RFID Tag Identification Protocol Implementing
Threshold-Based Dynamic Framed Slotted Aloha Policy

Submitted by

Ghassan Shaheen
(M.Sc. Eng., University Putra Malaysia, Malaysia, 2000)
(B.Sc. Eng., Jordan University of Science and Technology, Jordan, 1996)

A thesis submitted to the Faculty of Graduate
Studies and Research in partial fulfillment of the
requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Department of System and Computer Engineering
Ottawa-Carleton Institute for Electrical and Computer Engineering
Carleton University
Ottawa, Ontario, K1S 5B6

December, 2010

Your file Votre référence
ISBN: 978-0-494-79550-7
Our file Notre référence
ISBN: 978-0-494-79550-7

Abstract

In Radio Frequency Identification (RFID) system, one of the problems that have to be solved is the collision between tags which lowers the efficiency of the RFID system. There are different proposed algorithms to solve this problem. One of the popular anti-collision algorithms is ALOHA-type algorithms, which are simple and show good performance when the number of tags to read is small. However, they generally require exponentially increasing number of slots to identify the tags as the number of tag increases. In this thesis, we propose a new anti-collision algorithm based on Dynamic Framed Slotted Aloha (DFSA) called Threshold base DFSA (THDFSA), which adjusts the number of slots in the frame based on the number of remaining tags to be identified to give the optimal total number of slots required to identify all the tags. As a result, in the proposed method, the number of slots to read the tags increases linearly as the number of tags does. Simulation results show that the proposed algorithm reduces the total number of slots to identify the tags in comparison to the conventional algorithms for various numbers of tags.

Acknowledgments

I would like to express my grateful thanks to my supervisor Dr Jerome Talim, for his keen interest, guidance, encouragement, and support through out my research study, without which this work would not have been possible. I am grateful to him for his patience and feedback, which helped me to improve the quality of my work.

I wish also to express my sincere appreciation to my family members for their support. The encouragement words of my father and the prayers of my mother helped me achieve the milestones related to this work. I must also thank my wife for her support and understanding through out the long hours of my study. Last but not least, I dedicate this achievement to the soul of my sister who passed away at an early stage of my study.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION **10**

1.1 THESIS OBJECTIVE 11
1.2 THESIS CONTRIBUTION 12
1.3 THESIS ORGANIZATION 12

CHAPTER 2

RFID BACKGROUND **13**

2.1 FUNDAMENTALS OF RFID 13
2.2 RFID SYSTEM BASICS 14
2.3 ADVANTAGES OF RFID SYSTEM 16
2.4 RFID COMPONENTS 18
2.4.1 Reader or base station *18*
2.4.2 Tags *18*
2.4.3 Information system *19*
2.5 RFID TECHNOLOGY AND STANDARDS 19
2.5.1 Energy Source: Passive or active? *19*
2.5.2 Frequency *21*
2.5.3 Memory *24*
2.5.4 Standards *24*
2.6 RFID APPLICATIONS 25
2.6.1 Asset Tracking *25*
2.6.2 Manufacturing *26*
2.6.3 Supply Chain Management *26*
2.6.4 Retailing *26*
2.6.5 Payment Systems *26*
2.6.6 Security and Access Control *27*

CHAPTER 3

RFID COLLISION TYPES AND ANTI-COLLISION PROTOCOLS **28**

3.1 COLLISION TYPES 28
3.1.1 Tag-Tag Collision *28*
3.1.2 Reader Tag Collision *29*
3.1.3 Reader-Reader Collision *30*
3.2 RFID ANTI-COLLISION PROTOCOLS 31
3.2.1 Tree Based-Deterministic *33*
3.2.2 Aloha Based –Probabilistic Anti-Collision Protocols *37*

CHAPTER 4

THRESHOLD-BASED DYNAMIC FRAMED SLOTTED ALOHA POLICY **43**

4.1 MARKOV DECISION PROCESS (MDP) 44

4.2 FRAME SIZE CONTROL PROCESS 46
4.3 TRANSITIONAL PROBABILITY CALCULATION 49
4.4 TRANSITIONAL PROBABILITY CALCULATION EXAMPLE 51
4.5 THRESHOLD BASE DFSA ANALYSIS 59
4.5.1 General Approach Description 59
4.5.2 Two-Threshold Policy 61

CHAPTER 5

PERFORMANCE ANALYSIS AND RESULTS 64

5.1 SIMULATION ENVIRONMENT 64
5.2 FINDING THE OPTIMUM FRAME SIZE 65
5.3 TAG IDENTIFICATION PROCESS 70
5.4 PERFORMANCE COMPARISON 74

CHAPTER 6

CONCLUSION AND FUTURE WORK 78

6.1 CONCLUSION 78
6.2 FUTURE WORK 79

BIBLIOGRAPHY 80

LIST OF FIGURES

Figure 1: RFID System 14
Figure 2: RFID Functional Blocks 15
Figure 3: Tag Identification of Tree-Based Protocols. [2] (a) Tree expression of tag identification. (b)Tag identification of the binary tree protocol. 34
Figure 4: Examples of the Operation of Slotted ALOHA, Framed Slotted ALOHA, and Adaptive Framed Slotted ALOHA. [2] 38
Figure 5: The process of BFSA algorithm [2] 39
Figure 6: Frame size transition diagram 47
Figure 7: Example for state transitional probability calculations 53
Figure 8: Example for state transitional probability calculations (continued) 57
Figure 9: Two-Threshold Policy 60
Figure 10: Best frame size for medium number of tags, 20 time simulation 66
Figure 11: Best frame size for low number of tags, 20 time simulation 67
Figure 12: Best frame size for high number of tags, 20 time simulation 68
Figure 13: Best frame size for low number of tags, 50 time simulation 69
Figure 14: Best frame size for low number of tags, 50 time simulation 69
Figure 15: Best frame size for low number of tags, 100 time simulation 70
Figure 16: Comparison of FSA, Binary Tree, DFSA, Advanced DFSA, and THDFSA for low number of tags 76
Figure 17: Comparison of FSA, Binary Tree, DFSA, Advanced DFSA, and New THDFSA for large number of tags 76

LIST OF TABLES

Table 1: Tags Structure Classes 20
Table 2: RFID Operating Frequencies and Associated Characteristics 23
Table 3: Finding the best frame size to identify different number of tags 66
Table 4: Best frame size for (a) Low number of tags (b) Medium number of tags (c) High number of tags 68
Table 5 (a), (b), (c), (d): Different frame size for the number of remaining tags = 0.9*n 72
Table 6 (a), (b): Different frame size for the number of remaining tags = 0.6*n 72
Table 7: Simulation results of THDFSA algorithm 73
Table 8: Simulation results of other anti-collision protocols 74
Table 9: Number of slots needed using THDFSA for different simulation runs 77

ABBREVIATIONS AND ACRONYMS

RFID	Radio Frequency Identification
RTF	Reader Talks First
RF	Radio Frequency
IC	Integrated Circuit
EPC	Electronic Product Code
UHF	Ultra High Frequency
ISM	Industrial, Scientific and Medical
WORM	Write Once Read Many
TWA	Tree Walking Algorithm
STAC	Slotted Termination Adaptive Collection
BT	Binary Tree
DFSA	Dynamic Framed Slotted Aloha
ADFSA	Advance Dynamic Framed Slotted Aloha
MDP	Markov Decision Process
THFSA	Threshold Framed Slotted Aloha

Chapter 1

Introduction

Radio Frequency Identification (RFID) is the most reliable way to electronically identify, control, and track inventory items using RF communication. Today RFID systems are invisible to the users. The basic RFID system consists of a Reader and Transponders. The Reader or Transceiver is the unit acting as the master and supplies the RFID transponder with energy and triggers the communication signals to force the transponder to execute the requested action. The reader control can be either via a computer terminal or the automated execution of program scripts. In stationary installations, fixed readers are connected to power and communication lines, whereas in mobile applications, hand held readers are used. For further data exchange, the reader may be connected to a host computer or database [4].

The Transponder or Tag is the identification device which is located on the item to be identified. The tag mostly acts as a slave and relies on the reader to activate it using the "Reader Talks First" (RTF) concept. The reader supplies energy via the RF field and transmit requests/commands to instruct the tag about the action to be executed. The tag receives and decodes RF signals coming from the reader, executes the instructed action, and may respond with data or status information. An RFID reader recognizes an object through reading the identification number (ID) of the RFID tag attached to it [2]. To read tag IDs, the reader sends out a signal supplying instructions to tags. The tag transmits its own ID to the reader, and then the reader consults an external database with the ID to recognize the object. RFID is fast replacing bar code-based identification mechanisms because (1) communication between a reader and a tag is not

limited by the requirement of ''line-of-sight'' reading and (2) each tag is allowed to have a unique ID. Reader transmissions or tag transmissions lead to collision because readers and tags operate within the same frequency band due to cost considerations. Collisions make both communication overhead and transmission delay often lose their usefulness. As a result, either the reader may not recognize all objects or retransmissions are required for successful identification. Especially, since low-functional passive tags can neither detect collisions nor figure out neighboring tags, tag collision gives rise to the need for a tag anti-collision protocol that enables the recognition of tags with few collisions, and also executes in real time.

So far, several tag anti-collision algorithms have been proposed. Among them, the most widely used ones are framed slotted ALOHA algorithm and binary search algorithm. Due to its simple implementation, framed slotted ALOHA algorithm is used frequently [2]. This thesis contributes to the literature on identifying of tags with few collisions.

1.1 **Thesis Objective**

The objective of this thesis is to reduce the number of slots needed to identify tags in RFID system, by optimizing the initial frame size and modifying it based on the number of remaining tags. By reducing the number of slots, the total time required to identify all the tags will be reduced, and therefore improves the system efficiency.

1.2 **Thesis Contribution**

We developed an algorithm for reducing the number of slots required to identify the tags in an RFID system, the process is divided into two parts, first we found the optimum initial frame size to identify any given number of tags, and secondly, we developed an algorithm to modify the frame size based on number of remaining tags such that the number of collisions is minimum.

1.3 **Thesis Organization**

The rest of this thesis is organized as follows. Chapter 2 presents a formal review of the literature related to the fundamentals, components, technology and standards of RFID system. Chapter 3 provides a high level description of RFID collision types. Chapters four cover the RFID anti-collision protocols available in the literature, the focus is on framed slotted aloha anti-collision protocols. Chapter five described the work we have done, firstly we explain a dynamic process to control the frame size, and then we detail the heuristic process model for the threshold based anti-collision protocol. Chapter six demonstrates the performance analysis and results of the proposed algorithm and compares it with the solutions available in the literature. Chapter seven indicates our thoughts on the future direction of this research; and concludes the thesis

Chapter 2

RFID Background

In this literature, the fundamentals of Radio Frequency Identification (RFID) are reviewed, the focus of the review is on RFID anti-collision protocols, several anti-collision algorithms and their performances are discussed.

2.1 **Fundamentals of RFID**

RFID is the most reliable way to electronically identify, data capture, control, and track inventory items using RF communication. Today RFID is having a very broad use but most of the time such systems are invisible or are not recognized by the users.

The basic RFID system consists of a Reader and a Transponder. The Reader or Transceiver is the unit acting as the master and supplies the RFID transponder with energy and triggers the communication signals to force the transponder to execute the requested action. The reader control can be either via a computer terminal or the automatic execution of program scripts. In stationary installations, fixed readers are connected to power and communication lines, whereas in mobile applications, hand held readers are used. For further data exchange, the reader may be connected to a host computer or database as shown in figure 1, [4].

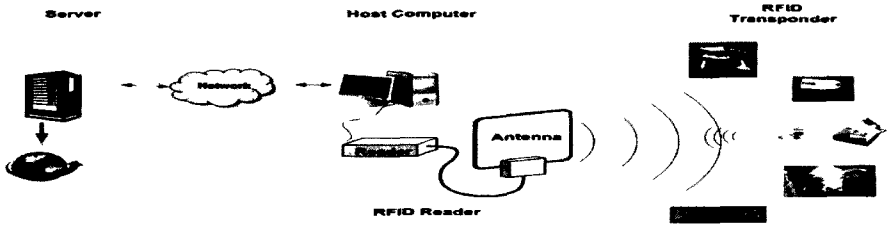

Figure 1: RFID System

The Transponder or Tag is the identification device which is located on the item to be identified. Most RFID transponders are without an internal power source (battery) and are called passive transponders. The power supply of a tag is the RF field generated by the reader. The tag generates its own supply voltage by rectifying the induced voltage from the Reader's RF signal. Active transponders have an integrated power source (internal battery) and behave the same way as passive devices but with increased performance. These tags are using the battery to supply the circuitry and to generate the response data. Their activation is mostly triggered by the reader signal.

Now RFID systems are widely used in applications with the primary task to identify items, but there are also new applications where higher security and computation as well as integrated sensors and actors are required. Due to the current cost structure of RFID systems, new application fields can be justified based on return of investment.

2.2 **RFID System Basics**

Transponders basically operate as active or passive devices. The functionality of both types is similar; the main difference is the increased performance in view of communication distance and computation capabilities of the active vs. the lower cost of the passive transponders. The integrated battery increases the cost of the transponder, limits the tag's life time, causes environmental issues over disposal, and limits the form factor and thickness of the tag. These disadvantages of the active transponders limit the applications where these tags can be used.

The Tag mostly acts as a slave and relies on the reader to activate it using the "Reader Talks First" (RTF) concept. The reader supplies energy via the Radio Frequency (RF) field and transmit requests/commands to instruct the tag about the action to be executed. The tag receives and decodes RF signals coming from the reader, executes the instructed action, and may respond with data or status information. The cost structure of the tag can be roughly split in costs for IC, antenna, assembly, and test. The electronics part or Integrated Circuit (IC) of the tag consists of some basic functional modules which are used to enable certain functionality as shown in figure.2:

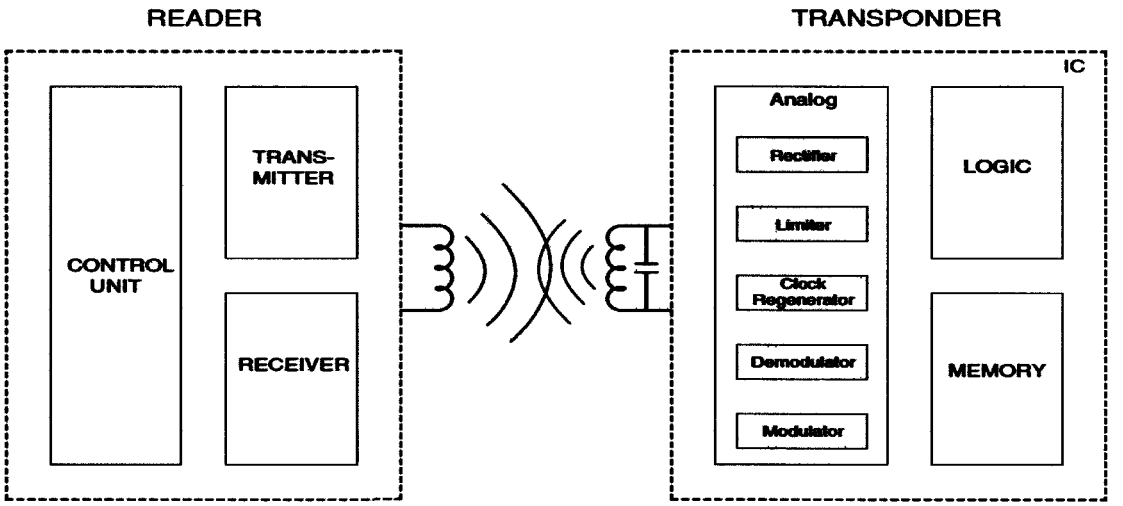

Figure 2: RFID Functional Blocks

- The induced voltage is rectified by the Rectifier to supply the IC with energy.
- The Limiter limits the RF voltage at the inputs pins to avoid over voltage which would destroy the circuitry.
- The Clock Regenerator extracts the frequency signal from the RF signal which is used as an internal clock.

