# Power Control for Optimizing RFID Tag Reading Rate in Multi-Reader Environment: A Study on Conveyor Belt System

By

XIAODAN FANG

Bachelor of Science in Computer Science
St. Francis Xavier University
Antigonish, NS, Canada
2007

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2009

Power Control for Optimizing RFID Tag Reading Rate in Multi-Reader Environment: A Study on Conveyor Belt System

Thesis Approved:

Dr. Venkatesh Sarangan

Thesis Advisor

Dr. Johnson Thomas

Dr. John Chandler

Dr. A. Gordon Emslie

Dean of the Graduate College

## ACKNOWLEDGMENTS

I would like to acknowledge the regular advice, guidance, encouragement and financial assistance of Dr. Venkatesh Sarangan, committee chairman. I also thank Dr. John Chandler and Dr. Johnson Thomas for being my graduate committee, providing valuable suggestions, and spending time in my thesis's correction. I would also like to thank the faculty and staff in the Computer Science Department for making my experience here a wonderful one.

Lastly, I would like to thank my family members, especially my mother, Yongzhi Cao, who taught me unconditional love; my father Zhongping Fang, who taught me the value of hard work; and my uncle Wansheng Li, who gave me generous support to study abroad. Without their support, I would not have finished the degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

# CHAPTER 1

# INTRODUCTION

## 1.1 Aim Of The Study

Today, RFID (Radio-frequency identification) is used in industry supply chain management to improve the efficiency of inventory tracking and management. Even some credit card companies (American Express) utilize this great technology on their credit cards (Blue Sky from American Express), which could greatly speed up customers' checking out payment process in stores. Several countries also have implemented RFID in passports to make passports more secure [8]. There are also many other prevalent applications of RFID, which we will talk about in more detail in Chapter 2.

In recent decades, with the increasing popularity of this technology in business enterprises, many scholars did research on RFID related topics, such as security of RFID systems, privacy enhancing technologies (PET), collision avoidance in RFID systems, tags reading rate in RFID systems, and so on. However, to make the system more efficient, most systems only use one or two readers to scan tagged items, which is far from enough. Hence many enterprises use multiple readers to achieve high performance. Other than that, companies increasingly make use of multiple readers in conveyor belt systems. Some airports have conveyor belt systems utilizing RFID technique to scan checked baggage for efficiently tracing suitcases of passengers.

According to the previous research on RFID, we believe that no one has researched in power control for optimizing RFID tag reading rate in Multi-reader conveyor belt systems. Therefore we researched on this issue in this thesis. In this thesis, we did

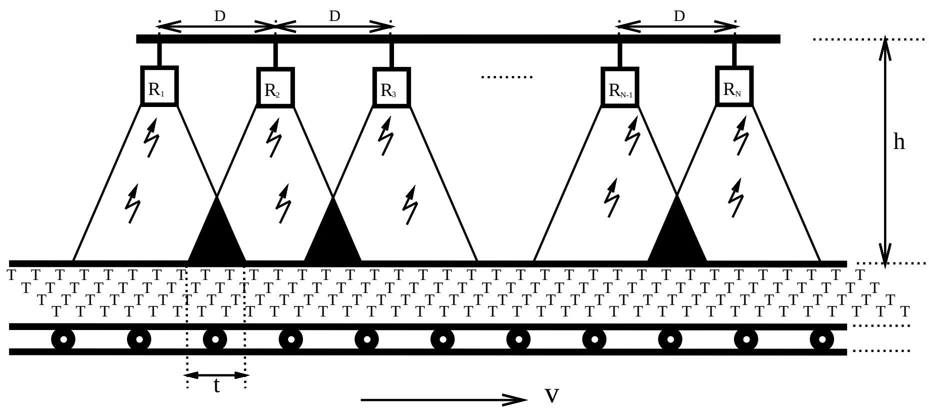

Figure 1.1: Distances of adjacent readers are constant D

some analysis in how the power level of readers affect the system's performance, in terms of the total area of one tag covered by a non-interference region (a tag can communicate well with only one reader in our case) on the conveyor belt. Two situations will be researched in this thesis. One of them (Figure 1.1) is the distance of adjacent readers are fixed, say some constant. The other one (Figure 1.2) is the distance of adjacent readers are not the same, but are random values that are all known to us when we set up the model. A detailed analysis will be performed on the above two situations in this thesis. After we obtain results theoretically, simulation programs (in Java) will be used to test the results and observe the performance. Finally, according to the results, a conclusions section followed by topics for future work will be presented in this thesis.

## 1.2 Related Work

The Master thesis [9] by Rupesh Bhochhibhoya from Oklahoma State University concentrates on mobile tag reading in a multi-reader RFID environment. Figure 1.3 below is the model he used in his thesis. In his thesis, he modeled the conveyor belt system and compared the performance of two models: one is a duty-cycle model (it is a time slotted model, which means group readers in more than one group, and no two

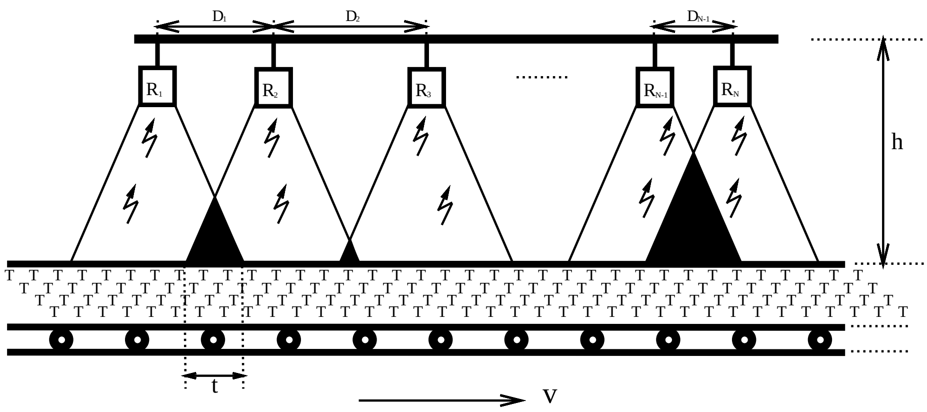


Figure 1.2: Distances of adjacent readers are different constants

groups function at the same time, but each reader in the same group functions simultaneously.); the other is a duty-cycle-free model (all readers work simultaneously). In Figure 1.3, the shaded area is where two adjacent readers interfere (we will explain this concept in section 3.4 later) with each other if the two adjacent readers operate simultaneously. When a tag goes through this region, it cannot respond to either of the two readers because it cannot differentiate the two signals from the left and right side readers. Hundreds of tags are transmitted rightward in speed of v (meters per second) on a conveyor belt.

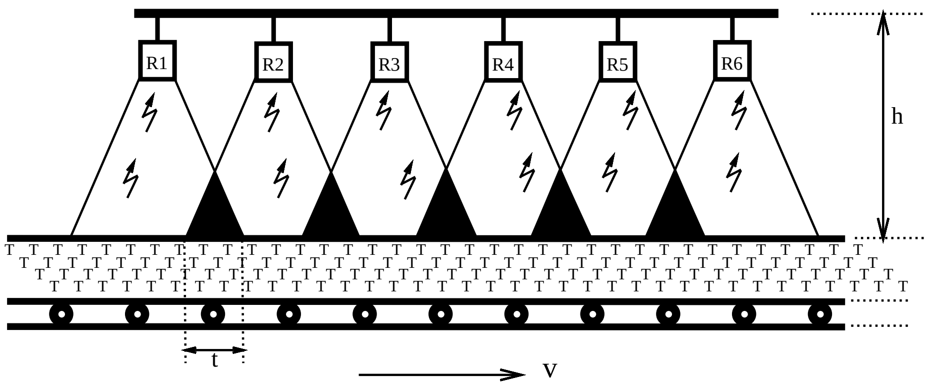


Figure 1.3: Mobile tags flow on conveyor belt (adapted from Bhochhibhoya's thesis)

In his thesis, Bhochhibhoya answered the question of what overlapping region is between two adjacent readers which guarantees that the duty cycle model will outperform the non-duty cycle model (improving the possibility of each tag being read). He also answered a few other questions that are not related with this thesis. By using a similar model, we also want do some work about the overlapping region, or focus on decreasing the reader-reader interference between adjacent readers. In this model, reader-reader interference could only occur between adjacent readers since there is no mutual interrogation area between any two readers far away from each other ("far away" means at least one reader separates them). To achieve this goal, Bhochhibhoya split six readers into two groups (Group 1: $R_1$, $R_3$, $R_5$; Group 2: $R_2$, $R_4$, $R_6$), and each group operates at different times, e.g., when time is at 1 second, readers in Group 1 work, while readers in Group 2 keep mute; when time is at 2.5 second, readers in Group 1 keep mute, while readers in Group 2 work. Therefore, it never happens that two adjacent readers operate simultaneously. As a result, there is no interference between adjacent readers due to this time-slotted operation mechanism.

Utilizing a time-slotted model is one way of decreasing interference between adjacent readers. Another possible way could be to control the power of each reader to eliminate or reduce the reader-reader interference, which is what this thesis will be concerned with. In addition, to make this problem more general, we increased the number of readers to N in our model.

## 1.3 Outline Of The Thesis

The remainder of this thesis will be organized as follows: Chapter 2 will present a general literature review of theory on RFID, such as the history of RFID, what RFID is, and RFID vs. Barcode. Then we introduce: RFID system components, how RFID works, security and privacy issues, and applications of RFID. Chapter 3 will describe RFID collisions in detail. We will cover how collisions arise, several outstanding Anti-Tag-Collision algorithms, and a handful of Anti-Reader-Collision algorithms. Chapter 4 will present the methodology used in this thesis step-by-step. In chapter 5, we provide the experimental results obtained. We give some conclusions based on the outcomes in Chapter 4. Chapter 6 will summarize the work has been done and present possible topics for future work related to this research. Lastly, the simulation program is listed in Appendix A.

# CHAPTER 2

# RFID BACKGROUND

## 2.1 RFID History

Some people might think RFID is a new technology. After you read this section (or look through the following Table 2.1), you might change your opinion. The history of RFID can be traced to World War II. Table 2.1 shows us the detailed history of RFID.

Table 2.1: History of RFID [1]

| Time | Events |
|---|---|
| World War II (1939) | IFF (Identification friend or foe) transponder technology to identify aircraft |
| 1950s to 1960s | US, Japan, and Europe scientists and researchers undertook research in RFID |
| 1973 | Mario W. Cardullo obtained first US patent for active RFID tag (rewritable)<br>Charles Walton received the patent for passive transponders for keyless entry system |
| 1970s | US government engaged some important work on RFID system<br>Los Alamos National Laboratory developed automated toll payment system<br>Some companies developed LF systems (utilized on cattle) |
| 1990s | IBM developed and patented UHF RFID system |
| 1999 | Auto-ID Center at MIT was set up supported by EAN International, Gillette, UCC and P&G<br>RFID was researched for supply chain purpose |
| 1999 to 2003 | Auto-ID Center was supported from thousands companies<br>More research labs were built up in several countries |
| 2003 | RFID Technology was licensed to UCC (Uniform Code Council) |

Although RFID has a long history, it is just being refreshed with research outcomes by worldwide researchers every year.

## 2.2 What Is RFID?

RFID represents Radio-Frequency Identification, which is a technology to identify and track tagged objects using radio waves. “Tagged” means to incorporate a tag in an object (mostly a product, sometimes used with an animal or a person). Similar to Barcode technology, RFID can read a tagged item using an RFID reader. However, it has some advantages compared with Barcode, which we will discuss in more detail in the next section.

A tag (Figure 2.1) also named smart label, is a small electronic device that contains a chip and an antenna. The chip is able to store some data related to the corresponding item, manipulate the radio frequency signal, and perform some other special usage. That data could be an EPC (Electronic Product Code), designed for uniquely identifying each object, normally 64 bits or 96 bits. The Antenna is for sending and receiving a signal [8].

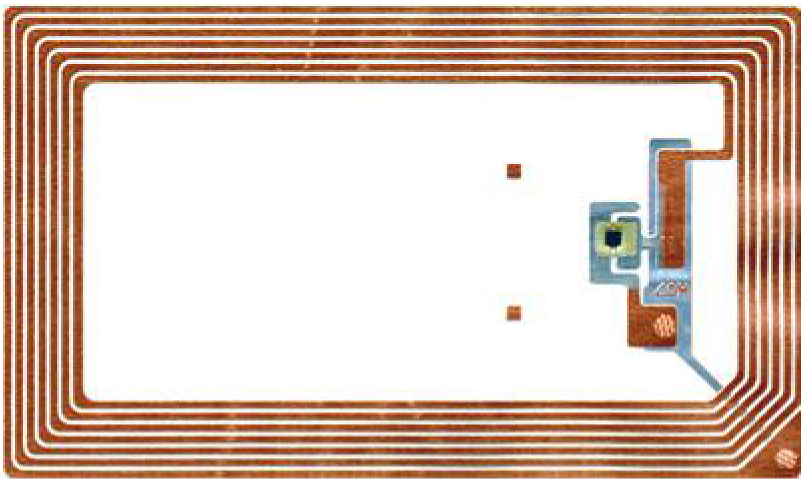

Figure 2.1: An RFID tag (adapted from [4])

Nowadays, tags are classified in three categories in terms of their source of power supply: passive tags, semi-passive tags (semi-active tags) and active tags [10]. Passive tags are very small and do not have any internal power supply. They also do not need to have batteries. Basically they response readers by utilizing the incoming radio frequency signal (such as from a reader nearby). This works in this way: when a tag is under a reader's coverage, the antenna will receive the electromagnetic wave, which will induce current in the coil. This will charge the capacitor of the chip. The capacitor will be the source of working power. The read distance of these tags could be up to a few meters. They have unlimited lifespan and the cost is fairly cheap, around 5 cents for each tag. Unlike passive tags, semi-passive tags have a small amount of battery. Therefore they have faster response and larger reading distance than passive tags. Compared to passive and semi-passive tags, active tags have a larger read distance because they have individual internal power sources. In addition to that, they also have more memory space [10]. Of course, they are expensive and large.

## 2.3 RFID vs. Barcode

RFID and Barcode are both used to identify products. However, there are several differences between the two technologies. First, Barcode is used for tracking a given class of products [11]. For example, in Wal-Mart, the same kinds of products have the same barcode information printed (all Diet Coke have the same barcode information). Also the barcode is exposed on the outside of the product. Normally, Barcode requires tags with a direct LOS (line of sight) from readers. The range to read a tag is up to several feet. Therefore the efficiency under Barcode is quite low. In RFID, information stored in tags are different among each objects. Using the same above example, each bottle of Diet Coke has its own tag information. Tags could be embedded inside the product instead of exposed on the outside of the product [12]. RFID does not require a direct LOS. Normally, tags can be read by readers up to hundreds of meters away. Barcodes are used once only, while RFID tags can be reused and they are RW (read-write) devices. Under RFID, hundreds of tags could be read in several seconds. As a result, the efficiency is greatly increased.

Undoubtably, RFID can have enormous market value, e.g., only Wal-Mart alone can save $8.6 billions a year [11]. Today, many companies are switching to RFID from Barcode because of these advantages that RFID technology have compared to Barcode.

## 2.4 RFID System Components

An RFID system consists of three main conponents: a tag (transponder), a reader (transceiver), and middleware [13].

As we already introduced in the last section, tags store information about objects. This information is queried by a transceiver when a tag is in a reader's range. Communication between a tag and a reader occurs through radio waves.

RFID readers are devices working in between middleware and tags. Its responsibility is to pass the information from a tag to middleware [13]. They also use antennas to communicate with tags. There are a variety of different readers: handheld readers (like the barcode scanner), "mobile" readers (embedded into mobile data collection devices), and fixed readers (like the readers fixed above a conveyor belt). They are used in different environments depending on the type of application.

