

國立臺灣大學電機資訊學院電信工程學研究所



碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

物聯網內以拓樸為基礎之自動設定演算法

Topology based Automatic Configuration
for the Internet of Things

鄧安哲

An-Che Teng

指導教授：周俊廷 博士

Advisor: Chun-Ting Chou, Ph.D.

中華民國 106 年 6 月

June 2017

Topology based Automatic Configuration for the Internet of Things



A Thesis
Presented to
The Academic Faculty

by

An-Che Teng

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in
Graduate Institute of Communication Engineering

National Taiwan University

January 2017

致謝

從台南到台北，兩年的碩士生涯在六月劃下了句點。軟體與開發，研究與創新，實實在在的兩年讓我受益匪淺。感謝爸爸、媽媽給予強而有力的後援，讓我不管在哪念書都能專注在學業。真的很謝謝你們。

非常感謝周俊廷教授兩年來的指導。從老師身上學到的遠遠超過課本上的知識。除了專業領域上的指導，學到更多的是如何成為一個好的工程師、好的設計師與一位好的銷售員。邏輯思考、簡報呈現、自我銷售等書本無法給予的軟技能，是除了專業知識以外更寶貴的訓練。也相當感謝老師在學習資源上所給予的支持，若沒有碩一的各種線上課程訓練，相信也不會有現在的我。

在實驗室中，與大家相處受到很多啟發。謝謝學長們的教導與教誨，首先謝謝俊佑，雖然隔了好幾屆，你依然不厭其煩的指導我各種系統與實作上的疑難雜症，即使你到了美國依然是實驗室的守護神。謝謝瑋辰當年的實驗室講解與之後常常一同對軟體開發進行討論，至今仍收穫良多。謝謝柏州，一起修課、做計畫，聽你講起創業的故事總是能激勵我們這些小學弟；你永不放棄的毅力與堅持、海綿般吸收知識的動力依然影響著我。謝謝承翰，從進實驗室就常和你討論程式開發，你清晰的邏輯與可靠的開發速度是我們在風雨飄搖的計畫中的燈塔。謝謝秉諭，有你在的地方永遠有歡笑與助理，許多疑難雜症總是需要問上知天文下知地理的你，而且答案永不落空。謝謝翔云，沉穩的你一直都是實驗室的避風港，也不曾吝於指導我們這些學弟，謝謝你對我們的照顧。謝謝乃文，謝謝妳總是供應實驗室糖果，很多時候這實在是太重要了。

同屆的夥伴們，謝謝恆哲，只要你力所能及的事情總是二話不說地幫忙，從計畫、一起當助教到解決各種問題，有你幫忙的時候就是安心。謝謝加富，講義氣的你幫了大家許多忙，很多需要揪團的時候也常常麻煩你。你認真的研究態度與思考，對我來說值得敬佩與參考；同時你廣泛的閱讀也讓我心中的書魂不曾熄滅。謝謝哲宇，你真的很好笑。Thank you Mathis, we are buddies! I learned a lot from you, and different cultures are really interesting. Thank you! 學弟們，謝謝理文，雖然你沒有基礎但依然想辦法把東西學好，辛苦你了。謝謝乃瑄，接下了嵌入式的開發與維護，每天的待命與呼叫也實在是辛苦你了，相信這會變成養分，讓你變成更好的工程師。謝謝以彥，在爐石上受了你相當多的指導，實在太猛了，在往後的資料與軟體之路請繼續加油。謝謝維哲，時常討論未來的發展與技能，認真的你除了工作與學業外記得要保持眼睛的健康，另外茶茶與可可真的很可愛。謝謝柏鈞，惦惦呷三碗公，感覺每天都很開心。加油啊大家！

最後感謝朋友們的支持，特別感謝無反射室的何賢奎博士給予的幫助。謝謝女朋友，謝謝妳在我忙碌的時候能體諒、疲憊的時候能安撫我的內心，碩士兩年謝謝妳的照顧，有妳的陪伴真的很開心。再次感謝你們，謝謝你們，這兩年來，謝謝。

中文摘要



在未來的無線通訊領域中，物聯網將成為下一個世代中的主角。不同於以往的人對人通訊，物聯網是由數以萬計的裝置和物件所連通而成的網路。也因為這龐大的數量，如何設定、管理與維護此難以想像的龐大網路成為了實現物聯網的關鍵。很明顯的，傳統人工或半自動的解決方案無法有效的解決上述問題。

在許多設定與管理的物聯網議題中，知曉某一已知位置上安裝的裝置之虛擬邏輯地址非常重要。舉例來說，使用者期望鄰近廚房電燈之開關能夠控制廚房之電燈，而此家用物聯網路之控制邏輯需要同時知道開關和電燈的實際位置和網路地址。雖然以人工的方式設定能解決小規模的實際位置和網路地址隻配對問題，但對於大規模之物聯網路來說，人工的方式依然無法規模化且不便進行管理

為了解決此問題，我們提出了分散式且可規模化的基於拓樸之自動設定演算法。有別於傳統以定位演算法之解決方案，此配對問題被我們視為一種排序問題。在我們的演算法下， N 個已知邏輯地址的裝置根據鄰近裝置之無線訊號資訊進行排序成為一個序列，而 N 個已知的實際位置根據安裝拓樸排序而成另一個序列。因此，此問題從二維配對問題轉化為一維的排序問題，相較於傳統定位演算法與圖像演算法來說進而更加簡單、更能被規模化以及具有更低的複雜度。

我們所研究模擬的結果顯示，在網狀拓樸中，我們提出的演算法在高達 0.8 測量誤差 (Measurement Error Rate, MER) 的條件限制下，能夠達到 80% 的成功率；而基於定位之三角定位演算法與 MDS-ICP 演算法只能在 0.5 與 0.2 的較低測量誤差下達到 80% 的成功率。在其他拓樸中我們的演算法依然較為可靠且複雜度更低。為了證明演算法在實際室內環境的可應用性，我們設計了類環令牌且基於 Simple Flooding 之通訊協定，並將其實作於安裝了 IEEE 802.15.4 模組與 STM32F0 晶片之 FCM2401 上。實驗結果也顯示我們的演算法於現實環境中運行良好且有效解決自動設定之問題。

關鍵字：物聯網、自動設定

ABSTRACT



The Internet of Things (IoT) is becoming main drive of the growth of wireless networks. Different from the human-to-human communication, IoT is a huge network that connects an enormous amount of physical objects. As a result, how to configure, manage, and maintain such a huge network becomes the key to realizing IoT. Obviously, the traditional manual or semi-automatic solutions will not be applicable to the emerging IoT networks.

Among many configuration/management issues in an IoT network, knowing the network (logical) address of a device installed at a certain physical position is important. For example, one would like a switch on the wall near the entrance of a kitchen to control the light fixtures in the kitchen. The control logic of this home network would need to know the network addresses of the switch and light fixtures at these two specific locations. Although one can manually provide the logical address and location "tuple" of each device to the control logic, such a solution is not scalable for an IoT network.

In this thesis, we propose a distributed, and scalable algorithm to solve this problem. Unlike the traditional positioning approaches, the problem is treated as a sorting problem. With the help of our algorithm, N known logical addresses (i.e., MAC addresses) are sorted as a sequence based on radio neighborhood information collected by the devices, while the N known physical addresses (installation locations) are sorted as another sequence based on the installation topology. As a result, the problem becomes a one-dimensional problem and is more scalable and simpler, in terms of computation complexity, than the traditional positioning-based or image-based approaches.

The simulation results show that our algorithm can achieve 80% success rate with 0.8 Measurement Error Rate (MER), while the trilateration and MDS-ICP can achieve the same success rate with only 0.5 MER and 0.2 MER, respectively for a grid topology. In order to demonstrate the feasibility, we also implement this algorithm in IEEE 802.15.4 wireless modules using a simple flooding protocol with a majority voting rule. The experiment results show that the algorithms work well in a real indoor environment.

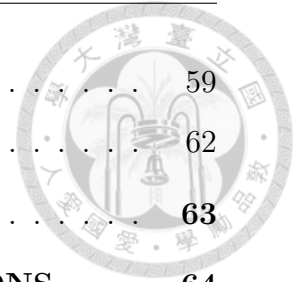
Keywords: the Internet of Things, automatic configuration

TABLE OF CONTENTS



ABSTRACT	ii
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER 1 INTRODUCTION	1
1.1 The Evolution of Internet of Things	1
1.2 Problem Statement	8
1.3 Contributions	11
1.4 Thesis Organization	11
CHAPTER 2 RELATED WORK	13
2.1 Trilateration Based Methods	13
2.2 Multidimensional Scaling based Methods	15
2.2.1 Multidimensional Scaling Algorithm	15
2.2.2 Iterative Closest Point Algorithm	17
2.2.3 Summary of the MDS-ICP algorithm	19
2.3 Summary	19
CHAPTER 3 PROPOSED SCHEME	21
3.1 System Setting	21
3.2 Concepts of the Algorithm	21
3.3 Topology-based Automatic Configuration Algorithm	23
3.4 Majority Mechanism	37
3.5 Anchor Location Determination Algorithm	38
CHAPTER 4 SIMULATION, ANALYSIS AND IMPLEMENTATION 45	
4.1 Simulation Settings	45
4.2 Performance among different locations of anchors	49
4.3 Performance among different algorithms	51
4.4 Analytical Performance Analysis	56

4.5	Protocol and Implementation	59
4.6	Summary	62
CHAPTER 5	CONCLUSIONS	63
APPENDIX A	— ALGORITHM TO FIND ALL REGIONS	64
APPENDIX B	— EXAMPLES OF OTHER TOPOLOGIES	73



LIST OF TABLES



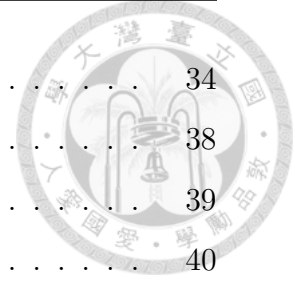
1	An example of the RSSI matrix between devices	16
2	Comparison between two algorithms	20
3	An example of the majority mechanism	37
4	Performance metrics among three algorithms	62

LIST OF FIGURES

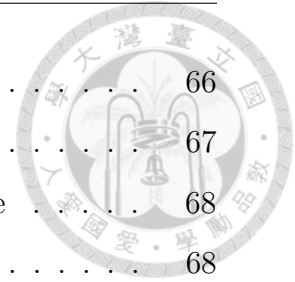


1	Smart light- location configuration example	6
2	An example of manual configuration with QR code	7
3	An example of the matching problem	8
4	Two sets of device-IDs and locations	9
5	An example of space information	10
6	Trilateration	14
7	An example of how trilateration solve the location configuration problem	15
8	The example result of multidimensional scaling with Table 1	16
9	An example of one iteration of the ICP algorithm	18
10	Key idea	22
11	Serialization of the location set - Initialization	23
12	Serialization of the location set - Find sequence number 1	24
13	Serialization of the location set - Find sequence number 2	25
14	Serialization of the location set - Find sequence number 3	25
15	Serialization of the location set - Find sequence number 4 (part1)	26
16	Serialization of the location set - Find sequence number 4 (part2)	27
17	Serialization of the location set - Find all sequence numbers	27
18	Simplified ambiguity list	28
19	Serialization of the device set - Initialization	29
20	Serialization of the device set - Find sequence number 1	30
21	Serialization of the device set - Find sequence number 2 (part1)	30
22	Serialization of the device set - Find sequence number 2 (part2)	31
23	Serialization of the device set - Find sequence number 4 (part1)	32
24	Serialization of the device set - Find sequence number 4 (part2)	32
25	Serialization of the device set - Find sequence number 4 (part3)	33
26	Serialization of the device set - Find all sequence numbers	33

27	Two correspondent sets	34
28	An example of the location of the anchor	38
29	The perpendicular bisector of two locations	39
30	All perpendicular bisectors for Figure 28	40
31	The definition of a region	40
32	Local max-min point - Case 1	41
33	Local max-min point - Case 2-1	42
34	Local max-min point - Case 2-2	42
35	Colored regions and best anchors	44
36	Relationship between MER and α	47
37	Relationship between MER and distance	48
38	Relationship between MER and p	48
39	Relationship between MER and σ	49
40	Picked locations of anchors	50
41	Performance among three different locations of anchors	51
42	Locations of anchors and devices for three algorithms	52
43	Comparison among three methods without Majority	54
44	Comparison among three methods with majority number=500	54
45	Comparison among three methods with majority number=1000	55
46	Comparison among three methods with majority number=5000	55
47	The reason why our algorithm is better	56
48	Comparison between simulation and analytical result	58
49	All $P(\text{Success}_{i \rightarrow i+1})$, $i = 1, 2, \dots, 7$	59
50	The concept of the protocol	60
51	The details of the protocol	60
52	FCM2401 Module	61
53	The implementation and indoor environment	61
54	Vertices, edges and regions	65
55	Simplified example for the adjacency matrix	66



56	The way to find all triangles	66
57	Remove degrees of edges	67
58	The graph and the adjacency matrix after extract a triangle	68
59	Fold the [3, 2] and [3, 4]	68
60	Fold the [5, 4] and [5, 6]	69
61	Extract the edge [3, 6]	69
62	The folded graph and folded adjacency matrix	70
63	The algorithm to find all regions	71
64	A more complicated topology	71
65	Recursively find all regions	72
66	Final result of finding the max-min point	72
67	Example of topology 1	73
68	Example of topology 2	74
69	Example of topology 3	75
70	Example of topology 4	76
71	Example of topology 5	77



CHAPTER 1



INTRODUCTION

The Internet of Things (IoT) is a huge network of enormous physical objects which can communicate with each other. Tremendous physical objects are being connected to the Internet to realize the concept of IoT. These physical objects are able to sense the surroundings, analyze data, and perform jobs collaboratively. IoT is also regarded as an extension of the Internet, expanding the communication from humans and humans to machines and machines. Everything which is not only physical but also virtual is possible to be connected by the IoT.

1.1 The Evolution of Internet of Things

IoT is evolving from the vertical applications to the integrated applications [1]. In the beginning, the most of IoT applications are domain-specific. For example, automobile manufacturers start to install sensors and robots in the assembly line of factories. These sensors and robots communicate with one another to collaboratively assemble the car, analyze the quality and transmit all data produced during the work to engineers and managers. The network of sensors and robots in the factories only serve their company and form a vertical application. A domain-specific application may only solve the problems with its own industry. This is the early stage of IoT.

Integrated applications are cross-industry applications based on different public information platforms and IoT architectures. These applications serve both individuals and enterprises. For example, future road-traffic systems may be composed of vehicle-to-vehicle networks, sensor networks on the road and global positioning systems. These future systems can support not only individuals who drive on the

road but also companies which may provide self-driving cars to carry passengers. Moreover, through a public information platform of this systems, automobile manufacturers, maintenance providers, and vehicle management agencies can share all these data to improve the vehicles and promote the safety of the road-traffic.

With evolving into the integrated applications, cooperating with other industries and creating more value are the current trend. Many countries consider IoT as strategic industries and chances to stimulate the economy. European Union (EU) has already spent more than 100 million Euros on IoT technologies. These investments will be used in the smart grid, intelligent transportation, smart cities, etc. South Korea also invested more than 20 million U.S. dollars in IoT fundamental technologies and researches. China even proposed the 12th Five-Year Plan for IoT development, and a total of 45 billion U.S. dollars will be invested for smart cities with more than 200 cities selected [2].

The various IoT applications are the key points to stimulate the economy. In addition to the applications mentioned above (smart cities, smart grid and intelligent transportation), here are some other typical IoT applications: 1) smart industry; 2) smart agriculture; 3) smart logistics; 4) smart home; 5) healthcare; 6) environment protection [1]. These applications are different from each other, but most of them consist of the following common capabilities:

1. Sensing

IoT systems can sense the environment with physical or chemical phenomena around the sensors. Typical sensing information includes temperature, humidity, velocity, light, motion and the magnetic fields. The sensors make the physical objects perceive the surroundings and have the ability to response to events around them. On the other hand, sensing information can also help the configuration of networks through the sensed data. It is the basic capability of IoT and has already been used widely in various scenarios before IoT showed up.



With the sensor technology advances, more and more different phenomena can be sensed and the precision of sensors will be better.

2. Remote Controlling

Not all IoT applications need to remotely control the physical objects, but this capability can not be replaced. Remote controlling enables the IoT systems to react to the surroundings physically. For example, smart home systems may remotely control the lights through the cloud server or users.

3. Location Configuration

Almost every IoT systems need to know the location information of physical objects to label the data sensed by the sensors because the data without the location information are too raw to analyze and create value. The location information can be dynamic or static. It depends on the requirements of IoT applications. For example, smart lights in the smart industry need to know their static location information because devices with only virtual addresses cannot be controlled by humans. The location information can obtain from GPS, RFID, cellular networks, absolute or relative position information between objects.

4. Networking

Networking is another important capability of IoT. Sensors need to transmit data to other sensors, gateways, and even remote servers. People also want to control the remote devices and send commands to do jobs. IoT systems must have network configuration capability to interoperate with each other and connect to the backbone network to provide different services.

The technologies of the four capabilities have existed for a long time. Sensor technologies are still evolving, and they are mature enough to support the current IoT scenarios. The most of the remote controlling requirements are also satisfied. However,

some of the requirements and scenarios of IoT are different from the past. Networking becomes more complicated because the infrastructures are more distributed, and the capability of devices are limited. The limited communication capability makes the configuration of a network more difficult. Also, the location configuration of tremendous sensors still, cannot be automated. The following will elaborate the challenges and constraints for realizing IoT [3].

- Scalability and Performance

The scalability of IoT is about adding new devices, services, and functions to systems without negatively affecting the current systems. The diversity of IoT is the difficult part to achieve the scalability. Different devices, architectures, and protocols now are not easy to build systems with the scalability because of the lack of standards. Another difficult part is about the limited capabilities of devices. The most of the devices in IoT only have limited resources. Energy, computation resources, wireless radio coverage, etc. are typical constraints of the IoT. These constraints affect not only the network performance but also the accuracy of location configuration. How to achieve the scalability and improve the performance are always important for IoT.

- Reliability and Availability

The availability means anytime and anywhere services, and the reliability refers to the high success rate of IoT service delivery. The two challenges must be implemented in both hardware and software. In IoT, hardware shall exist all the time and work normally, and software shall be able to provide services for huge amounts of users. For example, the communication networks of emergency response applications must be robust and can recover from the failure quickly. Also, the locating capability must be reliable to help the system work correctly and accurately. How to achieve the requirements above are still a challenge for



IoT.

- Management and Configuration

The management is the challenge due to the transformation of quantitative into qualitative changes. Billions of IoT end-devices will be deployed, and the nightmare of managing tremendous devices come along. For example, we know that monitoring the fault of connectivities among devices can control the reliability of services, but tracking a device in a huge distributed sensor network is difficult. We cannot directly get the information about the certain device because the radio coverage is limited. The data must be relayed from the device to monitors, and this process will consume power and cause latency.

The configuration is another potential problem for the deployment of IoT. Configuring the information of IoT applications among thousands of devices is also a nightmare. If we know the locations of devices, automating the configuration of devices is a promising way to solve the problem. However, the locations of devices also need to be configured, but the current technologies are still too immature to be adopted. New light-weight management protocols and configuration protocols are necessary and indispensable for IoT.

As mentioned above, the locations of devices are the key point for management and configuration. Figure 1 is an example problem about the location configuration.

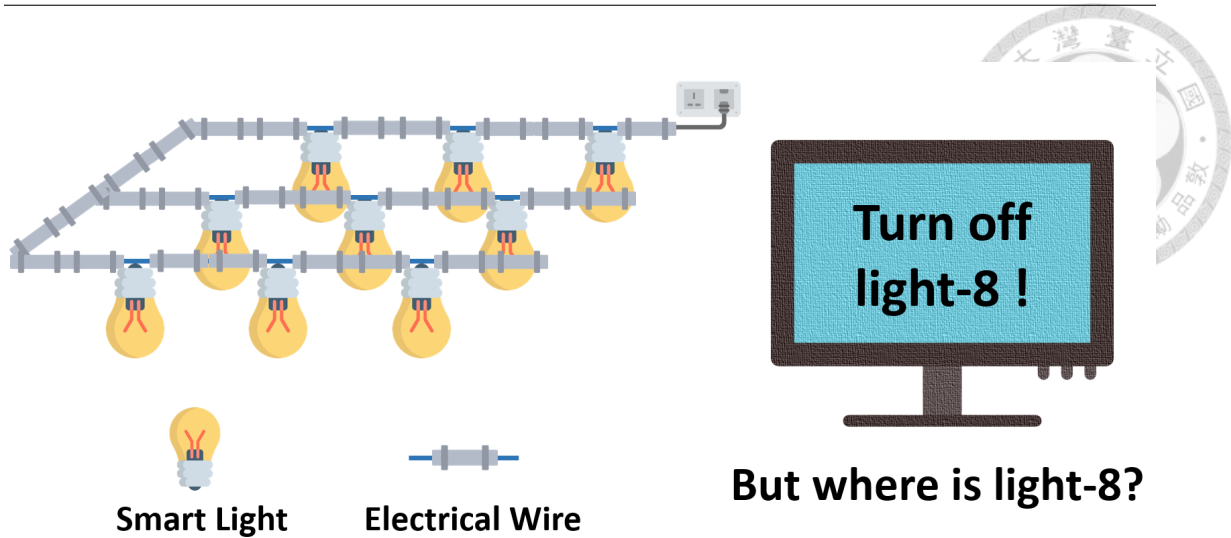


Figure 1: Smart light- location configuration example

Users get the message from the control panel and know that light-8 should be turned off. If users only have this message without any other information, they still do not know where the light-8 is. An intuitive solution is configuring the locations of devices manually. The mappings between cyberspace addresses and physical location information can be done by humans, and there are some technologies can make the location configuration easier, such as Quick Response Code (QR Code), Radio Frequency Identification (RFID), Near Field Communication (NFC) and Bluetooth Tag. Figure 2 shows how QR code can help the location configuration.

QR code can store cyberspace addresses information such as device IDs. Installation personnel can use QR code scanners to get the information and mark down the locations of devices. Then the QR code scanners can transmit the mapping between the location and the device ID to cloud servers, and the problem is solved.

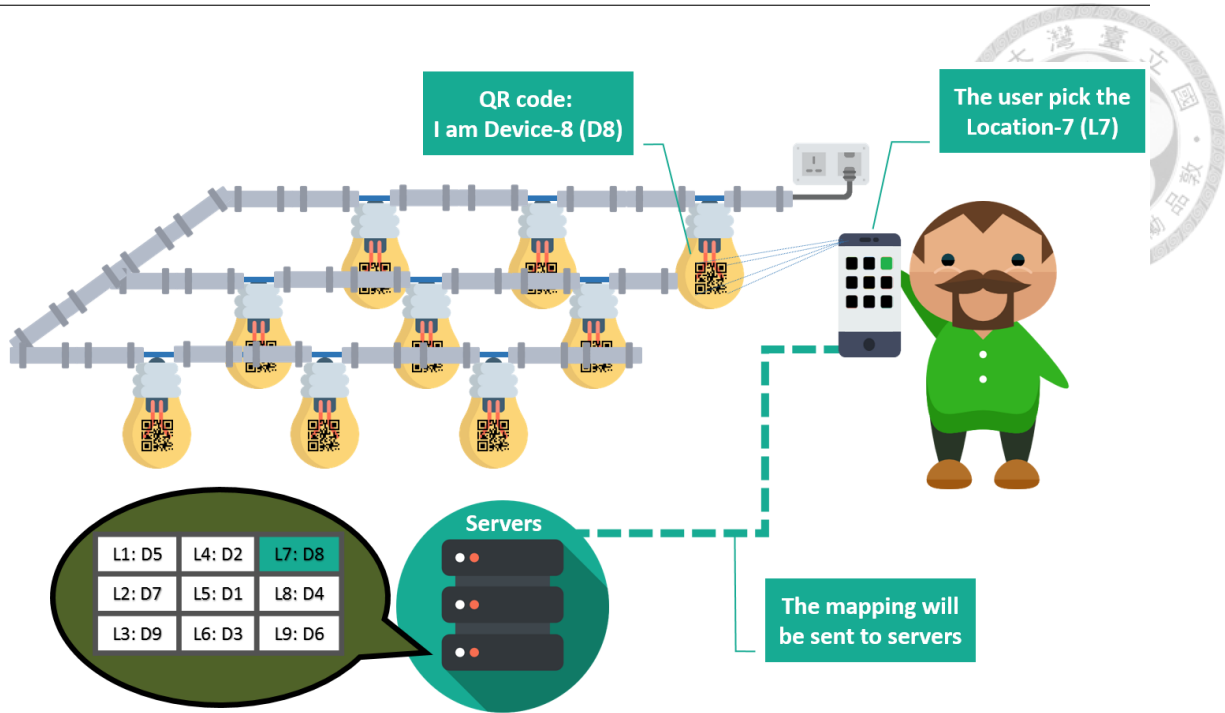


Figure 2: An example of manual configuration with QR code

However, the reliability, efficiency, scalability and manageability of manual configuration have some problems. First, the installation personnel should be trained before they start to do configuration and we cannot guarantee that all devices can be configured correctly by humans. Furthermore, the difficulty of manually configuring an IoT application is proportional to the number of devices. Second, when we need to replace the broken devices, we must redo the all configuration procedures manually. This will be a nightmare when companies deploy systems in our home because users may not be able to do configuration with new devices, especially for elders and patients. Thus, for all reasons above, we need an automatic location configuration method.

In this thesis, we want to solve the location configuration problem with an automatic location configuration algorithm. The location is the key attribute of IoT introduced in the previous section, and also important for the management of IoT. However, in contrast to the positioning problem, the location configuration problem

is ignored for a long time during the development of IoT. **Location configuration problems are not positioning problems.** IoT applications can know the deployment topology in advance no matter whether the topology is regular or irregular. Developers can use the known deployment topology to assist the location configuration, therefore, the location configuration problem becomes a “matching problem”: We know all device-IDs and all locations, but we do not know which device is located in a certain location, as shown in Figure 3. The next chapter will elaborate the problems and introduce some typical solutions for the configuration of locations.