- The Demodulator decodes the incoming data signal and generates a binary bit stream representing the command and data to be executed. These data are used by the IC to execute the requested activities.
- The Modulator modulates the decoded response data.
- The Logic part represents the microcontroller or digital circuitry of the tag.
- The Memory unit (mostly EEPROM) contains the tag specific data as well as additional memory where application specific data can be programmed.

The Reader consists of a control unit and the radio RF unit containing the transmitter and the receiver modules. In the control unit, the firmware and the hardware is implemented to control the reader activities such as communications with a host computer and the tag, as well as data processing. The transmitter generates the RF signal (frequency and power level) which is connected to the antenna resonance circuit. The receiver part receives the RF signal generated by the tag, demodulates and decodes the data, and sends the binary data to the control unit for further processing.

2.3 **Advantages of RFID System**

Currently, a revolution is occurring in RFID technology, and many companies create new implementations of RFID systems and new products related to this technology daily.

The main advantage of RFID technology is the automated identification and data capture that promises wholesale changes across a broad spectrum of business activities and aims to reduce the cost of the already used systems such as barcodes. For this reason, although RFID technology

was discovered many years ago, it has advanced and evolved only during the last decade since cost has been the main limitation in all implementations.

The advantages of the RFID systems in relation to other identification systems currently in use and especially barcode can be summarized as follows:

- Battery-less. Supply voltage derived from the RF field
- No line-of-sight required for the communication
- Large operating and communication range
- Read and Write capability of the transponder memory
- High communication speed
- High data capacity (user memory)
- High data security
- Data encryption/authentication capability
- Durability and reliability
- Resistant to environmental influence
- Reusability of the transponder
- Hands free operation
- Very low power
- Multiple simultaneous scans of items which reduce the time needed to collect the data.
- RFID systems can be used to track people and animals in real time, while this cannot be done with barcodes.
- A barcode is the same for all similar items, while with RFID technology; the same items can have different data, such as a different expiration date.

2.4 **RFID Components**

RFID wireless systems generally consist of three parts:

- Reader or base station: a fixed or mobile interrogator which communicates with tag.
- Tag: a small mobile communication circuit embedded on radiating element.
- Information system: a data base that gathers the information to be processed

2.4.1 Reader or base station

The RFID interrogator is the component of the system that facilitates and initiates communication with the tags. This module consists of both the transmission antenna as well as the receiving antenna. Through the transmission antenna a continuous wave RF signal is broadcast into the sensing environment. The tags then modulate this signal and reflect it back towards the interrogator where it is captured by the receiving antenna. Inside the interrogator, special hardware and software decode the received signal and extract the desired information [7].

2.4.2 Tags

There are two main components present in the RFID tag. Firstly, a small silicon chip or integrated circuit which contains a unique identification number (ID). Secondly, an antenna that can send and receive radio waves [8]. These two components can be tiny: the antenna consists of a flat, metallic conductive coil, and the chip is potentially less than half a millimeter. These two components are usually attached to a flat plastic tag that can be fixed to a physical item. These tags can be quite small, thin and, increasingly, easily embedded within packaging, plastic cards, tickets, clothing labels, pallets and books. There are two main types of tags: passive and active. Passive tags are currently the most widely deployed as they are the cheapest to produce.

2.4.3 Information system

Information system is considered by many to be the heart and soul of a comprehensive RFID system. The transfer of data between tags and readers is electronic. It's the software that allows us to actually tie electronic identity to production and management information, and share the information with others.

2.5 **RFID Technology and Standards**

The radio frequency part of RFID is the communication medium between tags and readers. With passive RFID tags, radio frequency is also used to deliver power to the tag, as they do not have on-board power systems [9].

RFID systems are designed to be asymmetric: readers are expensive and power hungry, whilst tags are cheap and require comparatively low levels of energy. In addition, there are three key elements that need to be borne in mind in any discussion of RFID systems: energy source (which determines if a tag is passive or active), frequency and memory.

2.5.1 Energy Source: Passive or active?

RFID tags come in a variety of different types according to their functionality, and these types have been defined in an RFID Class Structure by the Auto-ID Center (and later through Electronic Product Code Global (EPC Global) which has been subsequently refined and built on [18]. The basic structure defines five classes in ascending order as shown in Table 1.

Class	Class Layer Name	Functionality
1	Identity Tags	Purely passive, identification tags
2	Higher Functionality Tags	Purely passive, identification + some additional functionality (e.g. read/write memory)
3	Semi-Passive Tags	Addition of on board battery power
4	Active Tags	Communication with other active tags
5	Reader Tags	Able to provide power for and communicate with other tags, i.e. can act as a reader, transmitting and receiving radio waves

Table 1: Tags Structure Classes

Passive Tag Systems do not have an on-board power source so they have to obtain power from the reader in order to run the digital logic on the chip and issue a response to the reader. They can therefore only operate in the presence of a reader. The communication range is limited by the need for the reader to generate very strong signals to power the tag, which therefore limits the reader-to-tag range. In addition, the small amount of energy that the tag is able to harvest in order to power its response to the reader, means that the tag to reader range is also limited (to around four or five meters in UHF). However, as passive tags do not require a continuous power source they have a much longer lifecycle, and because of their minimal on-board circuitry they are much cheaper to produce. This means that passive RFID tags are more suitable for tagging individual product items for applications such as supermarket checkouts and smart cards. Semi-passive Tag Systems require the tag to use battery power for the digital logic on the chip, but still use harvested power for communication [9].

Semi passive tags are far more reliable and have greater read ranges than purely passive tags, but they also have shorter lives due to their reliance on battery power, are more fragile, and are significantly more expensive.

Active Tag Systems have an active RF transmitter (i.e. they are capable of peer-to-peer communication) and the tags use batteries to power the logic chip and to communicate with the reader (i.e. they do not use harvested power). Read range increases up to several kilometers and reliability improves; active tags can be read while moving at up to 100 miles an hour (e.g. in automatic toll-road payment systems) and the readers are capable of reading up to a thousand tags per second. Active tags can also be equipped with built-in sensors e.g. for monitoring temperature control and reporting unacceptable fluctuations on refrigerated products whilst in transit. They also have a much larger memory than passive tags and, due to their higher processing capabilities, are also more secure.

2.5.2 Frequency

RFID is fundamentally based on wireless communication, making use of radio waves, which form part of the electromagnetic spectrum (i.e. frequencies from 300 kHz to 3 GHz). It is not unlike two other wireless technologies, WiFi and Bluetooth. The three technologies are all designed for very different uses and therefore have different functionalities but there is shared ground between the three, with some hybrids starting to appear. RFID systems can utilize both WiFi and Bluetooth and need not see them as competitors.

RFID operates in unlicensed spectrum space, sometimes referred to as ISM (Industrial, Scientific and Medical) but the exact frequencies that constitute ISM may vary depending on the regulations in different countries. These operating frequencies are generally considered to be organized into four main frequency bands and Table 2 shows these different radio wave bands and the more common frequencies used for RFID systems [19]

Within a given frequency band the actual real-world communication range will vary widely depending on factors such as the operating environment, the detail of the antenna design and the available system power [20, 21].

Band	Low Frequency (LF)	High Frequency (HF)	Ultra High Frequency (UHF)	Microwave
Frequency	30-300 KHz	3-30 MHz	300 MHz-3 GHz	2-30 GHz
Typical RFID frequencies	125-134 KHz	13.56 MHz	433 MHz or 865-956 MHz 2.45 GHz	2.45 GHz
Approximate read Range	Less than 0.5 meter	Up to 1.5 meters	433 MHz = up to 100 meters 865-956 MHz = 0.5 to 5 meters	Up to 10m
Typical data transfer rate	Less than 1 Kbit/s	Approximately 25 Kbit/s	433-956 = 30Kbit/s 2.45 = 100 Kbit/s	Up to 100 Kbit/s
Characteristics	Short-range, low data transfer rate, penetrates water but not metal	Higher ranges, reasonable data rate, penetrates water but not metal	Long ranges, high data transfer rate, cannot penetrate water or metal	Long ranges, high data transfer rate, cannot penetrate water or metal
Typical use	Animal ID Car immobilizer	Smart label Contactless travel cards Access and security	Special animal tracking Logistics	Moving vehicle toll

Table 2: RFID Operating Frequencies and Associated Characteristics

There are two types of RFID system, each using different physical properties to enable communication between the reader and the tag [22]. The physics employed can become complex, but it is important to realize that it partly determines the operating range of the systems. RFID systems based on LF and HF frequencies make use of near field communication and the physical property of inductive coupling from a magnetic field. The reader creates a magnetic field between the reader and the tag and this induces an electric current in the tag's antenna,

which is used to power the integrated circuit and obtain the ID. The ID is communicated back to the reader by varying the load on the antenna's coil which changes the current drawn on the reader's communication coil. RFID systems based on UHF and higher frequencies use far field communication and the physical property of backscattering or reflected power. Far field communication is based on electric radio waves: the reader sends a continuous base signal frequency that is reflected back by the tag's antenna. During the process, the tag encodes the signal to be reflected with the information from the tag using a technique called modulation (i.e. shifting the amplitude or phase of the waves returned).

2.5.3 Memory

Tags come in a variety of forms with varying types of on-chip memory capability. Tags can be read-only (the unique ID code is permanently stored on the tag, also known as WORM: Write Once Read Many), read/write (allowing a user to change the ID and add additional data to the tag's memory), or they can be a combination, with a permanent tag ID and some storage space for the user's data. Passive tags typically have anywhere from 64 bits to 1 kilobyte of non-volatile memory. Active tags tend to have larger memories with a range of, typically, between 16 bytes and 128 kilobytes [20, 23]

2.5.4 Standards

The number and use of standards within RFID and its associated industries is quite complex, involves a number of bodies and is in a process of development. Standards have been produced to cover four key areas of RFID application and use:[24]

- Air interface standards (for basic tag-to-reader data communication),

- Data content and encoding (numbering schemes),
- Conformance (testing of RFID systems)
- Interoperability between applications and RFID systems

There are several standards bodies involved in the development and definition of RFID technologies including:

- International Organisation of Standardisation (ISO)
- EPC global Inc.
- European Telecommunications Standards Institute (ETSI)
- Federal Communications Commission (FCC)

2.6 **RFID Applications**

RFID has a wide and growing range of potential uses throughout industry, commerce, education and the public sector more widely. The main driver for the development of the technology is the capability to identify and track the movement of products through the supply chains. Current and potential uses of RFID include:

2.6.1 Asset Tracking

It's no surprise that asset tracking is one of the most common uses of RFID. Companies can put RFID tags on assets that are lost or stolen often, that are underutilized or that are just hard to locate at the time they are needed. Just about every type of RFID system is used for asset management.

2.6.2 Manufacturing

RFID has been used in manufacturing plants for more than a decade. It's used to track parts and work in process and to reduce defects, increase throughput and manage the production of different versions of the same product.

2.6.3 Supply Chain Management

RFID technology has been used in closed loop supply chains or to automate parts of the supply chain within a company's control for years. As standards emerge, companies are increasingly turning to RFID to track shipments among supply chain partners.

2.6.4 Retailing

Retailers such as Best Buy, Metro, Target, Tesco and Wal-Mart are in the forefront of RFID adoption. These retailers are currently focused on improving supply chain efficiency and making sure product is on the shelf when customers want to buy it.

2.6.5 Payment Systems

RFID is all the rage in the supply chain world, but the technology is also catching on as a convenient payment mechanism. One of the most popular uses of RFID today is to pay for road tolls without stopping. These active systems have caught on in many countries, and quick service restaurants are experimenting with using the same active RFID tags to pay for meals at drive-through windows.

2.6.6 Security and Access Control

RFID has long been used as an electronic key to control who has access to office buildings or areas within office buildings. The first access control systems used low-frequency RFID tags. Recently, vendors have introduced 13.56 MHz systems that offer longer read range. The advantage of RFID is it is convenient (an employee can hold up a badge to unlock a door, rather than looking for a key or swiping a magnetic stripe card) and because there is no contact between the card and reader, there is less wear and tear, and therefore less maintenance.

As RFID technology evolves and becomes less expensive and more robust, it's likely that companies and RFID vendors will develop many new applications to solve common and unique business problems

Chapter 3

RFID Collision Types and Anti-Collision Protocols

3.1 Collision Types

RFID is an automatic identification system that consists of two components; readers and tags. By reading all the tag IDs in the neighborhood and then consulting a backend database that provides a mapping between IDs and objects, the reader learns about the existence of corresponding objects in the neighborhood. This way RFID reader also acts as identification and/or proximity sensor. Simultaneous transmissions in RFID systems lead to collisions as the readers and tags typically operate on the same channel. Three types of collisions are possible.

3.1.1 Tag-Tag Collision

Tag-tag collision occurs when multiple tags respond to the same reader simultaneously. Due to multiple signals arriving at the same time, the reader may not be able to detect any tag. This problem prevents the reader from detecting all tags in its interrogation zone. A popular solution to this problem is the Tree Walking Algorithm (TWA) [10], which is generally used in UHF readers. In this protocol, the reader splits the entire ID space into two subsets and tries to identify the tags belonging to one of the subsets, recurring along the way until a subset has exactly one tag or no tags at all.

Due to larger turn around times at lower frequencies, TWA is not deemed suitable for HF readers. Instead, the HF readers use a slotted termination adaptive collection (STAC) protocol

[10] somewhat similar to the framed Aloha protocol. In STAC, tags respond at randomly selected slots whose beginning and end are controlled by the reader. The reader sends a begin round command with the number of slots in the round. Tags that are energized by the reader select a random slot number as the proposed reply slot and set their states to slotted read and counters to zero. This counter advances each time the reader sends an end slot command. A tag sends its response to the reader when its counter reaches the proposed reply slot. If the reader does not hear any tag in a slot, it sends a close slot command, which causes all tags to increment their counters. If the reader receives a response correctly, it closes the slot by issuing a x slot command which makes all tags to increment their counters and prompts the tag that was correctly heard to go into the xed slot state, after which the tag responds at this same slot in each round. If however, the reader hears a collision, it sends a close slot command forcing all tags to increment their counters, while those tags that had responded in this slot, realize that there was a collision since they did not receive the x slot command, and thus they select another slot for transmission.

3.1.2 Reader Tag Collision

Reader-tag collision occurs when the signal from a neighboring reader interferes with tag responses being received at another reader. This problem has been studied in the EPC-Global Class1 Gen1 and Gen2 standards for UHF readers [11, 12]. In Gen 1 standard, the reader-tag collision problem is mitigated by allowing frequency hopping in the UHF band or by time division multiple access. In Gen 2 the readers and tags operate on different frequencies so that the tag response does not interfere or collide with reader signals. Either solution requires fairly sophisticated technology.