Middleware sits between RFID systems and enterprises applications. It is responsible for managing the flow of data from readers and transmitting the data to back-end management systems. The other task are listed as follows: filtering data feeds to application software, generating inventory movement notifications, monitoring reader and tag network performance, capturing history, and analyzing tag reading events for application improvement [13].

## 2.5 How Does RFID Work?

RFID belongs to a group of technologies named as Automatic Identification and Data Capture (AIDC) [14]. AIDC functions automatically identify a object, collect data from them, and send these data upwards directly to computer systems that are provided with related database and softwares. An RFID system often works as in the following several steps below [14], and Figure 2.2 gives a diagram to illustrate the scenario.

- The RFID reader transmits a radio signal through its antenna (a reader may have several antennas)
- The radio wave emitted by a reader actives the RFID tag (mostly passive tags)
- Once the tag and reader authenticate each other, the tag sends its information (e.g., EPC) to the reader
- Finally, the reader transmits data to the database for processing (the database contains the object information associated with the serial number on the tag)

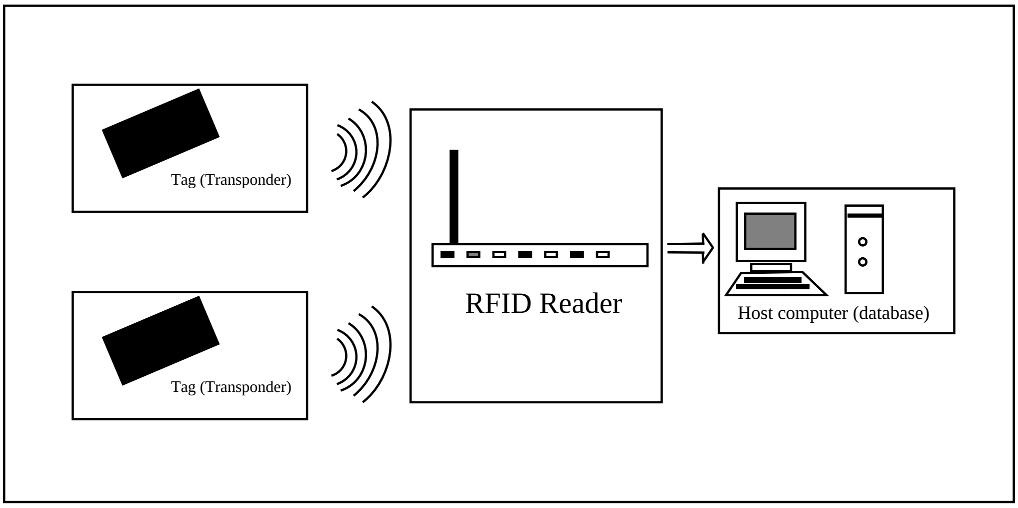


Figure 2.2: How RFID works

An RFID communication protocol has three layers: application layer (identification protocol), communication layer (medium access protocol), and physical layer (air interface, e.g., frequency, modulation) [5]. Figure 2.3 shows the RFID model protocol stack compared with that of OSI and TCP/IP.

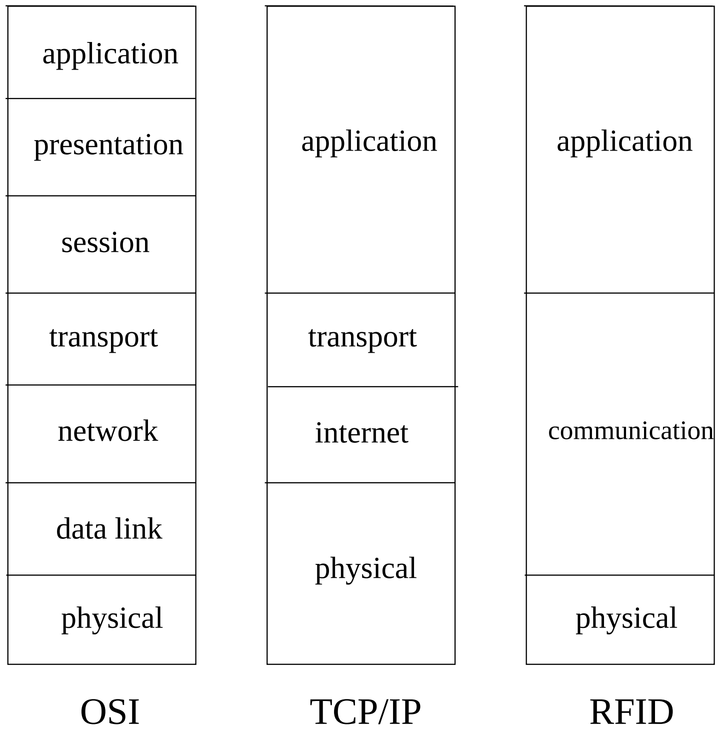


Figure 2.3: RFID model protocol [5]

## 2.6 Security And Privacy

The issues of privacy and security are interrelated, and also different with each other. Privacy is the ability of an RFID system to keep the **meaning** of the information transmitted from non-intended recipients [15]. Security is the ability of an RFID system to keep the information transmitted from non-intended recipients [15]. There are some examples here related with the issue of privacy: consumers can be tracked by the products they buy; travelers can be tracked by the passport they hold; readers can be tracked through the books they have checked out [15]. Security issues occur due to the fact that the readers and tags communicated with each other through open and unencrypted messages.

Even today, security and privacy are still very serious issues. For example, on a very wonderful day you shop at the Woodland Hills Mall in Tulsa, where there is some person named Allen who has a reader in hand. Allen is curious about what you have in your handbag or pocket. So he uses the reader to scan all the objects you have. Thus Allen might know very serious information about you, e.g., how much cash you have, your credit card information, your company information (if you take your badge with you by chance), etc. In this situation, the safety of yourself and your assets are both jeopardized by some bad guy. This above situation could occur because of several reasons [15]: (i) RFID tags can be read through materials, package, or items; as a result, consumers can never be sure about where the tag is hidden or when the tags are being scanned; (ii) RFID tags can be read at a small distance with no obvious action needed (you cannot see they are scanning your objects); (iii) Tags can be potentially active outside of the store; (iv) Data stored on tags is known to several different entities; when the information been transmitted, security problem occurs; (v) The smallest and cheapest passive tags do not have enough computing power to do data encryption, which also leads to privacy problems.

## 2.7 Applications

Although RFID technology had been used since World War II, the usage of RFID system is increasing rapidly every year. So far, RFID technology is deployed in many industries in inventory management, supply chain management, asset tracking, counterfeit prevention [14] (e.g., recognizing fake money in currency flow), security (e.g., controlling access to restricted areas), tracking persons, retail automation, vehicle theft protection, livestock identification, road tolling, ID badge, car parking access, public transportation system, logistics and distribution (e.g., tracking parcels from shipment to recipient), maintenance (e.g. monitoring patients) [8], and so forth.

# CHAPTER 3

# RFID COLLISIONS

## 3.1 How Collisions Arise?

In an RFID system, collisions arise when radio waves from one device interfere with radio waves from another device. There are three different types of collision: tag-tag collision, reader-reader collision, and reader-tag collision [6]. We will explain these three collisions respectively.

Tag-tag collision happens when multiple tags that are in the same interrogation region (the region surrounding a reader a tag can be successfully read without any collisions) of a reader respond to a reader's query simultaneously [6]. When there are a large number of tags in the interrogation region, it is prone to have this problem. Figure 3.1 illustrates this collision as an example. To schedule tags' response in a collision-free manner, we can use a framed Aloha protocol or tree-splitting protocol, which we will describe in the next section.

A reader-reader collision happens if two neighboring readers (with overlapping interrogation regions) operate simultaneously. So if a tag passes through the common interrogation region, the tag cannot differentiate between the two signals from two readers. Therefore, the tag cannot respond to any of the readers [6]. As an example, Figure 3.2 illustrates this collision. It is not applicable to use different channels with the interference readers due to the fact that tags cannot function in the right way if they are not designed to have the ability to work in different frequencies. Most tags cost 5 cents or so, and definitely they are not that "intelligent". One way to eliminate reader-reader collision is utilizing a TDMA (time division multiple

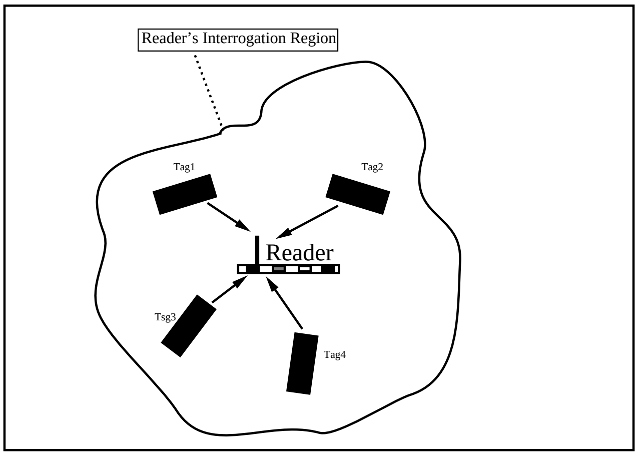


Figure 3.1: Tag-tag collision (adapted from [6])

access) mechanism (schedule conflicting readers to operate in different time slots). Another way is decreasing the power level of each reader to remove the overlapping interrogation region. In this thesis, we make use of the second method.

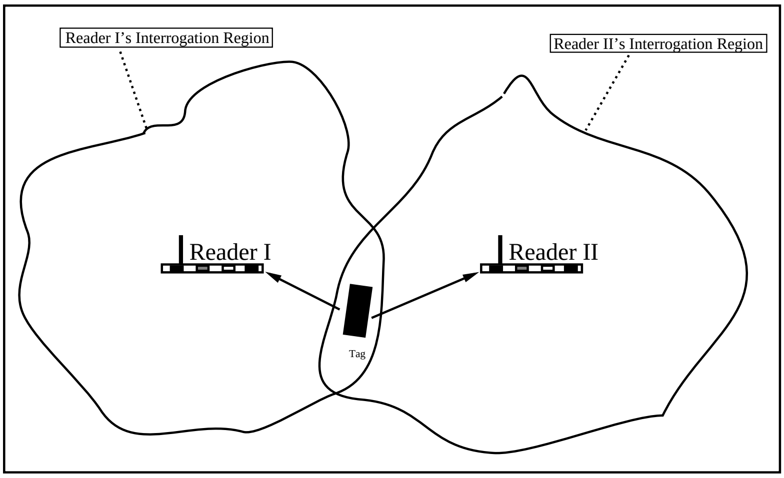


Figure 3.2: Reader-reader collision (adapted from [6])

Reader-tag collision happens when the signal from a nearby reader interferes with a tag response being received at another reader [6]. As a result, if a reader A is in the interference region of another reader B, the signal sent from tags to reader B could be distorted by reader A. Figure 3.3 illustrates this type of collision as an example.

Tag-reader collision can be removed by using frequency hopping (a technique of transmitting signals by rapidly switching a carrier chosen from many frequency channels, which could add frequency diversity [16]) in the UHF (ultra high frequency) band or deploy a TDMA mechanism as described in the last section.

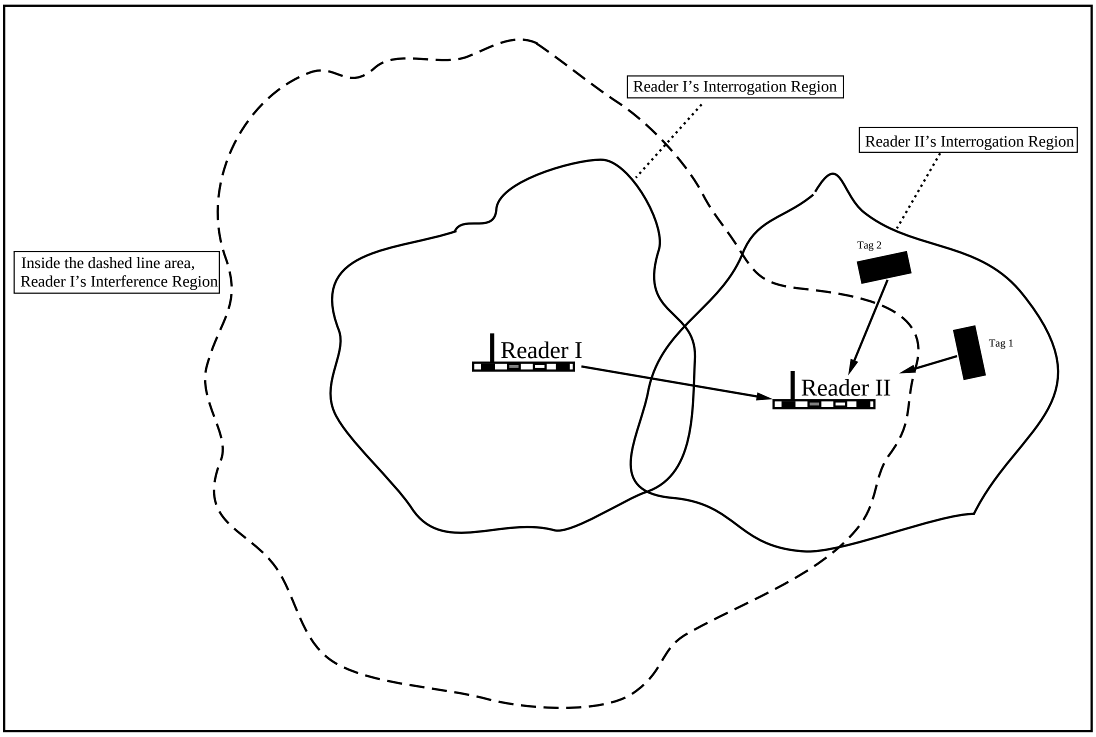


Figure 3.3: Reader-tag collision (adapted from [6])

### 3.2 Anti-tag-collision Algorithms

There are two basic types of anti-tag-collision protocols. They are probabilistic (e.g., slotted Aloha protocols, standardized for Class 1 Generation 1 and Class 1 Generation 2 RFID systems) and deterministic protocols (e.g., binary tree search based protocols, standardized for Class 0 Generation 1 RFID systems) [11]. In slotted Aloha protocols, a frame is divided into many time slots and every tag chooses a random time slot to transmit its information. The frame size (number of total time slots) is decided by the reader according to the population of tags, specifically the collisions detected in the previous query process. Binary tree search based protocols search for tag identifications that match a specific binary number [17]. In this protocol, the reader organizes the entire ID space of tags into a binary tree with each tag ID matched with a leaf. A reader traverses the tree in a depth-first order. At each node, the reader broadcasts a query message with a string that corresponds to the tag tree node. When a tag finds a message that matches its own ID prefix, it will respond to the reader. If multiple tags respond, collisions happen. Then the colliding tags are splitted by a randomly selected number 0 or 1 (every tag maintains a counter and a random number generator). The tags selecting 0 transmit immediately and tags selecting 1 will transmit later. The reader traverses down the tree in this way until all the tags are recognized [17]. Figure 3.4 illustrates the binary tree search protocol.

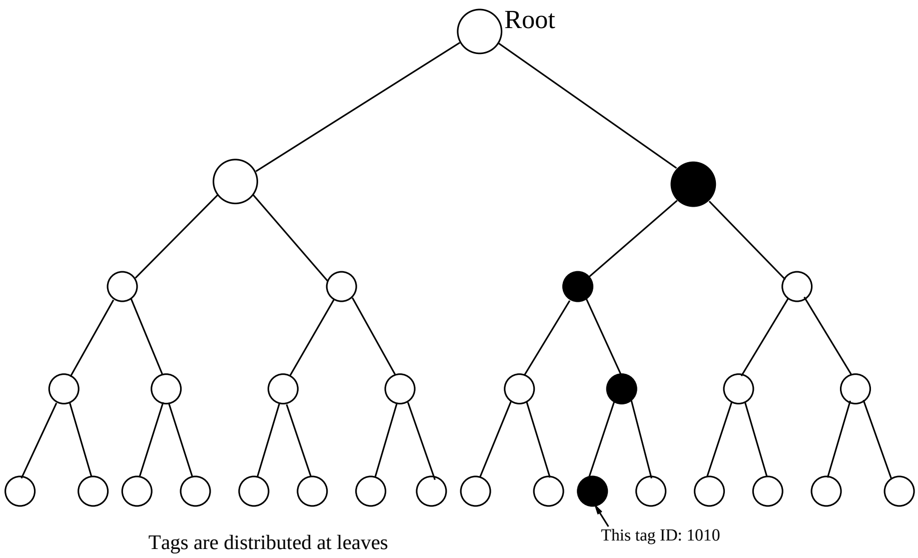


Figure 3.4: Binary tree search based protocol

We can see that deterministic protocols have relatively long identification delay, while probabilistic protocols are more efficient. However, probabilistic protocols cannot guarantee that each tag can be read (starvation problem), but this never happens in deterministic protocols.