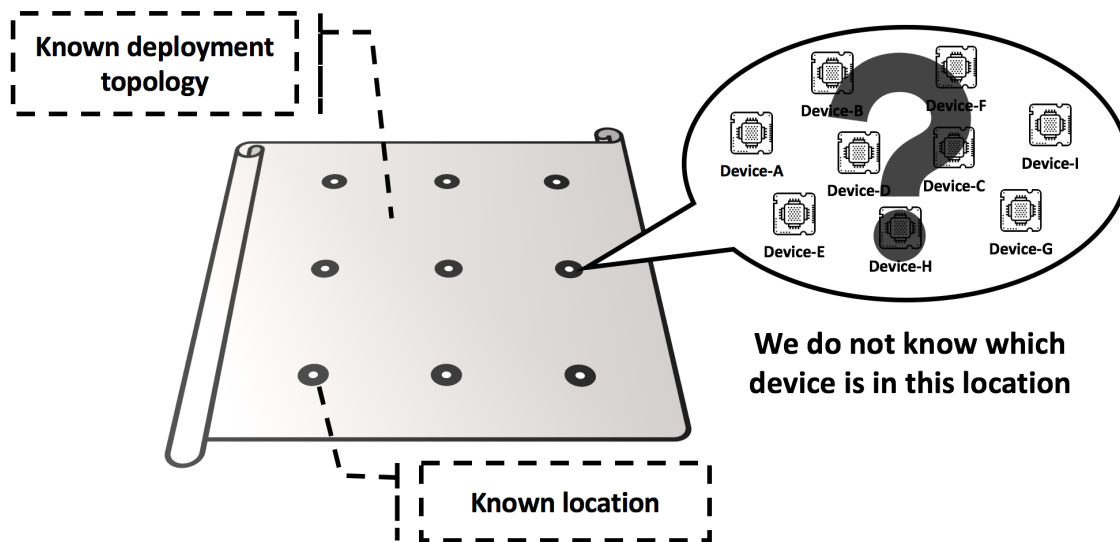


Figure 3: An example of the matching problem

1.2 Problem Statement

As mentioned in the previous section, location configuration is a “matching problem”: All device-IDs and all locations are known, but we do not know which device is located in a certain location. We can consider all device-IDs as one set, and all locations as the other set, as shown in Figure 4.

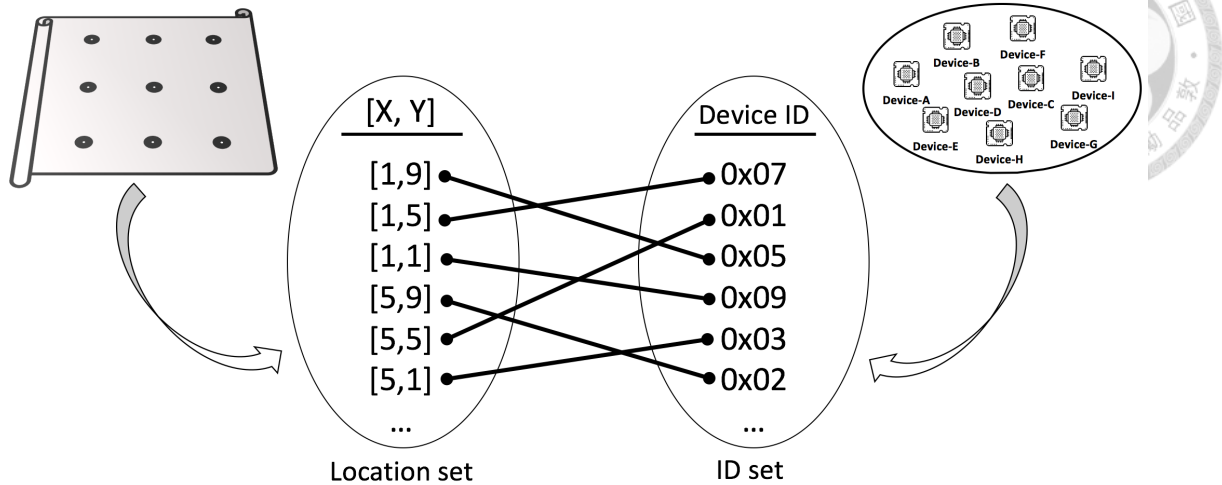


Figure 4: Two sets of device-IDs and locations

The number of the two sets must be the same because each device can only have one location. In other words, each element in the device-ID set is only mapped to one element in the location set. Thus, the location configuration is a one-to-one mapping problem with the two sets. Assume there are N elements in each set. There are total $N!$ combinations, and only one of them is the correct answer. The goal of automatic location configuration algorithms is to find that correct answer.

We can use some space information such as distance to help the location configuration. Figure 5 is an example. $L1, L2, \dots, L9$ represent nine locations, and $D1, D2, \dots, D9$ represent nine devices. The left of the dotted line is the correct mapping between locations and device-IDs, and the right of the dotted line is a simplified example of what we guess. Assume the devices can measure distances between other devices. For $D5$, the measured distance from $D1$ may be twice as large as the measured distance from $D6$, so devices can know that the guess is wrong.

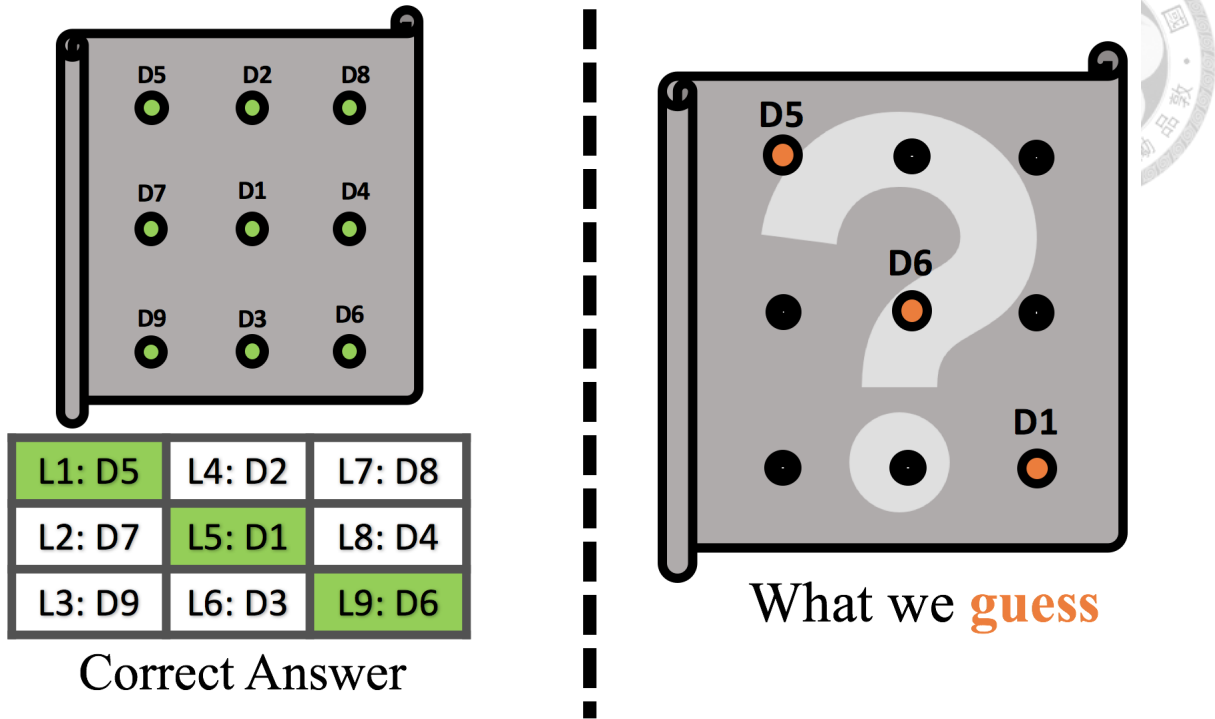


Figure 5: An example of space information

Distances between devices can really help us find the correct mappings, but how to measure distances is another problem. Using additional distance measurement modules such as ultrasound or infrared for location configuration is not cost-efficient because usually we only need to do location configuration once. On the other hand, most of the devices in IoT are equipped with wireless communication modules. Thus, the only thing we can use for free is Received Signal Strength Indicator (RSSI), even though RSSI has its limitations. Converting RSSIs into distances is very difficult because 1) we may not know the current wireless channel model of the environment [4], 2) even we know the model, the coefficients of the model for different environments are not the same so we always need to do massive measurements to estimate the model [5]. In spite of the disadvantages of RSSI, we still want to use it because it is the only thing we can use for free.

Although positioning problems are not the same as location configuration problems, typical positioning solutions provide many ways to use RSSI effectively. The next chapter will introduce typical solutions which can be used to solve location configuration problems.

1.3 Contributions

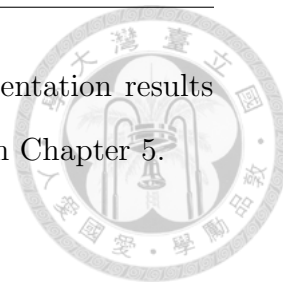
The main contributions of this thesis are listed as follows:

- Proposed a novel automatic configuration algorithm to solve the location configuration problem. In our designs, the algorithm only needs one anchor and do not need to convert RSSIs to distances. The algorithm can handle any topology and the complexity of the algorithm can be only $O(n)$. With the grid topology, the algorithm can achieve 80% success rate with the measurement error rate under 0.8, which is much better than other algorithms.
- Developed a method for choosing the optimized parameters of the algorithm. Simulated the performance of the proposed algorithm and compare the performance to trilateration and MDS-ICP. The simulation results show that, with the same success rate, our algorithm can tolerate much higher measurement error rate than trilateration and MDS-ICP.
- Implemented the proposed algorithm with a distributed protocol on the board STM32F0 which equipped with IEEE 802.15.4 wireless module.

1.4 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we will elaborate the problems of location configuration and introduce typical solutions which are used for automatic configuration and positioning. The proposed algorithm designs are

introduced in Chapter 3. In Chapter 4, the simulation and implementation results are presented. Finally, the conclusions and future works are drawn in Chapter 5.



CHAPTER 2



RELATED WORK

Location information is one of the most important information for IoT. Most of the devices in IoT need both a cyberspace address and a physical location information, and the two information need to be combined to provide services. In this chapter, we will introduce several possible solutions for location configuration. We will summarize pros and cons about the possible solutions and figure out the technical challenges in the end.

2.1 Trilateration Based Methods

Trilateration is the process of determining locations by measuring the distances between the anchor node and the target with the geometry of circles, spheres or triangles [6] [7] [8] [9] [10]. Anchor nodes here refer to the devices which have prior knowledge of their absolute positions, as shown in Figure 6. It is the most popular method used in positioning technologies and there exist many practical applications with trilateration.

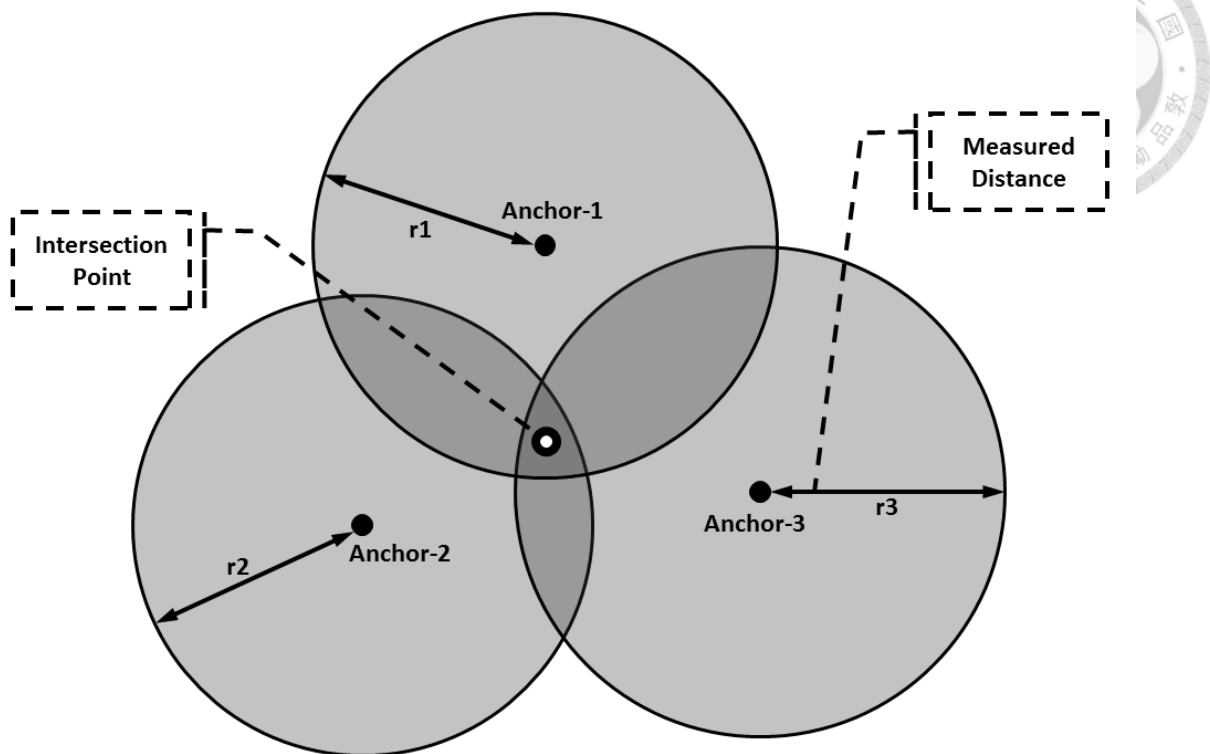


Figure 6: Trilateration

Figure 7 is an example of how trilateration can solve the location configuration problem. The system now wants to map the device D1 to a certain location. With the help of three anchors, D1 can use trilateration to estimate a location. Trilateration just gives us an estimation, not a mapping, so here we choose the closest known location L5 which has not mapped to another device as the mapped location of D1.

The advantages of trilateration are the relatively high accuracy and the relatively low complexity. Trilateration just need to get three distances and three locations of anchors to calculate the estimated location (with the complexity of $O(1)$), then choose a closest known location as the mapped location (with the complexity of $O(n)$). Because all devices need to do the same process, thus the total complexity is $O(n^2)$. However, the disadvantage of trilateration is fatal. Converting RSSIs into distances is too difficult as mentioned in the Section 1.2, and in most of the time, it

requires more effort than the manual location configuration. This is also the reason why trilateration is seldom used for location configuration problems.

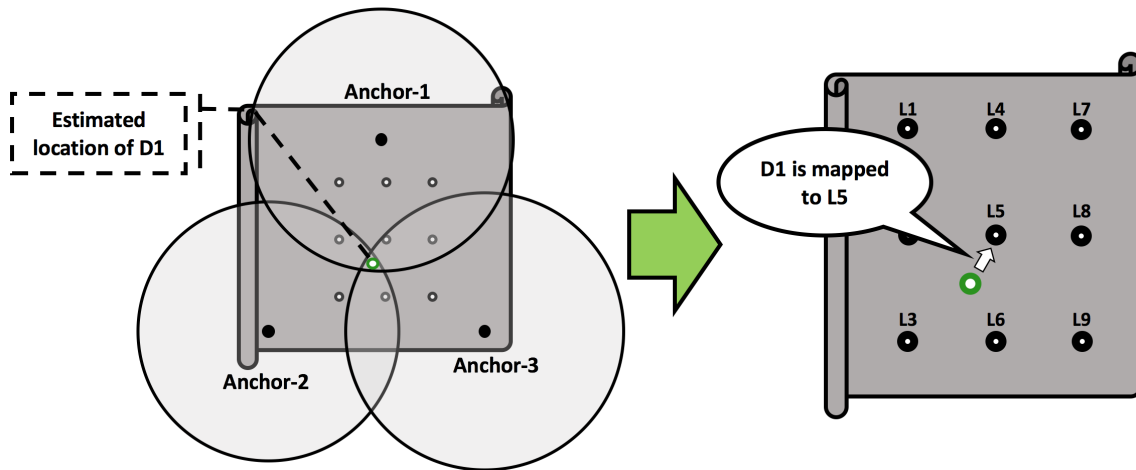


Figure 7: An example of how trilateration solve the location configuration problem

2.2 *Multidimensional Scaling based Methods*

In this section, we will introduce the multidimensional scaling (MDS), a well-known algorithm for data analysis and visualization. However, because of the limitations of MDS, it cannot completely solve the location configuration problem. Thus, we will also introduce a typical point registration algorithm, Iterative Closest Point (ICP) Algorithm, which is widely used in image registration problems to help MDS solve the location configuration problem.

2.2.1 Multidimensional Scaling Algorithm

Multidimensional Scaling (MDS) is widely used in data analysis to represent measurements or similarity among pairs of objects through the “distance” between objects. For example, the “distances” can be correlations among the attributes of objects, or some similarity measurements distances such as Euclidean distance, Manhattan distance, Mahalanobis distance, etc. The MDS representation is a plane with

Device	D1	D2	D3	...	D9
D1	0	189	184	...	171
D2	186	0	165	...	132
D3	188	159	0	...	180
...
D9	176	143	188	...	0



Table 1: An example of the RSSI matrix between devices

points which represent the objects. The points on the plane are closer, the correlation is more positive. Therefore MDS are usually used to visualize the relation between measurement targets. More details about theories and applications of MDS are available in [11].

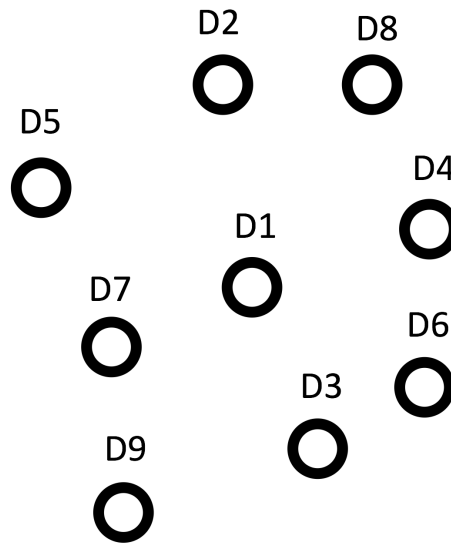


Figure 8: The example result of multidimensional scaling with Table 1

RSSI can be viewed as a similarity of distance. Using RSSIs as the physical relation among devices and calculating the MDS are another typical way to locate the positions [12] [13], as the example shown in Table 1 and Figure 8. The MDS result will be the similarity of the real topology, and the similarity depends on the measurement errors and conversion errors.

As we mentioned in the beginning, MDS has some limitations. MDS only provide

the similarities between objects, so the MDS result may be a twisted, scaled, rotated and flipped version of the original topology. How to transform back to the original one is still a problem. Using the point registration algorithms is one of the methods to do the transformation [14], and Iterative Closest Point (ICP) algorithm is a famous point registration algorithm which we will introduce in the next subsection.

2.2.2 Iterative Closest Point Algorithm

Iterative Closest Point (ICP) algorithm is widely used in image registration to reconstruct 2D or 3D surfaces from different scans [15] [16]. The principle of ICP is to minimize the difference between two sets of points. Here we continue from the MDS result and the real topology. In ICP, the real topology is called *reference*, which is kept fixed, while the MDS result is transformed to best match the reference. The algorithm iteratively updates the transform matrix which is composed of translation and rotation. The goal of the algorithm is to minimize an error function, usually the distance from the set of the MDS result to the set of the real topology.

The ICP here is composed of four steps, as shown in Figure 9. We will introduce the four steps of the algorithm with some mathematical formulas as follows:

1. For each point in the MDS result, find the closest point in the real topology

Assume \mathbb{M} is the set of the MDS result, and \mathbb{V} is the set of the real topology.
 $\forall m \in \mathbb{M}$ find a closest point $v \in \mathbb{V}$, which we called $match_v(m)$.

2. Estimate the combination of rotation and translation using a cost function that will best align each MDS result point to its match found in the previous step

Assume \mathbf{R} is the rotation matrix, and \mathbf{T} is the transform matrix. Update \mathbf{R} and \mathbf{T} that minimize

$$\frac{1}{N_m} \sum_{m \in \mathbb{M}} \|match_v(m) - (\mathbf{R} \times m + \mathbf{T})\|,$$

where N_m is the number of elements of \mathbb{M} .

We can obtain the \mathbf{T} by the following equations:

$$\mathbf{T} = \text{avg}(\mathbb{V}) - \mathbf{R} \times \text{avg}(\mathbb{M}),$$

and \mathbf{R} can be solved by cubic equation, linear list squares, or SVD

3. Update the MDS result with the obtained transformation

$$\forall m \in \mathbb{M}, \quad m_{new} = \mathbf{R} \times m_{old} + \mathbf{T}$$

4. Iterate the previous steps until the cost function converged

Redo the step-1 to step-3 until the cost function in step-2 converged

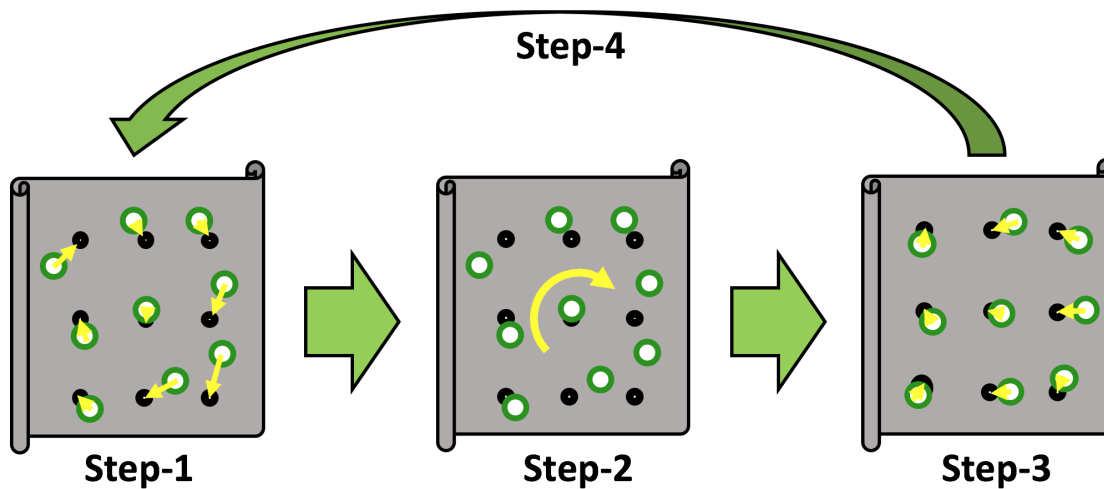


Figure 9: An example of one iteration of the ICP algorithm

After doing the ICP algorithm, each device can be mapped to the closest location one by one without duplicate. The next subsection will discuss the pros and cons about the MDS-ICP algorithm.

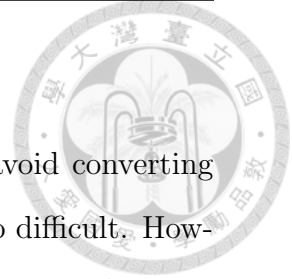
2.2.3 Summary of the MDS-ICP algorithm

The main advantage of the MDS-ICP algorithm is that it can avoid converting RSSIs to distances. It is very attractive because the converting is too difficult. However, there are many disadvantages about the MDS-ICP algorithm. First, the complexity is too high. The complexity of only MDS is $O(n^3)$, much less the complexity of MDS-ICP. Second, the accuracy of MDS-ICP is relatively lower than trilateration. The relationship between RSSI and distance are non-linear, so the distortion may be too severe to achieve enough accuracy. Third, if the original topology is symmetric, MDS-ICP cannot find the correct solution. This is a fatal disadvantage because it can not handle arbitrary topologies.

2.3 Summary

In this chapter, we introduced typical possible solutions for the location configuration. Two algorithms are summarized in Table 2. Converting RSSIs to distances is a critical problem for location configuration because in most of the time it requires more effort than manually location configuration. Accuracy is another critical problem because the location of a certain device must be fixed after deployed. These two requirements are a trade-off. Trilateration provides a relatively high accuracy but needs to convert RSSIs to distances. MDS-ICP can avoid converting the RSSI while it can only provide relative low accuracy.

Complexity is another issue. Lower complexity is better, because most of the devices in IoT may not have powerful computation resources. In addition, the complexity affects not only the performance but also the difficulty of implementation. As mentioned before, the complexity of MDS-ICP is greater than $O(n^3)$. Because MDS-ICP requires iteration, it is much more complex than the complexity of trilateration which is only $O(n^2)$.



The problem of scaled, rotated, and flipped result is another fatal issue for MDS-ICP. If the original topology is symmetric, ICP cannot ensure whether the result is correct or not. Refer to the previous 3x3 grid example. ICP only minimize the difference between two sets, but it cannot tell the direction. If the MDS result is just a 90-degree rotated version of the original topology, then ICP may consider it is the best match.

	Trilateration	MDS-ICP
Convert RSSIs to distances	Yes	No
Accuracy	High	Low
Complexity	$O(n^2)$	$> O(n^3)$
Scaled, Rotated and Flipped	No	Yes

Table 2: Comparison between two algorithms

Converting RSSIs to distances, accuracy, and complexity are three technical challenges for location configuration. Both trilateration and MDS-ICP cannot solve the three challenges simultaneously. Therefore, our goal is to design a whole new algorithm to achieve high accuracy, low complexity, and avoid converting RSSIs to distances for the location configuration.

The following chapters will introduce the solution proposed by this paper to solve the automatic location configuration problem with the help of the known topology and analyze the performance to compare with other existed solutions.

CHAPTER 3



PROPOSED SCHEME

As we mentioned in the previous chapter, we want to conquer the technical challenges: 1) converting RSSIs to distances, 2) high accuracy, 3) low complexity, and 4) can work with arbitrary topologies. In this chapter, we will first make a description of the system setting, and then introduce the concept and details of the proposed algorithm.

3.1 System Setting

We consider a set of installed devices and a set of known locations. We do not know which device is in a certain location, but we know that the number of known locations is the same as the number of devices. Devices are equipped with wireless modules and each device has a unique ID. Thus, devices can communicate with one another and get RSSIs of other devices. We assume the set of locations is on a 2-D plane for simplicity, and the deployment environment has boundaries.

3.2 Concepts of the Algorithm

According to the system setting, here we have two sets: one is the location set and the other is the ID set. Each unique ID represents a device, so the ID set can also be viewed as the device set. As we mentioned in Chapter 2, the location configuration is a one-to-one mapping problem. Thus, here we want to find all correct mappings between the location set and the device set.

Assume we have an algorithm to serialize the two sets. If each mapping of the two sets just contains the same sequence number, then the one-to-one mapping problem

is solved, as shown in Figure 10. In other words, we can use sequence numbers to map the two sets.