3.1.3 Reader-Reader Collision

A reader-reader collision occurs when a tag hears multiple readers at the same time. In this situation, the tag might be unable to respond to any reader at all. Colorwave [10] is one of the first works to address reader-reader collisions. In particular, it considers an interference graph over the readers, wherein there is an edge between two readers if they could lead to a reader-reader collision when transmitting simultaneously, and tries to randomly color the readers such that each pair of interfering readers have different colors. If each color represents a time slot, then the above coloring should eliminate reader-reader collisions. If conflicts arise (i.e., two interfering readers pick the same color or time slot), only one of them wins (i.e. sticks to the chosen color), the others pick another color again randomly. In [13], the authors suggest coloring of the interference graph using k colors, where k is the number of available channels. If the graph is not k-colorable using their suggested heuristic, then the authors suggest removal of certain edges and nodes from the interference graph using other heuristics which consider the size of the common interference regions between neighboring readers

3.2 **RFID Anti-Collision Protocols**

RFID is an automatic identification system which consists of readers and tags. An RFID reader recognizes an object through reading the identification number (ID) of the RFID tag attached to it [2]. To read tag IDs, the reader sends out a signal supplying instructions to tags. The tag transmits its own ID to the reader, and then the reader consults an external database with the ID to recognize the object. RFID is fast replacing bar code-based identification mechanisms because (1) communication between a reader and a tag is not limited by the requirement of ''line-of-sight'' reading and (2) each tag is allowed to have a unique ID.

Reader transmissions or tag transmissions lead to collision because readers and tags operate within the same frequency band due to cost considerations. Collisions are divided into reader collisions and tag collisions [2]. When neighboring readers interrogate a tag simultaneously reader signals collide and the tag cannot decode any reader signal. On the contrary, when multiple tags transmit IDs to a reader at the same time, tag signals collide and tag collision prevents the reader from recognizing any tag. Collisions make both communication overhead and transmission delay often lose their usefulness. As a result, either the reader may not recognize all objects or retransmissions are required for successful recognition. Especially, since low-functional passive tags can neither detect collisions nor figure out neighboring tags, tag collision gives rise to the need for a tag anti-collision protocol that enables the recognition of tags with few collisions, and also executes in real time.

Tag anti-collision protocols can be grouped into two broad categories, namely ALOHA based protocols and tree-based protocols. ALOHA-based protocols [5, 6, 11, 12, 14, 15, 16]

reduce the occurrence probability of tag collisions since each tag tries to transmit the ID at randomly selected time. ALOHA-based protocols, however, cannot completely prevent collisions, and hence they have the serious problem that a specific tag may not be identified for a long time, leading to the so-called ''tag starvation problem.'' On the other hand, in tree-based protocols such as the binary tree protocol and the query tree protocol, tag identification conceptually forms a tree [2].

Tree-based protocols [5, 6] split a set of tags into two subsets at a time and attempt to recognize the subsets one by one. By splitting until each set has only one tag, the reader can recognize all the tags in the reader's reading range. Tree-based protocols do not cause tag starvation, although they have relatively long identification delay as compared with ALOHA-based protocols. A good tag collision arbitration protocol for passive RFID tags should have the following characteristics:

The reader ought to recognize all the tags inside its own reading range. Tag starvation problem results in the failure of object tracking and monitoring. Since the reader, however, cannot estimate the number of tags precisely, the guarantee of recognizing all tags must be taken into consideration in the design of the tag anti-collision protocol.

The reader has to recognize tags promptly. Since an object with a tag is potentially mobile, tag identification must keep pace with the object's velocity. If tag identification is carried out slower than the object's velocity, the reader cannot recognize it and the RFID system fails in monitoring or tracking.

The tag should be recognized while consuming a small amount of resource. Since the passive tag is supplied with power by the reader's signal, tag's available power is limited. In addition, the tag has low computational capability and limited memory. Thus, the tag anti-collision protocol must load the tag with the least possible communication and computation overheads.

3.2.1 Tree Based-Deterministic

Tree-based tag anti-collision protocols perform tag identification in units of reading cycle. In a reading cycle, a reader transmits a query (or a feedback) to tags and then one or some of tags transmit ID to the reader. Since the passive tag cannot detect collision, the reader detects whether or not tag collision occurs among tag responses and determines the contents of the query (or the feedback) in the next reading cycle according to the result of the detection. On receiving a query (or a feedback) from the reader, the tag decides whether to transmit or not. Only if a single tag transmits in a reading cycle, the reader can recognize it successfully.

In tree-based protocols, the reader recognizes all the tags within its reading range during an identification frame, which consists of several reading cycles. The reader attempts to recognize a set of tags in a reading cycle. A set includes tags, which transmit at the same reading cycle. If a set has more than one tag, tag transmissions lead to collision. When tag collision occurs, the mechanisms split the set into two subsets by tag IDs or random binary numbers. After that, the reader attempts to recognize two subsets one by one in the same frame. By continuing the splitting procedure until each set has only one tag, tree-based protocols are capable of recognizing all the tags in the reader's range.

An identification frame in tree-based protocols can be represented by a tree structure as shown in Figure 3.

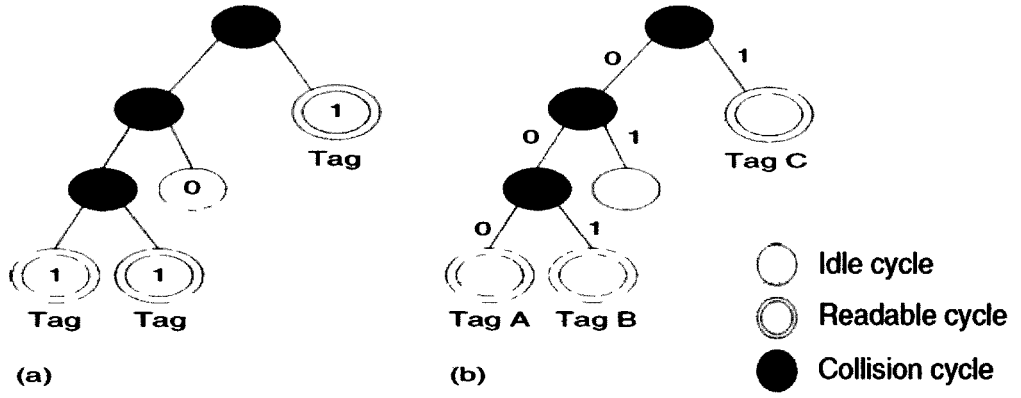

Figure 3: Tag Identification of Tree-Based Protocols. [2] (a) Tree expression of tag identification. (b)Tag identification of the binary tree protocol.

Each node in the tree corresponds to a reading cycle and a number in a node is the number of tag transmissions in that reading cycle. According to the number of tag transmissions in a reading cycle, reading cycles can be divided into three types as follows:

- Idle cycle: No transmission is attempted. The idle cycle does not make the reader fail to notice a tag, but it is a source of an unnecessary increment of identification delay.
- Readable cycle: Exactly one transmission is attempted. The reader recognizes a tag successfully.
- Collision cycle: More than one transmission is attempted. A tag collision occurs and the reader is unable to recognize any tags. The collision cycle defers tag identification and the tag's communication is pure overhead.

The reader sends a query (or a feedback) conducting the split of the set including conflicting tags. In a tree of an identification frame, only a node of a collision cycle has two child nodes because a set is split into two subsets in the collision cycle. Consequently, all intermediate nodes in the tree correspond to collision cycles and all the leaf nodes correspond to either readable cycles or idle cycles. Tag identification in tree-based protocols is coincident with a tree search starting at the root of the tree for finding nodes of readable cycles. The performance of tag identification is influenced significantly by how efficiently it splits the tag set.

3.2.1.1 *Binary Tree Protocol*

The binary tree (BT) protocol uses random binary numbers generated by colliding tags for the splitting procedure. The tag has a counter value initialized to 0 at the beginning of the frame. The tag transmits ID when the counter value is 0. Therefore, all tags, at the beginning of the frame, form one set and transmit concurrently. The reader transmits a feedback to inform tags of the occurrence of tag collision. According to the reader's feedback, all tags change their counter value. The tag randomly selects a binary number when its transmission causes collision (i.e., the counter value is 0). By adding the selected binary number to the counter value, a set is split into two subsets. When tag collisions occur, the tag which is not involved in collision (i.e., the counter value is not 0) increases its counter value by 1. When the reader's feedback indicates no collision, all tags decrease their counter values by 1. The tag infers the successful transmission from the following feedback indicating no collision. The tag recognized by a reader does not transmit any signal until the ongoing frame is terminated. Figure 3.b shows an example of tag identification of the binary tree protocol and the number by the side of the lines indicates the binary number selected randomly by conflicting tags.

The reader also has a counter to terminate a frame. It initializes the counter value with 0 in every frame. The counter value of the reader indicates the number of tag sets which are not yet recognized in a frame. If tag collision occurs, the reader adds 1 to its counter value since the number of tag sets, which the reader should recognize, increases. Otherwise, it decreases its counter value by 1. When the counter value is less than zero, the reader terminates the frame.

3.2.1.2 *Adaptive Splitting Protocol*

In tree-based tag anti-collision protocols, the tree search causes tag identification delay, and the reduction in identification delay can be accomplished by skipping of collision cycles. However, once a frame is started, the tree searches of the binary tree protocol depart from the root or the level 1 nodes of the tree and investigate all intermediate nodes wherein tag collisions occur. The unreasonable starting point of the tree search prolongs identification delay.

The basic idea of adaptive splitting protocols is to adaptively decide the starting point of the tree search with information on tags recognized in the last identification frame. At every identification frame, the tree search of tag identification starts from the nodes which were the leaf nodes of the tree in the last identification frame. Note that these starting nodes were readable cycles or idle cycles in the last frame.

To recognize arriving tags, the identification process traces down the path of the tree by inserting two child nodes of the current node into the tree. To handle unnecessary idle cycles induced by leaving tags, the identification process traces up the path of the tree by replacing two leaf nodes with their parent node. The key institution behind this approach is that in most

applications employing RFID tags, the set of objects encountered in successive readings from a particular reader does not change substantially and information from one reading can be used for the next.

3.2.2 Aloha Based –Probabilistic Anti-Collision Protocols

Probabilistic tag anti-collision protocols are based on ALOHA [2]. ALOHA is one of the basic medium access control mechanisms. In ALOHA, each tag generates a random number and waits for its transmission time according to the number chosen. If the data transmitted by a tag is not interfered by other data, the reader can identify the tag. A tag continues to do the same work after its transmission; generating a new random number and transmitting its own data after waiting for random amount of time. If during the interval two or more tags transmit, a complete or partial collision occurs. In order to solve partial collision problems, transmission time is divided into discrete time intervals in the slotted ALOHA [2]. All tags try to transmit their data after random back-off. If there are no partial collisions under the slotted ALOHA protocol, the slotted ALOHA doubles the channel utilization. A framed slotted ALOHA groups some slots into a frame, each frame having N slots. In a frame, each tag transmits its data only once. Under the framed slotted ALOHA, collisions caused by backlogged tags can be prevented.

Under ALOHA, slotted ALOHA, and framed slotted ALOHA, the waiting time for a tag is determined by a random function. The important factor which influences performance is the relationship between the number of tags and random space and the maximum value of the back-off timer. If the random space is larger than the number of tags in the reader's range, there exist many collision slots. On the other hand, if it is smaller than the number of tags, there are many

idle slots in the frame. It is important to set suitable random space based on the predicted number of tags.

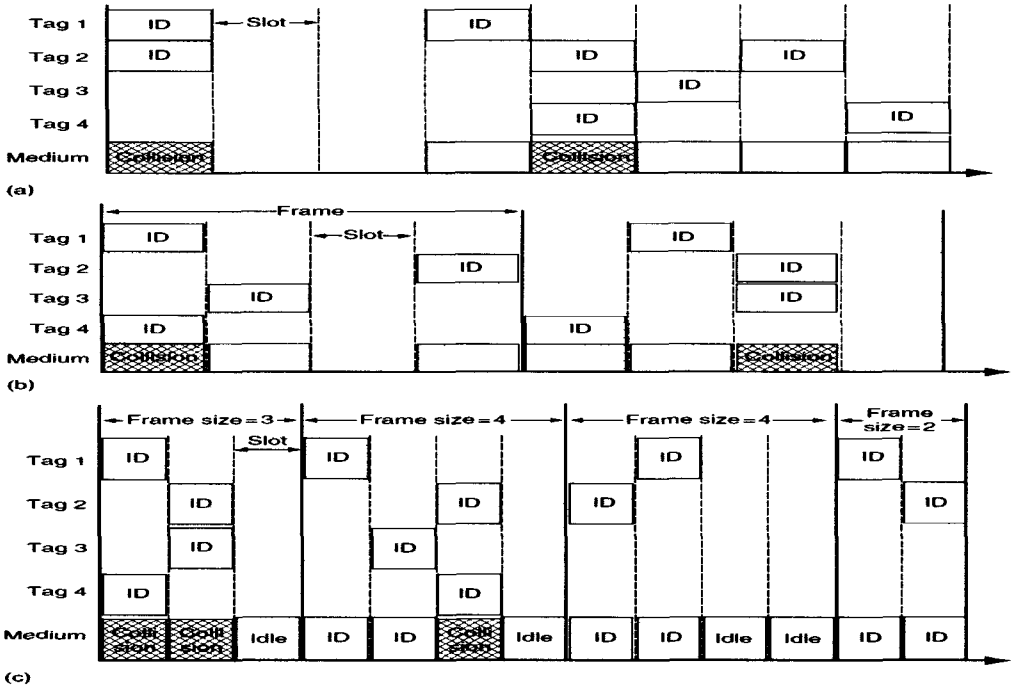

Figure 4: Examples of the Operation of Slotted ALOHA, Framed Slotted ALOHA, and Adaptive Framed Slotted ALOHA. [2]

(a) Slotted ALOHA. (b) Framed slotted ALOHA. (c) Adaptive framed slotted ALOHA.

Under the framed slotted ALOHA protocols, the frame size is the size of random space. It is easy to change the frame size at the start of a frame. There have been many proposed protocols which improve framed slotted ALOHA, called adaptive framed slotted ALOHA. Figure 4 shows examples of the operation of slotted ALOHA, framed slotted ALOHA, and adaptive framed slotted ALOHA. Next section briefly describes the existing framed slotted ALOHA anti-collision algorithms and compare their performance

3.2.2.1 *Basic Framed Slotted ALOHA (BFSA) Algorithm*

BFSA algorithm uses a fixed frame size and does not change the size during the process of tag identification. In BFSA, the reader offers information to the tags about the frame size and the random number which is used to select a slot in the frame. Each tag selects a slot number for access using the random number and responds to the slot number in the frame [1]. Figure 5 presents the process of BFSA algorithm. In the first read cycle, Tag 1 and Tag 3 simultaneously transmit their serial numbers in Slot 1. Tag 2 and Tag 5 transmit their serial numbers in Slot 2 respectively. As those are collided each other, i.e. tag collision, Tag 1, 2, 3 and 5 must respond next request of the reader. The reader can identify Tag 4 in the first reader cycle because there is only one tag response in the time Slot 3. In this example, the frame size is set to three slots.

