligent light control using sensor networks. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[62] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5):16–27, October 2000.

[63] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. *The Computer Journal*, 47(4):448–460, 2004.

[64] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Trans. Computer*, 52(5):545–557, 2003.

[65] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai. Wireless sensor networks for emergency navigation. *IEEE Computer*, 39(7):55–62, 2006.

[66] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *Proc. of IEEE Global Telecommunications Conference (Globecom)*, 2003.

[67] X. Wang, F. Silva, and J. Heidemann. Infrastructureless location aware configuration for sensor networks. In *Proc. of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2004.

[68] Y.-J. Wen, J. Granderson, and A. M. Agogino. Towards embedded wireless-networked intelligent daylighting systems for commercial buildings. In *Proc. of IEEE Int'l Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 2006.

[69] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.

[70] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proc. of IEEE INFOCOM*, 2005.

[71] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2002.

[72] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2004.

[73] H. Zhou, M. W. Mutka, and L. M. Ni. Reactive ID assignment for sensor networks. In *Proc. of IEEE Mobile Adhoc and Sensor Systems Conference (MASS)*, 2005.

[74] ZigBee specification version 2006, ZigBee document 064112, 2006.

# Vita

## 潘孟鉉 Meng-Shiuan Pan

**Contact Information**
Department of Computer Science
National Chiao Tung University
1001 Ta Hsueh Road, Hsinchu, Taiwan 300
Email: mspan@csie.nctu.edu.tw

## Education

Ph.D.: Computer Science, National Chiao Tung University (2003.9 ∼ 2008.5)
M.S.: Communication Engineering, National Tsing Hua University (2001.2 ∼ 2003.6)
B.S.: Electrical Engineering, National Chung Cheng University (1997.9 ∼ 2001.1)

## Awards

1. 教育部通訊競賽優等獎, "Indoor Security and Emergency Navigation Services by Wireless Sensor Networks", with Y.-Y. Tsai, C.-H. Tsai, C.-F. Huang, and Y.-C. Tseng 2005.

2. 入選第四屆 ACM MobiWAC 國際會議 Best paper award 候選名單, 2006.

3. 交通大學資訊工程所傑出研究獎, 2006.

4. 交通大學傑出教學助理, 2007

5. 第一屆智慧化居住空間情境模擬創作競賽第三名, "以無線感測網路為基礎之智慧型室內燈光調節系統", with 葉倫武, 廖家良, 林育萱, 曾煜棋, 2007

# Publication Lists

## Journal papers

1. Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai, "Wireless Sensor Networks for Emergency Navigation", *IEEE Computer*, vol. 39, no. 7, pp. 55- 62, July 2006. (SCI, EI)

2. M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng, "Emergency Guiding and Monitoring Applications in Indoor 3D Environments by Wireless Sensor Networks", *International Journal of Sensor Networks*, vol. 1, nos. 1/2, pp. 2-10, 2006.

3. M.-S. Pan and Y.-C. Tseng, "Quick Convergecast in ZigBee Beacon-enabled Tree-based Wireless Sensor Networks", *Computer Communications* (accepted). (SCIE, EI)

4. M.-S. Pan, L.-W. Yeh, Y.-A. Chen, Y.-H. Lin, and Y.-C. Tseng, "A WSN-Based Intelligent Light Control System Considering User Activities and Profiles", *IEEE Sensors Journal* (accepted). (SCIE, EI)

## Conference papers

1. M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng, "Implementation of an Emergency Guiding System by Wireless Sensor Networks", *IEEE Int'l Symposium on Network Computing Applications (NCA)*, 2006. (Fast Abstract)

2. Y.-C. Tseng and M.-S. Pan, "Quick Convergecast in ZigBee/IEEE 802.15.4 Tree-Based Wireless Sensor Networks", *ACM Int'l Workshop on Mobility Management and Wireless Access (MobiWAC)*, 2006. (selected as a candidate of the Best Paper Award) (EI)

3. M.-S. Pan and Y.-C. Tseng, "The Orphan Problem in ZigBee-based Wireless Sensor Networks", *ACM/IEEE Int'l Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2007.

4. M.-S. Pan, H.-W Fang, Y.-C. Liu, and Y.-C. Tseng, "Address Assignment and Routing Schemes in ZigBee-Based Long-Thin Wireless Sensor Networks", *IEEE VTC-spring*, 2008.

5. M.-S. Pan and Y.-C. Tseng, "Communication Protocols and Applications for ZigBee-Based Wireless Sensor Networks", *The Fourth Taiwanese-French Conference on Information Technology*, 2008.

6. L.-W. Yeh, M.-S. Pan, and Y.-C. Tseng, "Minimum Delay Two-Way Beacon Scheduling in ZigBee/IEEE 802.15.4 Tree-Based Wireless Sensor Networks", *IEEE Int'l Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 2008.

7. M.-S. Pan, L.-W. Yeh, Y.-A. Chen, Y.-H. Lin, and Y.-C. Tseng, “Design and Implementation of a WSN-Based Intelligent Light Control System”, *IEEE Int’l Workshop on Wireless Ad hoc and Sensor Networks (WWASN)*, 2008.

## Book Chapters

1. M.-S. Pan and Y.-C. Tseng, “ZigBee Wireless Sensor Networks and Their Applications”, *Sensor Network and Configuration: Fundamentals, Techniques, Platforms, and Experiments*, Springer-Verlag.

2. Y.-C. Tseng, M.-S. Pan, and C.-W. Yi, “Wireless Sensor Networks”, *McGraw Hill 2008 Yearbook of Science & Technology*, McGraw Hill.

## Patents

1. M.-S. Pan, L.-W. Yeh, Y.-C. Tseng, L.-C. Ko, H.-W. Fang, and C.-W. Teng, “Apparatus for a Beacon-enabled Wireless Network, Transmission Time Determining Method, and Computer Readable Medium Thereof”, Taiwan (pending), USA (pending), owned by III.

2. T.-C. Chien, M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng, “Dynamic Emergency Guiding Device, Method, and System”, Taiwan (pending), owned by NTHU.

3. M.-S. Pan, Y.-A. Chen, T.-C. Chien, Y.-F. Lee, and Y.-C. Tseng, “Automatic Lighting Control System and Method”, Taiwan (pending), USA (pending), China (pending), owned by ITRI.

4. H.-W. Fang, Y.-C. Liu, M.-S. Pan, Y.-C. Tseng, “Long Thin Network Routing and Address Assignment Methods”, Taiwan (pending), USA (pending), China (pending), owned by III.

5. M.-S. Pan, Y.-C. Tseng, H.-W. Fang, Y.-C. Liu, “Address Assignment and Routing Methods for Ultra Long Thin Networks”, Taiwan (pending), USA (pending), China (pending), owned by III.

## Book (in Chinese)

1. 曾煜棋、潘孟鉉、林致宇,“無線區域及個人網路：隨意及感測器網路之技術與應用”,知城科技, ISBN: 9868293316.

## Submitted papers

1. M.-S. Pan and Y.-C. Tseng, “The Orphan Problem in ZigBee-based Wireless Sensor Networks”, submitted to IEEE Trans. on Mobile Computing.

2. M.-S. Pan and Y.-C. Tseng, “ZigBee-Based Long-Thin Wireless Sensor Networks: Address Assignment and Routing Schemes”, submitted to IEEE Trans. on Wireless Communications.