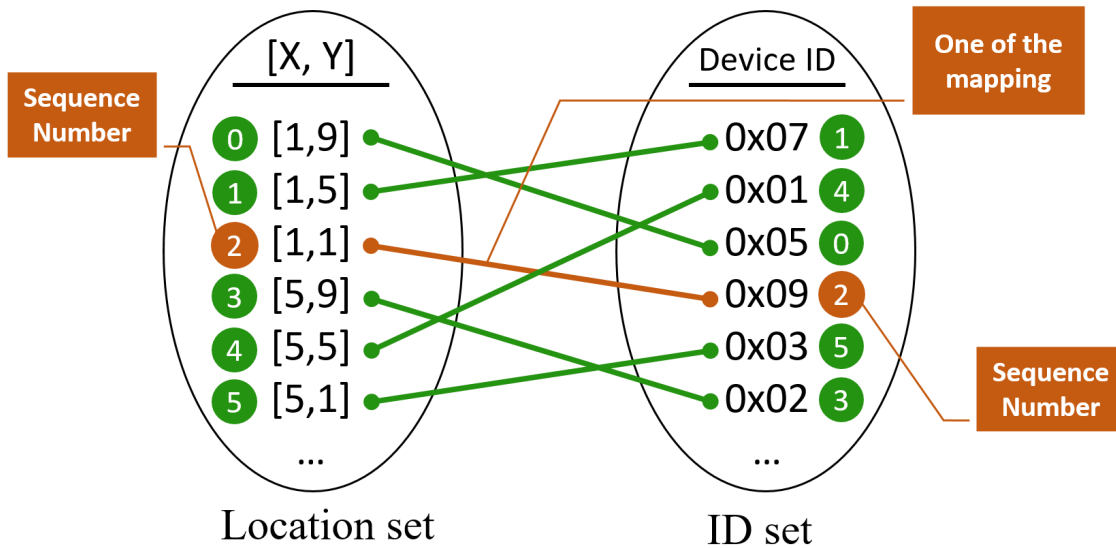
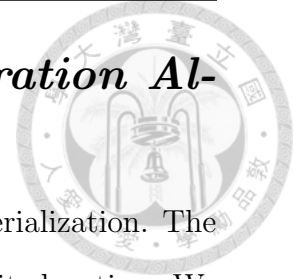


Figure 10: Key idea

For the device set, the RSSI is the only information we can use. For the location set, we have all known locations and all distances among locations are also known. Now assume a device has prior knowledge of its location and the sequence number of the device and location is “0”. The device starts to receive other devices’ packet and get RSSIs. We can know which device has the maximum RSSI value, and it means that the device with the maximum RSSI is “closest” to the device with sequence number 0. Also, from the location set, we can know which location is closest to the location with sequence number 0. Thus, the closest location is the location of the closest device, and we assign the sequence number “1” to this mapping. is the key idea of our algorithm, and we will elaborate the algorithm in the next section.

3.3 Topology-based Automatic Configuration Algorithm



In our algorithm, we need one anchor as a start-point to do the serialization. The anchor refers to the additional device which has prior knowledge of its location. We will introduce the importance of the anchor and how to determine the location of the anchor in Section 3.5. Here we use an example to introduce our algorithm step by step first, and then summary each step of the algorithm in the end.

The algorithm has two steps: 1) serialization of the location set and 2) serialization of the device set. We start from the location set. Figure 11 is the initial step of the serialization. The red circle with number 0 is the location of the anchor, and green circles are the other known locations. We assign the sequence number 0 to the location of the anchor.

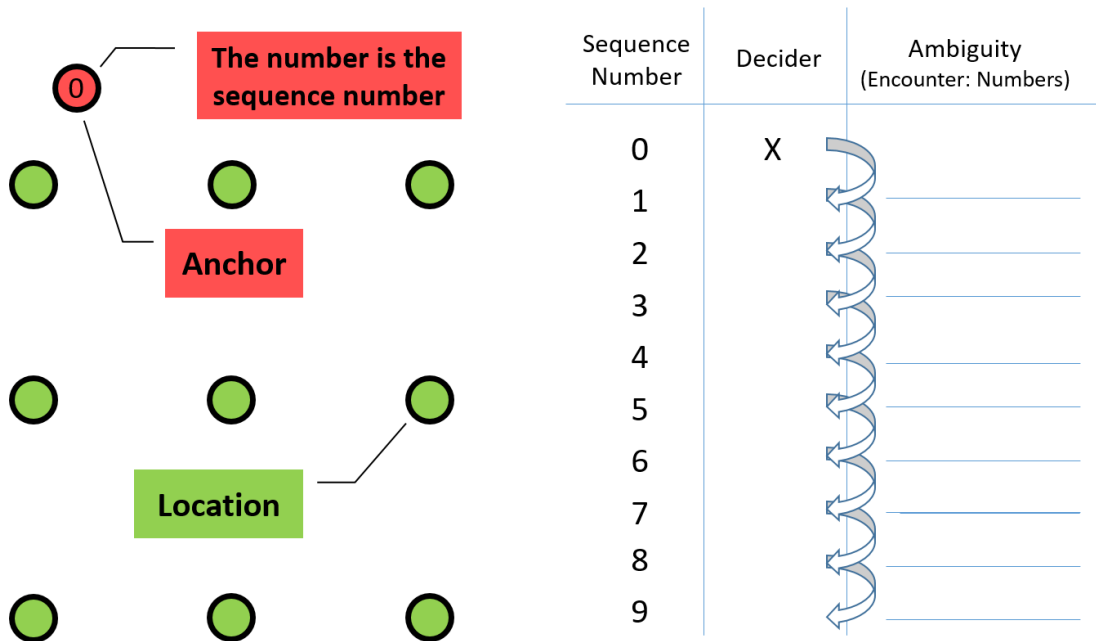


Figure 11: Serialization of the location set - Initialization

Next, we want to find the owner of sequence number 1. We choose the location which is closest to the anchor and does not have the sequence number. Here is



no ambiguity and we assign the sequence number 1 to that location, as shown in Figure 12. The ambiguity refers that there are more than two closest locations with the same distance.

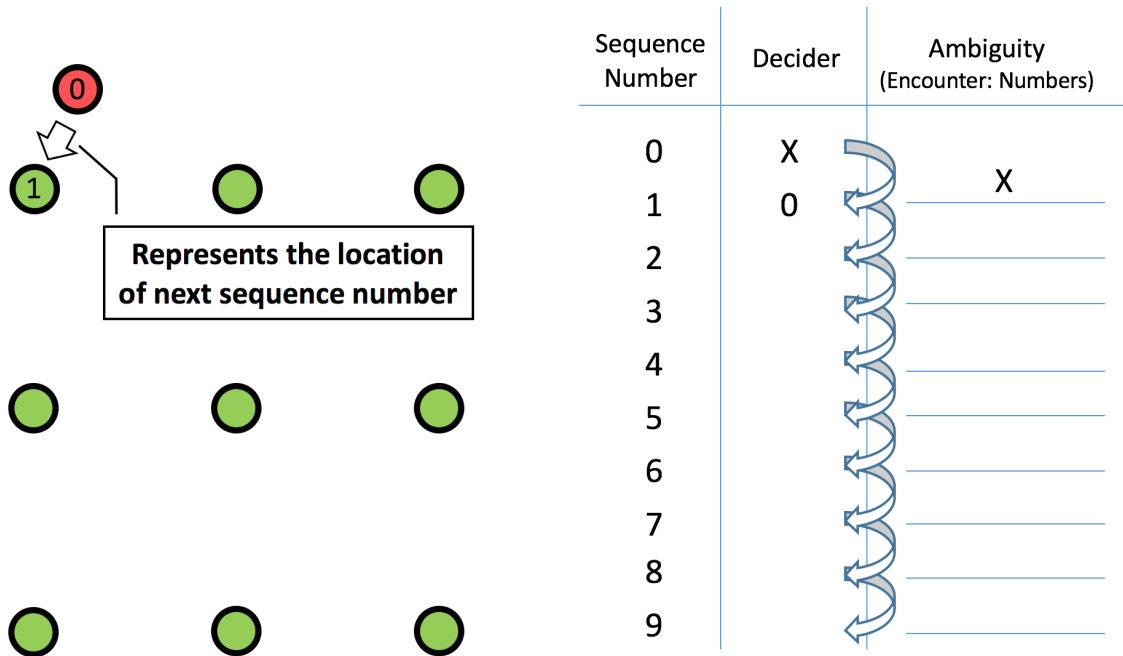


Figure 12: Serialization of the location set - Find sequence number 1

Now we want to find the owner of sequence number 2, but in Figure 13 we can see that there are two closest locations (red circles without numbers). In this situation, the current location will ask the previous location to choose the closest location from it. That is, in Figure 13, the location with sequence number 1 ask the location with sequence number 0 to choose the closest location. The dotted line shows which the location with sequence number 0 chooses. When a location find that there exists ambiguity, we record the sequence number and how many ambiguous locations, as shown on the right-hand side of Figure 13.

We can use the same procedure to find the owner of sequence number 3. The location of sequence number 2 encounter two closest locations, so it asks the previous location (sequence number 1) to determine the next one, as shown in Figure 14.

We have already serialized the location set from sequence number 0 to 3, and now we want to find the location of sequence number 4. The current location is the location with sequence number 3, and it finds that here are three ambiguous locations, as shown in Figure 15. We record the ambiguous information on the right-hand side of Figure 15. According to our algorithm, the current location must ask the previous location which has sequence number 2 to determine the location of sequence number 4. However, the location with sequence number 2 also encounters ambiguous locations, as shown in Figure 16. We also record the ambiguous information and recursively ask the previous location which has sequence number 1. Finally, the location with sequence number 1 can find the next location and the direction of the dotted arrow is the found location.

There are no ambiguities with sequence 5 to 9, and the procedure is almost the same as what we did from sequence number 0 to 1. We just find the closest locations one by one and the serialization of the location set is done, as shown in Figure 17.

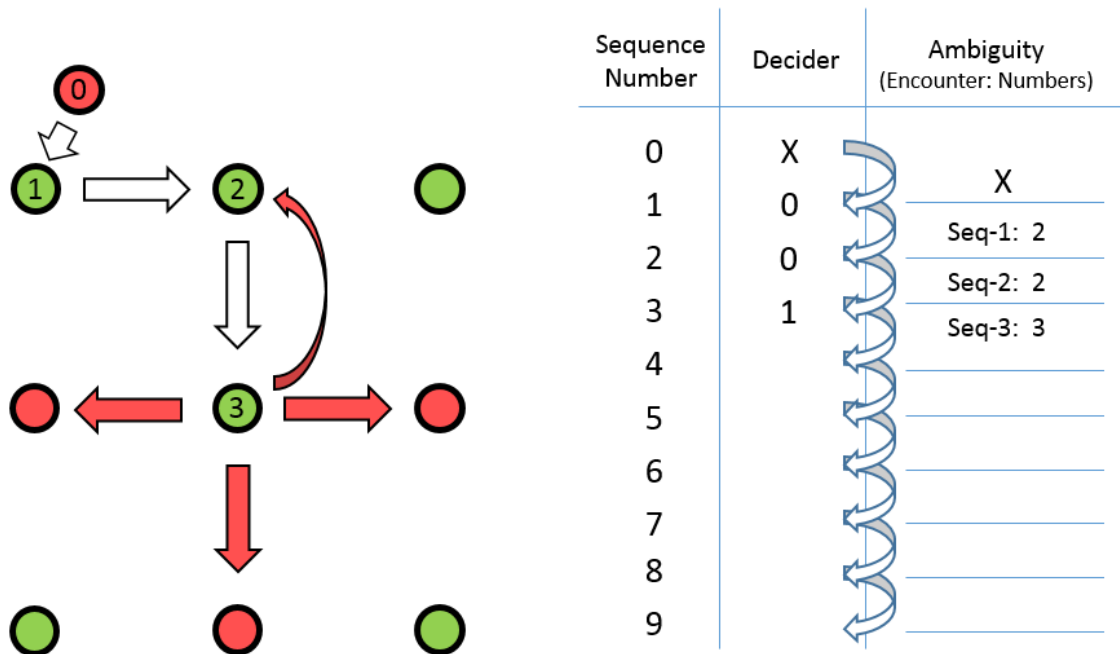
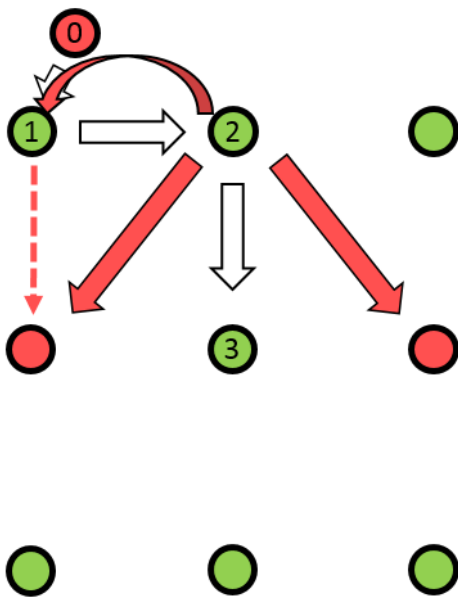
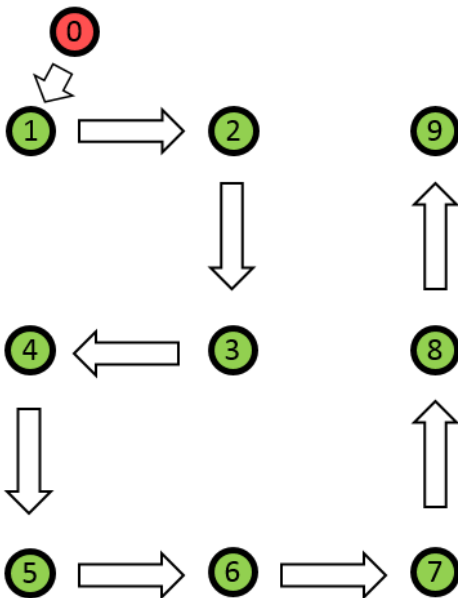


Figure 15: Serialization of the location set - Find sequence number 4 (part1)



Sequence Number	Decider	Ambiguity (Encounter: Numbers)
0	X	
1	0	X
2	0	Seq-1: 2
3	1	Seq-2: 2
4		Seq-3: 3
5		Seq-2: 2
6		
7		
8		
9		

Figure 16: Serialization of the location set - Find sequence number 4 (part2)



Sequence Number	Decider	Ambiguity (Encounter: Numbers)
0	X	
1	0	X
2	0	Seq-1: 2
3	1	Seq-2: 2
4	1	Seq-3: 3
5	4	Seq-2: 2
6		X
7	4	X
8	5	X
9	6	X
	7	X
	8	X

Figure 17: Serialization of the location set - Find all sequence numbers

After we serialize the location set, we get the sequence numbers of all locations and an “ambiguity list”. The ambiguity list refers to the list which records how many ambiguity locations when each location wants to determine the owner of the next sequence number. We simplify the ambiguity list as shown in Figure 18. The column of closest numbers means that when we want to find the sequence number $n+1$ from n , how many closest locations the sequence number n will encounter. The ambiguity list can guide the serialization of devices, and the device with the corresponding sequence number can know what it should do.

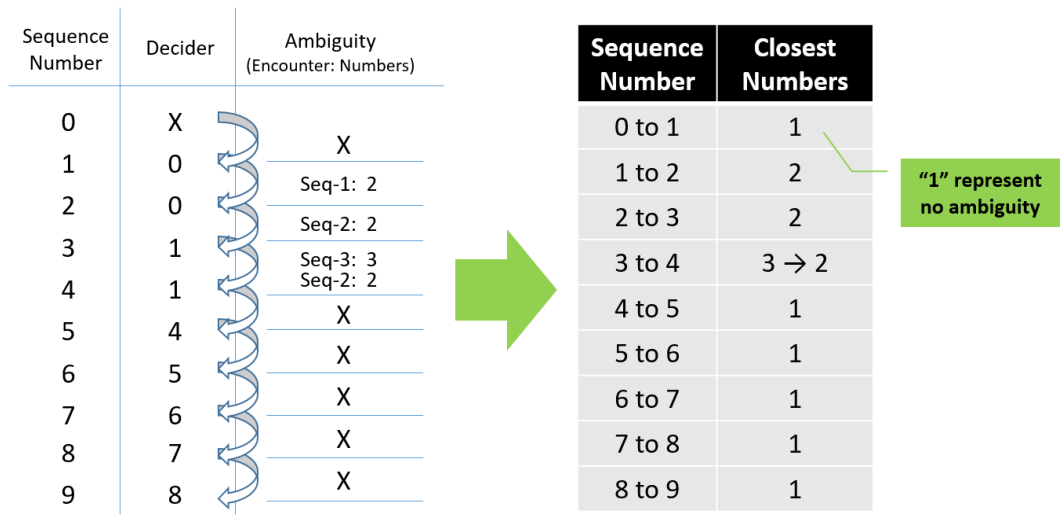


Figure 18: Simplified ambiguity list

Now we want to serialize the device set. We also start the serialization from the anchor device. As we mentioned in Section 3.2, RSSI is the only thing we can use to serialize the device set. We “repeat” the same process as what we did with the location set, but here we choose the device with the maximum received RSSI as the owner of the next sequence number. The following is the step-by-step example of serialization of device set.

The red icon with number 0 is the location of the anchor, and green icons are the other known devices. We assign the sequence number 0 to the anchor device for

initialization, as shown in Figure 19.

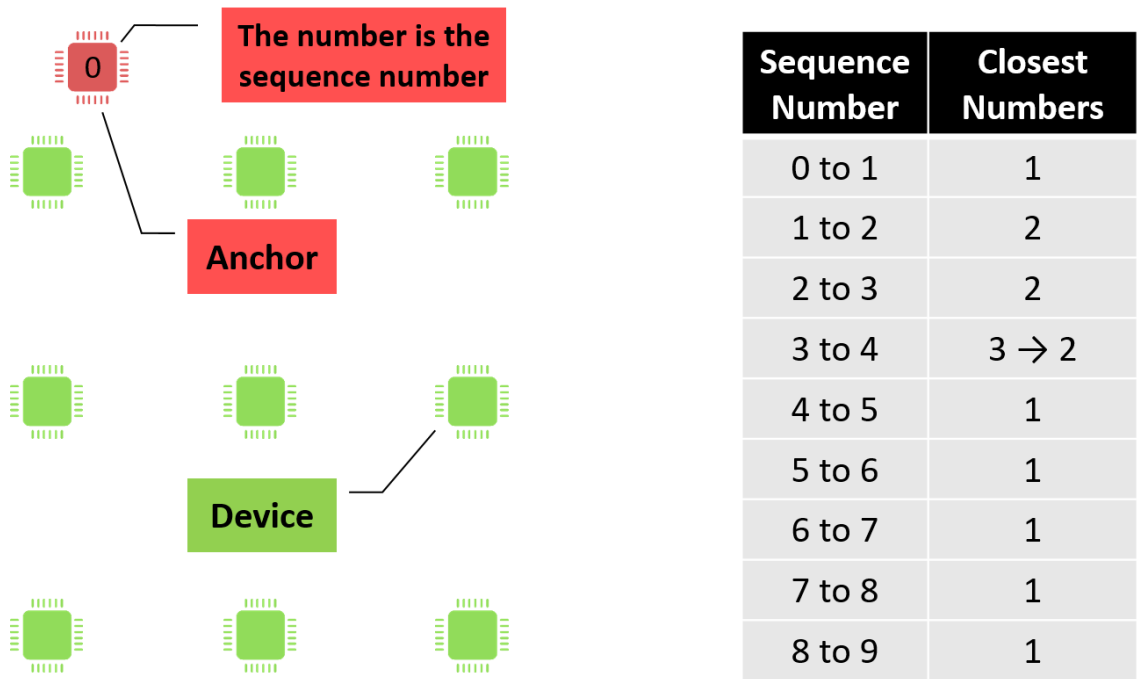


Figure 19: Serialization of the device set - Initialization

The next step is to find the owner of sequence number 1. First, we check the ambiguity list and find that we only need to find one “closest” device from sequence number 0 to 1, as shown on the right-hand side of Figure 20. The device with sequence number 0 now receive all packets from the other devices and assign sequence number 1 to the device with the maximum RSSI. In Figure 20, the red arrow refers to the packet with the maximum RSSI.

Now we want to find sequence number 2 from 1. We check the ambiguity list and find that we need to find two closest devices from sequence number 1 to 2, as shown on the right-hand side of Figure 21. Then we “repeat” what we did with the location set: the device with sequence number 1 ask the device with sequence number 0 to decide which one is closer, as shown in Figure 22. Finally the device with sequence number 0 choose the device with the red arrow as the owner of sequence number 2.

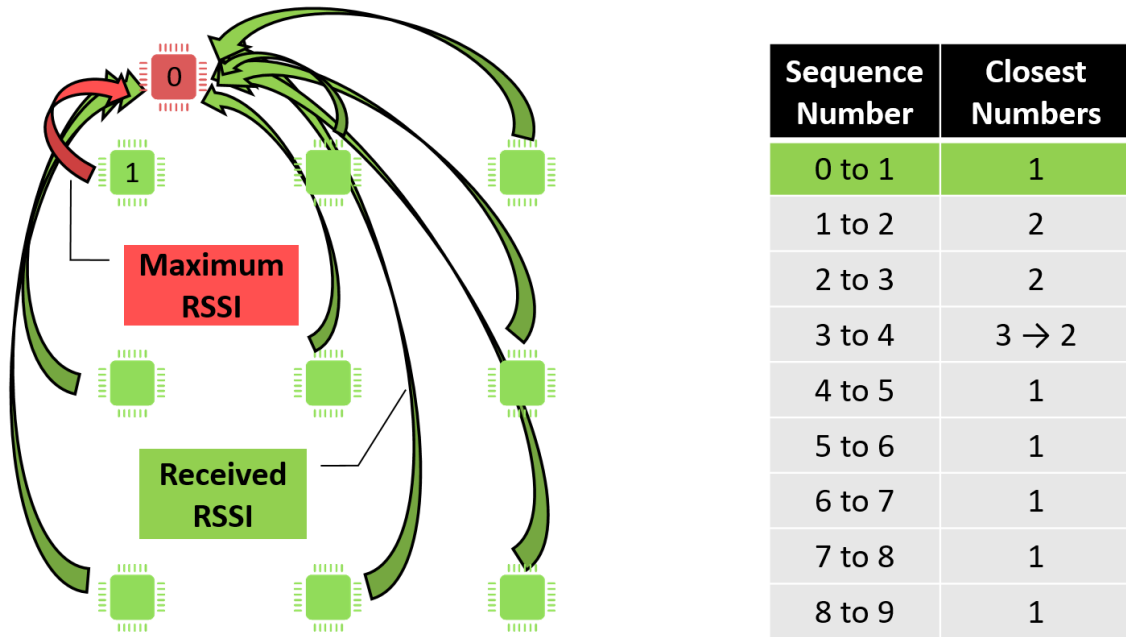


Figure 20: Serialization of the device set - Find sequence number 1

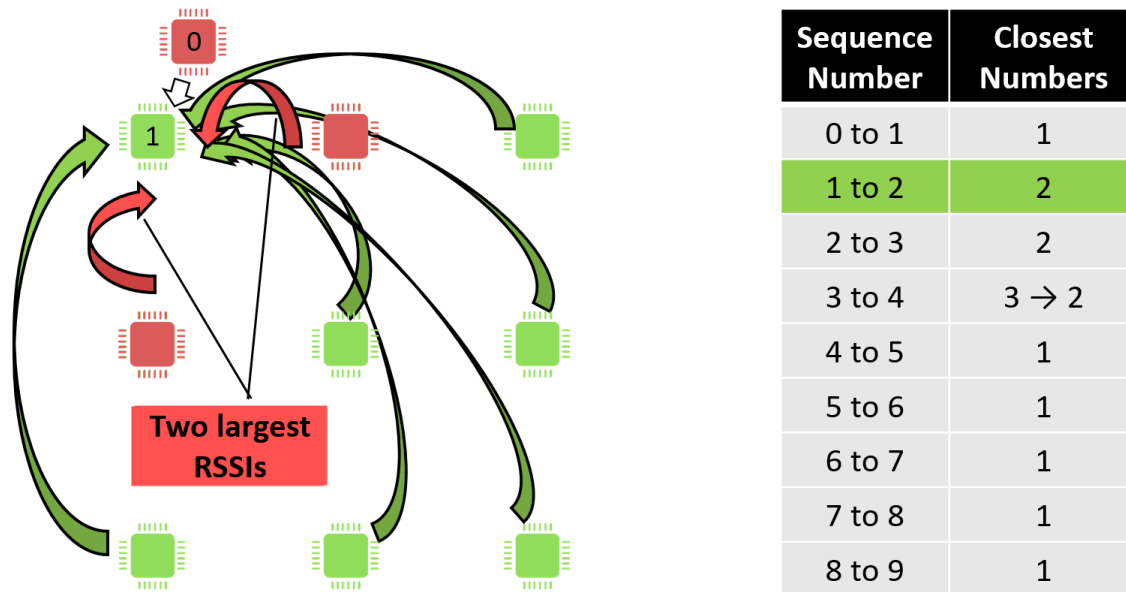


Figure 21: Serialization of the device set - Find sequence number 2 (part1)

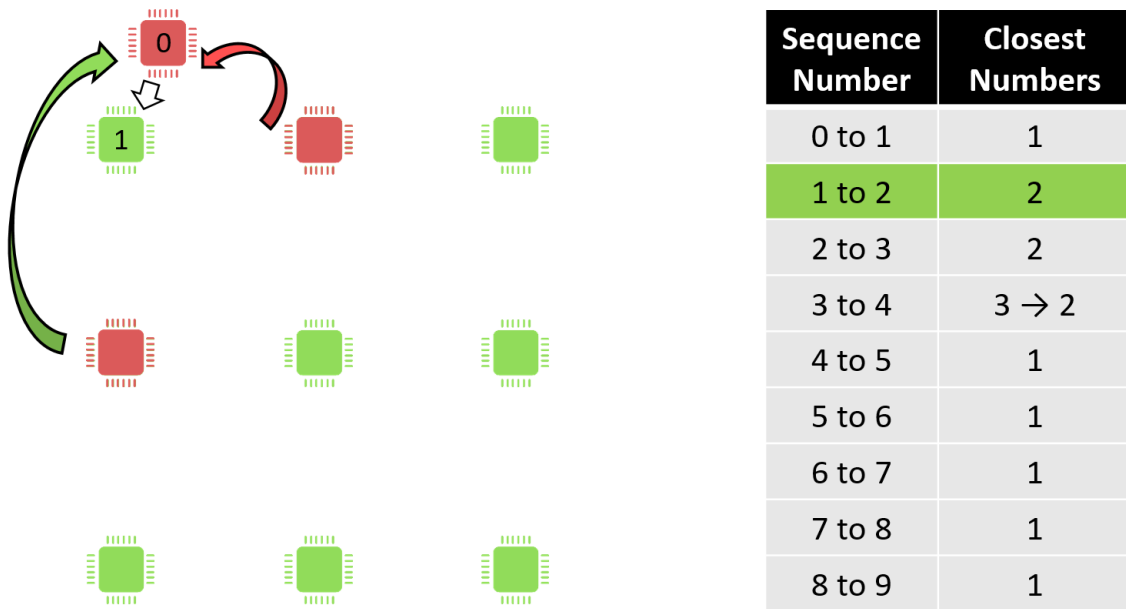
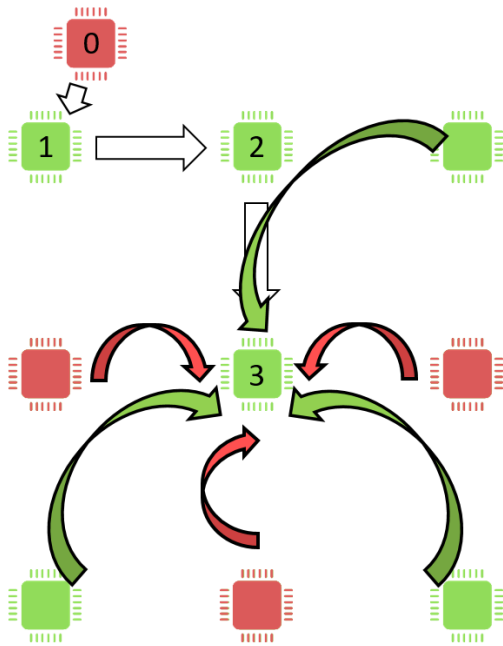


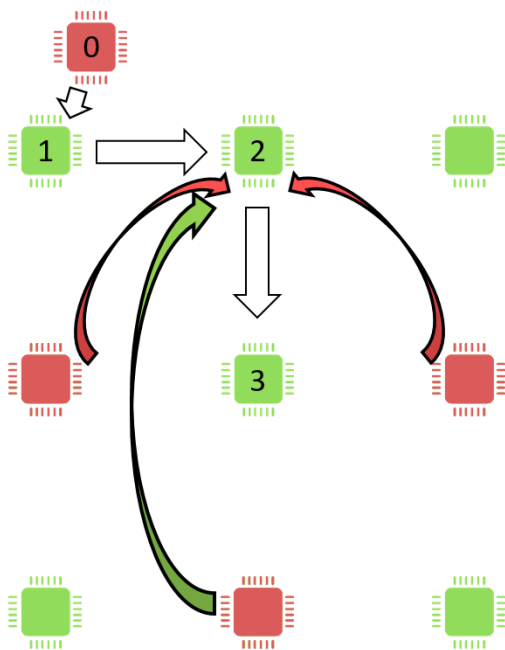
Figure 22: Serialization of the device set - Find sequence number 2 (part2)

We skip the process of finding sequence number 3 from 2 because it is almost the same as finding sequence number 2 from 1. Now we want to find sequence number 4 from 3, and we check the ambiguity list first. We can see that the device with sequence number 3 needs to find 3 closest devices, as shown in the right hand side of Figure 23. The device with sequence number 3 receive packets from the other devices which do not have sequence numbers, and then ask the previous devices to determine the three devices which have larger RSSIs. Before the previous devices (with sequence number 2) choose closest devices, it should check the ambiguity first. It found that it should choose the closest two of three devices, as shown in Figure 24. Thus, the device with sequence number 2 receive packets of that three devices and choose two devices with larger RSSIs. The device with sequence number 2 still cannot determine the closest device, so it should also ask the previous device. The ambiguity list shows that the previous device only needs to find the device with largest RSSI, so the device with sequence number 1 choose the device with the largest RSSI as the owner of sequence number 4, as shown in Figure 25.