Downlink	Request	1	2	3	Request	1	2	3
Uplink		**Collision**	**Collision**			**Collision**		
Tag1		10110010					10110010	
Tag2			10100011			10100011		10110011
Tag3		10110011						
Tag4				11110101				
Tag5			10111010			10111010		

Figure 5: The process of BFSA algorithm [2]

Since the frame size of BFSA algorithm is fixed, its implementation is simple; however, it has a weak point that drops efficiency of tag identification. For instance, no tag may be identified though the read cycle is repetitious if there are too many tags and all the slots may be filled with

collision. Or the waste of time slots generates if a large size frame is used in the case of small number of tags.

3.2.2.2 *Dynamic Framed Slotted ALOHA (DFSA) Algorithm*

DFSA algorithm changes the frame size for efficient tag identification. To determine the frame size, it uses the information such as the number of slots used to identify the tag and the number of the slots collided and so on. So DFSA algorithm can solve partially the problem of BFSA that is inefficient to identify the tag. DFSA algorithm has several versions depending on the methods changing the frame size. Among them, we will briefly explain the two popular methods appearing in [1].

The first algorithm regulates the frame size using the number of the empty slots, the slots with collision and the slots filled with one tag [1]. When the number of slots with collision is over the upper threshold, the reader increases the frame size. If the collision probability is smaller than the lower threshold, the reader decreases the frame size. Because the reader starts a read cycle with the minimum frame size, when the number of tag is small it can identify the tags efficiently without increasing the frame size much. When the number of tags is large, the reader changes its frame size so as to decrease the collision probability.

The second algorithm starts a read cycle with the initial frame size which is either two or four. If no tag is identified during the previous read cycle, it increases the frame size and starts another read cycle. It repeats this until at least one tag is identified. If a single tag is identified it

immediately stops the current read cycle and starts to read another tag with the initial minimum frame size [1].

DFSA algorithm can identify the tag efficiently because the reader regulates the frame size according to the number of tags. But, the frame size change alone cannot reduce sufficiently the tag collision when there are a number of tags because it cannot increase the frame size indefinitely. In the second method, when the number of tags is small, then it can identify all the tag without too much collision. However, if the number of tags is large, it needs exponentially increasing number of slots to identify the tags because it always starts with the initial minimum frame size after identifying a tag, regardless how many tags are unread.

3.2.2.3 *Advanced Framed Slotted ALOHA (AFSA) Algorithm*

AFSA algorithm estimates the number of tags and determines a proper frame size for the estimated number of tags and identifies tags using the determined frame size [5][6]. So it has better performance than BFSA algorithm. In AFSA, the number of tags is estimated using the result of a read cycle as the number of empty slots, slots filled with one tag, and slots with collision. AFSA algorithm uses an estimation function of the number of tag that indicate the outcome of a random experiment involving a random variable X is most likely somewhere near the expected value of X. The estimation function of the number of tags uses this property. Thus it measures the difference between the real results and the expected values to estimate the number of tags for which difference becomes minimal [4].

In AFSA algorithm, it was assumed that the tags already read respond during other read cycle. The AFSA algorithm calculates how many slots are needed to read 99% of the tags varying the frame size. Then it selects the frame size which gives the smallest number of slots. Because AFSA algorithm estimates the number of tags and determines the frame size to minimize the collision probability it is more efficient than the other algorithms. However, AFSA algorithm has the same problem that it cannot increase the frame size indefinitely as the number of tags increases. Thus, this algorithm works well if the number of tags is relatively small, however, if the number becomes large it begins to show poor performance [5][6]. Furthermore, this method cannot be applied to the tag that is deactivated once it is read.

Chapter 4

Threshold-Based Dynamic Framed Slotted Aloha Policy

Tag collision gives rise to the need for a tag anti-collision protocol that enables the recognition of tags with few collisions, and also executes in real time. The previous chapter shows that many anti-collision protocols have been investigated to solve the tag collision problem, and few solutions address the scalability issue with the number of tags is very large. To address the scalability issue with respect to the number of tags, we propose the Threshold-Base Dynamic Framed Slotted Aloha (THDFSA) protocol. Before we discuss the details of THDFSA protocol, we must explain the work we have done and inspired us to come out with this protocol.

An RFID reader recognizes an object through reading the identification number (ID) of the RFID tag attached to it. To read tag IDs, the reader sends out a signal supplying instructions to tags. The tag transmits its own ID to the reader, and then the reader consults an external database with the ID to recognize the object

Reader transmissions or tag transmissions lead to collision because readers and tags operate within the same frequency band due to cost considerations. Our focus here is on tag collision. When multiple tags transmit IDs to a reader at the same time, tag signals collide and tag collision prevents the reader from recognizing any tag. Since low-functional passive tags can neither detect collisions nor figure out neighboring tags, tag collision gives rise to the need for a tag anti-collision protocol that enables the recognition of tags with few collisions, and also executes in real time.

Our focus is on framed slotted Aloha anti collision protocol, where Tags are randomly allocated to slots within a frame. This results in some slots remaining empty, others containing two or more tags which results in a collision and no data can be retrieved from these tags, and others containing one tag in which case data is retrieved..

Our goal is to reduce the number of collisions and therefore minimize the total number of slots and time needed to identify given number of tags. We started analyzing the problem, and identifying its components by considering small number of tags and small frame size. We found that the most natural approach is to use Markov Decision Process (MDP) [25, 26]. We will briefly explain the component of this process and how we could benefit from using it in solving the tag collision problem.

4.1 **Markov Decision Process (MDP)**

MDP provide a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker. More precisely, a Markov Decision Process is a discrete time stochastic control process. At each time step, the process is in some state s, and the decision maker may choose any action a that is available in state s. The process responds at the next time step by randomly moving into a new state s', and giving the decision maker a corresponding reward $R_a(s,s')$.

The probability that the process chooses s' as its new state is influenced by the chosen action. Specifically, it is given by the state transition function $P_a(s,s')$. Thus, the next state s' depends on the current state s and the decision maker's action a. But given s and a, it is

conditionally independent of all previous states and actions; in other words, the state transitions of an MDP possess the Markov property. [25]. A MDP model contains:

- S: a finite set of states
- A: a finite set of actions
- $P_a(s,s')$: the probability that action a in a state s at a time t will lead to state s' at time $t + 1$
- $R_a(s,s')$: the immediate reward or the cost of the transition to state s' from state s with a transitional probability $P_a(s,s')$

In general, the behavior of any system that can be modeled as a Markov chain can have its behavior influenced trough MDP, such that the system state transition probabilities are controlled with the objective of either to maximize the long-run reward or minimize the long-run cost [25]. It turns out that given an optimization criterion, expressed through a once step cost or reward function, the solution of a problem formulated as an MDP is an optimal policy. The optimal policy is basically a mapping between system states and the best control actions to be taken, given the optimization criterion. The goal is to find a policy that maximize the rewards or minimize the cost.

We started to apply the above process to solve the tag collision problem, in our case the number of tags is the finite set of states. Three actions are proposed, increasing the frame size, decreasing the frame size, and keeping the frame size as it is. Our goal is then to find a policy that specify the best action we need to take when in state s that will result in the least number of slots (our cost), the following section describes this process.

4.2 **Frame Size Control Process**

In this section we describe a process that controls the decision of changing the frame size. For a frame size N(k), the policy will tell us how to fix the next frame size N(k+1) at the end of frame k so that we minimize the total number of wasted slots. For this analysis we assume the following:

- n: Total number of tags to be identified
- N(k): Frame size at frame k
- i(k): Number of tags identified in frame k
- j(k): Number of tags identified up to frame k
- a: The number of slots to be added or subtracted from N(k) which will give the next frame size N(k+1). We will set the value of a to be equal 1, 0, or -1
- m: Number of newly identified tags
- S_0: Empty slot, i.e. slot with empty tags
- S_1: Readable slot, i.e. slot with 1 tag
- S_c: Collided slot, i.e. slot that has 2 or more tags
- w: Number of wasted slots= empty slots (S_o) + collided slots (S_c)
- S= total number of slots = empty slots (S_o) + collided slots (S_c) + readable slots (S_1)

The objective of this process is to minimize the total number of unidentified tags at the end of k frames by taking the right action. Figure 6 provide an overview of the transition diagram

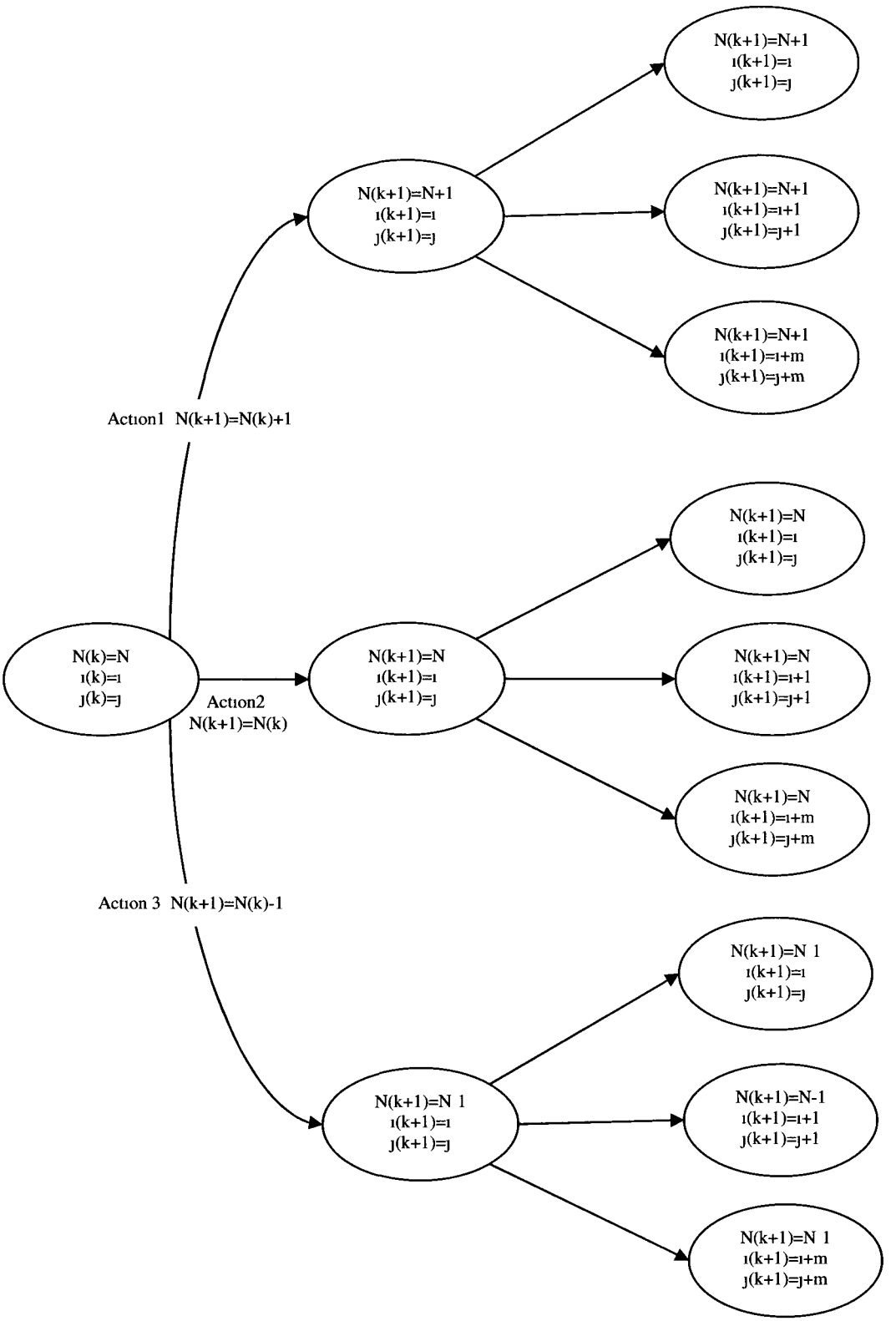

Figure 6: Frame size transition diagram

For n number of tags to be identified, the frame N(k) of size equals to N slots is used, based on the number of wasted slots (collided and empty) a decision is made to increase, decrease or maintain the same size for the frame. In order to make that decision we need a method to calculate the probability of collided, empty and slots with one tag, which will allow us to find the transitional probability from one state to the other. A decision can be then made based on the total cost value (the total number of wasted slots).

One of three actions is applied at each stage, in the first condition, the frame size is increased by a number of slots such that the number of newly identified tags is less or equal the new frame size (N+a) and less or equal the number of unidentified tags up to frame k.

- $N(k+1) = N(k) + a$
- $m <= N+a$ and $m <= n-j(k)$
- $n+a-m <= w <= n+a$

The second condition keeps the frame size unchanged, therefore the newly identified tags must be less or equal to the new frame size (N) and less or equal the number of unidentified tags up to frame k.

- $N(k+1) = N(k)$
- $m <= N$ and $m <= n-j(K)$
- $n-m <= w <= n$

In the third condition, the frame size is reduced by a number of slots such that the number of newly identified tags is less or equal new frame size (N-a) and less or equal the number of unidentified tags up to frame k.

- N(k+1) = N(k) – a
- m <= N-a and m <= n-j(k)
- n-a-m <= w <= n-a

Our task now is to calculate the transitional probabilities from one state to another for all three actions, and then Markov Decision Process can be used to calculate the best action to take such that the number of wasted slots and time required to identify all tags is minimum.

4.3 **Transitional Probability Calculation**

In order to calculate the total number of slots needed to identify a certain number of tags; we need to calculate the probability of the number of collided slots, empty slots, and slots with one tag for a given frame size and number of tags, which will allow us to find the transitional probability from one state to another, for example going from state N(k)=N, with the number of tags identified in frame k(i(k)=i) and the number of tags identified up to frame k (j(k)=j) to state N(k+1)=N, i(k+1)=i+1, j(k+1)=j+1

We derived a formula to calculate the probability of occurrence for a particular combination (S_o, S_1, S_c), the details of this derivation are as follows:

Considering that the RFID reader assigns a frame of N slots to n tags. Let S_0, S_1, and S_c denote the number of empty slots, slots with one tag, and collided slots respectively. The probability of occurrence for a particular combination (S_0, S_1, S_c) is what we want. Since each tag has N different choices (N Slots), all together there are N^n different outcomes for all the n tags. Now we count the number of different outcomes for a particular reading result (S_0, S_1, S_c). As a first step, we divide the N slots into 3 parts as $S_0 + S_1 + S_c = N$. The number of different outcomes is $\binom{N}{S_0, S_1, S_c}$.

Secondly, we consider the tags. We choose S_1 tags from the total population for the slot with one tag and leave the rest for the S_c collided slots. The number of different outcome is $\binom{n}{S_1}$. When we put the S_1 tags into S_1 slots, there are $S_1!$ different choices.