In the following sections, we will introduce several slotted Aloha protocols in more detail. They are the Basic Framed Slotted Aloha (BFSA) Algorithm, the Dynamic Framed Slotted Aloha (DFSA) Algorithm, the Enhanced Dynamic Framed Slotted Aloha (EDFSA) Algorithm, and the Accelerated Framed Slotted ALOHA (AFSA) Algorithm.

## 3.3 Frame Slotted ALOHA Anti-collision Algorithms

### 3.3.1 Basic Framed Slotted ALOHA (BFSA) Algorithm

Basic Framed Slotted Aloha is the simplest protocol of all probabilistic protocols. It uses a same constant value for frame size during all of the identification process. In each round, the reader provides the tags with information about the frame size. A tag randomly picks one of the *frame_size* slots and transmit its ID (EPC). After that, the reader provides (send out) a bitmap as acknowledgement [18]. As a result, the efficiency of the system is fairly low when the tag population is too large or small. When there is a large population of tags, too many collisions occur and in the other case, many slots are unused (wasted) and the identification takes unnecessarily long. Figure 3.5 illustrates an example of using a BFSA protocol. In Figure 3.5, the frame size is 3, so in each round there are three time slots available. In the first round, Tag 4 successfully transmits its ID to a reader after the reader initiates the query process. But there are two collisions happen, (Tag 1, Tag 3) and (Tag 2, Tag 5). In round 2, the ID of tag 1 and tag 5 are forwarded to the reader, but Tag 2 and Tag 3 collide. In the last round, finally the reader gets the information from Tag 2 and Tag 3. In this example, the reader needs to take 3 rounds to complete the tag reading process.

### 3.3.2 Dynamic Framed Slotted ALOHA (DFSA) Algorithm

The efficiency of BFSA drops significantly when there are large or small numbers of tags. Jae-Ryong Cha and Jae-Hyun Kim proposed the Dynamic Framed Slotted

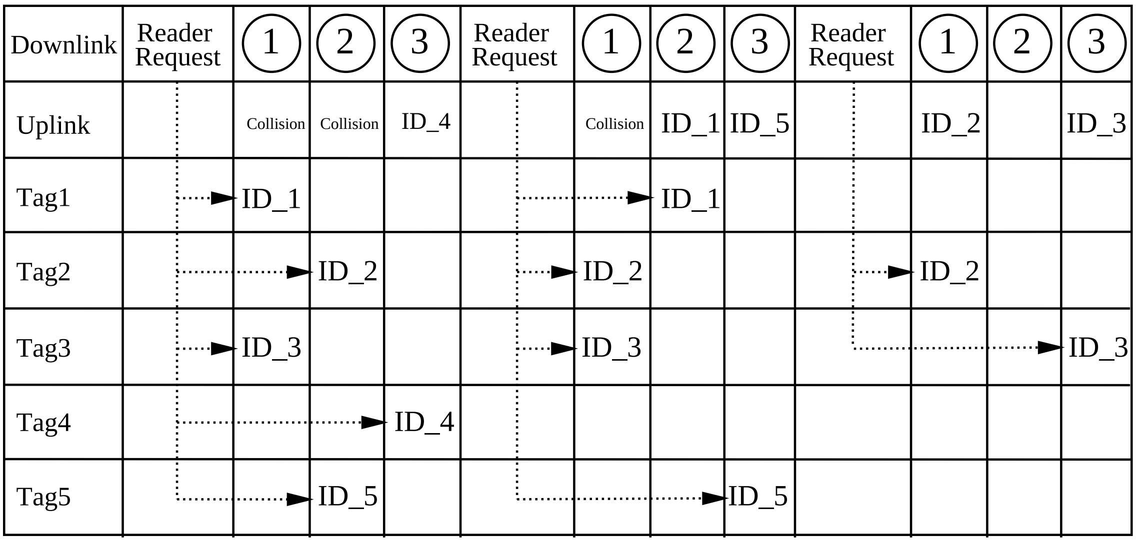


Figure 3.5: Basic Framed Slotted Aloha

Aloha algorithm (DFSA) using a tag estimation method (TEM) to estimate the population of tags near the reader [19]. To determine the number of tags around the reader, it uses the information obtained from the last round, e.g. the number of slots and the number of collisions. They use this method to dynamically assign the frame size based on the estimated number of tags. As a result, DFSA can partially increase the efficiency over BFSA. DFSA has several versions according to their different ways to find out the frame size.

Here are two main versions of the algorithm. The first one estimates the frame size by the information gotten from the last round, such as the number of idle slots (vacant slots), collisions slots, and successful slots (taken up by only one tag) [19]. In the beginning, the reader starts a read cycle (round) with the minimum frame size. If there is a small number of tags, then it works out perfectly. If the number of collision is greater than the upper bound (some threshold), the reader increases the frame size; else if the number of collision is less than some lower bound, the reader reduces the frame size. In the other algorithm, the reader starts with an initial frame size to be 2 or 4. It increases the frame size exponentially and starts a new read cycle if no tag is identified successfully in the preceding read cycle [19]. Continue with this process until at least one tag is identified. The reader stops the current read cycle and skips into the next read cycle with minimum frame size when a tag is identified successfully. In this algorithm, the reader always starts with the minimum frame size to identify the tags, no matter the number of unread tags. Therefore, compared with the last algorithm, this algorithm has a different way to change the frame size.

### 3.3.3 Enhanced Dynamic Framed Slotted ALOHA (EDFSA) Algorithm

In DFSA, the reader can increase the frame size infinitely when there is a large number of tags to be read. This will definitely delay the identification process, thus the number of unread tags is too large to achieve good system performance. Su-Ryun Lee, Sung-Don Joo, and Chae-Woo Lee proposed the Enhanced Dynamic Framed Slotted Aloha Algorithm (EDFSA) for RFID identification [2]. In this algorithm, to prevent the frame size from increasing infinitely when the number of tags is too large (great than some given maximum frame size), it separates the unread tags into several groups and allows tags in only one group to respond to the reader. So it restricts the number of participating tags in each round when the population of tags is quite large, to achieve the optimal number of tags responding the reader with the given frame size [2]. When the number of unread tags is too small, we reduce the frame size to achieve optimal system efficiency (Note: system efficiency is the ratio of successful slots to the current frame size). To achieve the best system performance, there is always an optimal frame size corresponding to some number of unread tags with a given maximum frame size.

In EDFSA, the readers start by estimating the number of unread tags in every read cycle, and then calculate the number of groups that lead to the optimal throughput. Also the number of groups depends on the maximum frame size and the number of unread tags. Assume N denotes the maximum frame size and K stands for the number of unread tags. Then the number of groups M is calculated by the formula: $M = \lceil \frac{K}{N} \rceil$ groups [2]. Table 3.1 gives the number of groups with different numbers of unread tags when the maximum frame size is 256. Then the reader broadcasts a message consisting of the number of tag groups and a random number to tags. After a tag receives the above information, it generates a new number from its ID and received random number, and divides this new generated number by the number of tag groups. Then just the tags with zero remainder respond to the reader. Repeat the above process when the reader performs the reading task [2].

Table 3.1: Number of unread tags vs. optimal frame size and Number of Groups [2]

| Number of unread tags | Frame Size | M |
|---|---|---|
| ⋮ | ⋮ | ⋮ |
| 1417–2831 | 256 | 8 |
| 708–1416 | 256 | 4 |
| 355–707 | 256 | 2 |
| 177–354 | 256 | 1 |
| 82–176 | 128 | 1 |
| 41–81 | 64 | 1 |
| 20–40 | 32 | 1 |
| 12–19 | 16 | 1 |
| 6–11 | 8 | 1 |
| ⋮ | ⋮ | ⋮ |

### 3.3.4 Accelerated Framed Slotted ALOHA (AFSA) Algorithm

In EDFSA, the probability of a slot being successful is maintained approximately euqual to the maximum possible value of 36.8%. Also there is a proportion of time wasted in transmission phase (data from tag to reader). Dr. V. Sarangan and M.R. Devarapalli proposed a novel algorithm named Accelerated Framed Slotted Aloha (AFSA) that reduces the tag reading process by using bitmaps and avoids wasted time due to collisions and idle slots [7].

In the reading process, each successfully identified tag goes through six different states [7]. All the tags begins in an *unpowered* (*passive*) state. The reader then sends the reset, oscillator calibration, and data sysmbol calibration signals to see which tags go to the *active* state [7]. If some tags lose of synchronization, they stay in the *unpowered* state. The reader then broadcasts the frame size and the number of groups to tags. All the tags in a selected group (tags have zero remainder) move to the *select* state while the others stay in *active* state. The tags in the *selected* state then try to reserve a slot for data transmission. If this process is successful, they switch to the *transmit* state; otherwise they just go back to the *active* state. Tags in the *transmit* state then transmit their ID (EPC) to the reader, after which they go to *Ack_Wait* state. Then they go to the *identified* state if the transmission is successful; otherwise, it goes back to the *active* state [7]. Figure 3.6 shows the tag state machine.

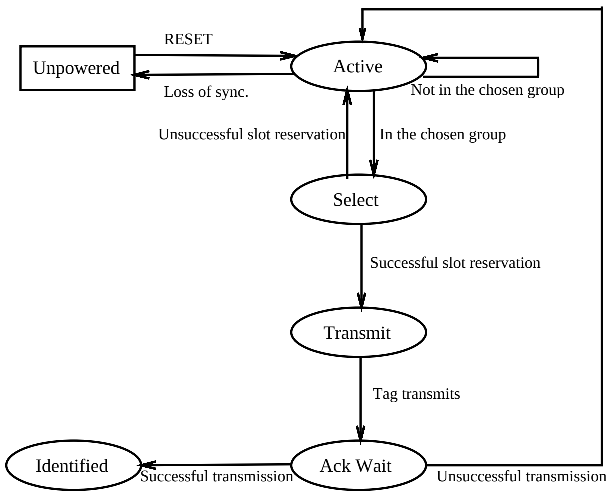


Figure 3.6: AFSA: tag state machine [7]

In addition to the three phases already contained in other slotted Aloha algorithms, AFSA has two more phases to complete a single cycle of tag reading. The five phases are advertisement phase, reservation phase, reservation summary phase, data transmission phase, and acknowledgement phase [7]. In the advertisement phase, the reader broadcasts the frame size N, the number of groups M, and a given value of n (the length of the bit sequence that tags transmit in the reservation phase; from [7], 2 gives the best performance when the number of tags is 256) to all tags within its interrogation region. In the Reservation phase, the tags in the select state transmit an n-bit sequence in their chosen slot (a tag randomly picks 1 of $2^n$ sequences and transmits the bit sequence). If the reader successfully receives the bit sequence in a slot, it just reserves that slot for the tag transmitting its data later. Otherwise, the reader treats the slot as a collision slot if it gets a garbled signal. In the reservation summary phase, the reader gives the tags feedback regarding the reservation phase by broadcasting a bitmap of length N [7]. For example, if N=8, bitmap 11001011 means that the n bit sequence was successfully received in slots 1, 2, 5, 7, and 8. Slots 3, 4, and 6 have one of two possibilities: collisions (more than one tag has chosen the same slot) or idle (the slot has not been chosen). In the data transmission phase, only tags that have successfully gotten a slot reserved can transmit their data. So in the last example, only tags that reserved one of the five slots (1, 2, 5, 7, and 8) transmit, and they transmit their data in the same order as shown of the five slots. Therefore the tag that reserved the first slot will transmit first, and the tag that reserved the fifth slot will transmit third. In the last acknowledge phase (final phase), the reader acknowledges the transmissions through a sequence of 0 and 1 bits to the tags (0 means successful, 1 means fail). Then the tags that get a "1" response become muted, and the ones with a "0" response go back to the *active* state and go through the process again.

We can see that AFSA tries to decrease the collision waste (96 bits per transmission) by transmitting fewer bits (about 2 or 3 bits) in the reservation phase to reserve a slot. If a slot successfully gets reserved by some tag, then the tag will transmit all of its data to the reader. In this way, it will greatly minimize the collision bit wastage in the slotted Aloha algorithms introduced above.

## 3.4 Anti-reader-collision Algorithms

### 3.4.1 Listen Before Talk

"Listen Before Talk" is a CSMA (carrier sense multiple access) based protocol. The reader must listen for the presence of any other signal within its intended sub-band of transmission for a fixed period time $\tau$ plus a random time of 0 to $\tau$ in eleven steps [3]. Table 3.2 shows the thresholds to determine the presence of another signal within the intended sub-band. After s sub-band has been chosen, the reader is permitted to use that sub-band time for up to some amount of time.

Table 3.2: Transmit and threshold power [3]

| ERP (W) | ERP (dBW) | Threshold (dBW) |
|---|---|---|
| Up to 0.1 | Up to -10 | less than -113 |
| 0.1 to 0.5 | -10 to -3 | less than -120 |
| 0.5 to 2.0 | -3 to 3 | less than -126 |
| Note: ERP(effective radiated power) is a standardized theoretical measurement of radio frequency energy. | | |

### 3.4.2 Colorwave

Colorwave (proposed by James Waldrop, Daniel W. Engles, and Sanjay E. Sarma) is a distributed online TDMA-based algorithm [20]. In this algorithm, each reader chooses a random time slot (color) for transmission. If a collision happens, then the reader selects a new timeslot and sends a kick to all of its neighbors telling them the selection of new timeslot. If any of the neighbors use the same slot again (choose the same color), the reader chooses a new color and repeats the process described above. In Colorwave, each reader monitors the percentage of successful transmissions. Colorwave also dynamically changes the maximum number of colors available at a reader according to the percentage of successful transmissions. Obviously, Colorwave requires time synchronization between all of the readers, and also the readers need to detect the collisions when collisions arise. More details can be found in [20].

### 3.4.3 HiQ

HiQ (Hierarchical Q-Learning, proposed by Junius Ho, Daniel W. Engles, and Sanjay E. Sarma) is a hierarchical online learning algorithm giving dynamic solutions for reader-collision problem [21]. It maximizes the number of simultaneously communicating readers, and also minimizes the number of reader collisions at the same time through learning the collision patterns between readers and effectively assigning frequencies among readers. HiQ uses three basic hierarchical tiers in its control structure. The three tiers are readers, R-servers, and Q-servers. R-servers assign the frequencies and time slots (communication resources) to readers for communication. Readers normally need to require the above communication resources before they start to communicate, and they are required to be able to detect the collisions with adjacent readers (but they are not expected to know which readers). Then readers report the number and types of collisions to their corresponding R-servers. An R-server is supposed to determine which readers are interfering with other readers, as reported by readers through the interference patterns. Then an R-server assigns the communication resources allocated by its master Q-learning server. Q-servers are the top level tier, and the smartest devices in the system. They might have a hierarchy too, in order to maintain flexibility and scalability. In this case, a single root Q-server is sitting there with global knowledge of all available communication resources. It uses a dynamic Q-learning based algorithm to allocate the communication resource to its children Q-servers and R-servers from the constraints learned in the system. More details can be found in [21].