Sequence Number	Closest Numbers
0 to 1	1
1 to 2	2
2 to 3	2
3 to 4	3 → 2
4 to 5	1
5 to 6	1
6 to 7	1
7 to 8	1
8 to 9	1

Figure 23: Serialization of the device set - Find sequence number 4 (part1)



Sequence Number	Closest Numbers
0 to 1	1
1 to 2	2
2 to 3	2
3 to 4	3 → 2
4 to 5	1
5 to 6	1
6 to 7	1
7 to 8	1
8 to 9	1

Figure 24: Serialization of the device set - Find sequence number 4 (part2)

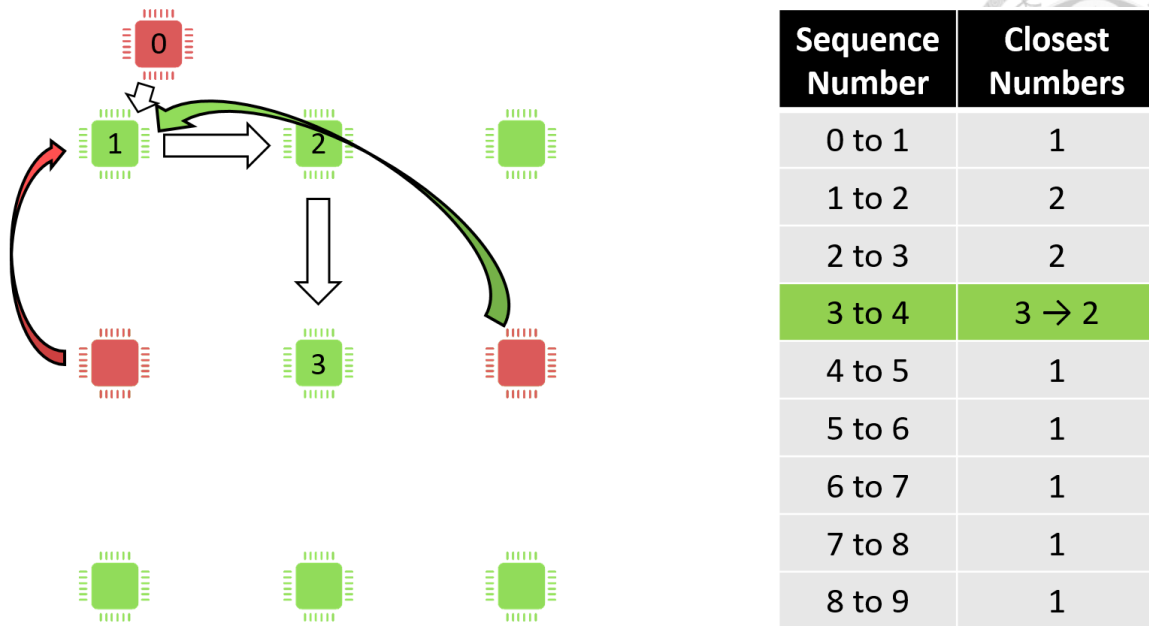


Figure 25: Serialization of the device set - Find sequence number 4 (part3)

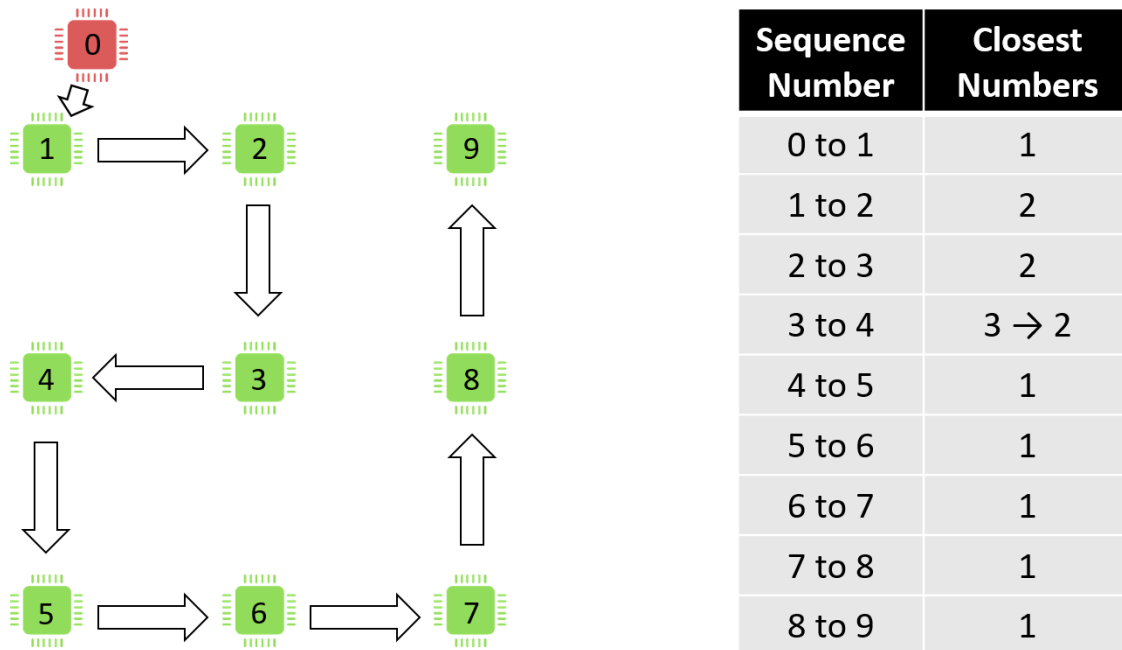


Figure 26: Serialization of the device set - Find all sequence numbers

The way to find the other sequence numbers remained is almost the same as finding sequence number 1 from 0, and the final result is in Figure 26. Until now we have serialized the location set and the device set. The device with the maximum RSSI means the closest device in the physical environment, and this is the same as finding the closest location. We can say that we use the same criteria (find the closest location/device) to serialize the two sets and get two correspondent sequences, as shown in Figure 27. The correspondent sequence number means a mapping, and finally, the location configuration is done.

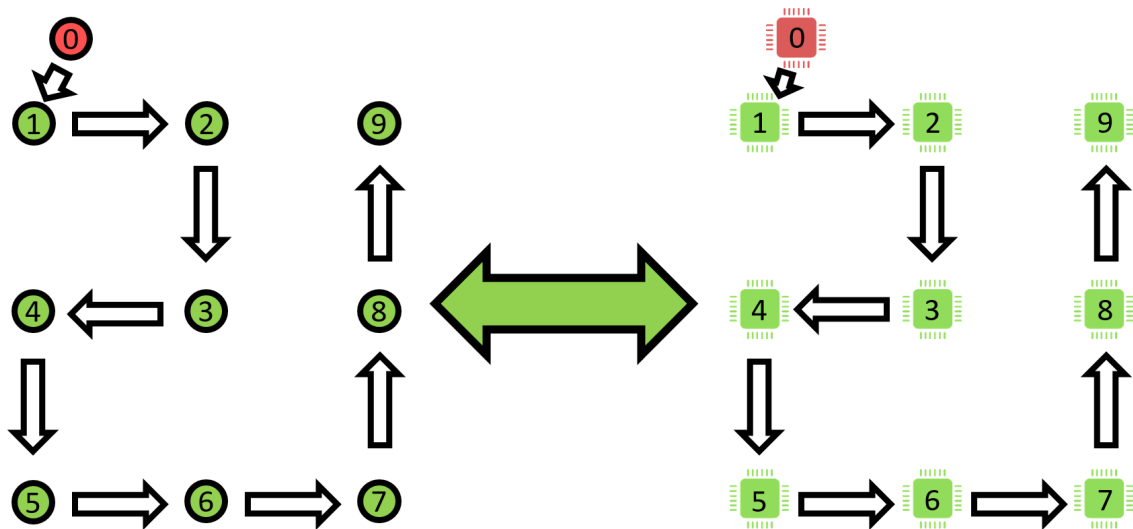
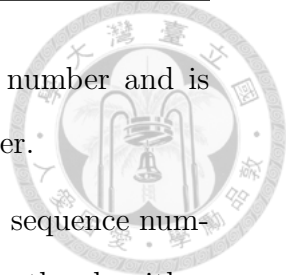


Figure 27: Two correspondent sets

The previous step-by-step example introduced how the topology-based configuration algorithm works. It can be used for not only the grid topology but also all other topologies. We summarize the algorithm in the following and discuss the problems remained in the end.

I. Serialization of the Location Set

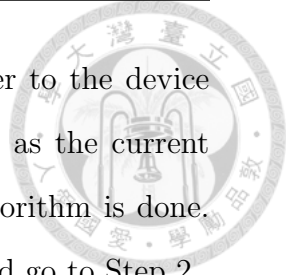
- Step 1. Assign the sequence number 0 to the location of the anchor. Set the current sequence number as 1.

- 
- Step 2. Choose the location which does not have a sequence number and is closest to the location with the current sequence number.
- Step 3. If there are only one closest location, assign the current sequence number to the location. If all devices have a sequence number, the algorithm is done. If at least one location has no sequence number, increase the current sequence number by one and go to Step 2.
- Step 4. If there are more than two closest locations with the same distance, record in the ambiguity list.
- Step 5. Choose the location which does not have a sequence number and is closest to the location with current sequence number-1. If there is only one closest location, go to Step 3. If there are more than two closest locations, go to Step 4.

After the serialization of the location set is done, we can get all sequence number of locations and an ambiguity list. The ambiguity list refers to the list which records how many ambiguity locations when each location wants to determine the owner of the next sequence number.

II. Serialization of the Device Set

- Step 1. Assign the sequence number 0 to the anchor device. Set the current sequence number as 1. Set the current device as the anchor.
- Step 2. The current device query the ambiguity list with the current sequence number to know how many other devices it should find. Set N as the number of devices it should find. The current device gets RSSIs from other devices which do not have a sequence number and choose N devices with larger RSSIs.

- 
- Step 3. If N is equal to 1, assign the current sequence number to the device found by the current device and set the found device as the current device. If all devices have a sequence number, the algorithm is done. If not, increment the current sequence number by 1 and go to Step 2.
- Step 4. If N is greater than 1, set the previous device as the current device and query the ambiguity list with the current sequence number to know how many devices the current device should find from the N devices. Set M as the new number of devices the current device should find. The current device gets RSSIs from the N devices and chooses M devices with larger RSSIs.
- Step 5. If M is equal to 1, set $N = M$ and go to Step 3.
- Step 6. If M is greater than 1, set $N = M$ and go to Step 4.

The topology-based configuration algorithm is composed of the two serializations above. The two sequences produced from the algorithm represent the one-to-one mappings and the location configuration is done. Each device is located on the location which has the same sequence number of the device.

However, there still exists some problems. First, it is impossible that devices can find the correct mapping all the time because of the fluctuation of RSSIs. It is possible that the RSSI of a closer device is smaller than other further devices in a certain moment. Second, if the anchor cannot determine two locations when we start to serialize the location set, the algorithm failed. How to decide a good location for the anchor is the other problem remained. We will introduce our solutions to solve the problems mentioned above in the following two sections.

3.4 *Majority Mechanism*

When we serialize the device set, we cannot assume that devices can always find the correct mapping. RSSI contains not only noise but also interference and the value of RSSI can be viewed as a Gaussian random variable [17]. Thus, it is possible that in a certain moment the RSSI of a closer device is smaller than another further device. In this situation, the serialization of the device set will be different from the serialization of the location set, and the mappings will be wrong. This is not a specific problem of our algorithm. All algorithms using RSSI will encounter this problem. We cannot avoid the fluctuation of RSSI.

However, we can “repeat” the algorithm for many times to see the statistical trend of all results of the algorithm. Assume we repeat the algorithm for 1000 times. We may get 5 results of the algorithm, and the occurrence frequency for each result will be different, as shown in Table 3. We choose the majority, C, as the final answer for the location configuration. The repeating number is called “majority number”.

Serialization Result	A	B	C	D	E
Occurrence Frequency	153	425	252	38	132

Table 3: An example of the majority mechanism

Using the majority mechanism can deal with the fluctuation of RSSI when the fluctuation is in a reasonable range. Also, not only our algorithm but also other algorithms can use the majority mechanism to promote the accuracy. The location configuration does not need to be real-time, so repeating the algorithm is valid for accuracy.



3.5 *Anchor Location Determination Algorithm*

The location of the anchor is the most important parameter for our algorithm. If the location of the anchor cannot find or cannot help others find the next locations, then the algorithm fail. For example, in Figure 28, all locations from L1 to L9 cannot be the anchor because every location cannot determine the next location with our algorithm. This is the reason why we need an anchor and we need to determine the location of the anchor by ourselves. On the other hand, the location of the anchor also affects the performance of the algorithm. The accuracy of the algorithm depends on the locations of the anchor, and we will see the simulation results of different locations in the next chapter.

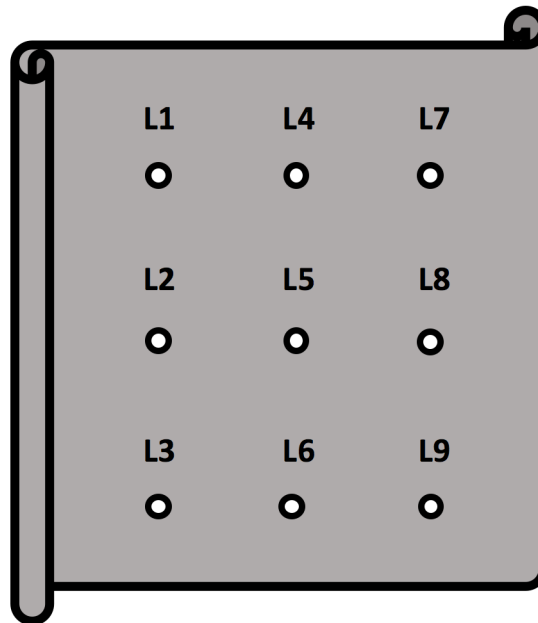


Figure 28: An example of the location of the anchor

As we mentioned, if the location of the anchor cannot determine the next location, the algorithm will fail. Thus, the problem is, when the location of the anchor cannot determine the next location? The answer is in the Figure 29. We can draw the perpendicular bisector of two certain locations. If the anchor is located on the

perpendicular bisector, the distances from the anchor to the two locations will be the same, and the anchor will be not able to distinguish the closest distance to these two locations.

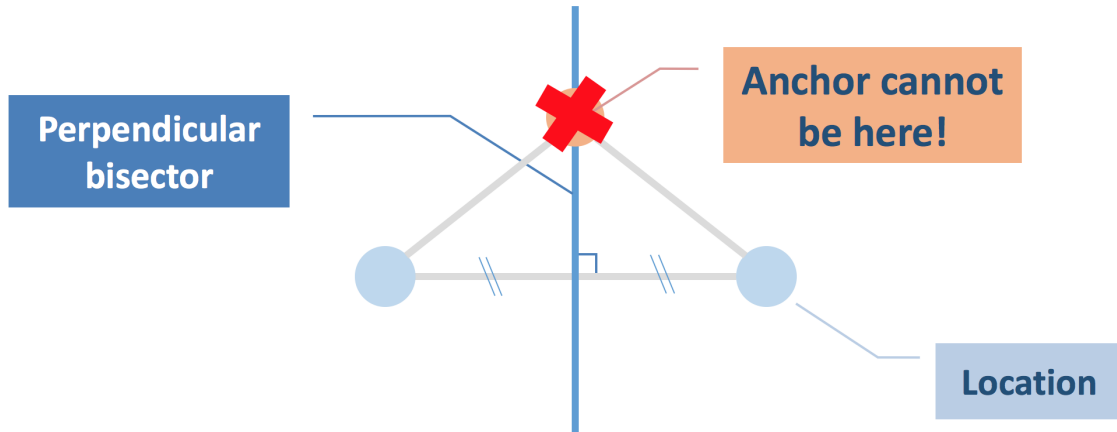


Figure 29: The perpendicular bisector of two locations

Thus, for the location set, anywhere the location is not on any perpendicular bisector can be the location of the anchor because any distance from the anchor to certain location will be different. However, for the device set, if the anchor is closer to a certain perpendicular bisector, it is more difficult to distinguish which device has larger RSSI because of the fluctuation of RSSI.

For the reasons above, the location of the anchor must be as far as possible to any perpendicular bisector. **Therefore, our objective is to find the best location which has the maximum of the minimum distance to any perpendicular bisector.** We call the best location as max-min point, and call the distance as max-min distance.

We can draw all perpendicular bisectors for all locations first. Figure 30 is an example of all perpendicular bisectors for Figure 28. The blue line represents the perpendicular bisector and the green line represents the boundary of the deployment environment. The night green circles represent location L1 to L9 in the Figure 28.

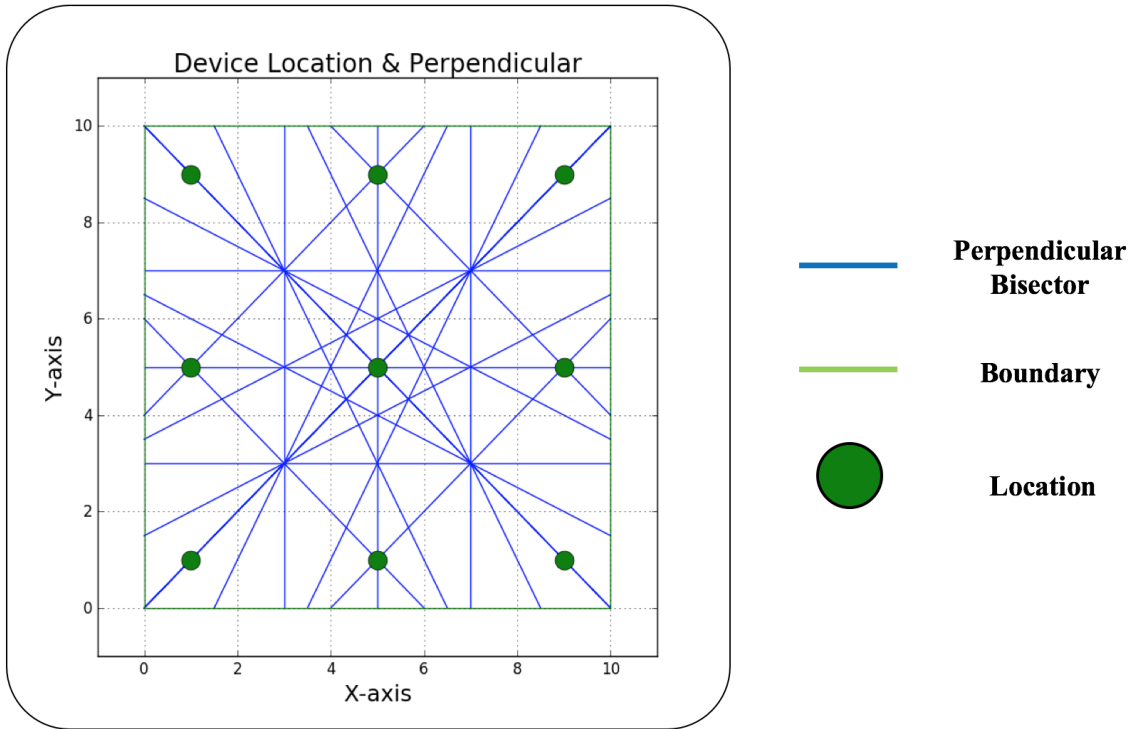


Figure 30: All perpendicular bisectors for Figure 28

We can see that there are a lot of “regions” divided by perpendicular bisectors and boundaries. The region refers to the area which does not contain any segment of perpendicular bisectors, as shown in Figure 31. The algorithm which can analyze the topology and find all regions is in Appendix A.

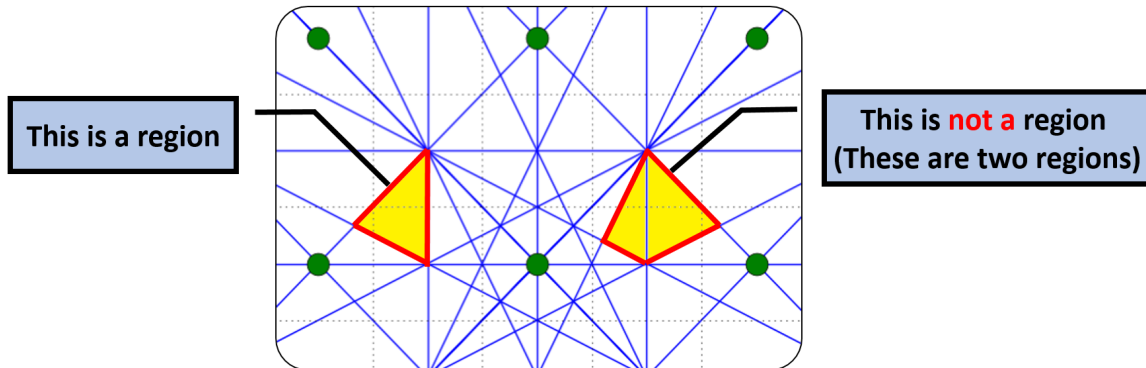


Figure 31: The definition of a region

Inside a certain region, the location with the maximum of the minimum distance is only related to the perpendicular bisectors surrounded, and we call the location as “local max-min point”. Similarly, we call the distance as “local max-min distance”. Obviously, the local max-min point which has the maximum local max-min distance is the best anchor location, which is also called max-min point, and therefore we can use “divide and conquer” with all regions to find the max-min point.

There are two kinds of regions. The first is the region which is surrounded only by perpendicular bisectors, and the second is the region which is surrounded by both perpendicular bisectors and boundaries. Here we discuss the two cases respectively.

Case 1. In a region which is surrounded only by perpendicular bisectors (the yellow region in Figure 32), the local max-min point is the center of the largest tangent circle in this region and the local max-min distance is the radius of that circle.

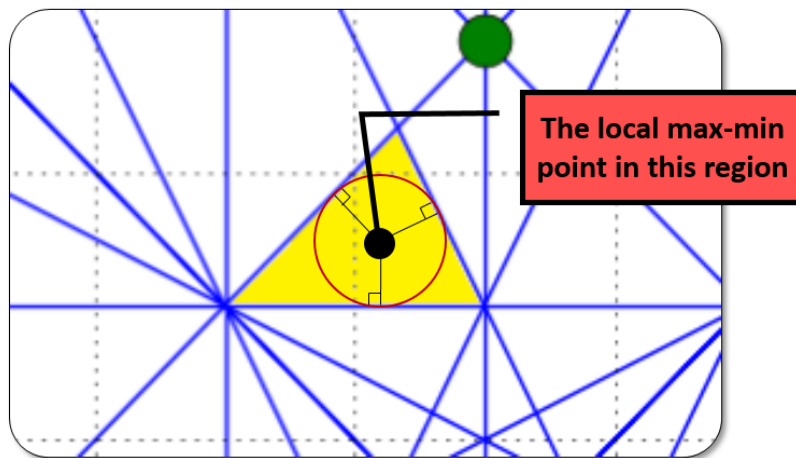


Figure 32: Local max-min point - Case 1

Case 2. In a region which is surrounded by perpendicular bisectors and boundaries, the local max-min point may be:

- I. The center of the largest tangent circle in this region, as shown in

Figure 33.