In the third step, we consider the collided tags. Let l_1, l_2, …,l_{Sc} denote the number of tags within this S_c slots. In other words, the *i-th* collided slot contains l_i tags. Then we have $l_1 + l_2 +\ldots+ l_{Sc} = n - S_1$ and l_1, l_2, …,$l_{Sc} >= 2$. For a particular combination of l_1, l_2, …,l_{Sc}, we have the number of different outcomes as

$$\binom{n - s_1}{l_1, l_2, \ldots, l_{sc}}$$

Then for all the possible combinations, the total number of outcomes is

$$\sum_{\substack{l_1, l_2, \ldots, l_{sc} \geq 2 \\ l_1 + l_2 + \ldots + l_{sc} = n - s_1}} \binom{n - s_1}{l_1, l_2, \ldots, l_{sc}}$$

Combining these three steps, we obtain the number of different outcome for a particular reading result as

$$\binom{N}{s_0, s_1, s_c} \frac{n!}{(n-s_t)!} \sum_{\substack{l_1, l_2, \\ l_1+l_2+ \ +l_{sc}=n-s_t}} \binom{n-s_1}{l_1, l_2, \ldots, l_{sc}} \quad l_{sc} \geq 2$$

Finally, the probability of occurrence for a particular combination (S_0, S1, Sc) can be calculated as:

$$\Pr\left(S_0 = s_0, S_t = s_t, S_c = s_c \middle| N(k) = N, n(k) = n\right) = \binom{N}{s_0, s_1, s_c} \frac{n!}{(n-s_t)! N^n} \sum_{\substack{l_1, l_2, \\ l_1+l_2+ \ +l_{sc}=n-s_t}} \binom{n-s_1}{l_1, l_2, \ldots, l_{sc}} \quad l_{sc} \geq 2$$

(Eq.1): Probability of occurrence for a particular combination (S_0, S1, Sc)

Equation (1) above was found in reference [17]; however there was no explanation of how this equation was obtained. Our derivation above provides a step by step derivation to calculate the probability of occurrence for a particular combination, this result will be used for our calculations in the next section.

Once we calculate the probability of occurrence for a particular combination, then we can compute the transitional probability from one state to another. The following section provides an example for doing such calculations.

4.4 **Transitional Probability Calculation Example**

In this example we demonstrate the calculation of the transitional probability from between states for a very low number of tags. We choose the initial value of the frame size to be equal 2 slots, and the number of tags to be identified equals 3 tags. We apply the control

process described earlier to find about the different set of possible states after applying the different actions. Figure 7 shows this control process of applying the actions on the frame size, the first action increases the frame size to 3 slots, the second action keeps the frame size at 2, and the third action reduces the frame size to 1 slot. These actions (action 1, action 2, and action 3 in the diagram) will be applied at the next set of states, for example, when we apply the three action on state N(k+1)=3, i(k+1)=1, j(k+1)=1, then we will either increase the frame size by one, keep it at 3, or decrement it by 1, which will result in new set of states having the values of (4,1,1), (3,1,1) or (2,1,1). The same procedure applies to all other states.

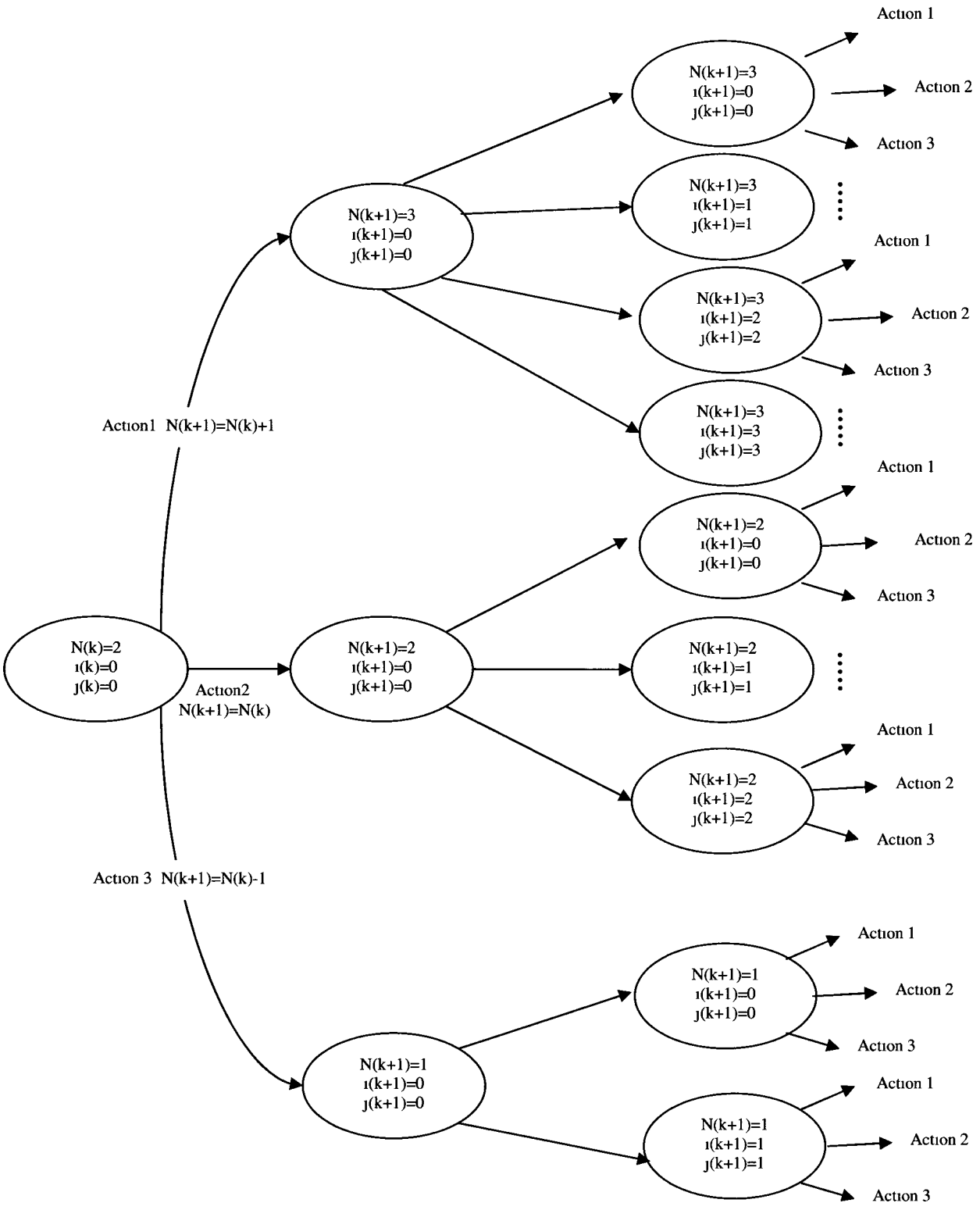

Figure 7: Example for state transitional probability calculations

When the frame size is 3, we can have one of the following four possibilities:

1. N=3, i=0, and j=0

   In order to calculate the probability of going from state (N=2,i=0,j=0) to state (N=3,i=0,j=0) we need to find what are the combination possibilities of S_0, S_1, and S_c for such transition.

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=3, i(k+1)=0, j(k+1)=0)$$
$$= P_r(S_0, S_1, S_c | N=3, n=3)$$

   The possible combinations for (S_0, S_1, S_c) are (0,0,3), (1,0,2), (2,0,1) and (3,0,0), of which combination (2,0,1) is the only valid option considering the condition $2 \times S_c \leq n - S_1$. The probability of occurrence for this combination can be calculated using equation 1 and is found to be equal to 0.111

2. N=3, i=1, and j=1

   In this case, the only valid combination of (S_0, S_1, S_c) for this transition is (1,1,1), therefore the probability of going from state (N=2,i=0,j=0) to state (N=3,i=1,j=1) is equal

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=3, i(k+1)=1, j(k+1)=1) =$$
$$P_r(S_0=1, S_1, =1, S_c=1 | N=3, n=3) = 0.667$$

3. N=3, i=2, and j=2

   In this case, there is no valid combination such that we can identify 2 tags, i.e. S_1=2 given n=3, and N=3. Therefore the probability of this transition will be equal to zero.

4. N=3, i=3, and j=3

   In this case, the only valid combination of (S_0, S_1, S_c) for this transition is (0,3,0), therefore the probability of going from state (N=2,i=0,j=0) to state (N=3,i=3,j=3) is equal

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=3, i(k+1)=1, j(k+1)=1)$$
$$= P_r(S_0=0, S_1,=3, S_c=0 \mid N=3, n=3) = 0.222$$

Similar to the above calculations, we can find the transitional probabilities for all other states, the values for the remaining states shown in Figure 7 are:

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=2, i(k+1)=0, j(k+1)=0)$$
$$= P_r(S_0=1, S_1,=0, S_c=1 \mid N=2, n=3) = 0.25$$

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=2, i(k+1)=1, j(k+1)=1)$$
$$= P_r(S_0=0, S_1,=1, S_c=1 \mid N=2, n=3) = 0.75$$

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=2, i(k+1)=0, j(k+1)=0)$$
$$= P_r(S_0=0, S_1,=2, S_c=0 \mid N=2, n=3) = 0.0$$

$$P_r(N(k)=2, i(k)=0, j(k)=0) \rightarrow P_r(N(k+1)=1, i(k+1)=0, j(k+1)=0)$$
$$= P_r(S_0=0, S_1,=0, S_c=1 \mid N=1, n=3) = 1.0$$

$$P_r(N(k)=2, i(k)=0,\ j(k)=0) \rightarrow P_r(N(k+1)=1, i(k+1)=1,\ j(k+1)=1) =$$
$$P_r(S_0=0, S_1,=1, S_c=0 \mid N=1, n=3) = 0.0$$

The control process will be applied to the new states, for example if we take case 2 above, we found that the transitional probability to go from state (2, 0, 0) to state (3, 1, 1) is 0.667. Figure 8 shows the transition states from states (3,1,1), applying the three actions again, the resulting new states will be (4,1,1), (3,1,1), and (2,1,1), then we need to find the transitional probabilities to all possible combinations, for example from (3,1,1) to (4,0,1), (4,1,2), or (4,2,3), these can be calculated using equation 1 and are found to be equal 0.25, 0, and 0.75 respectively. Since the number of tags in the system equals to 3, then there will be no further actions to take if we are at state (4,2,3), however if we are at state (4,0,1) then we need to further apply the actions in order to identify the remaining tags.

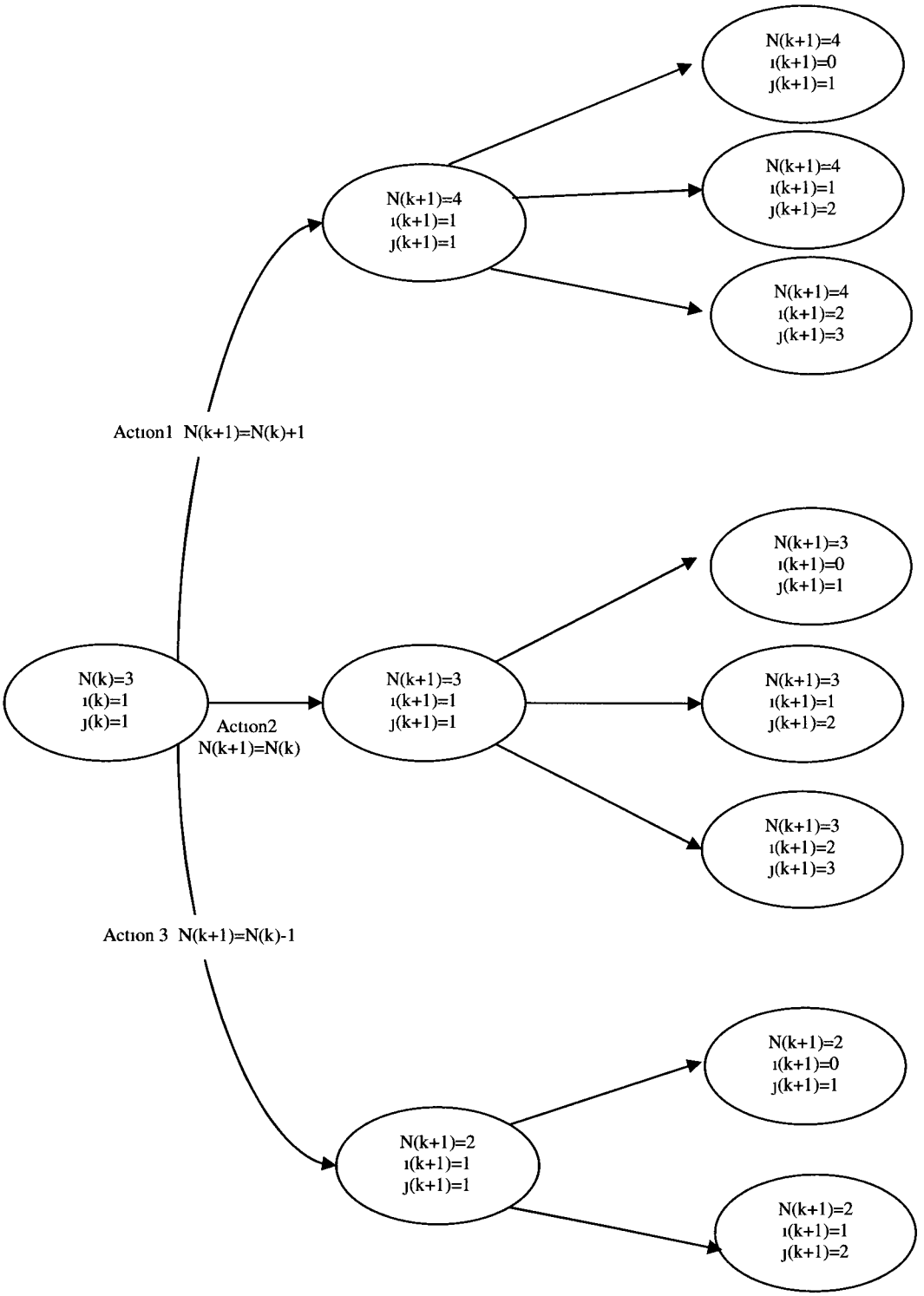

Figure 8: Example for state transitional probability calculations (continued)

This example shows that the resulting transition matrix is expected to be big considering the low number of tags, and it is likely that when the number of tags is bigger we would face a computation problem, this triggered us to move into considering a heuristic approach rather than the complicated exact approach, the next section provides the details of our proposed heuristic protocol which we called Threshold Based Dynamic Framed Slotted Aloha (THDFSA).

4.5 **Threshold Base DFSA Analysis**

The main objective is to minimize the time required to identify all the tags in a system as well as to minimize the number of wasted slots In order to achieve that, we need to find an answer to two questions, what is the optimum initial frame size to use in order to identify certain number of tags and how to vary the frame size in order to minimize the total number of slots needed to identify all the tags?

4.5.1 General Approach Description

To address the scalability issue with respect to the number of tags, we propose the Threshold-Base Dynamic Framed Slotted Aloha (THDFSA) protocol, which tackles the task into two steps: (1) First it determines the optimum initial frame size to use in order to identify a certain number of tags; (2) then it implements a policy to adjust the frame size in order to minimize the total number of slots needed to identify all the tags. AFSA updates the frame size for each read cycle, while our proposed protocol updates the frame size only when the number of estimated tags to identify drops below some threshold. The number of estimated remaining tags to identify is likely to be inaccurate (i.e. either it over-estimates or under-estimates the actual number of tags). This inaccuracy will be emphasized at each read cycle, in AFSA. A frame size that under-estimates the number of tags to identify would yield high collisions occurrences; on the other hand, a frame size resulting from an over-estimation of the number of tags would leave a large number of empty slots.