# CHAPTER 4

# PROPOSED METHOD AND PERFORMANCE ANALYSIS

## 4.1 How Does Power Affect A Reader's Read Diameter?

A reader's read diameter means the reader's read distance, which is the maximum distance a reader's signal can reach to read a tag. For example, the reading distance of a UHF RFID Reader (860–960MHz programmable, ISO 18000–6B) is 3 to 12 meters.

The power lever of a reader affects the reader's read diameter. Reader diameter is a function of a reader's power: $\boldsymbol{d = f(P)}$, where $d$ denotes the reader's read diameter, $P$ stands for the reader's power, and $f$ is monotonically increasing function. If the power level increases, the reader's read diameter increases, and vice versa. For simplicity, we let $d = \alpha \cdot P$ in our model, where $\alpha$ is a constant. For example, if $P' = 70\% \cdot P_0$, the read diameter under $P'$ is $0.7 \cdot d$ (d is the read diameter under $P_0$).

In the next following sections, we will introduction two topologies for multiple readers. The first one is readers with a constant spacing. The other one is readers with a uniform random distribution.

## 4.2 Readers With Constant Spacing

In this section, let us consider the situation where the distances between neighbor readers are the same, say the distance is D. We call this case "Readers with constant spacing". In the dense reader environment we know that the interference among readers greatly affects the performance of the tag-reading system. The same is true in the supply chain system, in particularly in the convey belt system. To reduce the interference between nearby readers, the factor we consider is the power of each reader. Specifically, we try to control or minimize the overlap region (interference zone) by adjusting the power of reader. In order to give you some idea, let us look at Figure 4.1 below. From Figure 4.1, each parameter (we will continue to use those names in later sections) is listed as follows, the velocity of the conveyor belt - v (meters per second); the height of each reader - h (meters); the distance between any neighbor readers - D (meters); the reading diameter of each reader (the maximum distance reader can read, under the maximum power) - L. The shaded (overlapped) area is the interference region between neighboring readers. Assume there are N readers in all. From left to right, the readers are numbered reader 1, 2, 3, ..., N.

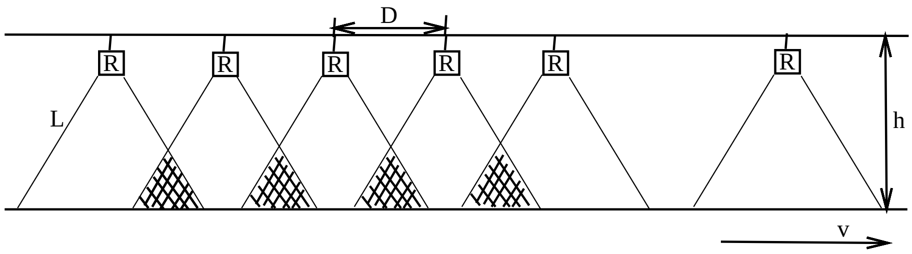


Figure 4.1: Each reader works under its maximum power

Figure 4.1 shows us the scenario when each reader works under maximum power (100% of original power). Now let us consider five scenarios when readers work under different level of power in the rest of this section.

**(I) Scenario 1**: Figure 4.1 is the right figure for this scenario, in which each reader works under maximum power. In this scenario, we know that the total time each tag spent in non-interference region is the sum of the times each tag spent under each reader. For example, the time a tag spent under Reader 1 is calculated by subtracting the overlapping area time from the overall time a tag spends under Reader 1. Specifically, the total time a tag spends under the non-interference region is:

$$t_1 = \sum_{i=1}^{N} t_0 - 2(N-1)t_{overlap} \tag{4.1}$$

and

$$t_0 = \frac{2\sqrt{L^2 - h^2}}{v}. \tag{4.2}$$

$t_0$ is the time each tag spends under each reader, and it is a constant for any specific topology.

In Equation (1), in order to calculate what $t_1$ is, we need to compute what $t_{overlap}$ is. In order to solve this problem, we take a snippet from Figure 4.1. Only two neighbor readers have overlapping area, as shown in Figure 4.2. In Figure 4.2, point A stands for one reader, point B stands for the other reader that is closest to point A reader on the right side. CE is the overlapping region between point A reader and point B reader. F is the midpoint of line AB and G is the midpoint of CE.

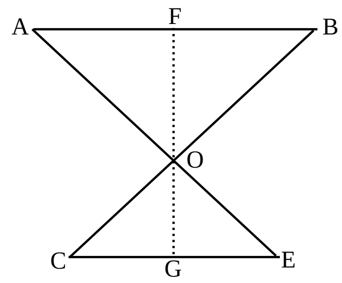


Figure 4.2: Overlapping region between adjacent readers in constant spacing

In Figure 4.2, we know that $|AB| = D$, which is the distance between any two neighbor readers. Also $|CE| = vt_{overlap}$, $|BC| = |AE| = L$ (the readers are working under their maximum power), and line FG is perpendicular to line AB and line CE. Obviously, $\Delta AOB$ and $\Delta COE$ are similar.

$$\begin{cases} \frac{|OG|}{|OF|} = \frac{|CE|}{|AB|} \\ \frac{|OC|}{|OB|} = \frac{|CE|}{|AB|} \\ |OG|^2 + |CG|^2 = |OC|^2 \end{cases} \tag{4.3}$$

Let $|OG| = m$, $|OC| = n$, also we already know that $|CE| = vt_{overlap}$, the above Simultaneous Equations 4.3 become

$$\begin{cases} \frac{m}{h-m} = \frac{vt_{overlap}}{D} \\ \frac{n}{L-n} = \frac{vt_{overlap}}{D} \\ m^2 + (\frac{1}{2}vt_{overlap})^2 = n^2 \end{cases} \tag{4.4}$$

After solving Simultaneous Equations 4.4, we get the result: $t_{overlap} = \frac{(2\sqrt{L^2-h^2})-D}{v}$, which we will frequently use in later sections.

**(II) Scenario 2**: In scenario 1, we let each reader work under its maximum power. In this scenario, we reduce the power of each reader. Therefore the overlapping region between neighboring readers decreases, and so does $t_{overlap}$. In this scenario, the power of each reader is changed to $x \cdot \{maximum\_power\}$. Figure 4.3 illustrates the scenario in which each reader's power has been reduced to $x \cdot \{maximum\_power\}$.

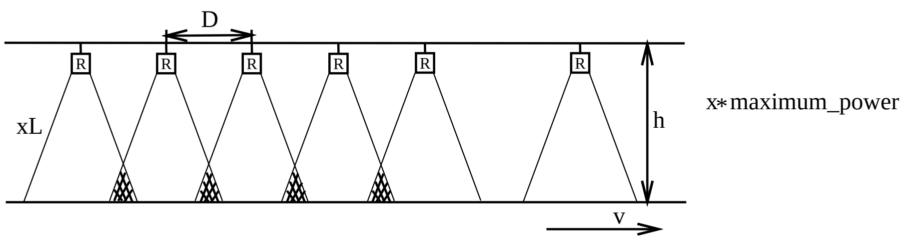


Figure 4.3: Reduce the power of each reader to $x \cdot \{maximum\_power\}$

In this scenario, the reader's reading distance becomes $x$·L. Therefore the total time each tag spends in the non-interference region is:

$$t_2 = \sum_{i=1}^{N} t'_0 - 2(N-1)t'_{overlap} \tag{4.5}$$

and

$$t'_0 = \frac{2\sqrt{x^2L^2 - h^2}}{v}. \tag{4.6}$$

$t_0$ is the time each tag spends under each reader. From the result we get from Scenario 1, we know that $t'_{overlap} = \frac{(2\sqrt{x^2L^2-h^2})-D}{v}$.

**(III) Scenario 3**: From Scenario 2, we further reduce the power of each reader, until the point when $t_{overlap} = 0$. This is illustrated in Figure 4.4.

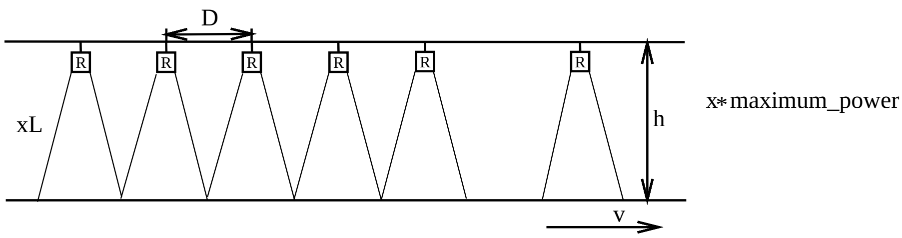


Figure 4.4: Reduce the power of each reader until $t_{overlap} = 0$

In this scenario, after reducing the power of each reader, the reading distance of each power is changed to $x$·L ($0 < x < 1$). We can also get the total time a tag spends in the non-interference region:

$$t_3 = \sum_{i=1}^{N} t'' \tag{4.7}$$

and

$$t'' = \frac{2\sqrt{x^2L^2 - h^2}}{v} = \frac{D}{v} \tag{4.8}$$

Now let us solve for x, in which case $x = f(D, L, h)$. By the Pythagorean theorem, this following equation holds: $(\frac{1}{2}D)^2 + h^2 = (xL)^2$. Thus we get $x = \frac{\sqrt{D^2+4h^2}}{2L}$, which also could be deduced by letting $t'_{overlap} = 0$ in Equation 4.6.

**(IV) Scenario 4**: From Scenario 3, let us further reduce the power of the readers. In this scenario, there will be some gaps between those pairs of neighboring readers.

Those gaps are not covered by radio waves from any of the readers. This is illustrated in Figure 4.5.

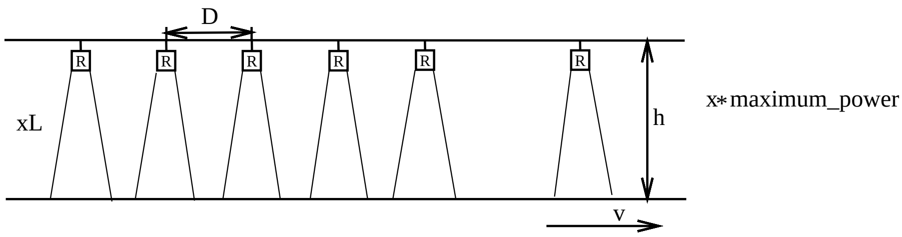


Figure 4.5: Gaps between neighboring readers

In this scenario, the total time a tag spends in the non-interference region is: $t_4 = \sum_{i=1}^{N} t'''$, and $t''' = \frac{2\sqrt{x^2L^2-h^2}}{v}$.

**(V) Scenario 5**: The last scenario is when each reader's reading distance is only h. Obviously, the total time a tag passes the radio wave zone is very limited, and is approximately zero. This is illustrated in Figure 4.6.

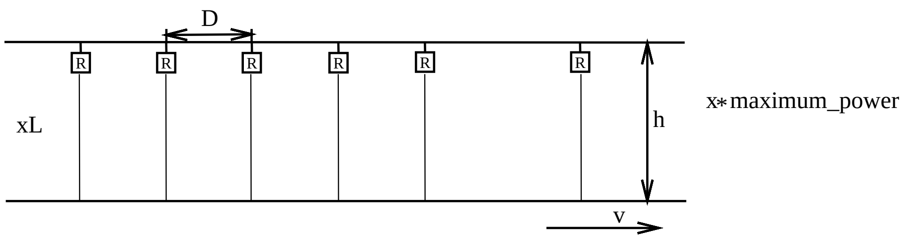


Figure 4.6: Extremely limited radio wave reach tags

## 4.3 Verification Regarding Readers With Constant Spacing Scenarios

In the last section we introduced five scenarios, each of which has different performance. The total time each tag spends in the non-interference region varies according to different scenarios, in which case all the readers work in the different level of power among these scenarios.

Intuitively, we think the system has optimum performance when $t_{overlap}$=0; that is to say the total time each tag spend under non-interference area is maximized. In that case tags are having a higher chance to be read by readers. Now let us prove this statement. In order to explain this problem clearly, we use Figure 4.7 that consists of a group of sub-figures.

Figure 4.7(a) through (e) show us the process when we gradually reduce the power of each reader. In Figure 4.7(a), power of each reader equals each reader's maximum power. In Figure 4.7(e), the power of each reader is the least amount of power among the five sub-figures. We know that $power_{fig(a)} > power_{fig(b)} > power_{fig(c)} > power_{fig(d)} > power_{fig(e)}$, and also $rd_{fig(a)} > rd_{fig(b)} > rd_{fig(c)} > rd_{fig(d)} > rd_{fig(e)}$. Power stands for reader's power under each scenario; rd stands for "reading distance" for each reader. For simplicity, we let each reader's reading distance be $x$·L ($0 < x \leqslant 1$). Obviously, in Figure 4.7(a), $x = 1$. From the last section, we solved that in Figure 4.7(b), $x = \frac{\sqrt{b^2+4h^2}}{2L}$. In Figure 4.7(c), x is less than the x of Figure 4.7(b). In Figure 4.7(d), $x = \frac{h}{L}$. In Figure 4.7(e), $x < \frac{h}{L}$.

Our goal is to prove when $x = \frac{\sqrt{b^2+4h^2}}{2L}$ ($t_{overlap=0}$), the system has the optimum performance. The proof is shown below. The variable t below is the total time a tag spends in the non-interference area. Also remember the range of x is $(0, 1]$. We discuss the following cases,

(i) When $x \in (0, \frac{h}{L})$, the reading distance of each reader is so small that tags cannot be reached by radio waves. Thus in this case, t = 0.

(ii) When $x = \frac{h}{L}$, the reading distance of each reader is h. In this case, tags can only be reached by radio waves in a discrete way (very limited radio wave can reach tags). So $t \rightarrow 0$.

(iii) When $x \in (\frac{h}{L}, \frac{\sqrt{D^2+4h^2}}{2L})$, there are some gaps between neighboring readers. Also these gaps cannot be reached by radio waves from any other reader. From the last section, we already know that

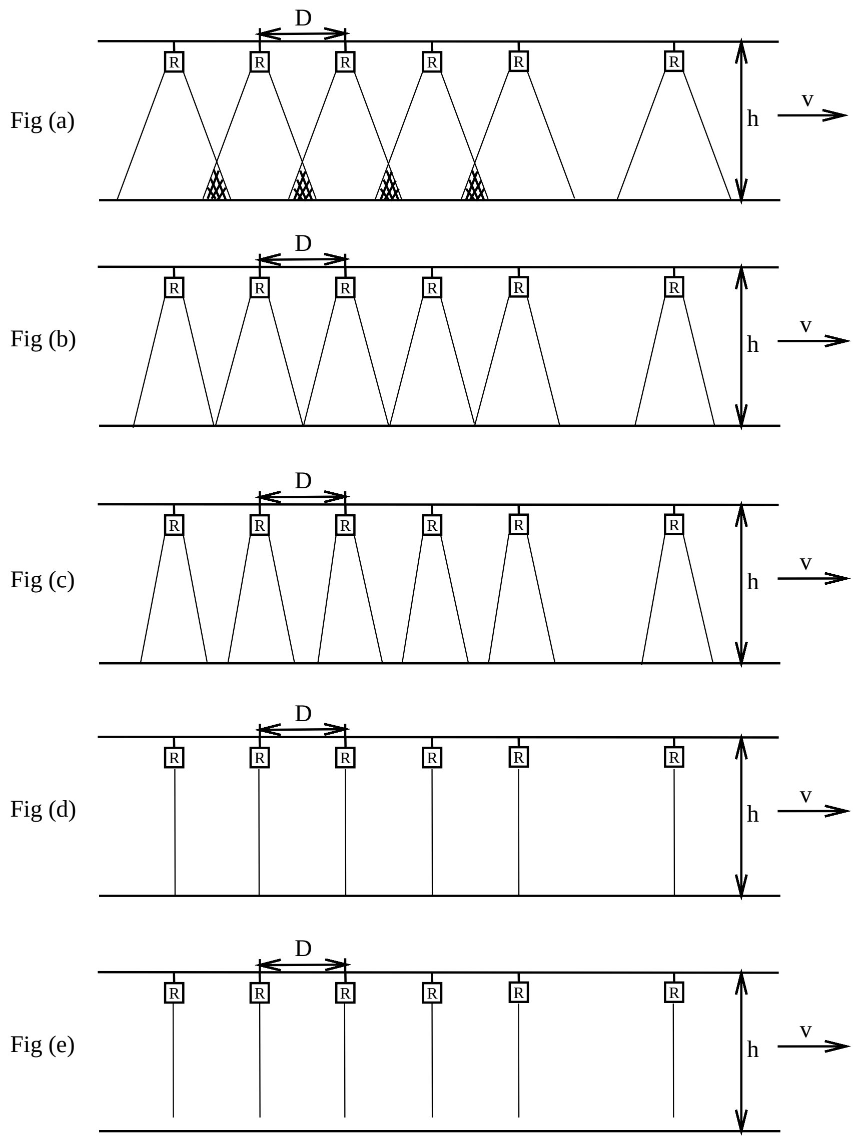


Figure 4.7: Reduce power of readers gradually

$$t = \sum_{i=1}^{N} t''' = \sum_{i=1}^{N} \frac{2\sqrt{x^2L^2 - h^2}}{v} = \frac{2N\sqrt{x^2L^2 - h^2}}{v}, \tag{4.9}$$

and

$$x \in (\frac{h}{L}, \frac{\sqrt{D^2 + 4h^2}}{2L}). \tag{4.10}$$

The above function t=f(x) is monotonically increasing. When x increases, t also increases. Therefore, if x is within this range $(\frac{h}{L}, \frac{\sqrt{D^2+4h^2}}{2L})$, $t_{max} \to f(\frac{\sqrt{D^2+4h^2}}{2L}) = \frac{DN}{v}$.