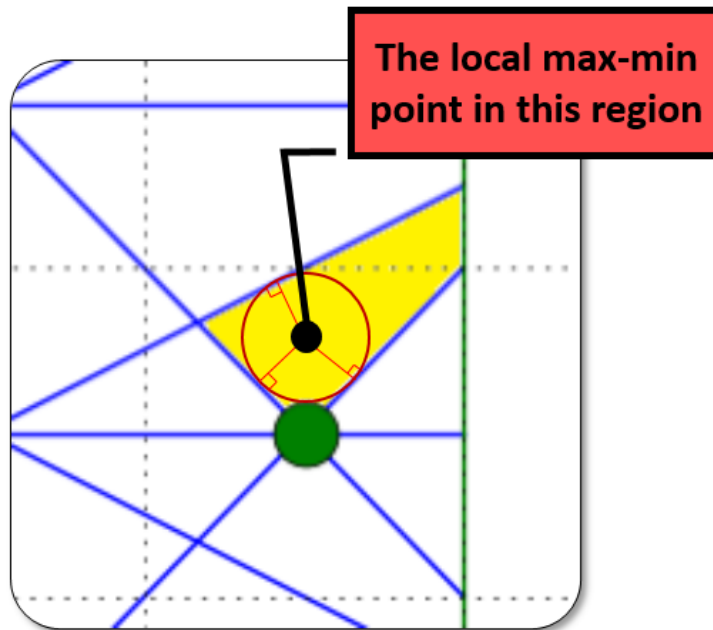


Figure 33: Local max-min point - Case 2-1

- II. The intersection point of a certain boundary and the angle bisector of two certain perpendicular bisectors. The intersection point is also the center of the largest tangent circle **whose center is in the region.**

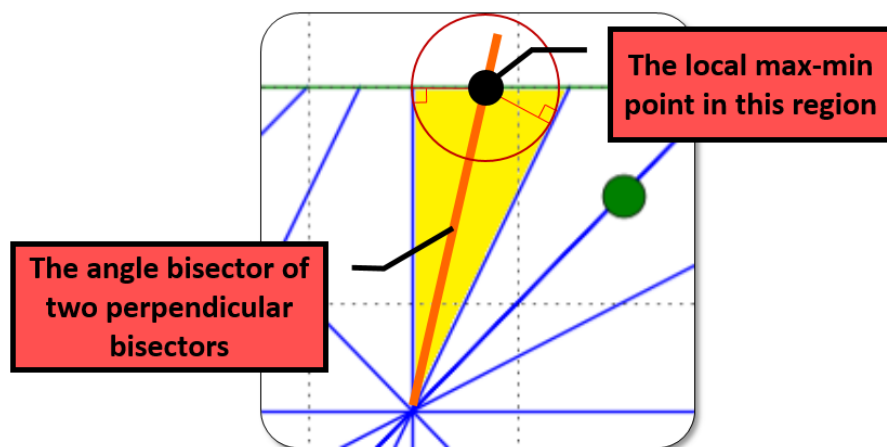


Figure 34: Local max-min point - Case 2-2

The circle does not need to be tangent with boundaries, so the circle can cross boundaries (green lines). In fact, the way to find the largest tangent circle has only two limitations: 1) the center of the circle must be in the region, and 2) the circle cannot cross any perpendicular bisector (blue lines). Thus, the circle in Figure 34 is valid.

We can find that the center of the largest tangent circle in a green region is also the intersection point of two certain angle bisectors of perpendicular bisectors. Therefore, the brute-force algorithm to find the local max-min point in a certain region is as follows:

- Step 1. Find all angle bisectors of all perpendicular bisectors in the region.
- Step 2. Find all intersection points of perpendicular bisectors and boundaries (if there exists)
- Step 3. The intersection point which has the maximum of all minimum distances is the local max-min point in the region.

As we mentioned above, the point with the maximum local max-min distance among all regions is the best anchor. Here we color all regions in Figure 30 with the value of local max-min distances, and the result is shown in Figure 35. The color bar on the right-hand side represents the relationship between colors and local max-min distances. White points in the figure represent local max-min points and the black points represent the best location of anchors, also known as max-min points. We can see that this topology has eight best anchors due to the symmetry. The performance simulation for different anchors is in the next chapter, and we will see how the location of anchor affect the performance of the algorithm.

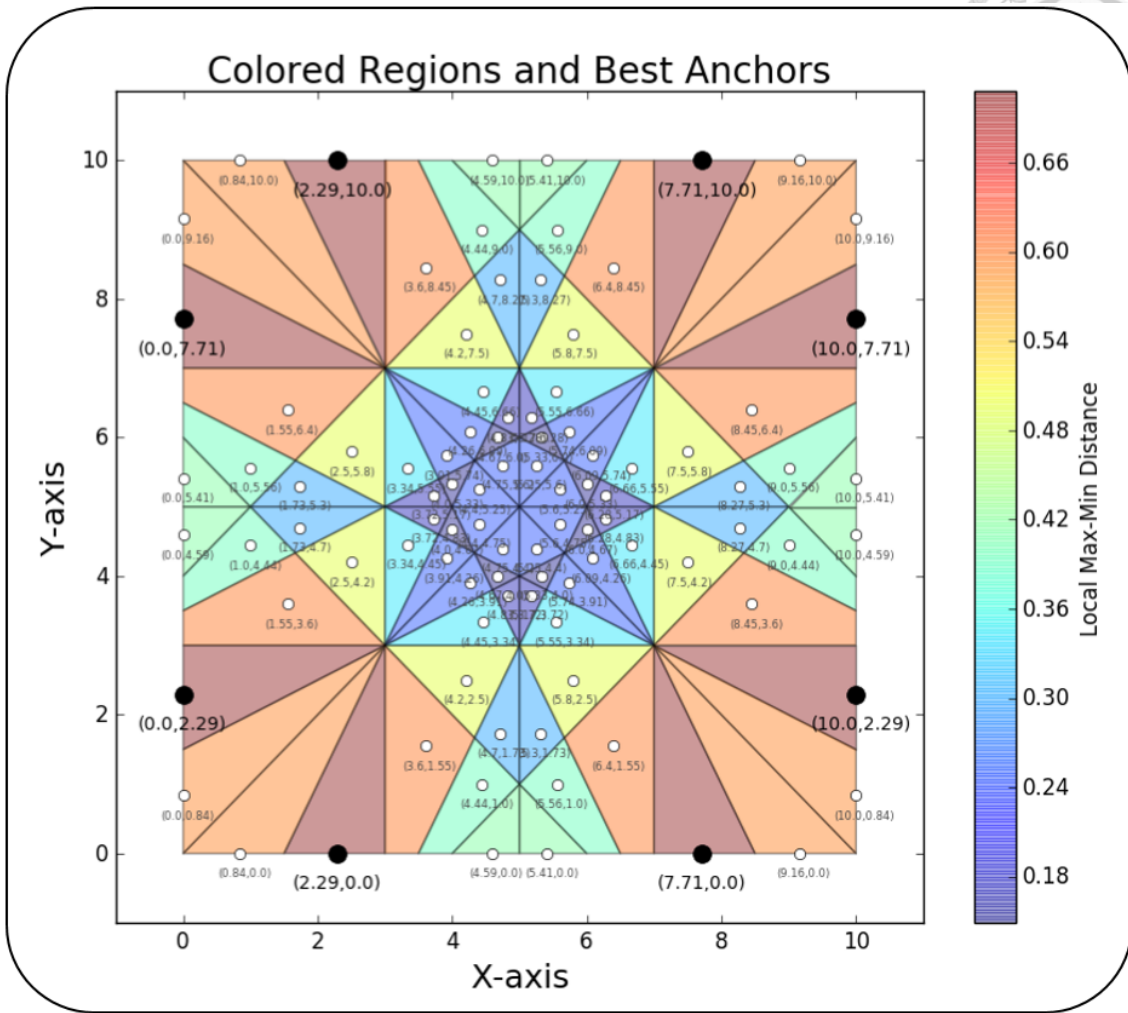


Figure 35: Colored regions and best anchors

Until now, we have introduced 1) topology-based configuration algorithm, 2) majority mechanism and 3) how to find the best anchor. In the next chapter, we will simulate the performance among different locations of anchors. Also, we will compare different algorithms with our solution and discuss the performance. Readers can find more examples of the anchor determination algorithm in Appendix B.

CHAPTER 4



SIMULATION, ANALYSIS AND IMPLEMENTATION

In this chapter, we will evaluate our algorithm in different points of view. We will introduce the simulation setting first, and then discuss the performance metrics of our algorithm. We will also discuss the analytical performance analysis. In the end, we will introduce the implementation and the protocol designed for our topology-based configuration algorithm.

4.1 Simulation Settings

In the following simulation, we adopt the well-known log-distance path loss model for RSSI as follows [17]:

$$P_R = P_T \frac{G_T G_R \gamma h^2}{4\pi d^p} \quad (4.1)$$

- P_R is the received power value, and P_T is the transmitted power
- G_R and G_T are the transmitter antenna gains and receiver antenna gains
- d is the distance between transmitter and receiver
- p is the path loss exponent
- γ is a model parameter for slow fading (log-normal distribution), also known as shadow fading
- h is a parameter modeling the fast fading, often referred to Rayleigh fading (NLoS condition) or Rician fading (LoS condition)

If the RSSI are averaged over a certain time interval, the fast fading term h can be approximated with $h=1$ and using logarithmic units, the model becomes

$$RSSI(dbm) = \alpha - 10p \log_{10}(d) + z(0, \sigma) \quad (4.2)$$

- $z(0, \sigma)$ represents the shadow fading which is a Gaussian random variable with zero mean and standard deviation
- α contains the averaged fast fading, P_T , G_T and G_R

As we mentioned above, we know that RSSI is a Gaussian random variable and $RSSI \sim N(\alpha - 10p \log_{10}(d), \sigma)$. We can assume the measured distance \hat{d} is another random variable transformed from RSSI with

$$\hat{d} = g(RSSI, \alpha, p) = 10^{\frac{\alpha - RSSI}{10p}}. \quad (4.3)$$

Here we define the *measurement error rate (MER)* as the following:

$$MER = \frac{\sqrt{Var(\hat{d})}}{E[\hat{d}]}. \quad (4.4)$$

We can derive the MER as follows:

$$\begin{aligned} MER &= \frac{\sqrt{Var(\hat{d})}}{E[\hat{d}]} \\ &= \sqrt{\frac{E[\hat{d}^2] - E^2[\hat{d}]}{E^2[\hat{d}]}} \\ &= \sqrt{\frac{E[\hat{d}^2]}{E^2[\hat{d}]} - 1}, \end{aligned} \quad (4.5)$$

where $E[\hat{d}^2]$ and $E^2[\hat{d}]$ are:



$$\begin{aligned}
 E[\hat{d}^2] &= \int_{-\infty}^{\infty} (g(RSSI, \alpha, p))^2 N(\alpha - 10 p \log_{10}(d), \sigma) dx \\
 &= \int_{-\infty}^{\infty} \left(10^{\frac{\alpha-x}{10p}}\right)^2 \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-(\alpha-10p\log_{10}(d)))^2}{2\sigma^2}} dx,
 \end{aligned} \tag{4.6}$$

and

$$\begin{aligned}
 E[\hat{d}] &= \int_{-\infty}^{\infty} g(RSSI, \alpha, p) N(\alpha - 10 p \log_{10}(d), \sigma) dx \\
 &= \int_{-\infty}^{\infty} \left(10^{\frac{\alpha-x}{10p}}\right) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-(\alpha-10p\log_{10}(d)))^2}{2\sigma^2}} dx.
 \end{aligned} \tag{4.7}$$

We can finally get the MER as follows:

$$\begin{aligned}
 MER &= \sqrt{\frac{E[\hat{d}^2]}{E^2[\hat{d}]} - 1} \\
 &= \sqrt{\frac{\int_{-\infty}^{\infty} \left(10^{\frac{\alpha-x}{10p}}\right)^2 \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-(\alpha-10p\log_{10}(d)))^2}{2\sigma^2}} dx}{\left(\int_{-\infty}^{\infty} \left(10^{\frac{\alpha-x}{10p}}\right) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-(\alpha-10p\log_{10}(d)))^2}{2\sigma^2}} dx\right)^2} - 1}
 \end{aligned} \tag{4.8}$$

Through the observation of the numerical integration, we find that MER is only the function of σ and p , and independent of α and d , as shown in Figure 36 to 39 .

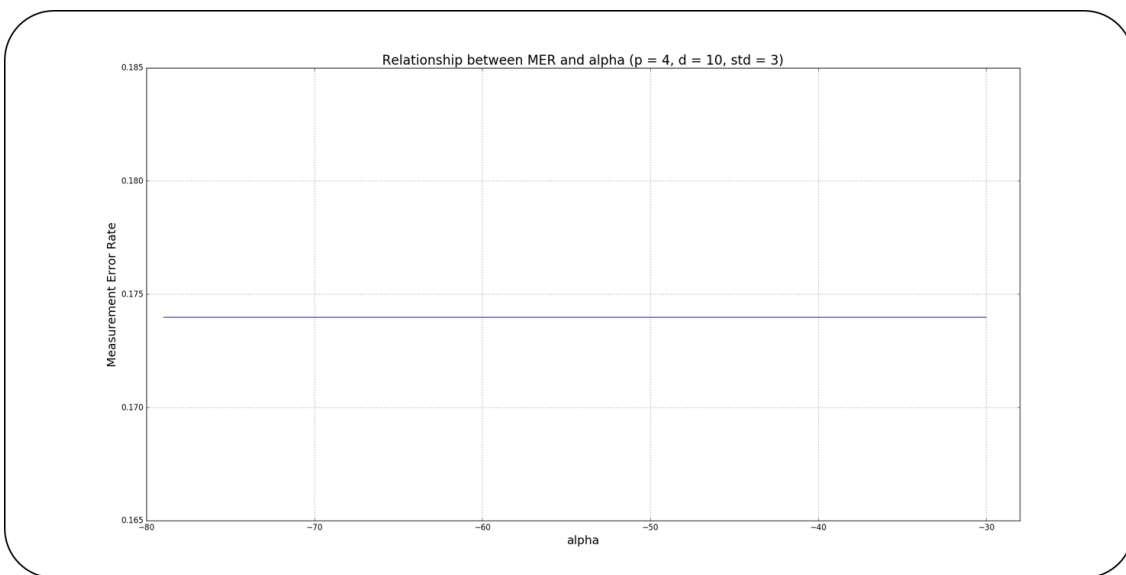


Figure 36: Relationship between MER and α

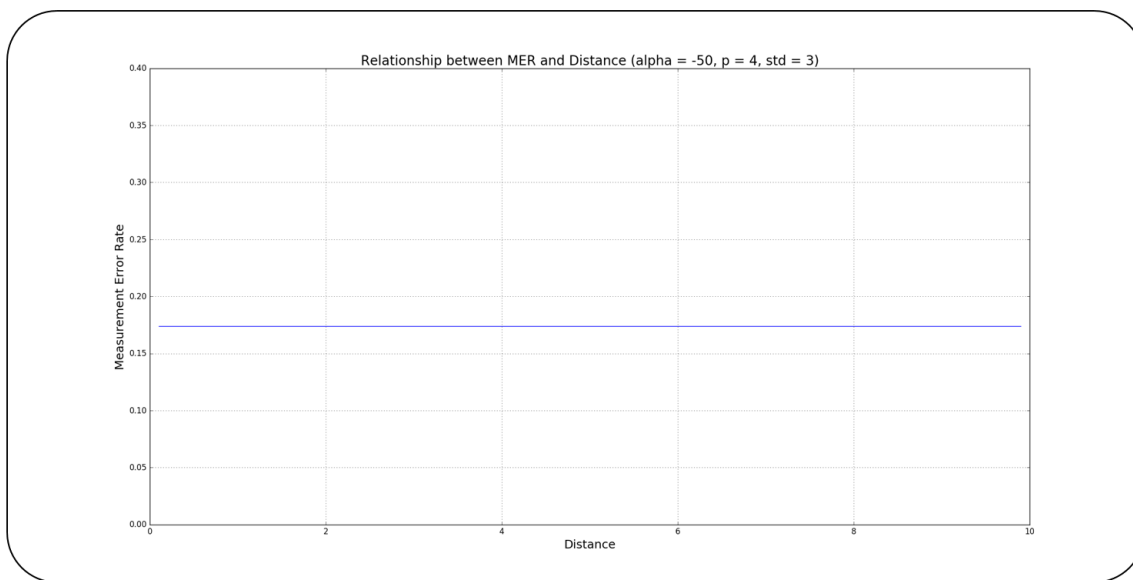
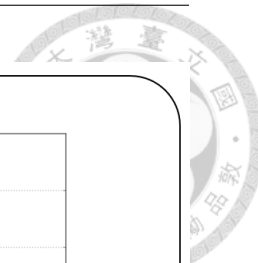


Figure 37: Relationship between MER and distance

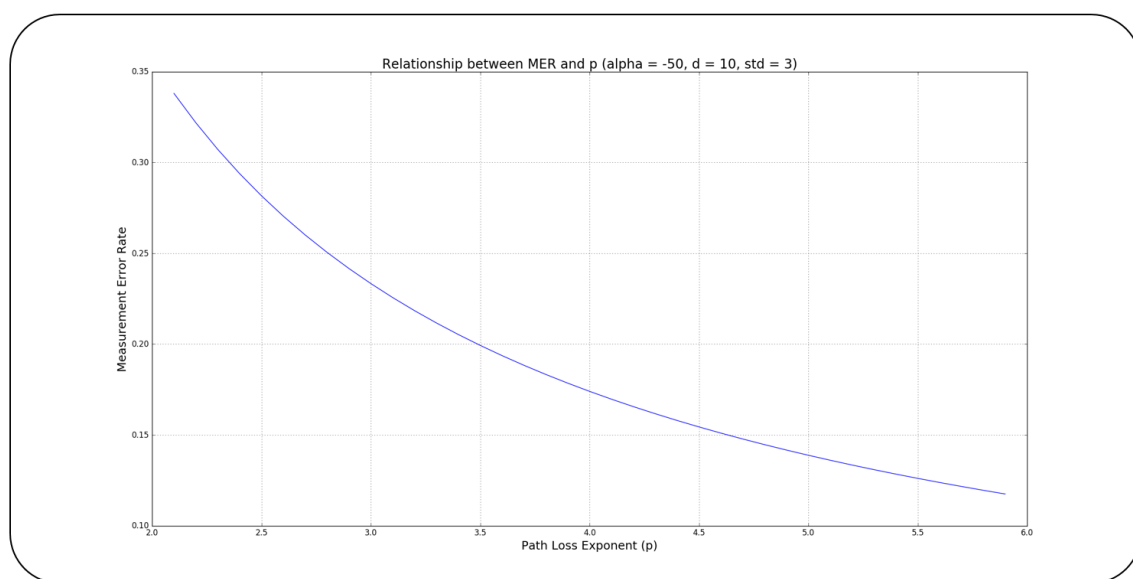


Figure 38: Relationship between MER and p

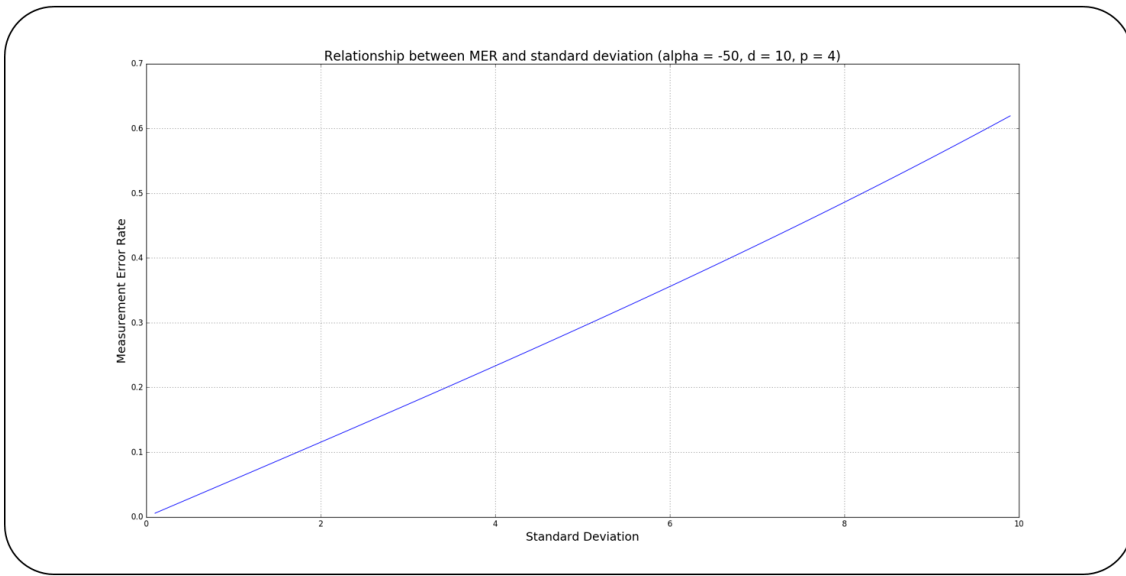


Figure 39: Relationship between MER and σ

Thus, MER can be written as $MER(\sigma, p)$. $MER(\sigma, p)$ is a good performance indicator:

- MER represents the fluctuation of measurements.
- MER is independent of the antenna gain and the transmit power.
- MER is also independent of the distance between transmitters and receivers. It means that, if the measured distance is larger, the fluctuation is also larger.

In the following simulation, we use the fixed $p = 4$ and change σ to get different MER to evaluate the performance of algorithms.

4.2 *Performance among different locations of anchors*

In this section, we use the simulation to verify the correctness of our anchor location determination algorithm. Here three local max-min points are picked. One is the best location of the anchor (black point), and the others are not, as shown in

Figure 40. The value of the local max-min distance can refer to the color bar on the right-hand side of the Figure.

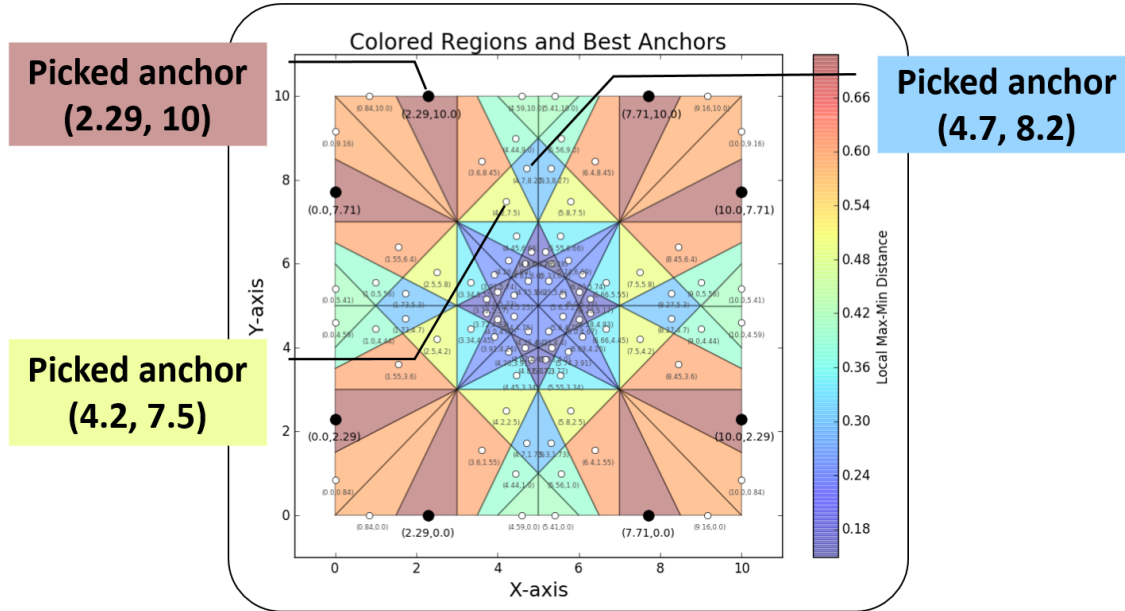


Figure 40: Picked locations of anchors

In the following simulation, the majority mechanism mentioned in the Chapter 3.4 is used. The majority number used here is 1000, which means that the algorithm is repeated for 1000 times to choose the majority of mappings as the final answer. For each MER, we calculate the success rate which is defined as follows:

$$Success\ Rate = \frac{Successful\ experiment\ times}{Total\ experiment\ times}. \quad (4.9)$$

Total experiment times here for each MER are 100, and the successful experiment means that the final answer of the configuration is correct. The simulation result is shown in Figure 41. The result shows that the best location of the anchor (black point) has the best performance among three locations, and the performance is related to the local max-min distance. This result verifies our anchor location determination algorithm. More examples of comparing different locations of anchors are in Appendix B.

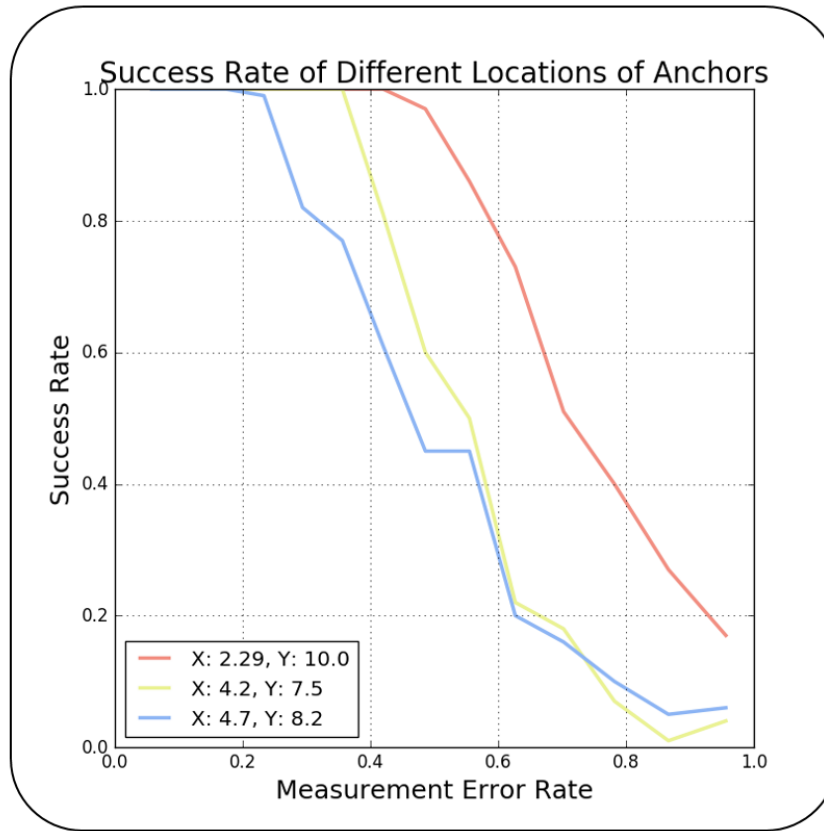


Figure 41: Performance among three different locations of anchors

4.3 Performance among different algorithms

We will compare three algorithms in this section: 1) topology-based configuration algorithm, 2) trilateration-based configuration, and 3) MDS-ICP-based configuration. First, we explain the simulation setting of each algorithm and then discuss the performance in the end.