THDFSA reduces the possible effect of inaccurate estimation by maintaining the frame size constant until a noticeable change in the number of tags to identify is observed. The protocol uses

the initial total number of tags to set the initial frame size. Then, the reader proceeds with the tag identification process, read cycle after read cycle, as implemented in AFSA. But instead of updating the frame size at the beginning of every read cycle, THDFSA updates the frame size only twice, at each of these times, the number of remaining tags is less than some threshold, as depicted in Figure 9.

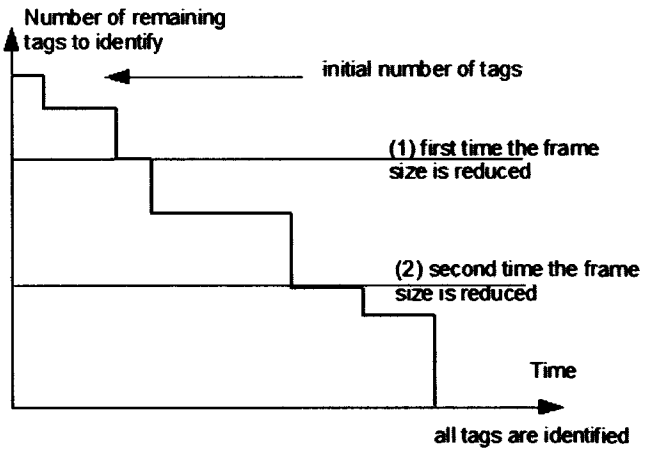

Figure 9: Two-Threshold Policy

This protocol may implement any number of thresholds. The more thresholds, the more frequently the frame size is updated. Without any threshold (i.e. the frame is constant throughout the tag identification process), THDFSA is in fact the Basic Slotted ALOHA. With a very large number of thresholds, THDFSA is likely to behave like Advanced Frame Slotted ALOHA. In other words, THDFSA (with two threshold) is a compromise between a policy which keeps the frame size constant, and the one that updates it too often (based on possible inaccurate estimations of the number of remaining tags).

4.5.2 Two-Threshold Policy

Let n(i) denote the estimated number of tags to identify, N(i) the frame size, and r(i) the total number of tags successfully identified, during read cycle i. N(i) is updated based on the estimated number of tags the reader still has to identify. Let us describe the frame size update policy:

Frame Size Update Policy:

```
n(1)                /*initial estimated tags number*/
N(1) = 0.5 · n      /* initial frame size*/

Read cycle i = 1
While n(i) /n(1) > TH1

    Start read cycle with tag identification
    and identify r(i)
    n(i+1) = n(i) – r(i)
    i = i+1
End

N(i) = R1 · N(i)

While n(i) / n(1) > TH2
    Start read cycle with tag identification
    and identify  r(i)
    n(i+1) = n(i) – r(i)
    i = i+1

End

N(i) = R2 · N(i)
While n(i) > 0
    Start read cycle with tag identification
    And identify r(i)
    n(i+1) = n(i) -  r(i)
    i = i+1

End
```

TH1 and TH2 (with TH1 > TH2) are the two thresholds, while R1 and R2 (with 1 > R1 > R2) are two ratios with respect to which the frame size is reduced. The proposed policy keeps the frame size unchanged as long as a 'large' number of tags remain to identify. And it is based on the implementation of two thresholds TH1 and TH2 which determine the appropriate moments to reduce the frame size. Within cycle i, the reader counts r(i), the number of tags it successfully identifies; at the end of cycle i, the remaining number of tags to identify is decremented by r(i).

The proposed policy is based on four parameters: the two thresholds (TH1 and TH2) and the two ratios (R1 and R2). Qualitatively, one may conjecture that fixing the thresholds "too large" and the ratios "close to 0" would yield a system where the frame size is reduced too early and where the collisions rates are high. Inversely, having the thresholds "too small" and the ratios "close to 1" produces a system with large frame sizes even when very few tags remain to identify.

In THDFSA algorithm, different parameters can be selected, such as the number of tags to be read and the number of slots in one frame. From the first simulation results we set the initial number of slots in the frame to be equal half the number of tags, and then the frame size is modified based on the number of remaining tags in the system, number remaining tags equals total number of tags minus number of tags identified. Once the remaining number of tags reaches certain threshold value (TH1) then the new frame size will be calculated as N_{new} = R1. N where R1 is the ratio by which the frame size is reduced. The process will continue and more tags will be identified, when the remaining number of tags is less or equal the second threshold value (TH2) a new frame size is calculated to be equal to R2.N_{new} where R1 > R2 and both R1 and R2

< 1. Tag identification continues until the remaining unidentified number of tags equals to zero, in which case all the tags in the system are identified.

The above process indicates that certain threshold values (TH1, TH2) are used to determine when to calculate the new frame size, moreover, to calculate the new frames size we need to know the value of R1 and R2. We simulated the THDFSA algorithm and analyzed the resulting data to find the values of TH1, TH2, R1, and R2 that allows us to identify all the tags with minimum number of slots. This analysis is explained in the next chapter.

Chapter 5

Performance Analysis and Results

In this chapter, we discuss the performance of THDFSA and compare it to the existing solutions of RFID tag collisions.

5.1 Simulation Environment

For the performance evaluation of THDFSA and to generate the graphs and results Matlab was used. In the first simulation, for an estimated number of tags we wanted to calculate the optimum initial frame size to identify all the tags with the lowest number of wasted slots. A frame consists of N number of slots, for n number of tags to be identified, each tag will randomly pick one slot in the frame to communicate with the reader, if two or more tags pick the same slot then a collision occur, the slot is wasted, and the tags are not identified. Also if a slot is not picked by any tag, then it is considered to be wasted. A tag can be identified if and only if it is the only tag in a slot. The reader then sends the next frame with the same number of slots to identify the remaining tags, the process continues until all the tags are identified

The second simulation was used to find the optimum threshold values to identify the tags with the least number of slots. The tag identification process is similar to the one explained above, however, in this simulation the frame size is not fixed. The frame size is adjusted based on the remaining number of unidentified tags in the system, the process of adjusting the frame size is explained in section 5.3.

5.2 **Finding the Optimum Frame Size**

The initialization of THDFSA requires an accurate estimation of n(1), the initial total number of tags to identify. Based on this estimation, the first frame is set to 0.5 n(1). In this section, we justify why the initial frame size is set to half of the n(1). In this simulation the frame size (N) is equal to number of tags to be identified multiplied by a constant value (r) where the $0.3 < r < 1.0$. We conducted simulations for "small" systems with n(1) ranging from 100 to 500, for "large" systems with n(1) from 1000 to 5000, and finally for "very large" systems with n(1) larger than 10000. For each scenario, we report the total number of slots (i.e. $\sum_i N(i)$) as a function of the ratio N(1)/n(1). Figures 10- 12 show that the tags can all be identified in the shortest amount of time when the ratio N(1)/n(1) is about 0.5 for most cases.

Running the simulation for 20 times for different number of tags, we found the best frame size such that min number of slots are required to identify all the tags. Table 3 and Figure 10 show a sample of the simulation results for tags ranging from 1000 to 5000. Another set of results were obtained for lower number of tags ranging from 100 to 500, also higher number of tags was considered in the range of 5000 to 20000 tags.

Frame Size N=r*n	Number of Tags (n)				
	1000	**2000**	**3000**	**4000**	**5000**
N=0.3*n	4710	9420	14355	19260	24150
N=0.4*n	4080	8200	12260	16240	20700
N=0.5*n	4000	8000	12200	16200	20500
N=0.6*n	4147	8287	12697	16807	21157
N=0.7*n	4270	8610	13020	17780	21525
N=0.8*n	4320	9040	14280	19200	23800
N=0.9*n	4595	9275	14180	18545	24080
N=1.0*n	5000	10000	15000	20000	26250

Table 3: Finding the best frame size to identify different number of tags

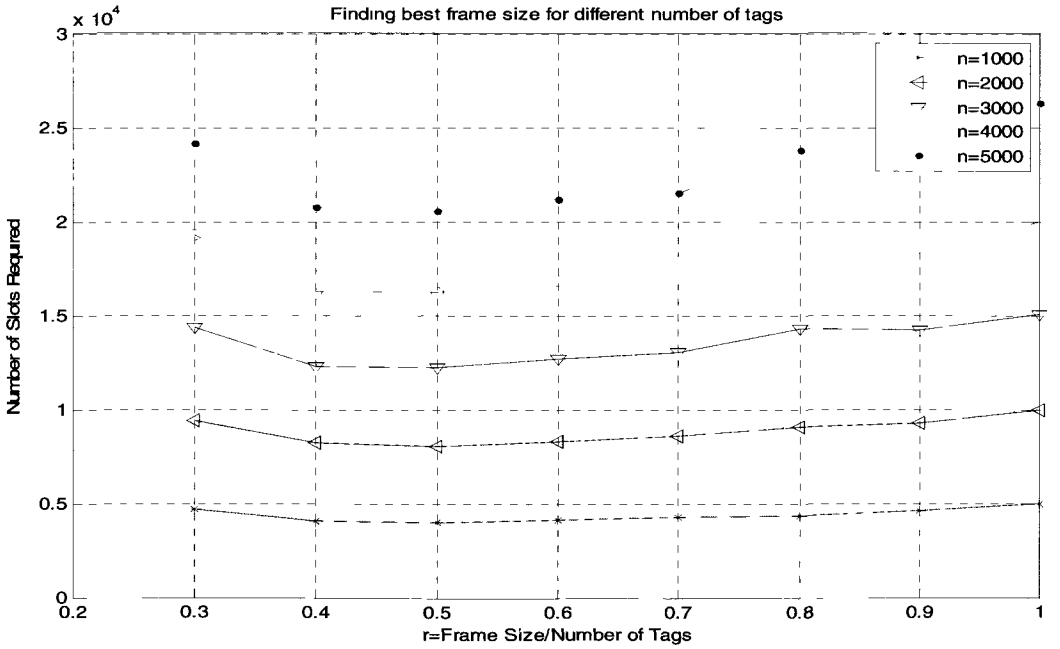

Figure 10: Best frame size for medium number of tags, 20 time simulation

Table 3 shows that the frame size (N) is calculated as the number of tags to be identified multiplied by the constant r. The results shows that the minimum number of slots needed to identify the tags would be when the frame size equals half the number of tags, i.e. when r = 0.5. On average, similar results were obtained for lower number of tags as shown in Figure 11, and also for higher number of tags as shown in Figure 12. A sample of the total number of slots needed for low, medium, and high number of tags is shown in Table 4 (a), (b), and (c). The simulation run of these results is 20 time.

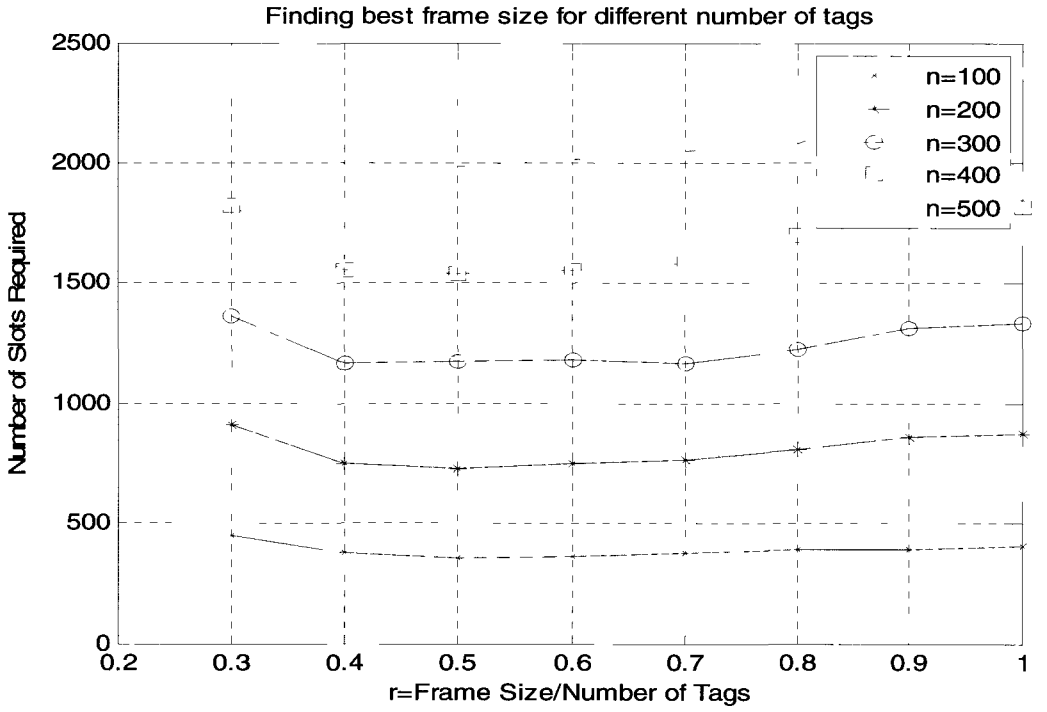

Figure 11: Best frame size for low number of tags, 20 time simulation

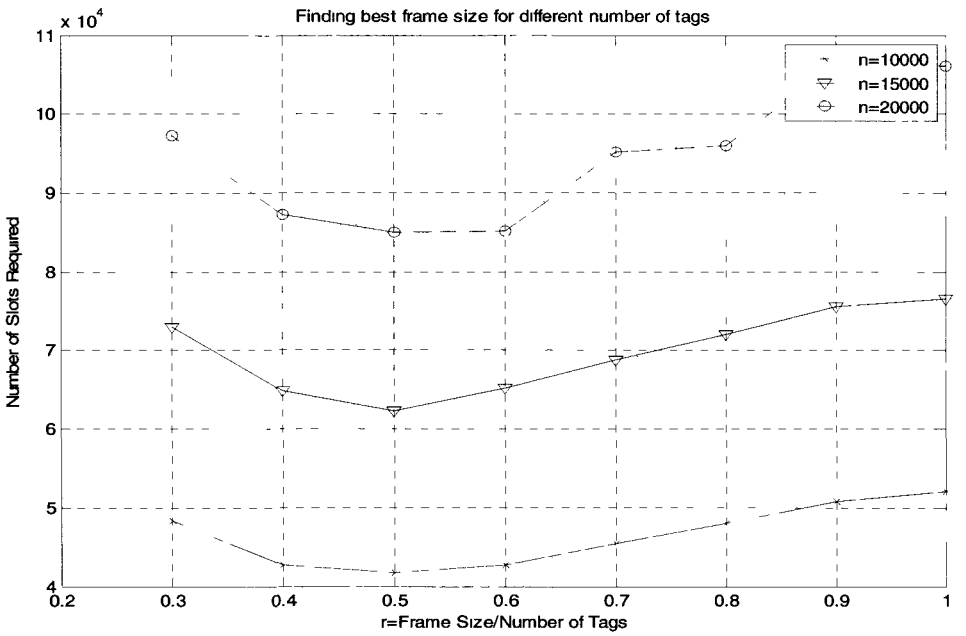

Figure 12: Best frame size for high number of tags, 20 time simulation

No of Tags (n)	100	200	300	400	500	1000
Best frame size N	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
Slots Needed	340	735	1134	1520	1938	4000

No of Tags (n)	1500	2000	2500	3000	4000	1500
Best frame size N	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
Slots Needed	6060	8000	10125	12200	16200	6060

No of Tags (n)	5000	10000	15000	20000
Best frame size N	0.5*n	0.5*n	0.5*n	0.5*n
Slots Needed	20250	41750	62250	85000

Table 4: Best frame size for (a) Low number of tags (b) Medium number of tags (c) High number of tags

Similar results were obtained when we ran the simulation for different number of times, Figures 13, 14, and 15 show these results for simulation run of 10, 50, and 100 time respectively, for the low range of number of tags.