(iv) When $x \in [\frac{\sqrt{D^2+4h^2}}{2L}, 1]$, either there is some overlapping area (the interference region) between any pair of neighbor readers or the overlapping area is zero. From the last section, we know that under this case, the following equation applies.

$$t = \sum_{i=1}^{N} t' - 2(N-1)t'_{overlap} \tag{4.11}$$

$$t' = \frac{2\sqrt{x^2L^2 - h^2}}{v} \tag{4.12}$$

and

$$t'_{overlap} = \frac{(2\sqrt{x^2L^2 - h^2}) - D}{v}. \tag{4.13}$$

Then we get the result:

$$t = \frac{2N\sqrt{x^2L^2 - h^2}}{v} - 2(N-1)\frac{(2\sqrt{x^2L^2 - h^2}) - D}{v} = \frac{2D(N-1) - (2N-4)\sqrt{x^2L^2 - h^2}}{v}, \tag{4.14}$$

and

$$x \in (\frac{\sqrt{D^2 + 4h^2}}{2L}, 1]. \tag{4.15}$$

Normally, the system uses more than two readers ($N > 2$). Obviously the above function t=f(x) is monotonically decreasing. When x decreases, t increases adversely. As a result, if x is within the range $(\frac{\sqrt{D^2+4h^2}}{2L}, 1]$, $t_{max} = f(\frac{\sqrt{D^2+4h^2}}{2L}) = \frac{DN}{v}$.

From the above discussion, we can conclude that the system has the best performance (the total time each tag spends in the non-interference area is maximized) when $x = \frac{\sqrt{D^2+4h^2}}{2L}$.

## 4.4 Readers With Uniform Random Distribution

We already analyzed the constant spacing scenario in the above sections. In this section, let us consider the situation when the distances between neighbor readers are not the same, say the distance set is $D = \{D_0, D_1, \ldots, D_{N-1}\}$. We call this case "Readers with uniform random distribution". Figure 1.2 illustrates this scenario. In this scenario, there are still N readers. The velocity is v, the height is h, and the distance between reader $R_i$ and $R_{i+1}$ is $D_i$. In this scenario, we develop an off-line algorithm to improve the system performance.

The algorithm is described as below.

- **Step 1:** Sort the distances in the set $\{D_0, D_1, \ldots, D_{N-1}\}$.

- **Step 2:** For reader $R_x$ with the smallest overlapping area (with largest distance, and both left and right sides with non-zero overlapping region), we reduce the power of $x^{th}$ reader, or the power of the $(x+1)^{th}$ reader until $t_{o.x.(x+1)} = 0$ or $t_{o.(x-1).x} = 0$ and $t_{o.(x+1).(x+2)} = 0$. That is we reduce the power of some reader until the overlapping area between this reader and another adjacent reader is zero. $t_{o.n.(n+1)}$ denotes the overlapping region between $R_n$ and $R_{n+1}$.

- **Step 3:** Repeat Step 1 until for all the readers, there is no overlapping area with adjacent readers or just one side overlapping area is zero (except for the leftmost and rightmost readers, since for the leftmost and rightmost readers, we can always reduce the leftmost and rightmost reader to make the overlap between $R_0$ and $R_1$, and the overlap between $R_{N-2}$ and $R_{N-1}$ to be zero).

After we apply the algorithm, we go to the first scenario of uniform random distribution: Scenario I (no gap between readers). At this point, we keep reducing the power level of readers whose overlapping area is non-zero (either left or right side) until the overlapping region is zero. Now there will be some gap generated on the other side (the side other than the side where the overlapping area just became zero). This is the other scenario we are concerned about: Scenario II (gap between readers). Our objective here is to analyze which scenario gives better performance (larger non-interference region or more total effective reading time).

## 4.5 Analysis For Uniform Random Distribution Scenarios

First, let us see how to compute $t_{overlap}$ between neighboring readers. Figure 4.8 is used to help us understand the notations for solving $t_{overlap}$. In Figure 4.8, reader $R_n$ sits on point A, the right neighbor of $R_n$ reader $R_i$ sits on point B. The shaded area is the overlapping region between reader $R_n$ and $R_i$. K is the midpoint of line EG, and J is the midpoint of line FG. The conveyor belt is moving rightward at a speed of v (meters per second). Distance between Reader $R_n$ and $R_i$ is $D_n$. Assume that the power level of reader $R_n$ is $x_n \cdot MAX\_POWER$, and that of $R_i$ is $x_i \cdot MAX\_POWER$, then the reading distance for reader $R_n$ is $x_n \cdot L_{max}$, and for reader $R_i$ is $x_i \cdot L_{max}$. From the figure, the follow equation holds:

$$|KG| + |FJ| = (|KF| + |FG| + |GJ|) + |FG| = D_n + |FG|. \tag{4.16}$$

We know that $|KG| = \sqrt{(x_n L_m ax)^2 - h^2}$ and $|FJ| = \sqrt{(x_i L_m ax)^2 - h^2}$. From the above equation, we get $|FG| = \sqrt{(x_n L_m ax)^2 - h^2} + \sqrt{(x_i L_m ax)^2 - h^2} - D_n$. Therefore, we get

$$t_{o.n.i} = \frac{\sqrt{(x_n L_{max})^2 - h^2} + \sqrt{(x_i L_{max})^2 - h^2} - D_n}{v}. \tag{4.17}$$

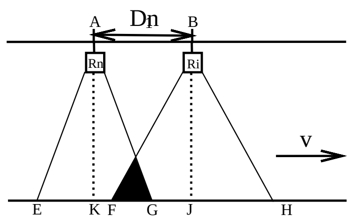


Figure 4.8: Overlapping region between adjacent readers in uniform random distribution

After applying the algorithm, we get scenario I, which looks like Figure 4.9, in which there are a couple of readers that have either left or right side overlap. Then we can get scenario II by reducing the power level of readers (with some overlap) to some amount. Here is a sample figure Figure 4.10 for scenario II.

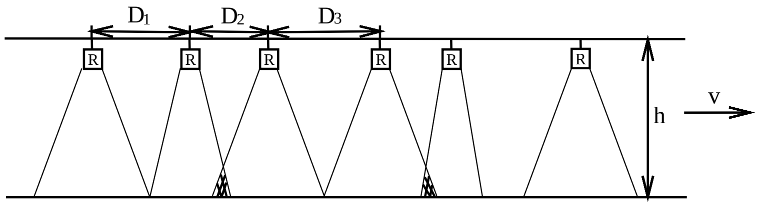


Figure 4.9: Scenario I: No gap between readers in non-uniform distribution

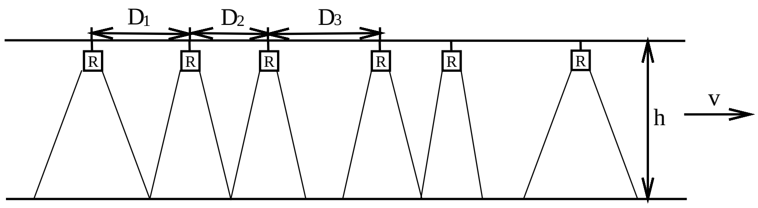


Figure 4.10: Scenario II: Gaps between readers in non-uniform distribution

For easy explanation, we take a snippet Figure 4.11 of the above two figures. Sub-figure (a) is taken from Figure 4.9, and sub-figure (b) is taken from Figure 4.10. In both top and bottom figures, the distance between $R_k$ and $R_{k+1}$ is $D_k$, the subtotal effective reading time of readers left to $R_k$ is $t_{left}$, and the subtotal effective reading time of readers right to $R_{k+1}$ is $t_{right}$. In Fig(a), there is overlap between reader $R_k$ and reader $R_{k+1}$, A is the middle point of line $|EF|$, and B is the middle point of line $|GH|$. The total effective time (Fig a) is:

$$Time_{effective}(Fig(a)) = t_{left} + t_{right} + \frac{|EF|}{v} + \frac{|GH|}{v} - 2 \cdot t_{o \cdot k \cdot (k+1)} \tag{4.18}$$

We already know that $t_{o \cdot k \cdot (k+1)} = \frac{|GF|}{v} = \frac{|AF|+|GB|-D_k}{v}$. Replacing $t_{o \cdot k \cdot (k+1)}$ in the last equation, we get

$$Time_{effective}(Fig(a)) = t_{left} + t_{right} + 2 \cdot \frac{D_k}{v} \quad (4.19)$$

In sub-figure (b), the power of reader $R_k$ is reduced to some amount in order to make $t_{o \cdot k \cdot (k+1)} = 0$. The total effective time (Fig b) is:

$$Time_{effective}(Fig(b)) = t_{left} + t_{right} + \frac{|E'G|}{v} + \frac{|GH|}{v} = t_{left} + t_{right} + 2 \cdot \frac{D_k}{v} \quad (4.20)$$

As a result, Scenarios I and II have the same total effective time. In chapter 5, we will test this statement in a simulation program.

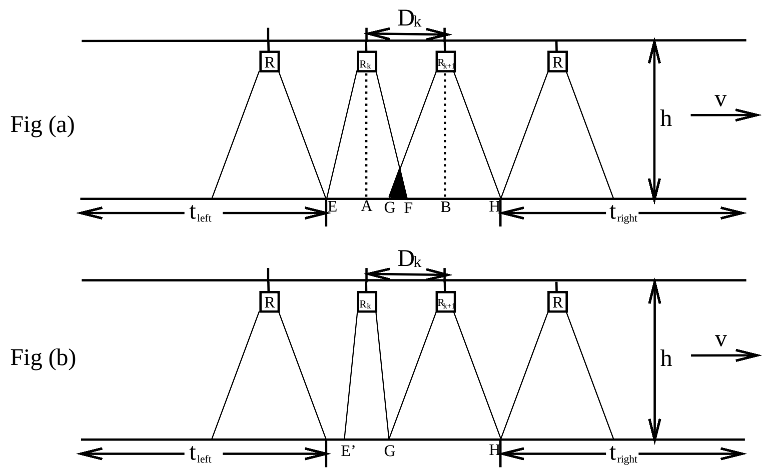


Figure 4.11: Snippet of above figures

# CHAPTER 5

# SIMULATION RESULTS

In this chapter, we will present the graphs and tables we obtained from a simulation program (in Java). Also we will write some observations here based on these results.

## 5.1 Results Involved In Constant Spacing Of The Readers

In this part, we test five different topologies. For simplicity, we use triple number group (h, d, L) to denote a topology, in which h is the height of readers, d is the distance between adjacent readers (in the constant spacing case, they are the same), L is the maximum reading distance for all readers (in this case, they also have the same reading distance). The five topologies are: $(1, 1, 1.3)$, $(1.5, 1, 1.8)$, $(2, 1, 2.2)$, $(1, 1.2, 1.5)$, and $(1, 1.4, 1.7)$. For the above five topologies, the conveyor belt has the same velocity of 4 meters per second, and the number of readers is 10. Table 5.1 which is an example table for topology (1, 1, 1.3) gives the total effective time of readers for each reader under different percentages of MAX_POWER. Only a small portion of data is listed here for succinctness. Table 5.2 lists out the prospective power level compared with the experimental power level in order to optimize the system performance (from chapter 4, we know that the percentage of reader's MAX_POWER is $x = \frac{\sqrt{D^2+4h^2}}{2L}$). Figure 5.1 shows us the total useful time for the above five topologies. Figure 5.1 illustrates the curves for total effective time and percentage of maximum power of each reader under all five topologies. In Figure 5.1, the curves go upwards when the power level increases from the minimum required power level (signal can reach tags) to the optimum power (with most effective time), and then go downwards when the power level is increased from optimum power to maximum power. From Table 5.2, we can see that the prospective results are consistent with the result we get from the simulation, which confirms the statement in chapter 4: the system has the best performance (total time each tag spends in the non-interference area is maximized) when $x = \frac{\sqrt{D^2+4h^2}}{2L}$ (x is a percentage of maximum power of each reader).