Simulation settings about trilateration-based configuration:

- Assume the trilateration-based configuration can transform RSSIs back to distances with a perfect transformation model.
- Three anchors of trilateration are located on the edge of the boundaries, as shown on the right-hand side of Figure 42.

Simulation settings about MDS-ICP-based configuration:

- Using only RSSI to do configuration is too inaccurate to do configuration, so for MDS-ICP, we still transform RSSIs back to distances with a perfect transformation model.
- MDS-ICP cannot handle the symmetric topology, so we add an anchor to break the symmetry. The location of the anchor is the same as the topology-based algorithm, as shown in the left-hand side of Figure 42.

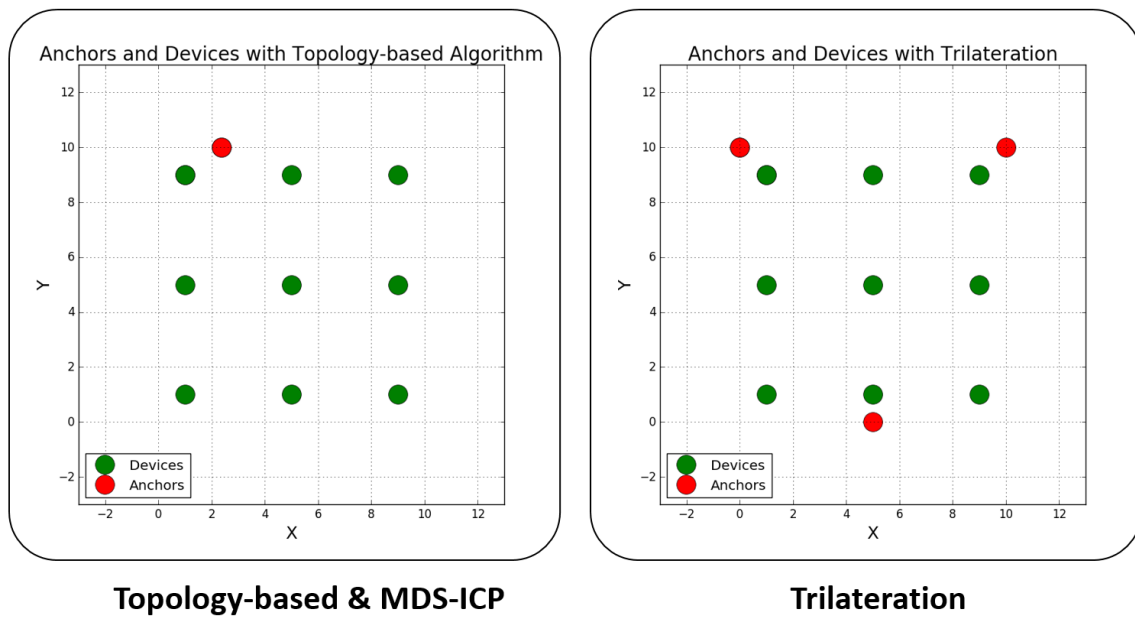


Figure 42: Locations of anchors and devices for three algorithms

In addition to the settings above, the followings are common settings for three algorithms:

- Make sure that each device will only be mapped to one location. For trilateration and MDS-ICP, if two devices find the same location, only one of them can pick that location and the other must pick another closest location.

- All three algorithms adopt the majority mechanism for accuracy and we will use different majority numbers in the following simulation.

Figure 43 to Figure 46 are the performances among three algorithms with different majority. Figure 43 is the performance without majority mechanism. We can see that MDS-ICP cannot achieve 80% success rate even when MER is very low, and the performance of our algorithm is slightly better than trilateration. However, with the majority number become 500, all performances of three algorithms become better. The difference of performances among three algorithms also becomes clear. Our solution can achieve 80% success rate with 0.5 MER, while trilateration and MDS-ICP can only achieve 80% success rate with about 0.35 MER and 0.2 MER respectively.

When the majority number become 1000, we can see that the performance of MDS-ICP is the same as the majority number equals to 500. Here we also simulate the MDS-ICP with only RSSI, and the result shows that it is too inaccurate to use. On the other hand, our solution and trilateration can achieve 80% success rate with about 0.55 MER and 0.38 MER respectively. Next, when the majority number becomes 5000, we can see that the difference of performances among three algorithms becomes much larger. Our solution can achieve 80% success rate with 0.8 MER, while trilateration can only achieve 80% with 0.5 MER. The performance of MDS-ICP is still the same as when the majority number equals to 500.

We can see that as the majority number becomes larger, the performance of our algorithm becomes better than other algorithms. The key point is about how to find each mapping. For trilateration and MDS-ICP, when they estimate the location of a device, the estimated location can not exceed a limited region, as shown in Figure 47. This will cause errors when MER becomes larger. Even the majority number becomes larger, the problem cannot be solved. However, our algorithm does not have the limited region problem because we only compare values between RSSIs. Even if the MER becomes larger, the statistical trend of two RSSIs with different



distances will be not the same. This is the reason why our algorithm is better than the others.

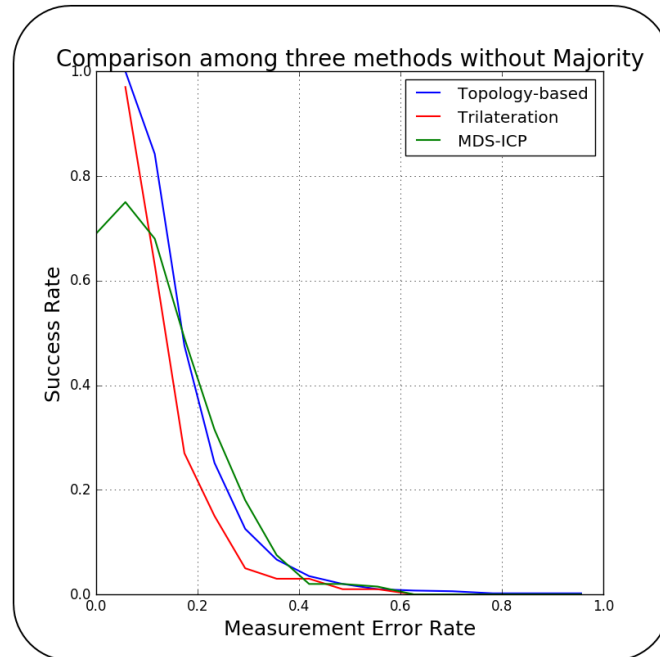


Figure 43: Comparison among three methods without Majority

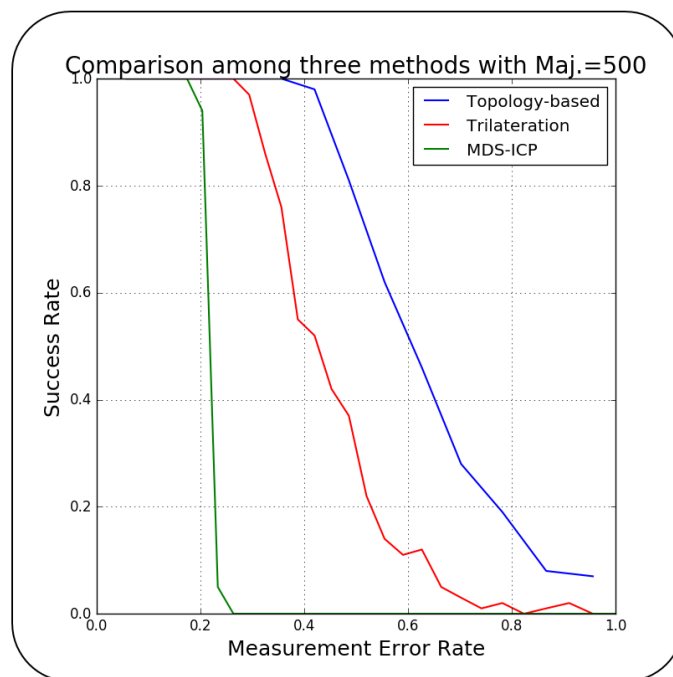


Figure 44: Comparison among three methods with majority number=500

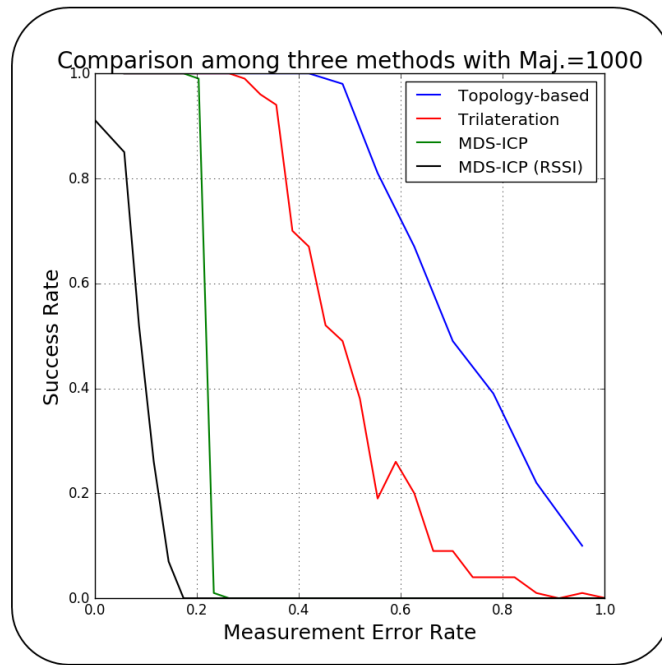


Figure 45: Comparison among three methods with majority number=1000

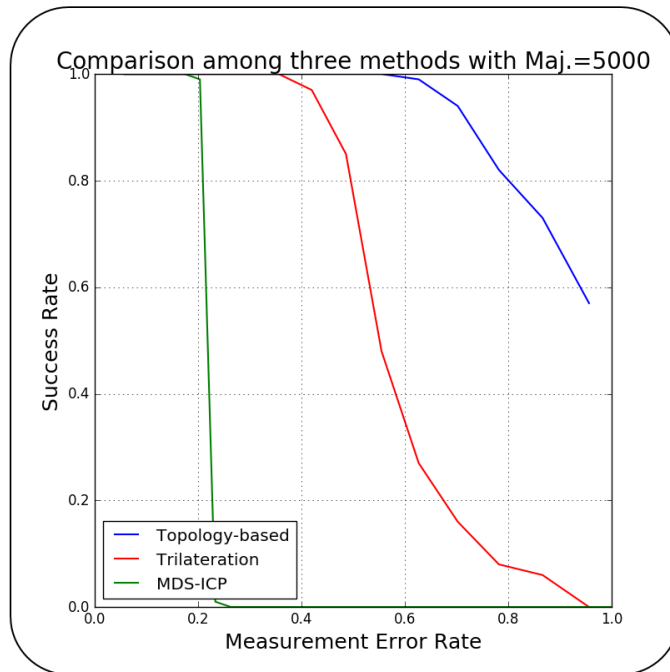


Figure 46: Comparison among three methods with majority number=5000

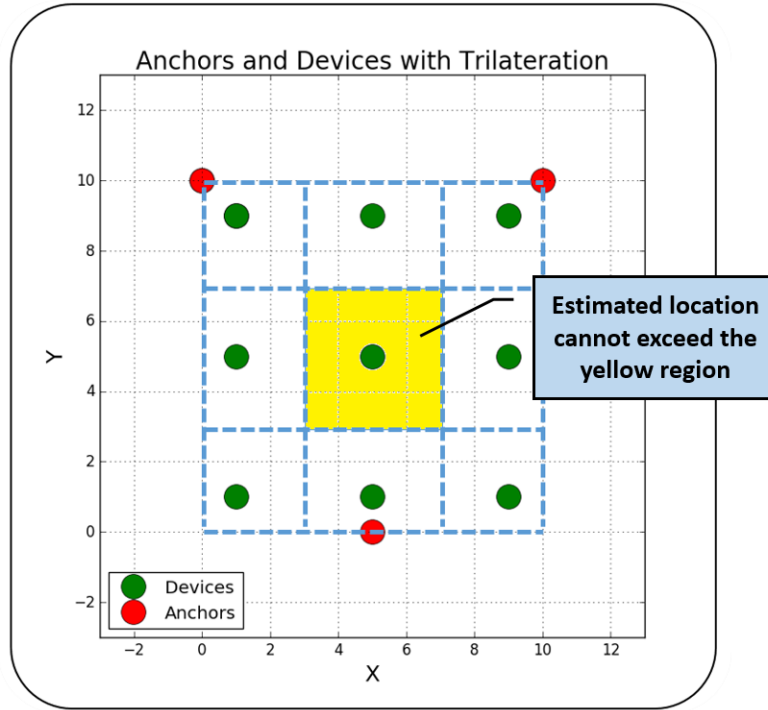


Figure 47: The reason why our algorithm is better

4.4 Analytical Performance Analysis

In this section, we try to analyze the success rate of the topology of previous example without majority. The location of the anchor is the same as Figure ???. The RSSI between different devices is a Gaussian random variable defined as $R_{i,j}$, where i and j are sequence numbers of devices. Assume α and p are the same as Section 4.1, $R_{i,j}$ can be defined by σ and the distance between the two devices. Assume we have two Gaussian random variable $A = R_{i,j}$ and $B = R_{m,n}$, the formula to compare the two random variables are as follows:

$$\begin{aligned}
 P(A < B) &= \int_0^\infty \int_0^y f_B(y) f_A(x) dx dy \\
 &= \int_0^\infty f_B(y) Pr(A < y) dy.
 \end{aligned}
 \tag{4.10}$$

Here we define the event *Success* as finding the correct configuration and define

$Success_{i \rightarrow i+1}$ as the event that we successfully find the next sequence number of device from sequence number i . The probability of $Success$ can be defined as follows:

$$P(Success) = \prod_{i=0}^7 P(Success_{i \rightarrow i+1}) \quad (4.11)$$

Refer to Figure 20, the $R_{0,1}$ must be greater than all other RSSI. Thus, the probability $P(Success_{0 \rightarrow 1})$ can be defined as :

$$P(Success_{0 \rightarrow 1}) = \prod_{i=2}^9 P(R_{0,1} > R_{0,i}). \quad (4.12)$$

Refer to Figure 21, we can see that $R_{1,2}$ must be at least greater than all other RSSI except for $R_{1,4}$. For simplicity, we assume $R_{1,4}$ is also greater than all other RSSI except for $R_{1,2}$. Then, $R_{0,2}$ must be greater than $R_{0,4}$ in Figure 22. Therefore, the upper-bound of probability $P(Success_{1 \rightarrow 2})$ can be defined as :

$$P(Success_{1 \rightarrow 2})_{upper-bound} = \left(\prod_{i=3, i \neq 4}^9 P(R_{1,2} > R_{1,i}) \right) \times P(R_{0,2} > R_{0,4}). \quad (4.13)$$

Similarly, the upper-bound of probability $P(Success_{2 \rightarrow 3})$ can be defined as:

$$P(Success_{2 \rightarrow 3})_{upper-bound} = \left(\prod_{i=4, i \neq 9}^9 P(R_{2,3} > R_{2,i}) \right) \times P(R_{1,3} > R_{1,9}). \quad (4.14)$$

Next, we want to calculate $P(Success_{3 \rightarrow 4})$. Refer to Figure 23, we only consider that $R_{3,4}$ is greater than $R_{3,5}$, $R_{3,7}$, and $R_{3,9}$ for simplicity. From Figure 24 and 25, we know that $R_{3,4}$ and $R_{3,8}$ must be greater than $R_{3,6}$ and $R_{1,4}$ must be greater than $R_{1,8}$. Then the probability can be defined as:

$$\begin{aligned} P(Success_{3 \rightarrow 4})_{upper-bound} &= \left(\prod_{i=5, i \neq 6, 8}^9 P(R_{3,4} > R_{3,i}) \right) \times P(R_{2,4} > R_{2,6}) \\ &\quad \times P(R_{2,8} > R_{2,6}) \times P(R_{1,4} > R_{1,8}). \end{aligned} \quad (4.15)$$

For the rest probabilities, devices only need to find the closest devices and the probability can be defined as:

$$P(\text{Success}_{i \rightarrow i+1}) = \prod_{j=i+2}^9 P(R_{i,i+1} > R_{i,j}), \text{ where } 4 \leq i \leq 7. \quad (4.16)$$

The curve of the analytical result and the simulation result are shown in Figure 48. We can see that the analytical result corresponds with the simulation.

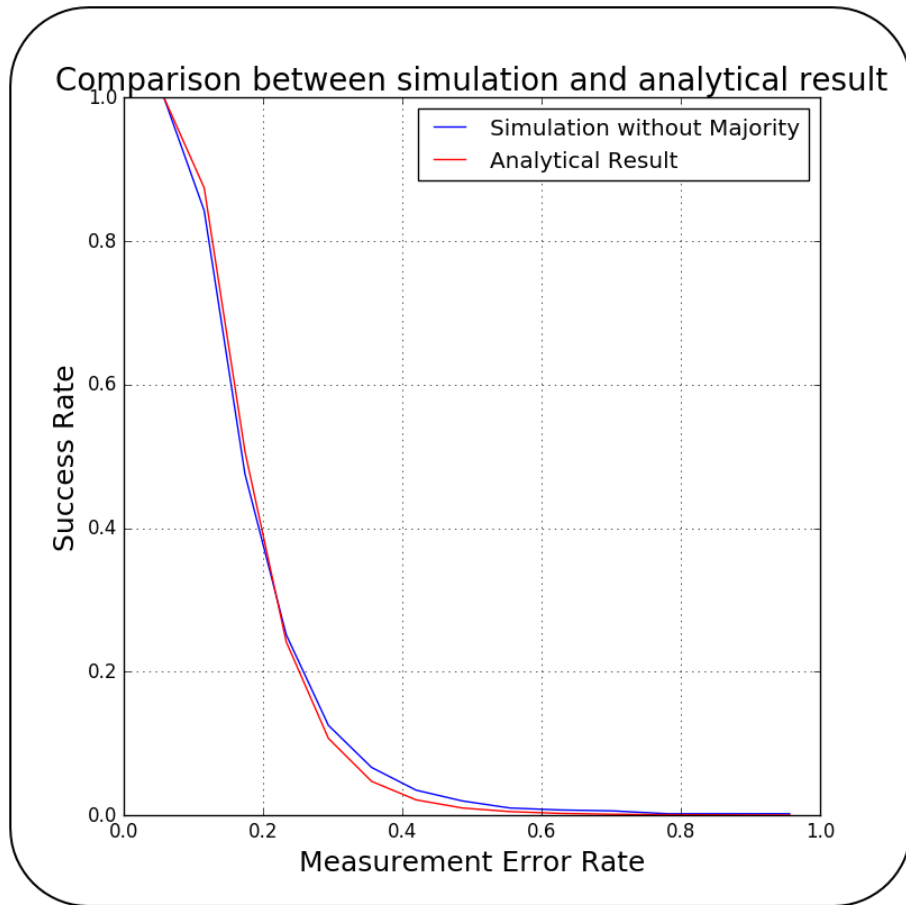


Figure 48: Comparison between simulation and analytical result

We also plot all $P(\text{Success}_{i \rightarrow i+1})$, as shown in Figure 49. Obviously, the bottleneck of this topology is at sequence number 3. With $\text{MER}=0.2$, $P(\text{Success}_{3 \rightarrow 4})$ is under 0.6 while other $P(\text{Success}_{i \rightarrow i+1})$ are greater than 0.8. This makes sense because from sequence number 3 to 4, it encounters the most ambiguity number. Also, only

sequence number 3 need to ask previous devices twice.

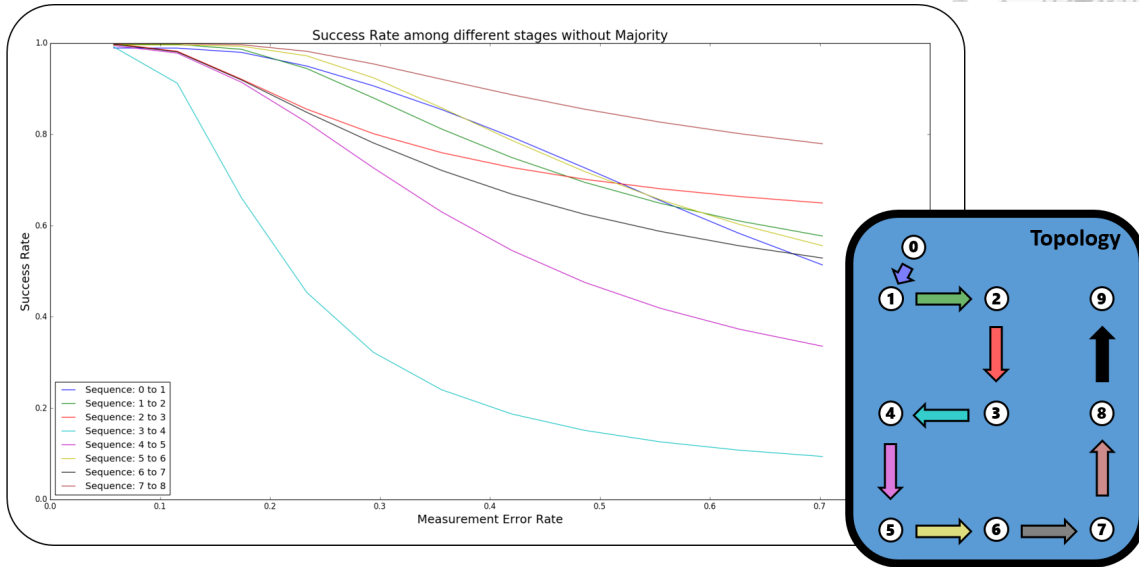


Figure 49: All $P(Success_{i \rightarrow i+1})$, $i = 1, 2, \dots, 7$

If we can use two anchors, we can assign the second anchor to the location of sequence number 3 as a “checkpoint”. From doing this, we can avoid the bottleneck of this topology and become more robust. Other topologies can use the similar analyze method to calculate all $P(Success_{i \rightarrow i+1})$ and find the bottleneck of that topology.

4.5 Protocol and Implementation

In addition to designing the topology-based algorithm, we also design a simple protocol for the algorithm. We adopt the concept of the token ring protocol, as shown in Figure 50.

In our protocol, only the device which holds the token can action, and we call that device as the token holder. The way to decide the next token holder is based on our algorithm. We assume that all devices hold the ambiguity list for the current topology. The current token holder uses the topology-based configuration algorithm to find the device of the next sequence number, and then send the token to that device. Thus, the next device becomes the token holder. Until the final device, it will

use flooding to send the token back to the anchor, and this round is done. The system repeats this procedure for majority number times which is set by programmers and each device will record the result of every round. The details of what each device will do are shown in Figure 51.

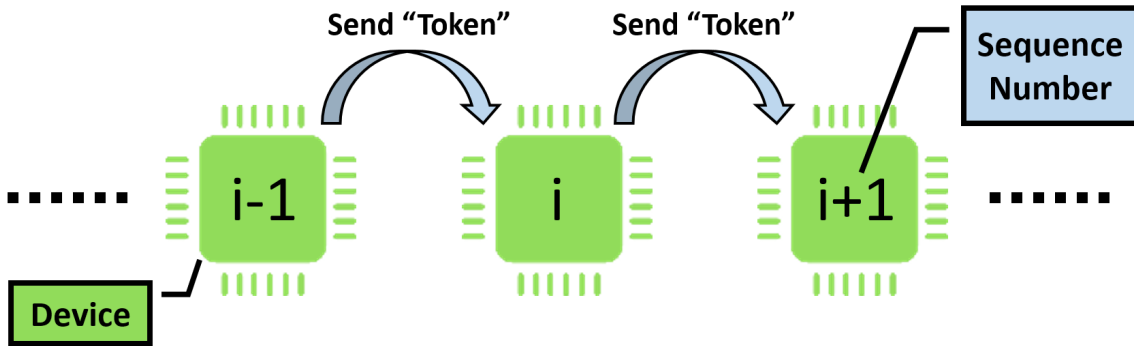
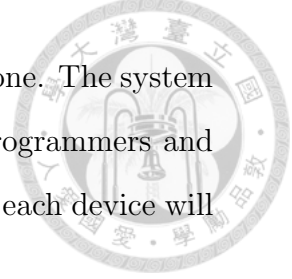


Figure 50: The concept of the protocol

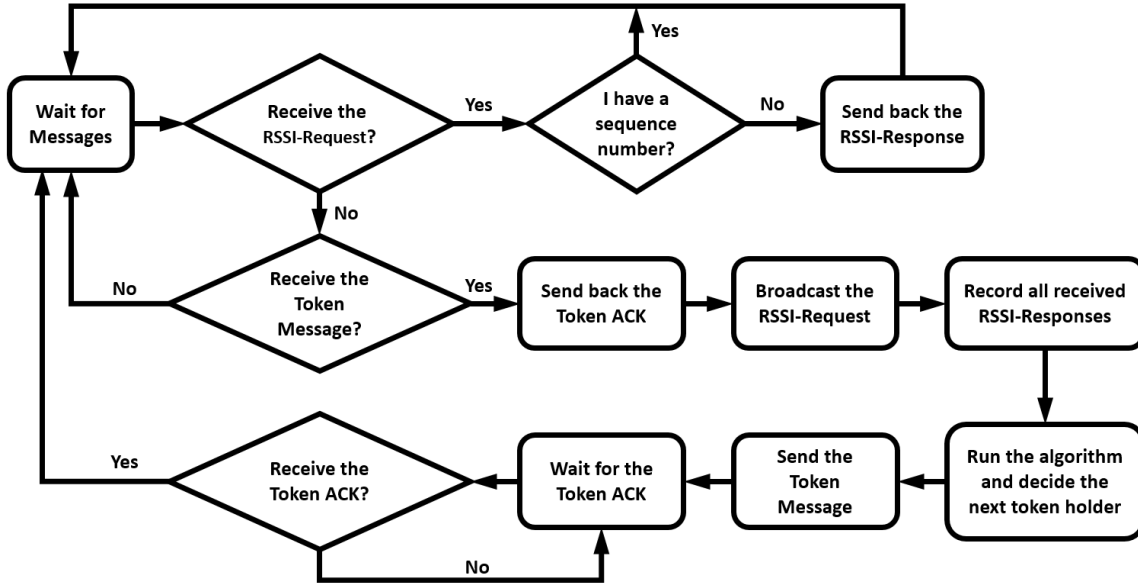


Figure 51: The details of the protocol

There is a good property of this protocol: it can concurrently do multiple rounds. Devices which have a sequence number will do nothing until the end of this round, so these devices can start the next round. Concurrency can accelerate the configuration

speed of our algorithm. Other algorithms cannot achieve concurrency because they need to check if there are two devices in one location or not.