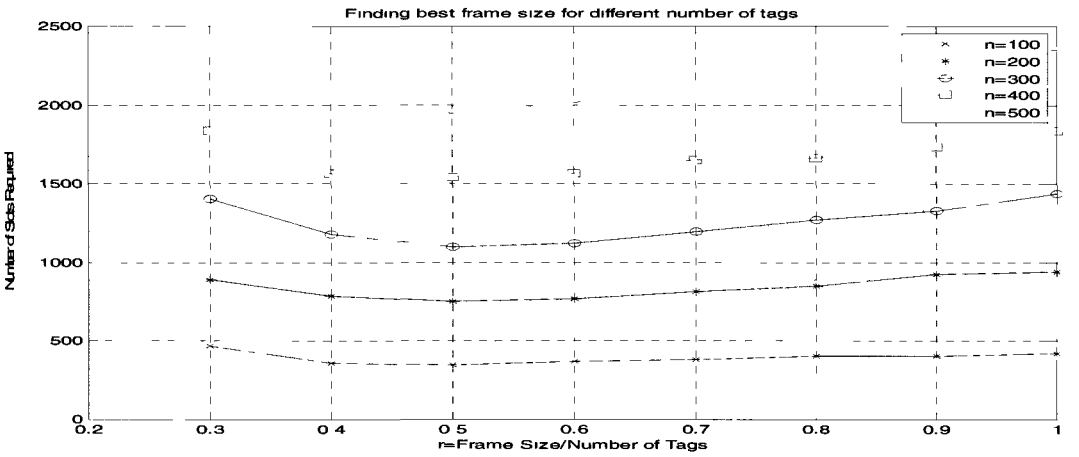

Figure 13: Best frame size for low number of tags, 50 time simulation

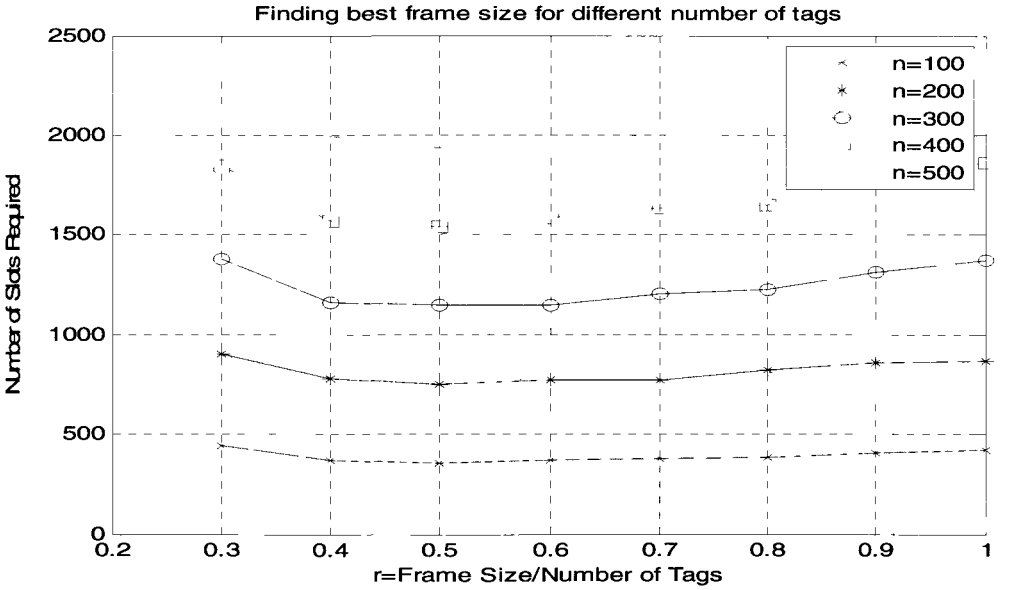

Figure 14: Best frame size for low number of tags, 50 time simulation

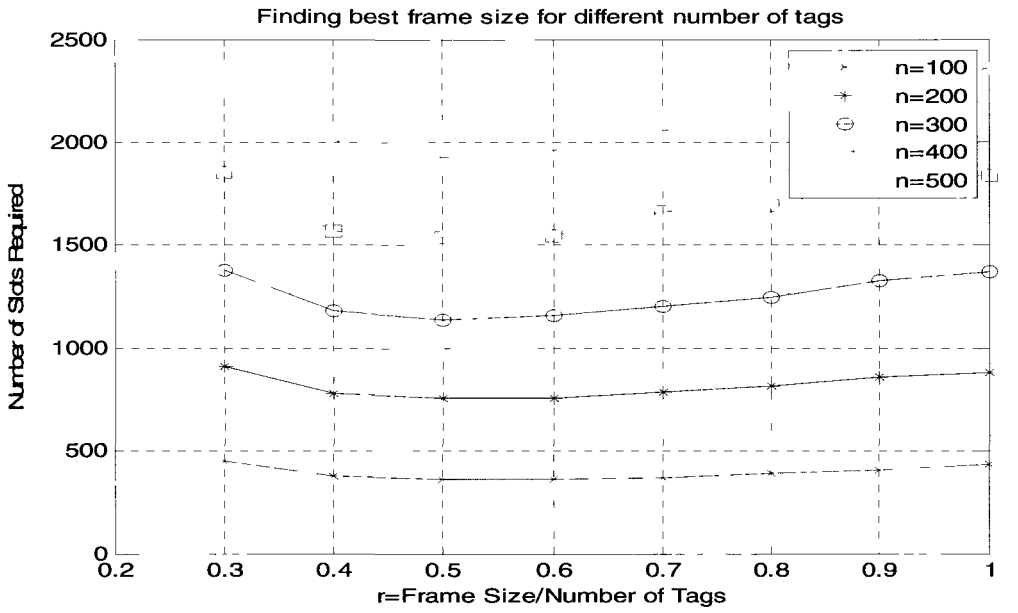

Figure 15: Best frame size for low number of tags, 100 time simulation

5.3 **Tag Identification Process**

In the second simulation, different parameters of the algorithm can be selected, such as the number of tags to be read and the number of slots in one frame. From the previous simulation results we set the initial number of slots to be equal half the number of tags, and then the frame size is reduced based on the number of remaining tags in the system. Once the remaining number of tags reaches the value of TH1 a new frame of size equal to R1*N is used to identify the remaining tags, the process of tag identification continues until the remaining number of tags equals the second threshold value of TH2, at this point another frame size is calculated to be equal to R2*N. Tags identification continues until all tags are identified.

The results are obtained from simulations (based on MATLAB). The protocol is based on the specification of four parameters TH1, TH2, R1 and R2. Assigning arbitrary values to those

parameters is likely to yield low performance. The main performance metric in our analysis is the total number of slots required to identify all tags i.e. $\sum_i N(i)$ The numerical optimization presented in this thesis is derived from exhaustive search of the best combination of the four parameters. To reduce the total number of combinations, we consider only discrete values for each. Let n(1) denote the initial number of tags to identify, N(1) the initial frame size, N(i) the frame size, TH1, TH2 the threshold values, and R1, R2 the two ratios with respect to which the frame size is reduced. Let us describe the process of finding the optimal initial frame size:

Two-Threshold Policy:

For a given initial number of tags to identify: n(1)
Set N(1) = 0.5 n(1)

For TH1 = 0.1, 0.2, 0.3, …, 1.0 Do
    For R1 = 0.1, 0.2, …, 1.0 Do
        For TH2 = 0.1, 0.2, …, TH1 Do
            For R2 = 0.1, 0.2, …, 1.0 Do

            Simulate the system
            And Determine $\sum_i N(i)$
            End
        End
    End
End

We simulated the above process and produced a large set of results. We started by fixing the remaining number of tags and varying the value by which the frame size is reduced, and then we varied the number of remaining tags and fixed the value by which the frame size is reduced. Samples of our results are shown in Tables 5, and 6. These results are the average of 20 time simulation.

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (Nnew)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	338	352	361	386	423

(a)

No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (Nnew)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1094	1092	1113	1134	1328

(b)

No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (Nnew)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	3915	3840	3885	3990	4450

(c)

No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (Nnew)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	20250	20000	19600	20550	22875

(d)

Table 5 (a), (b), (c), (d): Different frame size for the number of remaining tags = 0.9*n

No of Tags (n)	100	100	100	100	100	100
Remaining Tags (TH1)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (Nnew)	0.8*N	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	319	323	314	304	325	358

(a)

No of Tags (n)	500	500	500	500	500	500
Remaining Tags (TH1)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (Nnew)	0.8*N	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1780	1769	1655	1669	1705	1899

(b)

Table 6 (a), (b): Different frame size for the number of remaining tags = 0.6*n

The analysis of the data collected resulted in finding the values of TH1, TH2, R1 and R2 that produces the minimum number of slots, these values are 0.5, 0.2, 0.4, and 0.2 respectively. That means when the number of remaining tags is less or equal 0.5 of the initial number of tags to be identified, the frame size is reduced to be equal 0.4 of the original frame size (N), and when the remaining tags is less or equal 0.2 of the initial number of tags, the frame size is further reduced to be equal to 0.2 of the initial frame size (N). These values resulted in the minimum total number of slots used to identify all the tags. Appendix A provides sample of the data collected to find the best values for TH1 and R1, for example, for 1000 tags to be identified, when TH1 is larger or less than 0.5 and R1 is larger or less than 0.4 the number of slots needed was higher than when TH1=0.5 and R1=0.4. Appendix B provides the data collected to find the best values for TH2 and R2, the lowest number of tags was found when TH2=0.2 and R2=0.2

Several results in the simulation of the THDFSA algorithm are listed in Table 7. The results shown are the average of twenty time simulation.

Number of Tags	100	200	300	400	500	1000	2000	3000	5000	10000
Number of Slots	261	544	849	1098	1375	2755	5640	8475	14200	28500

Table 7: Simulation results of THDFSA algorithm

Table 7 shows the number of slots needed to identify different number of tags ranging from 100 to 10000. The ratio of the number of tags to be identified to the number of slots needed is around 0.38 for low number of tags and 0.35 -0.36 for high number of tags, which outperforms the ratios for the other algorithms. The performance comparison with those algorithms is detailed in the next section.

5.4 **Performance Comparison**

In this section, we compare the performance of Binary Tree, Framed slotted ALOHA (FSA), Dynamic Framed Slotted ALOHA (DFSA) and Advanced DFSA protocols [3] with that of our threshold-based policy. The performance metric used in the comparison is defined as the total number of slots necessary to identify all the tags.

Table 8 shows simulation results for other algorithms available in the literature [3]. According to the simulation results of FSA algorithm, when the number of slots per frame is smaller than the number of tags, there is so many collision that it costs so many slots to identify tags; when the number of slots per frame is bigger than the number of slots, and along with the increase of the number of slots per frame, it costs more and more slots totally because of the waste in empty slots. So when the slot number is close to the tags number, reader can identify tags more efficiently. Table 8 FSA values are for a frame size of 512 slots.

Total Slots	**Number of Tags**								
	20	**50**	**100**	**200**	**300**	**500**	**1000**	**1500**	**2000**
FSA (512)	512	512	768	-	1330	1788	-	-	-
DFSA	62	157	356	694	1146	1855	-	-	-
BT	80	181	373	684	984	1763	3881	5485	7716
ADFSA	56	164	299	643	964	1682	3401	4920	6527
THDFSA	48	134	261	544	849	1375	2755	4260	5640

Table 8: Simulation results of other anti-collision protocols

According to the simulation result of DFSA algorithm, the total numbers of slots change smoothly. So the DFSA algorithm can solve the FSA problem of inefficiency. When the number of tags is small, the DFSA algorithm is better than the BT algorithm; when the number of tags is large, the BT algorithm is more efficient than the DFSA algorithm. As shown in the table, the ADFSA algorithm is better than the FSA, DFSA, and BT algorithms. Our THDFSA has produces even better results compared to ADFSA in terms of the total number of slots required to identify any number of tags, which means less time required to identify the tags and better system efficiency. Reducing the time required to identify all the tags is essential in many RFID applications, such as in supply chain management and retail industry.

Figure 16 compares the THDFSA algorithm results with those of FSA, DFSA, BT, and ADFSA. The figure shows that the THDFSA algorithm produces better values compared to FSA with frame size of 512 (FSA 512), BT, DFSA, and Advanced DFSA protocols presented in for the low range values of number of tags. Number of tags in this figure is between 100 to 500 tags.

Figure 17 shows that the THDFSA algorithm is also better than DFSA, Binary Tree, and ADFSA for larger number of tags. Number of tags in this figure is in the range of 500 to 2000 tags.

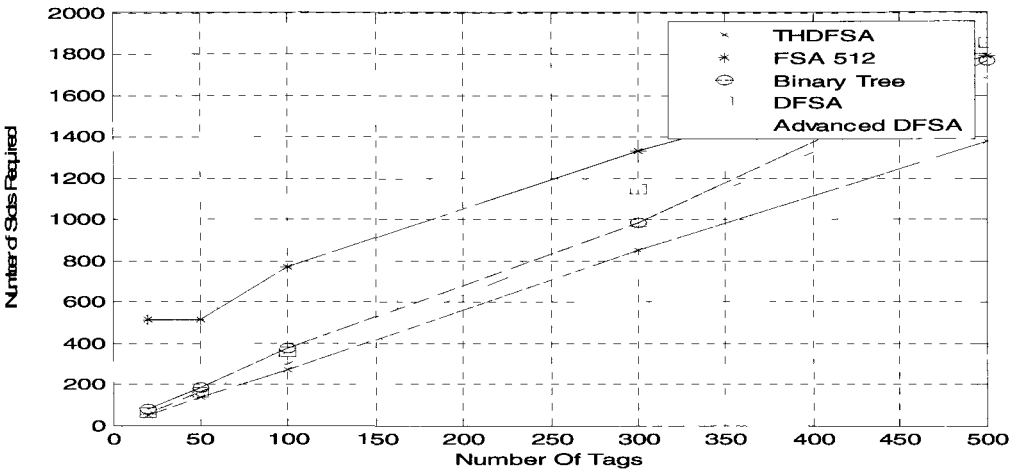

Figure 16: Comparison of FSA, Binary Tree, DFSA, Advanced DFSA, and THDFSA for low number of tags

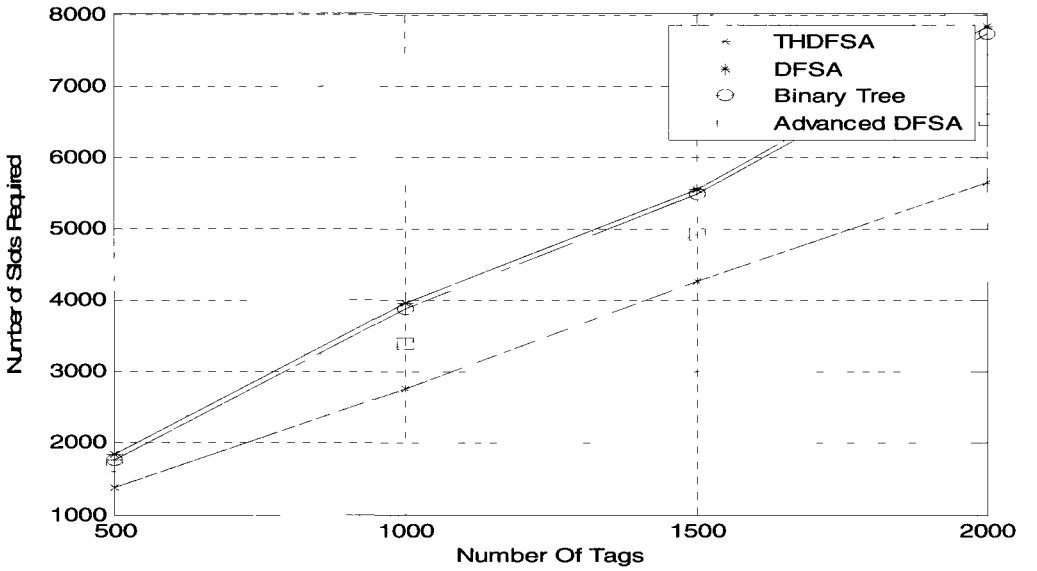

Figure 17: Comparison of FSA, Binary Tree, DFSA, Advanced DFSA, and New THDFSA for large number of tags

Table 9 shows the total number of slots needed to identify different number of tags using THDFSA for different number of simulation runs. The results show that similar values were obtained when the simulation was run for 10, 20, 50, and 100 time.