Table 5.1: Effective time vs. percentage of MAX_POWER under constant spcaing

| Percentage(x) | Effective_time (1, 1, 1.3) |
|---|---|
| 76.923077 | 0.0 |
| 77.023077 | 1.020135 |
| 77.313077 | 2.016502 |
| 77.793077 | 3.016482 |
| 78.453077 | 4.008771 |
| 79.293077 | 5.002769 |
| 80.313077 | 6.002738 |
| 81.503077 | 7.003571 |
| 82.853077 | 8.003071 |
| 84.353077 | 9.000214 |
| 86.003077 | 9.999785 |
| 88.243077 | 9.006120 |
| 90.703077 | 8.003250 |
| 93.323077 | 7.009336 |
| 96.153077 | 6.000267 |
| 99.113077 | 5.000013 |
| 99.993077 | 4.711655 |

## 5.2 Results Involved In Uniform Random Distribution Of The Readers

In this part, we compare the performance of two scenarios: **Scenario (I)** no gap between readers under non-uniform distribution (using the algorithm we introduced in chapter 4); **Scenario (II)** based on what we get from Scenario (I), keep reducing the power of a reader that has an overlap region with its neighbors until the overlap

Table 5.2: Prospective optimum power level percentage vs. experimental result

| Topology | Prospective Percentage: $\frac{\sqrt{D^2+4h^2}}{2L} * 100\%$ | Experimental percentage |
|---|---|---|
| (1, 1, 1.3) | 86.0000% | 86.0031% |
| (1.5, 1, 1.8) | 87.8410% | 87.8433% |
| (2, 1, 2.2) | 93.7069% | 93.7091% |
| (1, 1.2, 1.5) | 77.7460% | 77.7477% |
| (1, 1.4, 1.7) | 71.8033% | 71.8035% |

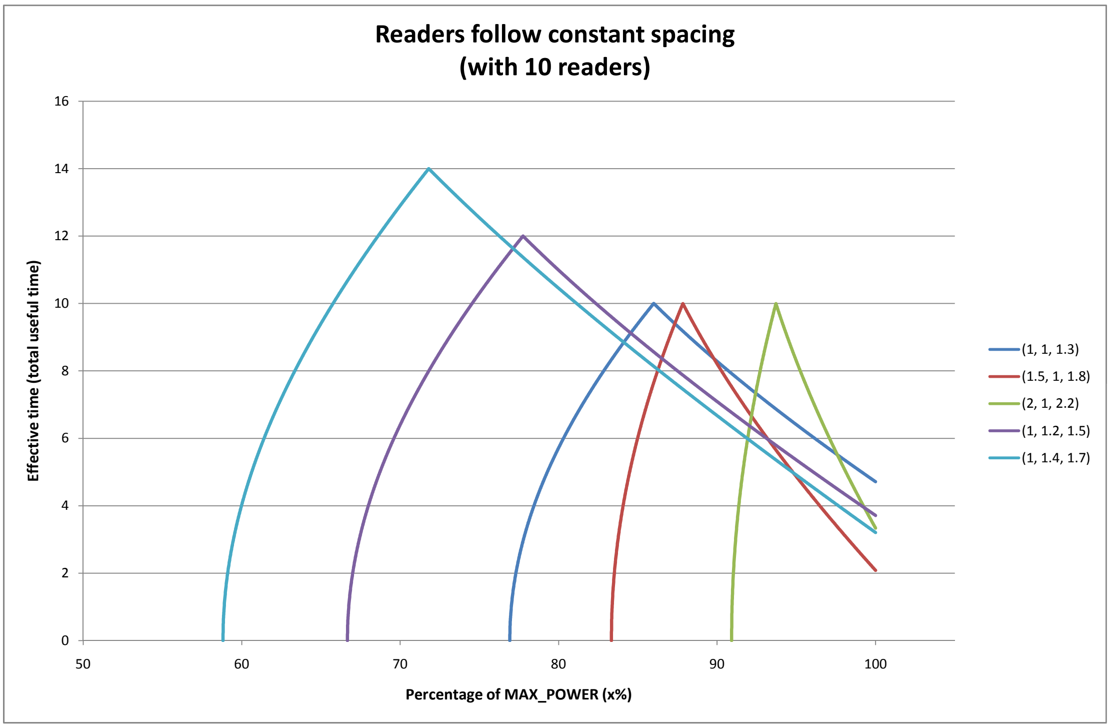


Figure 5.1: Effective time (total useful time) for different percentage of MAX_POWER with ten readers

region is zero (there are some gap regions on the side other than the overlap region side). For simplicity, we use URD (no gap) and URD (with gap) to denote the two scenarios respectively (URD means uniform random distribution), $R_i$ means the $(i+1)^{th}$ reader (from left to right), and we use a double number group (h, L) to denote a topology, in which h is the height of readers, and L is the maximum reading distance for all readers. Now let us explain how to set up the distance between adjacent readers: by using h and L, we can derive the range of distances between neighbor readers. Because when h and L are known, the distance between any pair of adjacent readers cannot less than a lower bound (if not, the middle reader of any three continuous readers can be turned off since it is nonsense to put a reader there), also this distance cannot be greater than an upper bound because there is some gap (a region is not covered by any reader) between neighboring readers even when readers work under maximum power. Obviously, we can easily figure out that the lower bound is $\sqrt{L^2 - h^2}$, and the upper bound is $2\sqrt{L^2 - h^2}$. In the simulation program, we randomly pick a value between the lower bound and upper bound.

We did experiments on six different topologies: (1, 1.3), (1, 1.6), (1, 1.9), (2, 2.2), (2, 2.5), and (2, 2.8). The number of readers is still ten. Table 5.4 lists out the power level, left overlap and right overlap for each reader in all six topologies. From the table, we observe that the right overlap of R3 and R7 (the left overlap of R4 and R8 respectively) in topology (1, 1.3), the right overlap of R2 and R4 (the left overlap of R3 and R5 respectively) in topology (1, 1.6), the right overlap of R1 and R5 (the left overlap of R2 and R6 respectively) in topology (1, 1.9), the right overlap of R4, R6, and R7 (the left overlap of R5, R7, and R8 respectively) in topology (2, 2.2), the right overlap of R2 and R7 (the left overlap of R3 and R8 respectively) in topology (2, 2.5), and the right overlap of R2 and R5 (the left overlap of R3 and R6 respectively) in topology (2, 2.8), are greater than zero. Also there is not a single reader that has both left overlap and right overlap greater than zero. Because Scenario (II) is based on Scenario (I), after completing the simulation of Scenario (II), the power of those readers whose left (right) overlap is not zero will be decreased until the left (right) overlap is zero. For example, in topology (1, 1.3), the power of R3 became 92.0390% of MAX_POWER and the power of R7 became 77.6368% of MAX_POWER; in topology (2, 2.5), the power of R2 became 96.5966% of MAX_POWER and the power of R7 became 82.8856% of MAX_POWER.

Table 5.3: Effective time (Scenario I vs. Scenario II) under uniform random distribution

| Values | **(1, 1.3)** | **(1, 1.6)** | **(1, 1.9)** | **(2, 2.2)** | **(2, 2.5)** | **(2, 2.8)** |
|---|---|---|---|---|---|---|
| **Distance[R0, R1]** | 1.0383 | 1.5998 | 2.3989 | 0.9538 | 1.8996 | 2.1880 |
| **Distance[R1, R2]** | 1.2940 | 2.1423 | 2.2737 | 1.7497 | 2.2659 | 2.9570 |
| **Distance[R2, R3]** | 1.6501 | 1.5531 | 1.6662 | 1.7914 | 2.1123 | 2.9371 |
| **Distance[R3, R4]** | 1.1231 | 2.4835 | 2.1042 | 1.3501 | 2.0632 | 2.3648 |
| **Distance[R4, R5]** | 1.0693 | 1.6064 | 3.1220 | 1.0062 | 2.8018 | 2.6786 |
| **Distance[R5, R6]** | 1.4339 | 1.7750 | 2.5637 | 1.8204 | 2.9975 | 2.1770 |
| **Distance[R6, R7]** | 1.2154 | 1.4484 | 2.3718 | 1.1017 | 2.1153 | 3.4483 |
| **Distance[R7, R8]** | 0.8422 | 1.4485 | 2.5013 | 1.5240 | 1.8425 | 2.7943 |
| **Distance[R8, R9]** | 1.5363 | 1.9507 | 1.9624 | 1.1787 | 2.8005 | 1.9734 |
| **Effective Time(Scenario 1)** | 3.043801 | 4.079183 | 5.286559 | 3.015894 | 5.828100 | 5.977059 |
| **Effective Time(Scenario 2)** | 3.043801 | 4.079183 | 5.286559 | 3.015894 | 5.828100 | 5.977059 |

Table 5.3 shows us the detailed topology attributes, values and the effective time for both above Scenario (I) and (II). From this table, we can see that the effective times for both scenarios (with gap and without gap) are exactly the same for all six tested topologies. So there is no advantage to use the second model (URD with gap). However, from the perspective of power consuming, URD (with gap) model does save more power than URD (no gap) model.

Table 5.4: Power level Left overlap right overlap of readers in NUD (no gap)

| Topology(1, 1.3) | percentage of MAX_POWER | left overlap | right overlap | Topology(1, 1.6) | percentage of MAX_POWER | left overlap | right overlap |
|---|---|---|---|---|---|---|---|
| **R0** | 88.3060% | 0.0 | 0.0 | **R0** | 76.5222% | 0.0 | 0.0 |
| **R1** | 85.1448% | 0.0 | 0.0 | **R1** | 83.8072% | 0.0 | 0.0 |
| **R2** | 99.4510% | 0.0 | 0.0 | **R2** | 100% | 0.0 | 0.9304 |
| **R3** | 100% | 0.0 | 0.1737 | **R3** | 99.2952% | 0.9304 | 0.0 |
| **R4** | 84.8699% | 0.1737 | 0.0 | **R4** | 100% | 0.0 | 0.7159 |
| **R5** | 89.8337% | 0.0 | 0.0 | **R5** | 91.6870% | 0.7159 | 0.0 |
| **R6** | 100% | 0.0 | 0.0 | **R6** | 76.3488% | 0.0 | 0.0 |
| **R7** | 82.4190% | 0.0 | 0.2482 | **R7** | 78.0041% | 0.0 | 0.0 |
| **R8** | 94.1461% | 0.2482 | 0.0 | **R8** | 76.3521% | 0.0 | 0.0 |
| **R9** | 100% | 0.0 | 0.0 | **R9** | 100% | 0.0 | 0.0 |

| Topology(1, 1.9) | percentage of MAX_POWER | left overlap | right overlap | Topology(2, 2.2) | percentage of MAX_POWER | left overlap | right overlap |
|---|---|---|---|---|---|---|---|
| **R0** | 66.8560% | 0.0 | 0.0 | **R0** | 90.9800% | 0.0 | 0.0 |
| **R1** | 100% | 0.0 | 0.4103 | **R1** | 99.2255% | 0.0 | 0.0 |
| **R2** | 77.0209% | 0.4103 | 0.0 | **R2** | 99.2270% | 0.0 | 0.0 |
| **R3** | 61.3194% | 0.0 | 0.0 | **R3** | 100% | 0.0 | 0.0 |
| **R4** | 95.1641% | 0.0 | 0.0 | **R4** | 93.0207% | 0.0 | 0.3313 |
| **R5** | 100% | 0.0 | 0.5379 | **R5** | 99.7627% | 0.3313 | 0.0 |
| **R6** | 94.2730% | 0.5379 | 0.0 | **R6** | 100% | 0.0 | 0.4223 |
| **R7** | 70.3102% | 0.0 | 0.0 | **R7** | 95.0105% | 0.4223 | 0.2482 |
| **R8** | 100% | 0.0 | 0.0 | **R8** | 100% | 0.2482 | 0.0 |
| **R9** | 55.7074% | 0.0 | 0.0 | **R9** | 91.6867% | 0.0 | 0.0 |

| Topology(2, 2.5) | percentage of MAX_POWER | left overlap | right overlap | Topology(2, 2.8) | percentage of MAX_POWER | left overlap | right overlap |
|---|---|---|---|---|---|---|---|
| R0 | 76.5222% | 0.0 | 0.0 | R0 | 83.1264% | 0.0 | 0.0 |
| R1 | 83.8072% | 0.0 | 0.0 | R1 | 79.8184% | 0.0 | 0.0 |
| R2 | 100% | 0.0 | 0.1466 | R2 | 100% | 0.0 | 0.6682 |
| R3 | 99.2952% | 0.1466 | 0.0 | R3 | 92.5029% | 0.6682 | 0.0 |
| R4 | 100% | 0.0 | 0.0 | R4 | 75.9048% | 0.0 | 0.0 |
| R5 | 91.6870% | 0.0 | 0.0 | R5 | 100% | 0.0 | 1.2713 |
| R6 | 76.3488% | 0.0 | 0.0 | R6 | 89.0446% | 1.2713 | 0.0 |
| R7 | 78.0041% | 0.0 | 0.0733 | R7 | 100% | 0.0 | 0.0 |
| R8 | 76.3521% | 0.0733 | 0.0 | R8 | 77.4003% | 0.0 | 0.0 |
| R9 | 100% | 0.0 | 0.0 | R9 | 82.1928% | 0.0 | 0.0 |

# CHAPTER 6

# CONCLUSIONS AND FUTURE WORK

In this thesis, we started with introducing the concepts of RFID, then focus on the collision problems in RFID, particularly, we concentrated in those anti-collision algorithms (collisions specifically refers to tag-tag collisions and reader-reader collisions). Then we proposed the methodology for power control in multi-reader environment. Two main system topologies (in term of distance between adjacent readers) were considered: readers with uniform distribution (same distance between any two adjacent readers) and readers with non-uniform distribution (random distance between any two adjacent readers, which means the distances are just arbitrary values). We did detailed analysis on both topologies, and figured out what the power level of readers should be in order to maximize the total effective time (total time tag spend in non-interference region) in uniform distribution topology. In non-uniform distribution topology, we developed an off-line algorithm to improve the system performance (increase the total effective time). We found that when power percentage of MAX_POWER is $\frac{\sqrt{D^2+4\cdot h^2}}{2\cdot L}$, in which D is the distance between neighbors, h is the height of readers, L is the maximum reading distance of readers (each reader has the same value of maximum reading distance). When each reader works under this power level, the overlaps all readers are zero. Then we compared performance of two scenarios (one with zero-gap, the other with gaps) after applying the off-line algorithm. We found that the total effective time for both are the same. But in terms of power consuming, the with gap scenario is more power conserving than the no gap scenario. Finally we run simulation program (code is available in Appendix A) to test what we prospected.

Future work could be done in (i) solving the scenario when readers and tags are both mobile (more complicate); (ii) considering other factors affecting the performance of the system other than power of readers; (iii) making the analysis generic not only limited to conveyor belt system.

## BIBLIOGRAPHY


[1] Unknown, "RFID and its history." Website, 2007. Available at `http://www.rfid-weblog.com/50226711/rfid_and_its_history.php`.

[2] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag," in *Proceedings of the Second International Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS)*, (San Diego, USA), July 2005.

[3] K. S. Leong, M. L. Ng, A. Grasso, and P. Cole, "Synchronization of RFID Readers for Dense RFID Reader Environments," in *Applications and the Internet Workshops 2006*, pp. 4–51, January 2006.

[4] Unknown, "An RFID tag." Website, unknown. Available at `http://www.jltrfid.com/public/editor/rfid_panoramica/tag.jpg`.

[5] G. Avoine and P. Oechslin, "RFID Traceability: A Multilayer Problem." Presentation, 2005. Available at `http://lasecwww.epfl.ch/ gavoine/rfid`.

[6] Z. Zhou, H. Gupta, S. Das, and X. Zhu, "Slotted Scheduled Tag Access in Multi-reader RFID Systems," in *Proceeding of the IEEE International Conference on Network Protocols (ICNP'07)*, (Beijing, China), pp. 61–70, October 2007.

[7] V. Sarangan, M. Devarapalli, and S. Radhakrishnan, "A Framework for Fast RFID Tag Reading in Static and Mobile Environments," *Computer Networks*, vol. 52, pp. 1058–1073, 2008.

[8] Wiki, "Radio-frequency Identification (wikipedia)." Website, 2009. Available at http://en.wikipedia.org/wiki/RFID.

[9] R. Bhochhibhoya, "Mobile Tag Reading in a Multi-reader RFID Environment," Master's thesis, Oklahoma State University, 2008.

[10] R. Lowdown, "Classification of RFID tags." Website, 2006. Available at http://www.rfidlowdown.com/2006/04/classification.html.

[11] V. Sarangan, "Fast RFID Tag Reading in Dense Environments." Lecture, 2008. Not available now.

[12] E. C. P. Ltd, "RFID vs Barcodes." Website, 2004. Available at http://www.electrocom.com.au/pdfs/RFIDvsBarcodes.pdf.

[13] V. D. Hunt, A. Puglia, and M. Puglia, *RFID: A Guide to Radio Frequency Identification*. Wiley-Interscience, kindle ed., 2007.

[14] Unknown, "RFID basics - how does RFID work." Website, 2004. Available at http://www.abrfid.com/rfid/articles/rfid-basics.aspx.

[15] J. Bank, "Understanding RFID Privacy and Security." Website, 2008. Available at http://www.rfidnews.org/understanding-rfid-privacy-and-security.

[16] Wiki, "Frequency-hopping Spread Spectrum (wikipedia)." Website, 2009. Available at http://en.wikipedia.org/wiki/Frequency-hopping_spread_spectrum.

[17] D. R. Hush and C. Wood, "Analysis of Tree Algorithms for RFID Arbitration," in *Proceedings of IEEE International Symposium on Information Theory*, (Boston, USA), pp. 107–117, 1998.

[18] F. C. Schoute, "Dynamic Frame Length ALOHA," *IEEE Transcations on Communications*, vol. 31, no. 4, pp. 565–568, 1983.

[19] J. R. Cha and J. Kim, "Dynamic Framed Slotted ALOHA Algorithms Using Fast Tag Estimation Method for RFID Systems," in *Proceedings of the Third IEEE Consumer Communications and Networking Conference, CCNC*, (Las Vegas, USA), pp. 768–772, 2006.