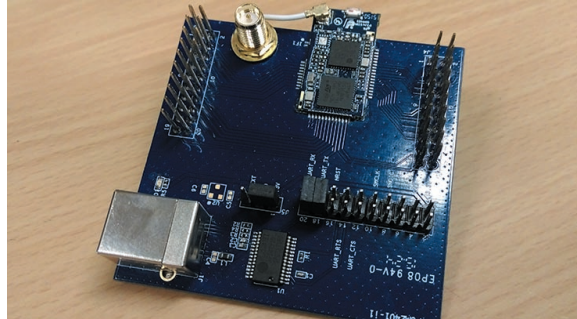


Figure 52: FCM2401 Module

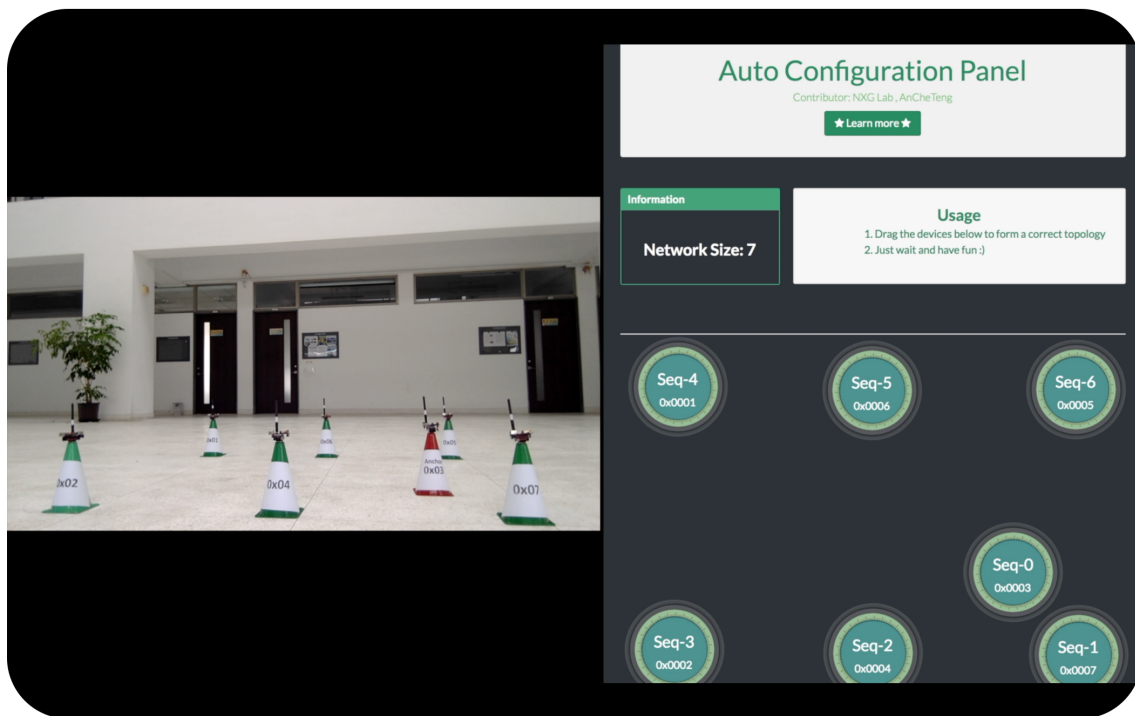


Figure 53: The implementation and indoor environment

The algorithm is easy to implement in the distributed way. We have also implemented the proposed algorithm with the protocol mentioned above on the board FCM2401 which equipped with STM32F0 chip and IEEE 802.15.4 wireless module.

The implementation shows that our algorithm can really be used indoors and without any RSSI model measurement.



4.6 Summary

The performance metrics among three algorithms is in Table 4. The calculation of complexity of topology-based configuration algorithm is as follows. Each device needs to compare at most N RSSIs, where N is the total number of devices. We have N devices, so the complexity is $O(n^2)$. However, if the protocol proposed is used, the complexity can reduce to $O(n)$ due to the concurrency. The performance metrics show that our algorithm is the most accurate and do not have any trade off.

	Trilateration	MDS-ICP	Topology based
Convert RSSIs to distances	Yes	No	No
Accuracy	Medium	Low	High
Complexity	$O(n^2)$	$> O(n^3)$	$O(n^2)$ or $O(n)$
Scaled, Rotated and Flipped	No	Yes	No

Table 4: Performance metrics among three algorithms

CHAPTER 5



CONCLUSIONS

In this thesis, we proposed a topology-based algorithm for location configuration of IoT applications which have a prior topology information. In our solution, developers do not need to estimate the transformation model between RSSIs and distances and the accuracy is very high. Simulations show that, with the grid topology, the performance of our solution can achieve 80% success rate with 0.8 MER, which is much better than other algorithms. Moreover, the complexity can be only $O(n)$ with the protocol designed by us.

The protocol we designed is based on token ring and it can simply achieve concurrency to accelerate the configuration speed. The implementation shows that our algorithm can really be used indoors and without any RSSI model measurement.

Our solution can be used in not only smart agriculture but also many applications such as smart lights in a factory, smart street lights, sensors of smart health-care systems, etc. With our solution, developers and service providers can avoid labor-intensive RSSI measurements and achieve high success rate to configure locations of devices. For the most of IoT applications, configurations of devices will be much easier and effective with the help of our location configuration solution.

APPENDIX A



ALGORITHM TO FIND ALL REGIONS

We design an algorithm to analyze a topology and find all regions without overlapping. We first introduce some definitions with Figure 54. Notice that the blue lines are perpendicular bisectors and the green lines are boundaries, as shown in Figure 30.

- Vertex

Intersect points of perpendicular bisectors. In Figure 54, all red points are vertices.

- Edge

A line connects only two vertices. In Figure 54, edges are a part of perpendicular bisectors or boundaries. Thus, edges are green or blue.

- Region

An area enclosed by edges. Each region is a polygon, and any polygon does not intersect with any edge.

- Degree of an edge

Times to be used to construct a region. In Figure 54, the degree of blue edges is two, and the degree of green edges is one.

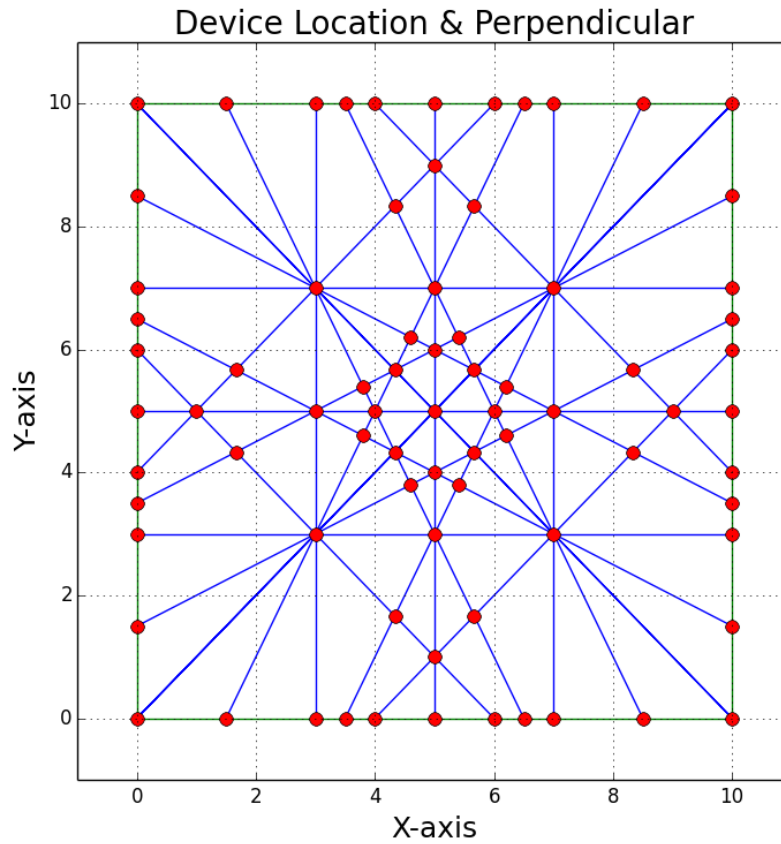


Figure 54: Vertices, edges and regions

Now we want to convert Figure 54 to an adjacency matrix. Figure 55 is a simplified example for converting a graph to an adjacency matrix. The order of each column and row of the matrix represent each vertex in the graph. If there exists an edge between vertex-2 and vertex-4, we record the degree of that edge at $[2, 4]$ and $[4, 2]$ in the matrix. We can see that because the degree of the edge is two, the value of $[2, 4]$ in the matrix is two. Notice that the adjacency matrix is a symmetric matrix.

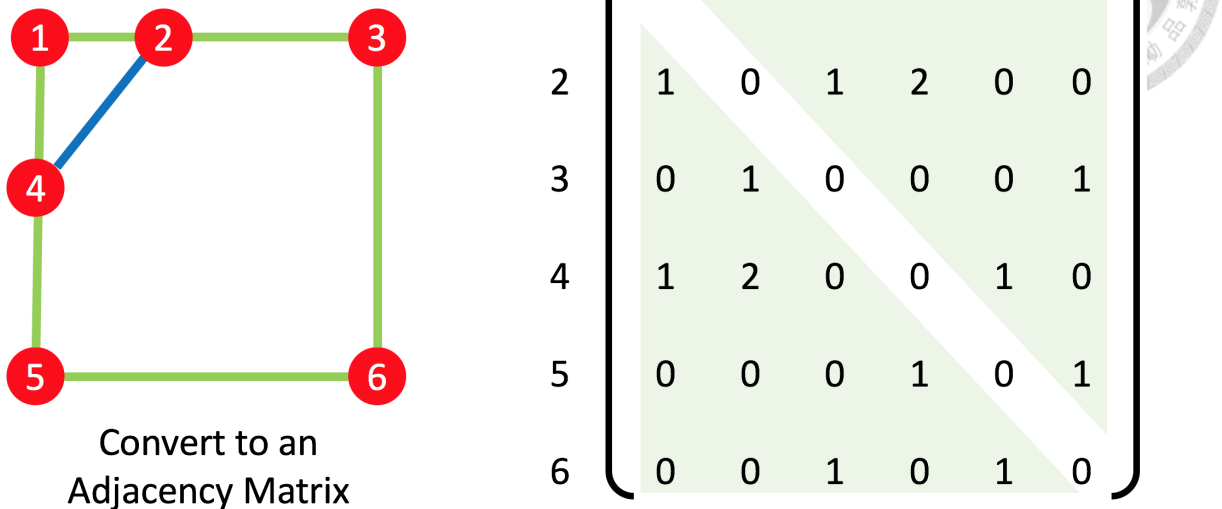


Figure 55: Simplified example for the adjacency matrix

All regions can be found with the adjacency matrix. First, we extract all triangles, because all triangles are regions. As shown in Figure 56, we can find a triangle from one vertex to other two vertices. Here the three edges and three vertices represent a region. We record the region and delete one degree of each edge in the adjacency matrix, as shown in Figure 57.

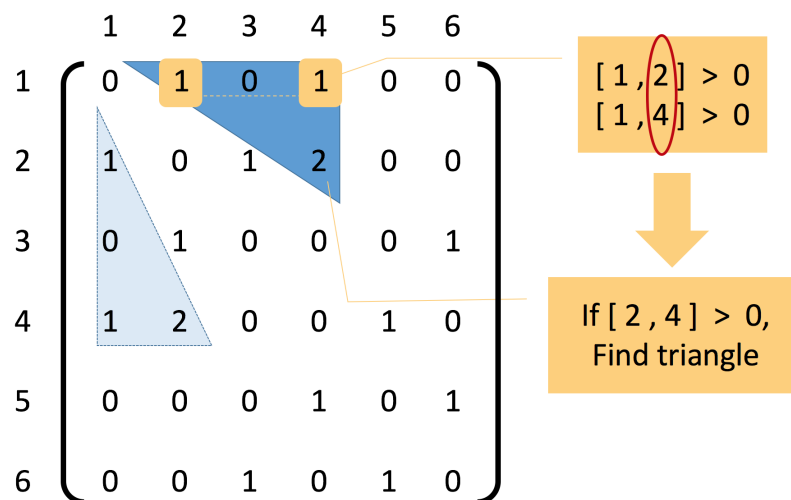


Figure 56: The way to find all triangles



$$\begin{array}{c}
 \\
 \\
 1 \\
 2 \\
 3 \\
 4 \\
 5 \\
 6
 \end{array}
 \begin{pmatrix}
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 1 \\
 0 & 1 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 1 & 0 & 1 \\
 0 & 0 & 1 & 0 & 1 & 0
 \end{pmatrix}$$

Figure 57: Remove degrees of edges

After extracted a triangle, the graph and the adjacency matrix is shown in Figure 58. Now the remained region is a polygon with five edges. It is too difficult to find a polygon with five edges just as what we do with triangles, so we use an alternative way to extract the polygon. We “fold” the adjacency matrix as the following rules:

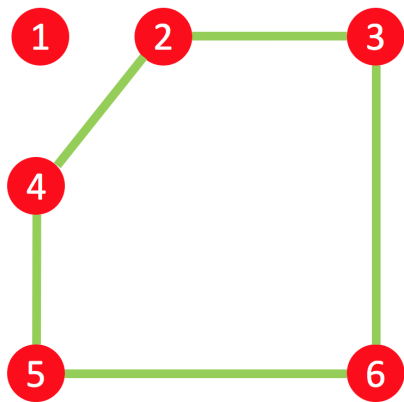
Step-1 Find a vertex which is connected by only two edges. The two another vertices of the two edges form a folded edge. For example, in Figure 59, if we find a vertex-2 which has two edges $[3, 2]$ and $[4, 2]$, vertex-3 and vertex-4 form a folded edge $[3, 4]$.

Step-2 Remove degrees of the two edges on the original adjacency matrix and record them.

Step-3 Add the degree of the folded edge to the folded adjacency matrix

Step-4 Repeat Step-1 to 3 until the original adjacency matrix becomes a zero matrix

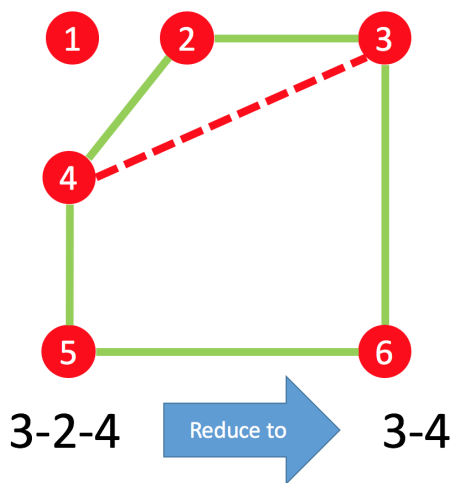
Figure 59 to 61 are each step of our algorithm, and the folded result is in Figure 62.



The Graph
(after extracted)

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	1	1	0	0
3	0	1	0	0	0	1
4	0	1	0	0	1	0
5	0	0	0	1	0	1
6	0	0	1	0	1	0

Figure 58: The graph and the adjacency matrix after extract a triangle



	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	1	1	0	0
3	0	1	0	0	0	1
4	0	1	0	0	1	0
5	0	0	0	1	0	1
6	0	0	1	0	1	0

Take 3-2-4

Figure 59: Fold the $[3, 2]$ and $[3, 4]$

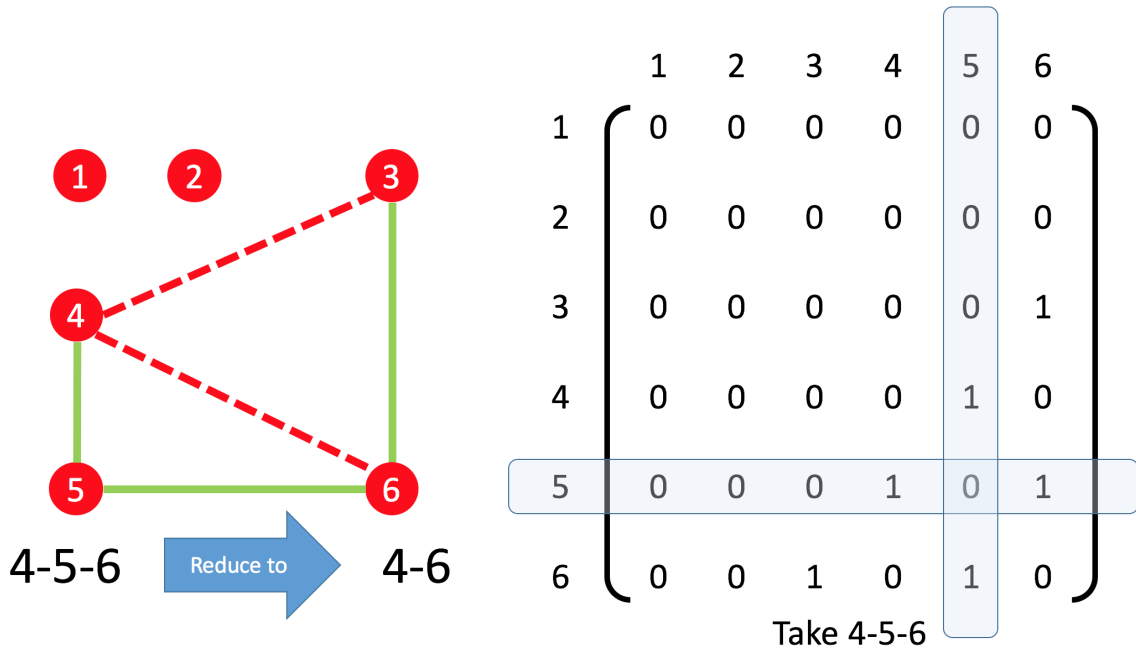


Figure 60: Fold the $[5, 4]$ and $[5, 6]$

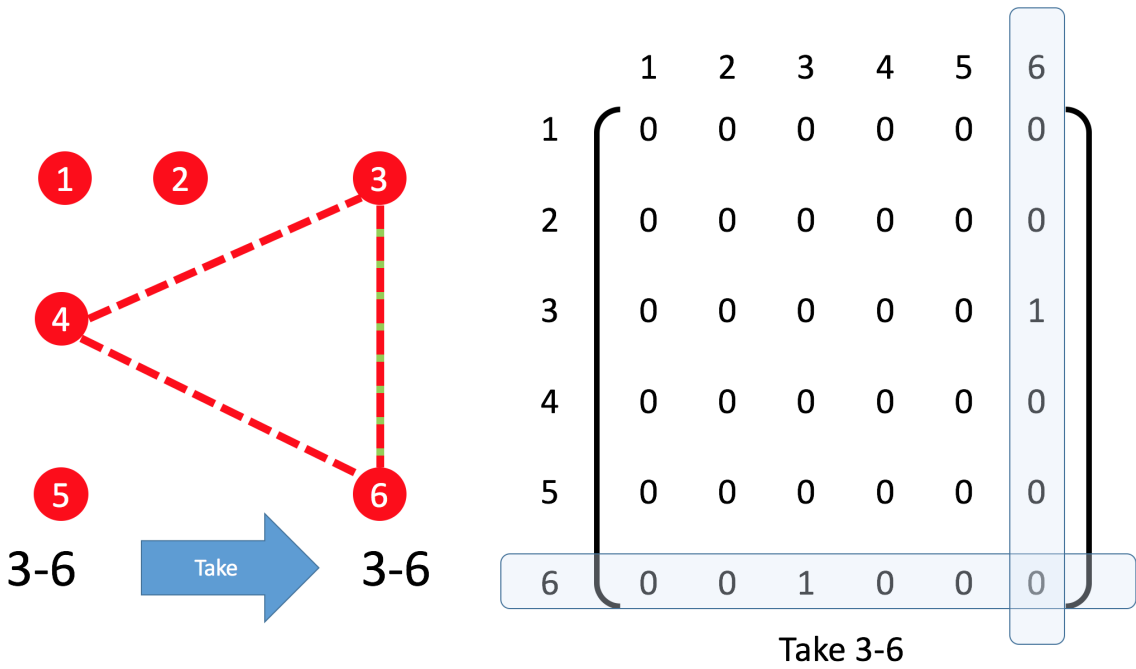


Figure 61: Extract the edge $[3, 6]$

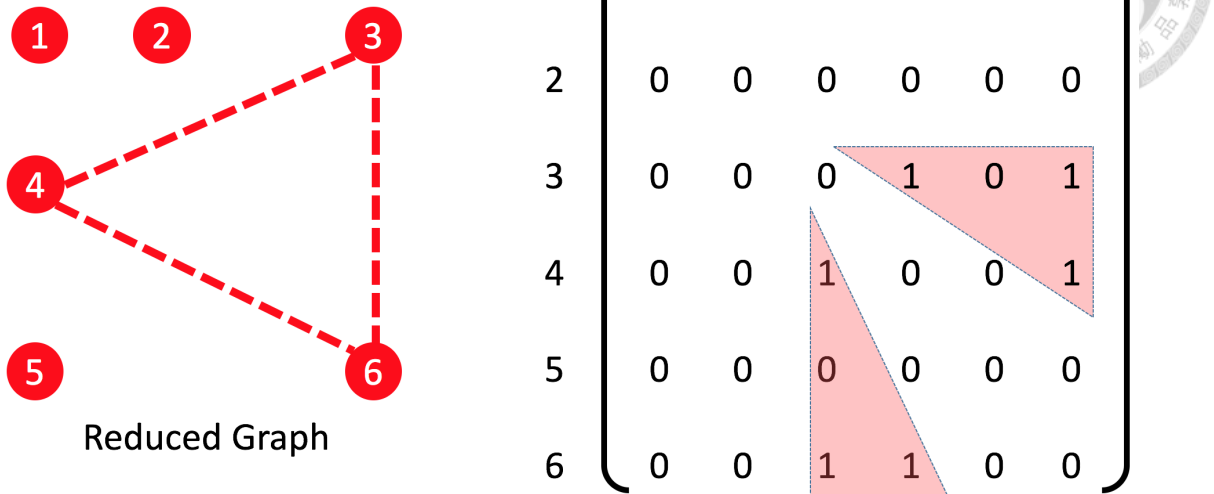


Figure 62: The folded graph and folded adjacency matrix

We can extract the triangle from the folded adjacency matrix and then transform the triangle back to the polygon in Figure 58. If the polygon folded is still not a triangle, just fold it recursively. We can repeat the folding and extract triangles to find all regions and the complete algorithm is shown in Figure 63. Here we also show another more complicated example in Figure 64 to 66.

Euler planar theorem is another way to check the correctness of the number of all regions. The graph is a planar graph because of the definition of vertices. There is no edge intersection and the each edge does not cross another edge. Thus, the graph is a planar graph. Euler planar theorem is as follows:

Euler Planar Theorem. *If a finite, connected, planar graph is drawn in the plane without any edge intersections, and v is the number of vertices, e is the number of edges and r is the number of regions (regions bounded by edges, including the outer, infinitely large region), then*

$$v - e + f = 2 \quad (\text{A.1})$$

Because our algorithm does not consider the outer, infinitely large region, we use $v - e + f = 1$ as the formula to check the number of regions.