[20] J. Waldrop, D. Engels, and S. Sarma, "Colorwave: A MAC for RFID Reader Networks," in *Proceeding of WCNC*, 2003.

[21] J. Ho, D. Engels, and S. Sarma, "HiQ: A Hierarchical Q-learning Algorithm to Solve The Reader Collision Problem," in *SAINTW*, 2006.

# APPENDIX A

# SIMULATION PROGRAM

Listing A.1: Global.java

```
/**
 * PROGRAM: Global.java
 * This class Global declares all the global variables.
 * @author Xiaodan Fang
 */
import java.util.Vector;

class Global {

        public static Vector<Reader> readerGroup = new Vector<Reader>();
        public static int numReaders;   // the total number of readers in model
        public static double height;    // height of readers

        // distance to be modified generate other distance
        public static double distanceBase;
        public static double [] distance;       // distance set of adjacent readers

        public static double velocity;  // velocity of the conveyor belt
        public static boolean uniform = false;  // true if readers uniform distributed
        public static double eff_time_uniform;  // effective time under uniform distribution
        public static double eff_time_nonuniform; // effective time under non-uniform distribution

        // UD means uniform distribution
        public static Vector<String> results = new Vector<String>(); // Store the results for UD

}
```

Listing A.2: Log.java

```
/**
 * PROGRAM: Log.java
 * This class Log declare all the attributes and methods for a Log.
 * @author Xiaodan Fang
 */
import java.io.*;

public class Log {

        private static FileWriter filewrite; // file writer

        /**
         * This method generates a new log file
         * @param fileName the file name operate on
         * @exception java.io.FileNotFoundException the file doesn't exist
         */
        public static void Generate(String fileName)
        {
                try{
                        filewrite = new FileWriter(fileName);
                }
                catch(Exception e){
                        System.out.println("Couldn't open log file " + fileName);
                }
        }

        /**
         * This method write to a log file
         * @param message the string to be written to a file
         */
        public static void write(String message)
        {
                try{
                        filewrite.write(message + "\r\n");
                        filewrite.flush();
                }
                catch(Exception e){
                        System.out.println("Couldn't write log file");
                }
        }

        /**
         * This method close a log file
         */
        public static void close()
        {
                try{
                        filewrite.close();
                }
                catch(Exception e){
                        System.out.println("Couldn't close log file");
                }
        }
}
```

Listing A.3: Reader.java

```
/**
 * PROGRAM: Reader.java
 * This class Reader define attributes for a reader and overloads method
 * toString() for printing a Reader object.
 * @author Xiaodan Fang
 */
public class Reader {

        static double MAXP = 100;           // maximum power
        static double MAXL = 2.8;           // maximum read distance
        int id;                             // id
        double power = MAXP;                // current power AND default power is MAXP
        double leftOverlap;                         // left interference lane
        double rightOverlap;                // right interference lane
        double effectiveLane;               // effective reading lane
        double lane;                                // total range (under a reader)

        /**
         * Class constructor
         * @param id of a reader
         */
        Reader(int idParam)
        {
                id = idParam;
        }

        /**
         * overloading method for printing object purpose.
         * @return value detailed information of a reader
         */
        public String toString()
        {
                String value = "Reader Id: " + id + "\n";
            value = value + "Reader power: " + power + "\n";
            value = value + "Reader lane: " + lane + "\n";
            value = value + "Reader leftOverlap: " + leftOverlap + "\n";
            value = value + "Reader rightOverlap: " + rightOverlap + "\n";
            value = value + "Reader effectiveLane: " + effectiveLane + "\n";
              return value;
        }


}
```

Listing A.4: Algo.java

```java
/**
 * PROGRAM: Algo.java
 * This class implement the developed algorithm for NUD scenarios.
 * NUD denotes non-uniform distribution.
 * @author Xiaodan Fang
 */
public class Algo {

        static final double BOUND = 0.00001; // BOUND used for comparison

        /**
         * This method implement the Scenario I (no gap)
         * @return value total effective time for all readers
         */
        public static double scen1() // the scenario with no gap
        {
                double value = 0.0;

                initialUpdate(); // update all readers profile

                while(!isDoneSce1())
                {
                        // "1"-initially modify left side reader power
                        // "2"-initially modify right side reader power
                        // "3"-initially modify both side reader power
                        // in this simulation, always use 1. If need to use others,
                        // add some line of code is necessary.
                        int flag = 1;
                        int tempId = findId();
                        int idNeedModify = -1; // no reader needs to be modified

                        if(flag==1)
                                idNeedModify = tempId;
                        if(flag==2)
                                idNeedModify = tempId+1;
                        if(flag==3)
                                idNeedModify = tempId;

                        // debug print starts
                        System.out.println("flag="+ flag + ",idNeedModify="+idNeedModify);
                        System.out.println(Global.readerGroup.elementAt(idNeedModify));
                        // debug print ends

                        if(idNeedModify==-1)
                                break;
                        else
                                reducePowerToBound(flag, idNeedModify);

                        if(flag==3)
                        {
                                if(!leftmost(idNeedModify))
                                        updateReader_NUD(idNeedModify-1);
                                if(!rightmost(idNeedModify+1))
                                        updateReader_NUD(idNeedModify+2);
```

```java
                        if(leftmost(idNeedModify))
                            updateReader_NUD(idNeedModify-1);
                    }
                    if(flag==2)
                    {
                        if(!leftmost(idNeedModify))
                            updateReader_NUD(idNeedModify-1);
                        if(!rightmost(idNeedModify))
                            updateReader_NUD(idNeedModify+1);

                    }
                    if(flag==1)
                    {
                        if(!leftmost(idNeedModify))
                            updateReader_NUD(idNeedModify-1);
                        if(!rightmost(idNeedModify))
                            updateReader_NUD(idNeedModify+1);
                    }
                }

                // debug
                System.out.println("======start_debug_scen#1======");
                for(int i=0; i<Global.readerGroup.size(); i++)
                    System.out.println(Global.readerGroup.elementAt(i));
                System.out.println("======end_debug_scen#1======");
                // end debug

                value = MainClass.calculate_uniform();

                return value;
        }

        /**
         * This method reduce a specified reader's power to BOUND.
         * @param flagParam 1, 2 or 3
         * @param idParam id of the reader
         */
        private static void reducePowerToBound(int flagParam,int idParam)
        {
                double temp;

                // reduce rightOverlap to BOUND
                if(Global.readerGroup.elementAt(idParam).rightOverlap <
                        Global.readerGroup.elementAt(idParam).leftOverlap)
                {
                    if(idParam!=Global.readerGroup.size()-1)
                        temp = (Math.sqrt(Math.pow(Global.distance[idParam] -
                                Global.readerGroup.elementAt(idParam+1).lane/2, 2)+
                                Math.pow(Global.height, 2)))*Reader.MAXP/Reader.MAXL;
                    else
                        temp = (Math.sqrt(Math.pow(Global.distance[idParam-1] -
                               Global.readerGroup.elementAt(idParam-1).lane/2, 2)+
                               Math.pow(Global.height, 2)))*Reader.MAXP/Reader.MAXL;

                    updateReader_NUD(idParam, temp);
```

```
                }
                else
                {
                        if(idParam!=0)
                                temp = (Math.sqrt(Math.pow(Global.distance[idParam-1] -
                                        Global.readerGroup.elementAt(idParam-1).lane/2, 2)+
                                        Math.pow(Global.height, 2)))*Reader.MAXP/Reader.MAXL;
                        else
                                temp = (Math.sqrt(Math.pow(Global.distance[idParam] -
                                        Global.readerGroup.elementAt(idParam+1).lane/2, 2)+
                                        Math.pow(Global.height, 2)))*Reader.MAXP/Reader.MAXL;

                        updateReader_NUD(idParam, temp);
                }
        }

        /**
         * This method check if a specified reader has overlaps on both side
         * @param idParam
         * @return true(both sides overlap);
         * @return false(at least one side no overlap);
         */
        private static boolean bothSideCross(int idParam)
        {
                if(Global.readerGroup.elementAt(idParam).leftOverlap>BOUND
                  && Global.readerGroup.elementAt(idParam).rightOverlap>BOUND)
                        return true;
                return false;
        }

        /**
         * This method find the reader need to operate on in Scenario I.
         * @return id of the reader
         */
        private static int findId()
        {
                double min = 1000;
                int temp = -1;
                for(int i=1; i<Global.readerGroup.size()-1; i++)
                {
                        if(Global.readerGroup.elementAt(i).leftOverlap>BOUND
                                && Global.readerGroup.elementAt(i).rightOverlap>BOUND
                                        && Global.readerGroup.elementAt(i).rightOverlap<min)
                        {
                                min = Global.readerGroup.elementAt(i).rightOverlap;
                                temp = i;
                        }
                }
                if(Global.readerGroup.elementAt(0).rightOverlap>BOUND &&
                        Global.readerGroup.elementAt(0).rightOverlap<min)
                {
                        min = Global.readerGroup.elementAt(0).rightOverlap;
                        temp = 0;
                }
                if(Global.readerGroup.lastElement().leftOverlap>BOUND &&
```

```
                        Global.readerGroup.lastElement().leftOverlap<min)
                        temp = Global.readerGroup.size()-1;
                return temp;
        }

        /**
         * This method check if finish the Scenario I simulation
         * @return true (no reader has both side overlap and left-end
         *                and right-end reader doesn't have overlap);
         * @return false (at least one reader has both side overlap and
         *                left-end and right-end reader has overlap);
         */
        private static boolean isDoneSce1()
        {
                for(int i=1; i<Global.readerGroup.size()-1; i++)
                {
                        if( Global.readerGroup.elementAt(i).leftOverlap>BOUND
                                 && Global.readerGroup.elementAt(i).rightOverlap>BOUND)
                                return false;
                }
                if( Global.readerGroup.firstElement().rightOverlap>BOUND ||
                        Global.readerGroup.lastElement().leftOverlap>BOUND )
                   return false;
                return true;
        }

        /**
         * This method check if finish the Scenario II simulation
         * @return true (no overlap left);
         * @return false (still have overlap left);
         */
        private static boolean isDoneSce2()
        {
                for(int i=0; i<Global.readerGroup.size(); i++)
                {
                        if( Global.readerGroup.elementAt(i).leftOverlap>BOUND
                                        || Global.readerGroup.elementAt(i).rightOverlap>BOUND)
                                return false;
                }
                return true;
        }

        /**
         * This method initialize all readers profile when power is MAX_POWER.
         */
        private static void initialUpdate()
        {
                for(int i=0; i<Global.readerGroup.size(); i++)
                        updateReader_NUD(i, Reader.MAXP);
        }

        /**
         * This method check if the specified reader is leftmost reader.
         * @param idParam id of the specified reader
         * @return true (if leftmost), false(if not)
```

```
     */
    private static boolean leftmost(int idParam)
    {
        return (idParam==0);
    }

    /**
     * This method check if the specified reader is rightmost reader.
     * @param idParam id of the specified reader
     * @return true (if rightmost), false(if not)
     */
    private static boolean rightmost(int idParam)
    {
        return (idParam==Global.readerGroup.size()-1);
    }

    /**
     * This method check if the specified reader have left overlap.
     * @param idParam id of the specified reader
     * @return true (if have left overlap), false(if not)
     */
    private static boolean hasLeftOverlap(int i)
    {
        double dLeft = (Global.readerGroup.elementAt(i-1).power*Reader.MAXL)/
                                                    Reader.MAXP;

        double d = (Global.readerGroup.elementAt(i).power*Reader.MAXL)/Reader.MAXP;
        if( Math.sqrt(Math.pow(dLeft, 2)-Math.pow(Global.height, 2)) +
                Math.sqrt(Math.pow(d, 2)-Math.pow(Global.height, 2)) >
            Global.distance[i-1] )
                return true;
        return false;
    }

    /**
     * This method check if the specified reader have right overlap.
     * @param idParam id of the specified reader
     * @return true (if have right overlap), false(if not)
     */
    private static boolean hasRightOverlap(int i)
    {
        double dRight = (Global.readerGroup.elementAt(i+1).power*Reader.MAXL)/Reader.MAXP;
        double d = (Global.readerGroup.elementAt(i).power * Reader.MAXL) / Reader.MAXP;
        if( Math.sqrt(Math.pow(d, 2)-Math.pow(Global.height, 2)) +
                Math.sqrt(Math.pow(dRight, 2)-Math.pow(Global.height, 2)) >
            Global.distance[i] )
                return true;
        return false;
    }

    /**
     * This method update one specified reader's profile.
     * @param i id of the specified reader
     */
    private static void updateReader_NUD(int i)
```

```
        {
            double lane = Global.readerGroup.elementAt(i).lane;

            if(leftmost(i)) // leftmost reader
            {
                double dRight = (Global.readerGroup.elementAt(i+1).power*Reader.MAXL)/
                                                                        Reader.MAXP;

                Global.readerGroup.elementAt(i).leftOverlap = 0;
                if(hasRightOverlap(i)) // there is overlap region with right-side reader
                {
                    double rightCross = ( Math.sqrt(Math.pow(dRight,2)-
                                          Math.pow(Global.height,2)) )
                                    + ( lane/2 ) - Global.distance[i];

                    Global.readerGroup.elementAt(i).rightOverlap = rightCross;
                    Global.readerGroup.elementAt(i).effectiveLane = lane-rightCross;
                }
                else // there is no overlap with right-side reader
                {
                    Global.readerGroup.elementAt(i).rightOverlap = 0;
                    Global.readerGroup.elementAt(i).effectiveLane = lane;
                }

            }
            else if(rightmost(i)) // rightmost reader
            {
                double dLeft = (Global.readerGroup.elementAt(i-1).power*Reader.MAXL)/
                                                                        Reader.MAXP;

                Global.readerGroup.elementAt(i).rightOverlap = 0;
                if(hasLeftOverlap(i)) // there is overlap region with the left-side reader
                {
                    double leftCross = ( Math.sqrt(Math.pow(dLeft,2)-
                                           Math.pow(Global.height,2)) )
                                     + ( lane/2 ) - Global.distance[i-1];

                    Global.readerGroup.elementAt(i).leftOverlap = leftCross;
                    Global.readerGroup.elementAt(i).effectiveLane = lane-leftCross;
                }
                else // there is no overlap region with left-side reader
                {
                    Global.readerGroup.elementAt(i).leftOverlap = 0;
                    Global.readerGroup.elementAt(i).effectiveLane = lane;
                }
            }
            else // middle readers
            {
                double dLeft = (Global.readerGroup.elementAt(i-1).power*Reader.MAXL)/
                                                                    Reader.MAXP;

                double dRight = (Global.readerGroup.elementAt(i+1).power*Reader.MAXL)/
                                                                Reader.MAXP;

                if(hasRightOverlap(i)) // there is overlap region with right-side reader
```

```
                        {
                                double rightCross = ( Math.sqrt(Math.pow(dRight,2)-
                                                        Math.pow(Global.height,2)) )
                                                + ( lane/2 ) - Global.distance[i];

                                Global.readerGroup.elementAt(i).rightOverlap = rightCross;
                        }
                        else // there is no overlap with right-side reader
                                Global.readerGroup.elementAt(i).rightOverlap = 0;

                        if(hasLeftOverlap(i)) // has overlap region with the left-side reader
                        {
                                double leftCross =  ( Math.sqrt(Math.pow(dLeft,2)-
                                                        Math.pow(Global.height,2)) )
                                                  + ( lane/2 ) - Global.distance[i-1];

                                Global.readerGroup.elementAt(i).leftOverlap = leftCross;
                        }
                        else // there is no overlap region with left-side reader
                                Global.readerGroup.elementAt(i).leftOverlap = 0;

                        Global.readerGroup.elementAt(i).effectiveLane = lane -
                                        Global.readerGroup.elementAt(i).leftOverlap -
                                        Global.readerGroup.elementAt(i).rightOverlap;

                }

        }

        /**
         * This method update one specified reader's profile.
         * @param i id of the specified reader
         * @param powerParam new power of the specified reader
         */
        private static void updateReader_NUD(int i, double powerParam)
        {
                double d = (powerParam * Reader.MAXL) / Reader.MAXP;
                double lane = 2*Math.sqrt(Math.pow(d,2)-Math.pow(Global.height,2));
                Global.readerGroup.elementAt(i).power = powerParam;
                Global.readerGroup.elementAt(i).lane = lane;

                if(leftmost(i)) // leftmost reader
                {
                        double dRight = (Global.readerGroup.elementAt(i+1).power*Reader.MAXL)/
                                                                        Reader.MAXP;

                        Global.readerGroup.elementAt(i).leftOverlap = 0;
                        if(hasRightOverlap(i)) // there is overlap region with right-side reader
                        {
                                double rightCross = ( Math.sqrt(Math.pow(dRight,2)-
                                                Math.pow(Global.height,2)) )
                                                + ( lane/2 ) - Global.distance[i];

                                Global.readerGroup.elementAt(i).rightOverlap = rightCross;
                                Global.readerGroup.elementAt(i).effectiveLane = lane-rightCross;
```

```
                }
                else // there is no overlap with right-side reader
                {
                        Global.readerGroup.elementAt(i).rightOverlap = 0;
                        Global.readerGroup.elementAt(i).effectiveLane = lane;
                }

        }
        else if(rightmost(i)) // rightmost reader
        {
                double dLeft = (Global.readerGroup.elementAt(i-1).power*Reader.MAXL)/
                                                                Reader.MAXP;

                Global.readerGroup.elementAt(i).rightOverlap = 0;
                if(hasLeftOverlap(i)) // has overlap region with the left-side reader
                {
                        double leftCross =  ( Math.sqrt(Math.pow(dLeft,2)-
                                                Math.pow(Global.height,2)) )
                                        + ( lane/2 ) - Global.distance[i-1];

                        Global.readerGroup.elementAt(i).leftOverlap = leftCross;
                        Global.readerGroup.elementAt(i).effectiveLane = lane-leftCross;
                }
                else // there is no overlap region with left-side reader
                {
                        Global.readerGroup.elementAt(i).leftOverlap = 0;
                        Global.readerGroup.elementAt(i).effectiveLane = lane;
                }
        }
        else // middle readers
        {
                double dLeft = (Global.readerGroup.elementAt(i-1).power*Reader.MAXL)/
                                                                        Reader.MAXP;

                double dRight = (Global.readerGroup.elementAt(i+1).power*Reader.MAXL)/
                                                                        Reader.MAXP;

                if(hasRightOverlap(i)) // there is overlap region with right-side reader
                {
                        double rightCross = ( Math.sqrt(Math.pow(dRight,2)-
                                        Math.pow(Global.height,2)) )
                                + ( lane/2 ) - Global.distance[i];

                        Global.readerGroup.elementAt(i).rightOverlap = rightCross;
                }
                else // there is no overlap with right-side reader
                        Global.readerGroup.elementAt(i).rightOverlap = 0;

                if(hasLeftOverlap(i)) // has overlap region with the left-side reader
                {
                        double leftCross =  ( Math.sqrt(Math.pow(dLeft,2)-
                                        Math.pow(Global.height,2)) )
                                        + ( lane/2 ) - Global.distance[i-1];

                        Global.readerGroup.elementAt(i).leftOverlap = leftCross;
```

```
                        }
                        else // there is no overlap region with left-side reader
                                Global.readerGroup.elementAt(i).leftOverlap = 0;

                        Global.readerGroup.elementAt(i).effectiveLane = lane -
                                                    Global.readerGroup.elementAt(i).leftOverlap -
                                                    Global.readerGroup.elementAt(i).rightOverlap;

                }

        }

        /**
         * This method find the id of reader need to operate on in Scenario II.
         * @return id of the reader
         */
        private static int findIdScen2()
        {
                int temp = -1;
                for(int i=1; i<Global.readerGroup.size()-1; i++)
                {
                        if(Global.readerGroup.elementAt(i).leftOverlap>BOUND ||
                                Global.readerGroup.elementAt(i).rightOverlap>BOUND)
                        {
                                temp = i;
                                return temp;
                        }
                }
                return temp;
        }

        /**
         * This method reduce the power of the specified reader to BOUND
         * @param flagParam 1, 2, or 3
         * @param idParam id of the specified reader
         */
        private static void reducePowerToBoundScen2(int flagParam,int idParam)
        {
                double temp;
                if(Global.readerGroup.elementAt(idParam).leftOverlap > BOUND)
                {
                        temp = (Math.sqrt(Math.pow(Global.distance[idParam-1] -
                                        Global.readerGroup.elementAt(idParam-1).lane/2, 2)+
                                        Math.pow(Global.height, 2)))*Reader.MAXP/Reader.MAXL;
                        updateReader_NUD(idParam, temp);
                }
                if(Global.readerGroup.elementAt(idParam).rightOverlap > BOUND)
                {
                        temp = (Math.sqrt(Math.pow(Global.distance[idParam] -
                                        Global.readerGroup.elementAt(idParam+1).lane/2, 2)+
                                        Math.pow(Global.height, 2)))*Reader.MAXP/Reader.MAXL;
                        updateReader_NUD(idParam, temp);
                }
        }

```

```
	/**
	 * This method implement the Scenario II (with gap exists)
	 * @return value total effective time for all readers
	 */
	public static double scen2()
	{
		double value = 0.0;

		while(!isDoneSce2())
		{
			int flag = 1; // flag can be 1, 2, or 3
			int tempId = findIdScen2();
			int idNeedModify = -1;

			if(flag==1)
				idNeedModify = tempId;
			if(flag==2)
				idNeedModify = tempId+1;
			if(flag==3)
				idNeedModify = tempId;

			// debug print starts
			System.out.println("flag="+ flag + ",idNeedModify="+idNeedModify);
			System.out.println(Global.readerGroup.elementAt(idNeedModify));
			// debug print ends

			if(idNeedModify==-1)
				break;
			else
				reducePowerToBoundScen2(flag, idNeedModify);

			if(flag==3)
			{

				if(!leftmost(idNeedModify))
					updateReader_NUD(idNeedModify-1);
				if(!rightmost(idNeedModify+1))
					updateReader_NUD(idNeedModify+2);
				if(leftmost(idNeedModify))
					updateReader_NUD(idNeedModify-1);
			}
			if(flag==2)
			{
				if(!leftmost(idNeedModify))
					updateReader_NUD(idNeedModify-1);
				if(!rightmost(idNeedModify))
					updateReader_NUD(idNeedModify+1);

			}
			if(flag==1)
			{
				if(!leftmost(idNeedModify))
					updateReader_NUD(idNeedModify-1);
				if(!rightmost(idNeedModify))
					updateReader_NUD(idNeedModify+1);
```

```
                }
        }

        // debug
        System.out.println("======start_debug_scen#2======");
        for(int i=0; i<Global.readerGroup.size(); i++)
                System.out.println(Global.readerGroup.elementAt(i));
        System.out.println("======end_debug_scen#2======");
        // end debug

        value = MainClass.calculate_uniform();

        return value;
    }
}
```

Listing A.5: MainClass.java

```
/**
 * PROGRAM: MainClass.java
 * This program simulate the two scenarios introduced in the thesis,
 * uniform distribution & non-uniform distribution.
 * @author Xiaodan Fang
 */
import java.util.Random;
import java.util.Scanner;

public class MainClass {

        /**
         * main method starting point of this program
         * @param args parameters for this program
         */
        public static void main(String [] args)
        {

                Scanner scan = new Scanner(System.in);
                Random rand = new Random();

                Log.Generate("simCopy.in"); // log file for the topology

                // start read in system configuration in the order
                // of number of readers, height of the readers,
                // speed of the conveyor belt, and then the distance
                // array for distances of adjacent readers
                if(scan.hasNextInt())
                        Global.numReaders = scan.nextInt(); // number of readers
                else{
                        System.out.println("Error: read in num of readers");
                        System.exit(0);
                }
                Log.write("" + Global.numReaders);

                if(scan.hasNextDouble())
                        Global.height = scan.nextDouble();      // height of readers
                else{
                        System.out.println("Error: read in height of readers");
                        System.exit(0);
                }
                Log.write("" + Global.height);

                if(scan.hasNextDouble())
                        Global.velocity = scan.nextDouble();// velocity of conveyor belt
                else{
                        System.out.println("Error: read in velocity of conveyor belt");
                        System.exit(0);
                }
                Log.write("" + Global.velocity);

                if(scan.hasNextDouble())
                        Global.distanceBase = scan.nextDouble();// velocity of conveyor belt
                else{
```

```java
            System.out.println("Error: read in distance base");
            System.exit(0);
        }
        Log.write("" + Global.distanceBase);

        Global.distance = new double[Global.numReaders-1];

        for(int i=0; i<Global.numReaders-1; i++)
        {
            if(Global.uniform)
                Global.distance[i] = Global.distanceBase;
            else
            {
                double lowerBound = Math.sqrt(Math.pow(Reader.MAXL,2)-
                                                Math.pow(Global.height, 2));
                Global.distance[i] = lowerBound + lowerBound*rand.nextDouble();
            }
        } // end read in topology data
        for(int i=0; i<Global.numReaders-1; i++)
            Log.write("Distance[R"+i+",R"+(i+1)+"]: " + Global.distance[i]);
        Log.close();

        // summary of the system topology
        System.out.println("Number of readers:\t" + Global.numReaders);
        System.out.println("Height of reader:\t" + Global.height);
        System.out.println("Velocity of belt:\t" + Global.velocity);
        for(int i=0; i<Global.numReaders-1; i++)
            System.out.println("Distance[R"+i+",R"+(i+1)+"]:\t" +
                                                Global.distance[i]);
        // end of summary

        for(int i=0; i<Global.numReaders; i++)
        {
            Reader obj = new Reader(i);
            Global.readerGroup.addElement(obj); // Reader id range 0~(numReader-1)
        }
        if(Global.uniform) // UD scenario
        {
            // the minimum power reader have to make reader range at least h
            double powerMin = (Global.height * Reader.MAXP)/Reader.MAXL;
            Global.eff_time_uniform = 0;

            while(powerMin<Reader.MAXP)
            {
                updateReaders(powerMin);
                double temp = calculate_uniform();
                Global.results.add(powerMin+":"+temp);
                if(temp > Global.eff_time_uniform)
                    Global.eff_time_uniform = temp;
                powerMin += 0.01;
            }
            System.out.println("maximum efficient time:\t" +
                                    Global.eff_time_uniform/Global.velocity);
            printResult();
        }
```

```
            else // NUD scenario
            {
                System.out.println("efficient_time_for_sce#1(no_gap):\t" +
                                                    Algo.scen1()/Global.velocity);
                System.out.println("efficient_time_for_sce#2(with_gap):\t" +
                                                    Algo.scen2()/Global.velocity);
            }
    } // end of method main

    /**
     * This method results to a file named results_uniform
     */
    static void printResult()
    {
        Log.Generate("results_uniform");
        for(int i=0; i<Global.results.size(); i++)
            Log.write(Global.results.elementAt(i));
        Log.close();
    }

    /**
     * This method update all reader's some profile given a new power level.
     * @param powerParam the new power of a reader
     */
    static void updateReaders(double powerParam)
    {
        double readDistance = (powerParam * Reader.MAXL) / Reader.MAXP;
        double lane = 2*Math.sqrt(Math.pow(readDistance, 2)-
                                                Math.pow(Global.height, 2));
        if(lane>Global.distanceBase) // there is overlap between neighbor readers
            overlap(readDistance, lane);
        else // there is no overlap between neighbor readers
            overlapZero(lane);
    }

    /**
     * This method update some of reader's profile and overlap of its neighbors
     * (if have any) if there is overlap between given reader and its neighbor.
     * @param d new reading distance of a reader
     * @param lane new total range of a reader
     */
    static void overlap(double d, double lane)
    {
        double temp = 2*Math.sqrt(Math.pow(d,2)-Math.pow(Global.height,2))-
                                                        Global.distanceBase;
        for(int i=0; i<Global.readerGroup.size(); i++)
        {
            Global.readerGroup.elementAt(i).lane = lane;
            Global.readerGroup.elementAt(i).leftOverlap = temp;
            Global.readerGroup.elementAt(i).rightOverlap = temp;

            if(i==0){       // leftmost reader
                Global.readerGroup.elementAt(i).leftOverlap = 0;
                Global.readerGroup.elementAt(i).effectiveLane = lane-temp;
            }
```

```
                        else if(i==(Global.readerGroup.size()-1)){      // rightmost reader
                                Global.readerGroup.elementAt(i).rightOverlap = 0;
                                Global.readerGroup.elementAt(i).effectiveLane = lane-temp;
                        }
                        else    // middle readers
                                Global.readerGroup.elementAt(i).effectiveLane = lane-2*temp;
                }
        }

        /**
         * This method update some of reader's profile when no overlap between itself
         * and its neighbors.
         * @param lane new total range of a reader
         */
        static void overlapZero(double lane)
        {
                for(int i=0; i<Global.readerGroup.size(); i++)
                {
                        Global.readerGroup.elementAt(i).leftOverlap = 0;
                        Global.readerGroup.elementAt(i).rightOverlap = 0;
                        Global.readerGroup.elementAt(i).effectiveLane = lane;
                }
        }

        /**
         * This method calculate the sum of all readers' effective lanes.
         * @return value total effective lane of the system
         */
        static double calculate_uniform()
        {
                double value=0;
                for(int i=0; i<Global.readerGroup.size(); i++)
                        value += Global.readerGroup.elementAt(i).effectiveLane;
                return value;
        }
}
```

VITA

Xiaodan Fang

Candidate for the Degree of

Master of Science

Thesis: Power Control for Optimizing RFID Tag Reading Rate in Multi-Reader Environment: A Study on Conveyor Belt System

Major Field: Computer Science

Biographical:

Personal Data: Born in Xinyang, Henan, China on May 23, 1983.

Education:

Received the B.S. degree from St. Francis Xavier University, Antigonish, NS, Canada, 2007, in Computer Science.
Completed the requirements for the degree of Master of Science with a major in Computer Science at Oklahoma State University in December, 2009.

Experience:

Programmer at CCOK Inc., Tulsa, May 2008 - August 2008.
Research Assistant, OSU Department of Computer Science, Stillwater, January 2008 - May 2008.
Teaching Assistant, Department of Computer Science in Oklahoma State University, Stillwater, January 2008 - Present.
Research Assistant, St. Francis Xavier University, Canada, May 2006 - September 2006. Computer Lab Assistant, St. Francis Xavier University, Canada, September 2006 - May 2007.
Television Operator, EastLink Television Inc., Canada, December 2006 - July 2007.

Name: Xiaodan Fang Date of Degree: December, 2009

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: Power Control for Optimizing RFID Tag Reading Rate in Multi-Reader Environment: A Study on Conveyor Belt System

Pages in Study: 67 Candidate for the Degree of Master of Science

Major Field: Computer Science

Radio Frequency Identification (RFID) systems provide a mechanism to identify and track tagged objects using radio waves. It is an alternative for existing "bar code" identification of objects. However, to make the RFID system more efficient, most systems only use one or two readers to scan tagged items, which is far from enough. Hence many enterprises use multiple readers to achieve high performance. Other than that, companies increasing make use of multiple readers in conveyor belt systems. Some airports have conveyor belt systems utilizing RFID techniques to scan checked baggage for efficiently tracing suitcases of passengers.

The main objective of this thesis is to analyze how to achieve the best passive RFID tag reading performance in a supply chain system (specifically, we focus on the conveyor belt system), in which case each tag would have the maximum time in the non-interference region in a dense reader environment. Two situations will be considered in this thesis. The first one is when the distances between neighboring readers are constant. The other is when the distances between neighboring readers are uniformly randomly distributed. A detailed analysis is performed on the above two cases in this thesis.

ADVISOR'S APPROVAL: Dr. Venkatesh Sarangan