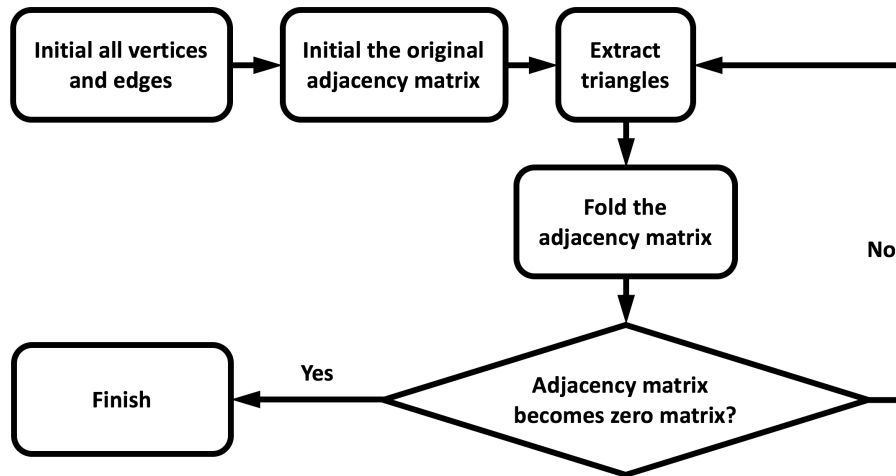


Figure 63: The algorithm to find all regions

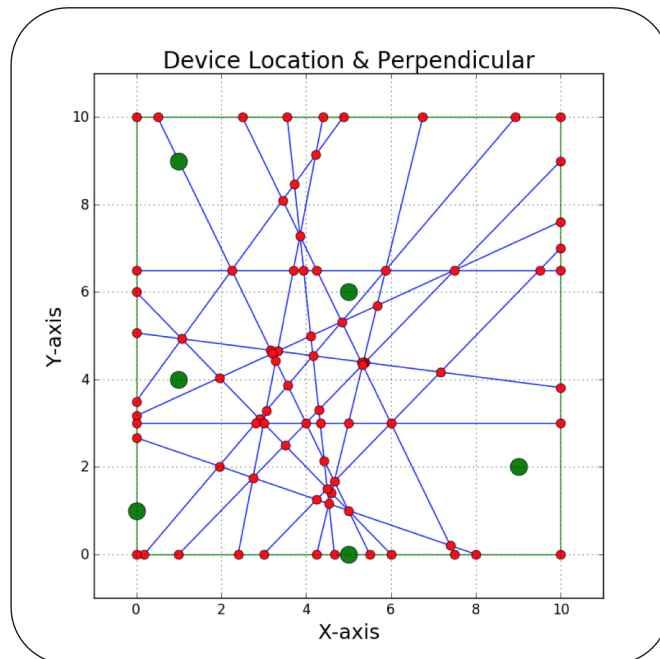


Figure 64: A more complicated topology

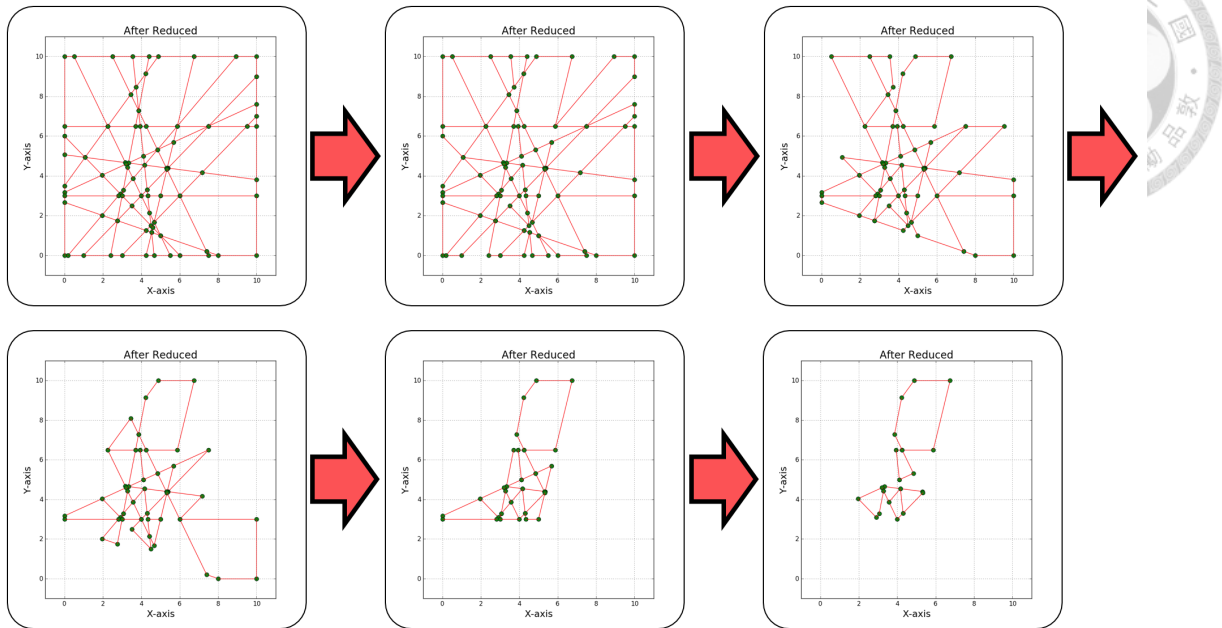


Figure 65: Recursively find all regions

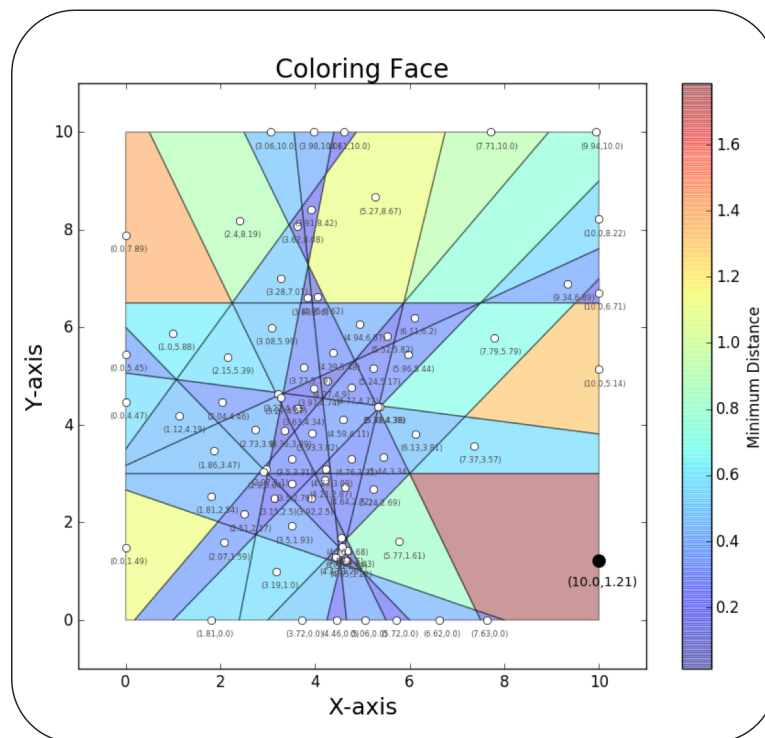


Figure 66: Final result of finding the max-min point

APPENDIX B



EXAMPLES OF OTHER TOPOLOGIES

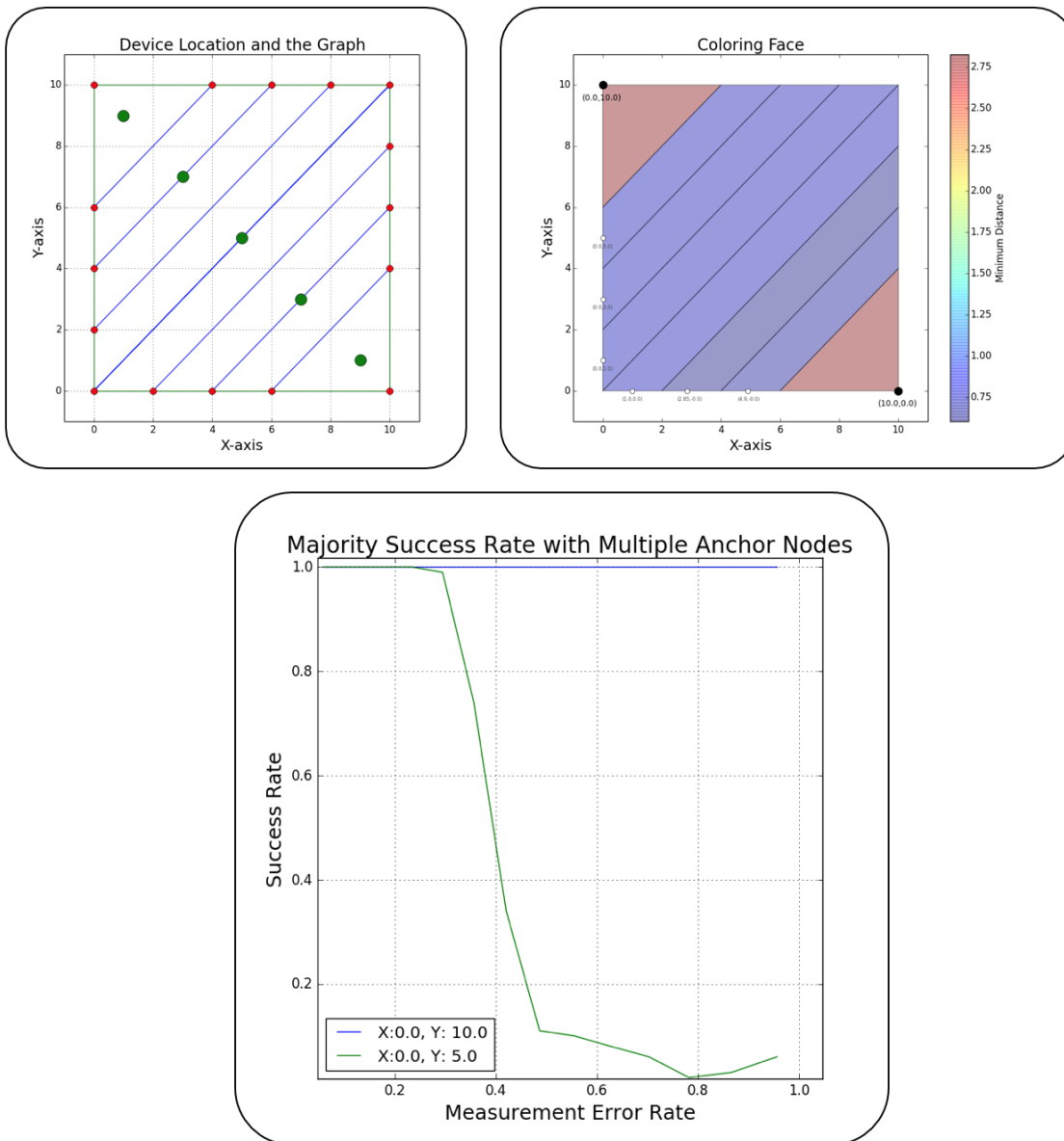


Figure 67: Example of topology 1

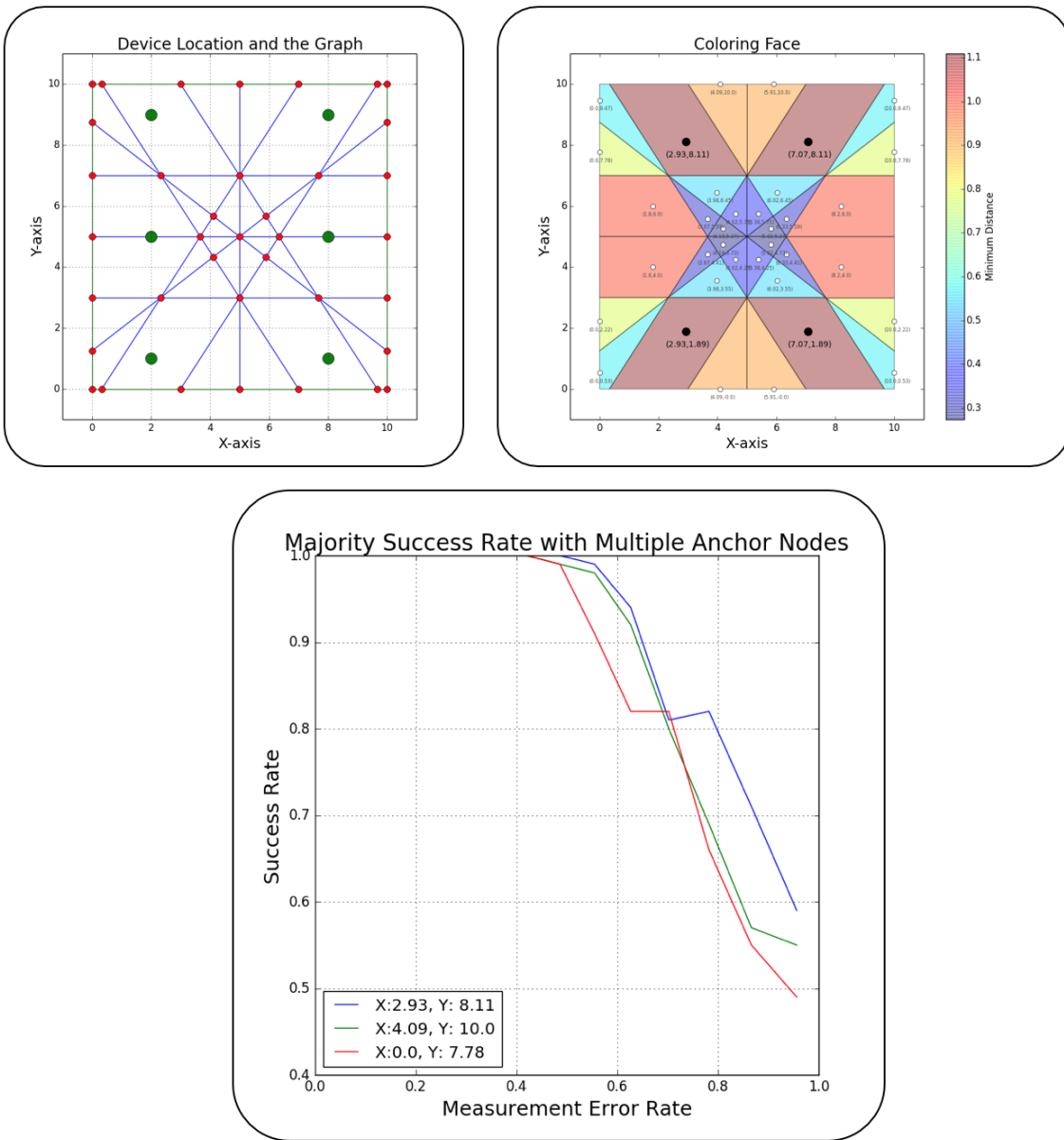
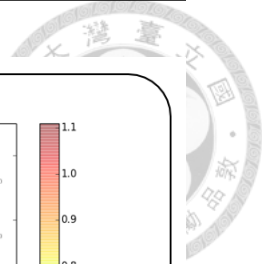


Figure 68: Example of topology 2

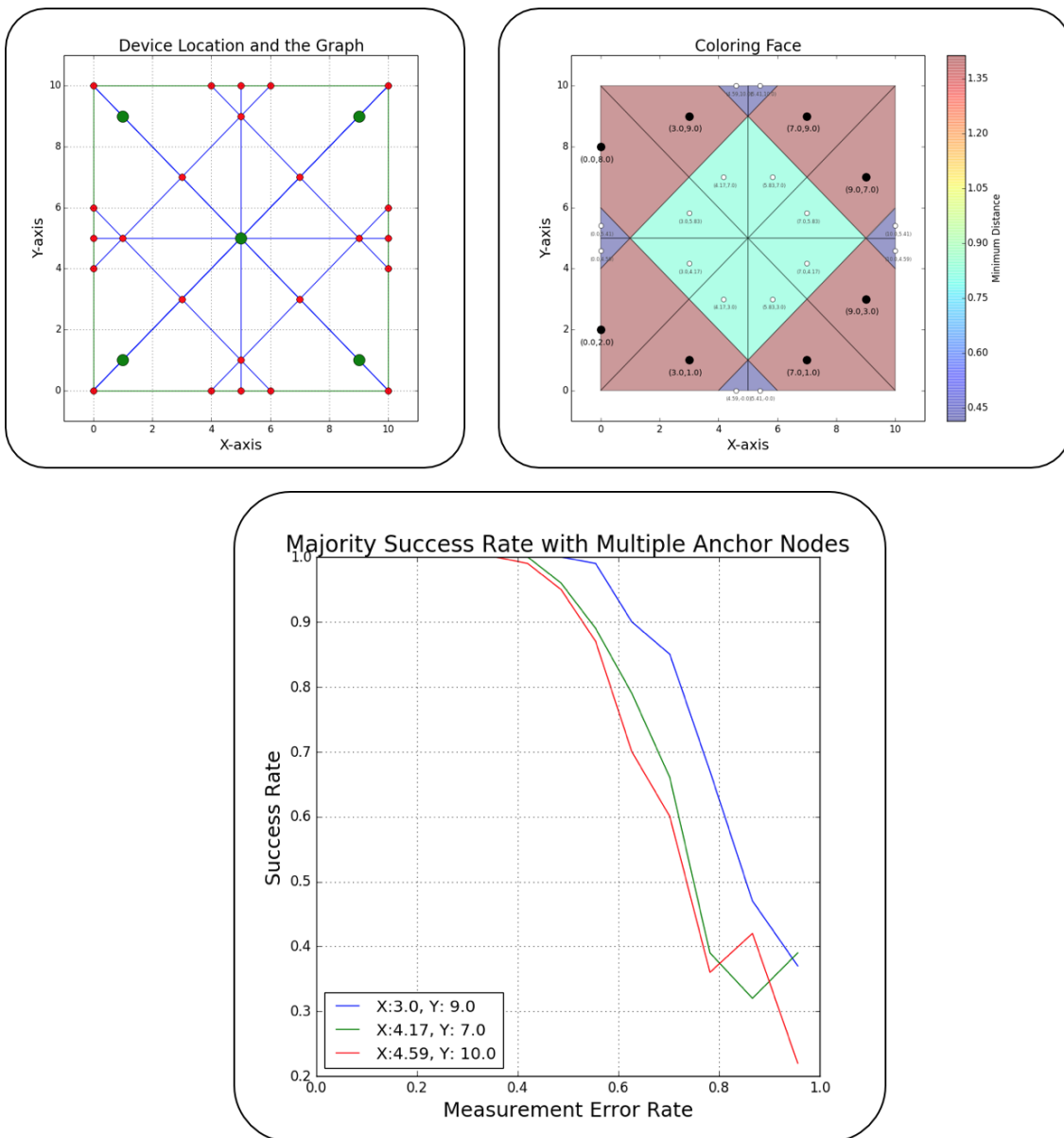


Figure 69: Example of topology 3

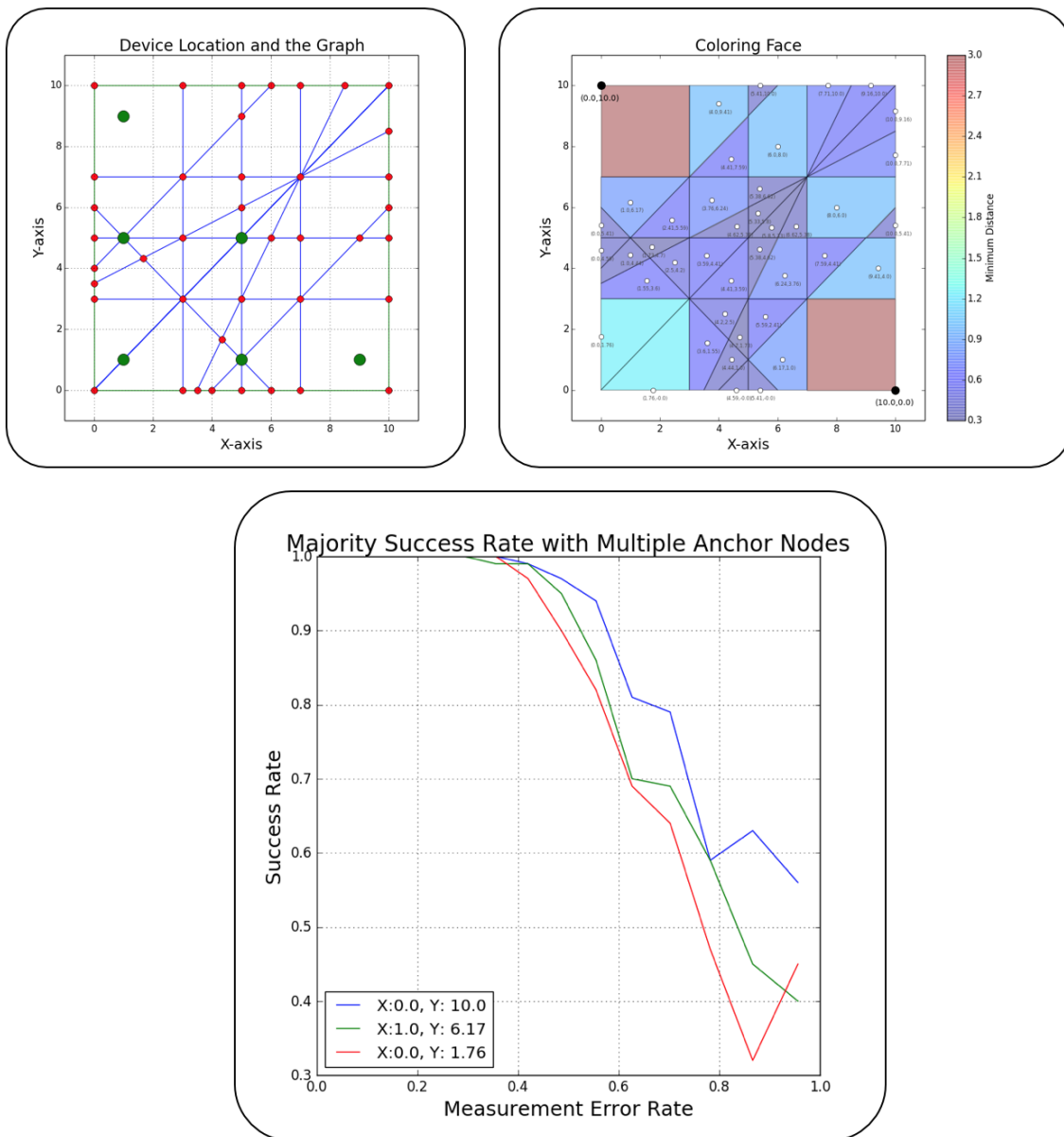


Figure 70: Example of topology 4

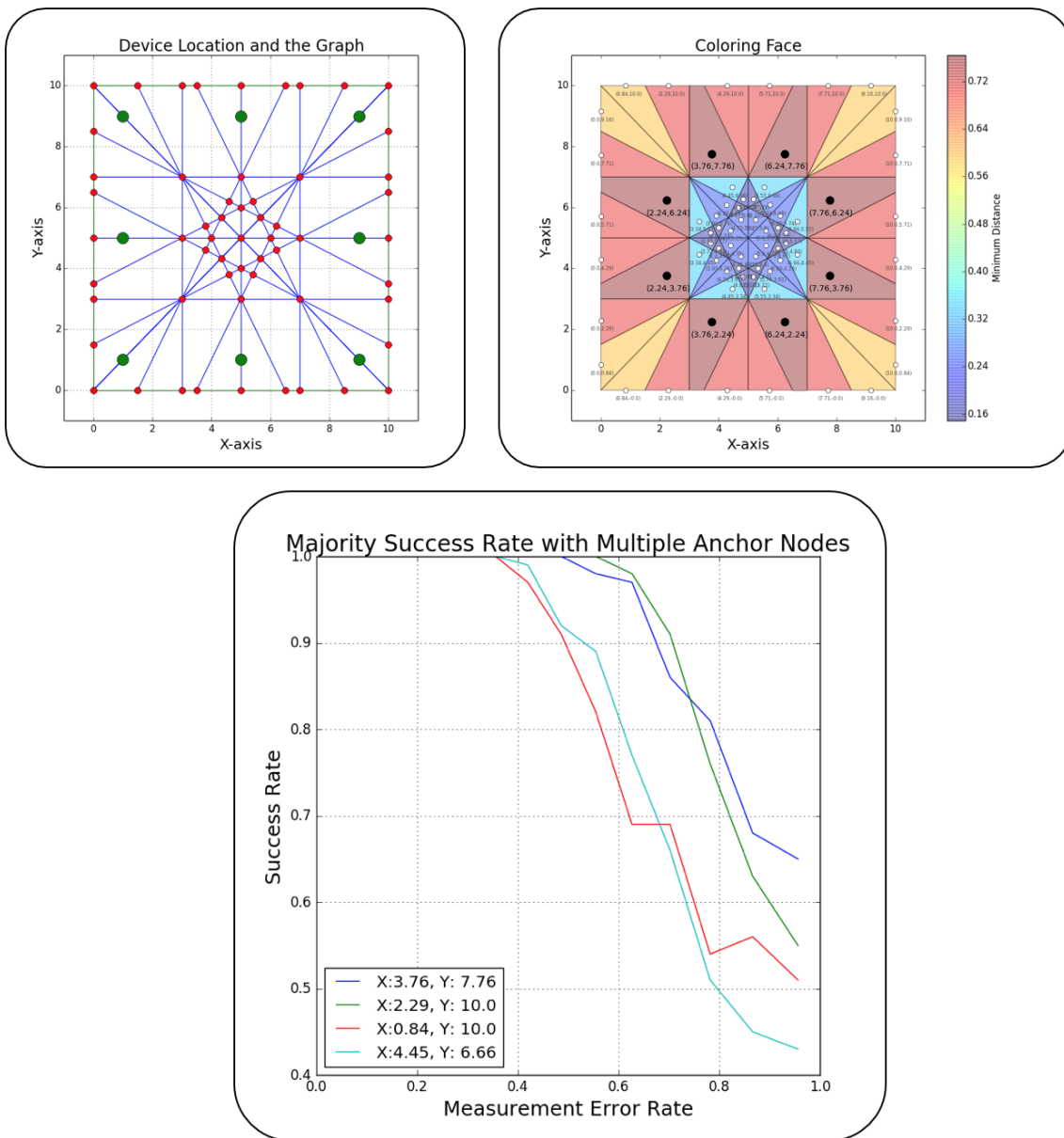



Figure 71: Example of topology 5

REFERENCES



- [1] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things journal*, vol. 1, no. 4, pp. 349–359, 2014.
- [2] "Top 6 Fast Growing Tech Industries in China - Internet of Things and Social Media." <http://www.thexnode.com/blog/top-6-fast-growing-tech-industries-in-china-internet-of-things>.
- [3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
- [4] J. Xu, W. Liu, F. Lang, Y. Zhang, C. Wang, *et al.*, "Distance measurement model based on rssi in wsn.," *Wireless Sensor Network*, vol. 2, no. 8, pp. 606–611, 2010.
- [5] S. Shrestha, E. Laitinen, J. Talvitie, and E. S. Lohan, "Rssi channel effects in cellular and wlan positioning," in *Positioning Navigation and Communication (WPNC), 2012 9th Workshop on*, pp. 187–192, IEEE, 2012.
- [6] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 166–179, ACM, 2001.
- [7] L. Doherty, L. El Ghaoui, *et al.*, "Convex position estimation in wireless sensor networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1655–1663, IEEE, 2001.

- 
- [8] D. Moore, J. Leonard, D. Rus, and S. Teller, “Robust distributed network localization with noisy range measurements,” in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 50–61, ACM, 2004.
- [9] K. A. Khalid and T. A. Gulliver, “Range-based localization in wireless networks using decision trees,” in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 131–135, IEEE, 2010.
- [10] R. Nagpal, H. Shrobe, and J. Bachrach, “Organizing a global coordinate system from local information on an ad hoc sensor network,” in *Information Processing in Sensor Networks*, pp. 333–348, Springer, 2003.
- [11] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.
- [12] X. Ji and H. Zha, “Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling,” in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2652–2661, IEEE, 2004.
- [13] Y. Shang and W. Ruml, “Improved mds-based localization,” in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2640–2651, IEEE, 2004.
- [14] B. Zitova and J. Flusser, “Image registration methods: a survey,” *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.
- [15] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.
- [16] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Robotics-DL tentative*, pp. 586–606, International Society for Optics and Photonics, 1992.

- [17] R. Mautz, "Indoor positioning technologies," 2012.