Total Slots THDFSA	Number of Tags								
	20	**50**	**100**	**200**	**300**	**500**	**1000**	**1500**	**2000**
10 Simulation Runs	53	128	267	548	828	1372	2750	4140	5620
20 Simulation Runs	48	134	261	544	849	1375	2755	4260	5640
50 Simulation Runs	52	132	268	552	827	1391	2780	4206	5612
100 Simulation Runs	50	133	270	547	835	1388	2791	4206	5622

Table 9: Number of slots needed using THDFSA for different simulation runs

The above tables and figures show that THDFSA performs better that all other algorithms when considering the total number of slots and therefore the overall time required to identify a given number of tags, for example, to fully identify 1000 tags, THDFSA uses 2755 slots compared to 3401 slots for ADFSA. Next chapter concludes our work and provide some future work recommendations.

Chapter 6

Conclusion and Future Work

This chapter presents what has been accomplished in this thesis and what remains to be addressed in the future.

6.1 Conclusion

An analysis for RFID tag-tag collision problem using MDP was conducted. The goal is to reduce the number of collisions and therefore minimize the total number of slots and time needed to identify given number of tags. We started analyzing the problem, and identifying its components by considering small number of tags and small frame size. The result of our analysis and computation showed that the resulting transition matrix is expected to be big considering the low number of tags, and it is likely that when the number of tags is bigger we would face a computation problem, this triggered us to move into considering a heuristic approach rather than the complicated exact approach

An enhanced version of framed slotted ALOHA, based on the implementation of two thresholds is proposed to solve the tag collision problem in RFID systems. Our proposed THDFSA uses the initial total number of tags to set the initial frame size, our simulation results showed that the initial number of slots in a frame should be equal to half the number of tags to be identified.

The second part of ADFSA is the mechanism to control the frame size. Our proposed policy is based on four parameters: the two thresholds (TH1 and TH2) and the two ratios (R1 and R2).

The frame size is updated only twice during the tag identification process; the two thresholds define the appropriate times of the frame size update. Using exhaustive search, we determine the set of parameters to use in order to minimize the overall number of slots required to identify all tags.

The threshold-based policy exhibits better performance than existing solutions, such as Advanced Framed Slotted ALOHA, in particular in systems with very large number of tags.

6.2 **Future work**

We are investigating two directions to extend our protocol: (1) the number of thresholds could be optimized based on the initial number of tags to identify. The protocol may implement any number of thresholds, whose effect on the system performance needs further analysis. (2) The second direction of research deals with the determination of the parameters (the thresholds TH1, TH2... and the respective ratios R1, R2 ...)using a more effective method than the exhaustive search. The frame control process using MDP we described in chapter 5 will be the focus of our future research to further enhance tag identification process.

Bibliography

[1] **Su-Ryun Lee; Sung-Don Joo; Chae-Woo Lee**; "*An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification*". Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on 17-21 July 2005 Page(s):166 –172

[2] **K. Finkenzeller**. *RFID handbook - Second Edition.* JOHN WILEY & SONS, 195–219, 2003.

[3] **Tao Cheng; Li Jin** "*Analysis and simulation of RFID anti-collision algorithms*". Advanced Communication Technology, 2007 The 9th International Conference on, 12-14 Feb. 2007 Page(s) 697 - 701

[4] **Paris Kitsos and Yan Zhang**. *RFID Security.* Springer Publishing Company, 2008

[5] **H. Vogt**. *Multiple Object Identification with Passive RFID Tags.* 2002 IEEE International Conference on Systems, Man and Cybernetics. October 2002.

[6] **H. Vogt**. *Efficient Object Identification with Passive RFID Tags.* Proc. Pervasive 2002. 98–113. 2002.

[7] **Anil Rohatgi**. *Implementation and Applications of an Anti-Collision Differential-Offset Spread Spectrum RFID System*, 2006

[8] **Nikolaos Alchasidis**. *Data Integrity in RFID Systems.* 2006

[9] **Matt Ward**. *RFID Frequency, Standards, Adoption, and Innovation.* 2006

[10] **Shweta Jain, Samir R. Das**. *Collision Avoidance in a Dense RFID Network. WiNTECH'06,* September 29, 2006

[11] EPC Generation 1 Tag Data Standards Version 1.1 Rev.1.27," EPC Global, Tech. Rep., May 2005.

[12] EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 1.0.9," EPC Global, Tech. Rep., January 2005.

[13] **V. Deolalikar, M. Mesarina, J. Recker, D. Das, and S. Pradhan**, *Perturbative Time and Frequency Allocations for RFID Reader Networks,*" in HP Lab Technical Report, 2005.

[14] **J. Zhai and G. Wang**, *An anti-collision algorithm using two-functioned estimation for RFID tags,* in Proceedings of International Conference on Computational Science and its Applications, LNCS 3483, pp. 702–711, May 2005.

[15] UCODE, Philips Semiconductors, http://www.semiconductors.philips.com, 2005. Accessed in December 2010

[16] **F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min**, *Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems*, in Proceedings of the International Symposium on Low Power Electronics and Design, pp. 357–362, August 2004.

[17] **Lei ZhuTak-Shing Peter Yum**. *The Optimization of Framed Aloha based RFID Algorithms. MSWiM'09,* October 26–29, 2009, Tenerife, Canary Islands, Spain

[18] **ENGELS, D.W., SARMA, S. E.** 2005. *Standardization of requirements within the RFID class structure framework.* Technical Report: Auto-ID Center. Cambridge, Massachusetts.

[19] IEE. 2005. Radio Frequency Identification Device Technology (RFID) Factfile. The Institution of Electrical Engineers. http://www.iee.org/Policy/sectorpanels/control/rfid.cfm. Accessed in December 2010

[20] **DRESSEN, D**. *Considerations for RFID technology selection.* Atmel Applications journal. Corporate communication: Atmel Corporation, iss. 3, Summer 2004.

[21] **PARET, D**. *Technical state of art of 'Radio Frequency Identification – RFID' and implications regarding standardisation, regulations, human exposure, privacy.* Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies (sOc- EUSAI '05), pp. 9–11. ACM Press: New York, NY.

[22] **THOMPSON, D**. 2006. *RFID technical tutorial.* The Journal of Computing Sciences in Colleges, vol. 21, no. 5, pp. 8–9. Consortium for Computing Sciences in Colleges: USA.

[23] http://www.zebra.com/id/zebra/na/en/index/rfid/faqs/rfid_tag_characteristics.html. Accessed in December 2010

[24] RFID Journal. 2006. A summary of RFID standards. RFIDJournal.com. Available online at: http://www.rfidjournal.com/article/articleview/1335/1/129/. Accessed in December 2010

[25] **Richard S. Sutton and Andrew G. Barto** " *Reinforcement Learning: An Introduction*" , MIT Press, Cambridge, MA, 1998

[26] **Puterman, Martin L.** *Markov Decision Process, Discrete Stochastic Programming*. S.1: Wiley-Interscience, 1994.

[27] **Yang Z.Y, Chen J., Mao Z.***"Study on the Performance of Tag Tag Collision Avoidance Algorithms in RFID Systems"* 2009 Third Asia International Conference on Modelling & Simulation.

[28] **Yan X., Yin Z., Xiong L.**, *"A Comparative Study on the Performance of the RFID Tag Collision Resolution Protocols"* 2008 Second International Conference on Future Generation Communication and Networking, 2008 IEEE

[29] **Qiaoling T., Xuecheng Z., Dongsheng L., Yifei D.** "Modeling the Anti-collision Process of RFID System by Markov Chain" 2007 IEEE

[30] **R. Angeles**, "*RFID Technologies:Supply Chain Applications and Implementation Issues*", Info. Sys. Mgmt., Winter, 2005, pp. 51-65.

[31] **N. C. Wu, M. A. Nystrom, T. R. Lin, and H. C. Yu**, "*Challenges to Global RFID Adoption,*" *Technovation,* vol. 26, pp. 1317-1323, 2006.

Appendix A

Sample of the data collected to calculate the threshold values, TH1 and R1

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	338	352	361	386	423
No of Tags (n)	200	200	200	200	200
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	720	738	721	756	840
No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1094	1092	1113	1134	1328
No of Tags (n)	400	400	400	400	400
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1476	1488	1442	1512	1780
No of Tags (n)	500	500	500	500	500
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1845	1860	1803	1935	2250
No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	3915	3840	3885	3990	4450
No of Tags (n)	2000	2000	2000	2000	2000
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	7920	7760	7840	7920	9100
No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	20250	20000	19600	20550	22875
No of Tags (n)	10000	10000	10000	10000	10000
Remaining Tags (TH1*n)	0.9*n	0.9*n	0.9*n	0.9*n	0.9*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	40500	40000	38850	41700	46000

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	343	342	334	318	363
No of Tags (n)	200	200	200	200	200
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	739	692	688	694	710
No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1109	1122	1085	1059	1095
No of Tags (n)	400	400	400	400	400
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1460	1432	1390	1436	1490
No of Tags (n)	500	500	500	500	500
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	1848	1850	1773	1750	1825
No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	3875	3660	3650	3680	3750
No of Tags (n)	2000	2000	2000	2000	2000
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	7480	7560	7370	7300	7600
No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	20050	19100	18950	19000	19000
No of Tags (n)	10000	10000	10000	10000	10000
Remaining Tags (TH1*n)	0.8*n	0.8*n	0.8*n	0.8*n	0.8*n
New frame size (R1*N)	0.9*N	0.8*N	0.7*N	0.6*N	0.5*N
Number of slots needed	41000	39400	39300	38000	38500

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	331	329	320	345	454
No of Tags (n)	200	200	200	200	200
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	666	664	648	722	860
No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1004	1007	1009	1085	1370
No of Tags (n)	400	400	400	400	400
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1370	1362	1340	1406	1730
No of Tags (n)	500	500	500	500	500
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1728	1698	1700	1713	2408
No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	3470	3460	3438	3610	4430
No of Tags (n)	2000	2000	2000	2000	2000
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	7180	6950	6825	7110	8200
No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	18175	17700	17500	17350	20650
No of Tags (n)	10000	10000	10000	10000	10000
Remaining Tags (TH1*n)	0.7*n	0.7*n	0.7*n	0.7*n	0.7*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	37125	35650	35000	35000	39550

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	323	314	304	325	358
No of Tags (n)	200	200	200	200	200
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	671	651	623	662	730
No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	983	975	964	1037	1142
No of Tags (n)	400	400	400	400	400
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1380	1348	1310	1370	1534
No of Tags (n)	500	500	500	500	500
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1769	1655	1669	1705	1899
No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	3503	3400	3350	3410	3858
No of Tags (n)	2000	2000	2000	2000	2000
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	7075	6920	6850	6920	7910
No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	18563	17450	17438	17350	19400
No of Tags (n)	10000	10000	10000	10000	10000
Remaining Tags (TH1*n)	0.6*n	0.6*n	0.6*n	0.6*n	0.6*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	37300	35350	34750	35000	39175

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	314	312	308	294	302
No of Tags (n)	200	200	200	200	200
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	674	646	603	607	621
No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1005	978	926	900	912
No of Tags (n)	400	400	400	400	400
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1360	1282	1245	1210	1232
No of Tags (n)	500	500	500	500	500
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1679	1658	1600	1510	1516
No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	3470	3315	3200	3110	3170
No of Tags (n)	2000	2000	2000	2000	2000
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	7165	6780	6550	6180	6215
No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	18175	17100	16438	15700	15825
No of Tags (n)	10000	10000	10000	10000	10000
Remaining Tags (TH1*n)	0.5*n	0.5*n	0.5*n	0.5*n	0.5*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	36175	35100	33125	31300	31390

No of Tags (n)	100	100	100	100	100
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	325	313	310	298	304
No of Tags (n)	200	200	200	200	200
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	683	665	625	616	629
No of Tags (n)	300	300	300	300	300
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1037	989	931	912	923
No of Tags (n)	400	400	400	400	400
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1372	1318	1260	1226	1240
No of Tags (n)	500	500	500	500	500
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	1698	1675	1631	1555	1579
No of Tags (n)	1000	1000	1000	1000	1000
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	3548	3375	3213	3150	3180
No of Tags (n)	2000	2000	2000	2000	2000
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	7195	6790	6580	6220	6265
No of Tags (n)	5000	5000	5000	5000	5000
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	18198	17275	16550	15900	15975
No of Tags (n)	10000	10000	10000	10000	10000
Remaining Tags (TH1*n)	0.4*n	0.4*n	0.4*n	0.4*n	0.4*n
New frame size (R1*N)	0.7*N	0.6*N	0.5*N	0.4*N	0.3*N
Number of slots needed	36245	35500	33310	31700	31900

Appendix B

Sample of the data collected to calculate the threshold values, TH2 and R2

No of Tags (n)	100	100	100
Remaining Tags (TH2*n)	0.5*n	0.5*n	0.5*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	299	423	6938
No of Tags (n)	500	500	500
Remaining Tags (TH2*n)	0.5*n	0.5*n	0.5*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	1508	1795	9860
No of Tags (n)	1000	1000	1000
Remaining Tags (TH2*n)	0.5*n	0.5*n	0.5*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	3030	3680	23055
No of Tags (n)	2000	2000	2000
Remaining Tags (TH2*n)	0.5*n	0.5*n	0.5*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	6090	6760	25020
No of Tags (n)	5000	5000	5000
Remaining Tags (TH2*n)	0.5*n	0.5*n	0.5*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	15300	16950	61150
No of Tags (n)	100	100	100
Remaining Tags (TH2*n)	0.4*n	0.4*n	0.4*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	291	329	2349
No of Tags (n)	500	500	500
Remaining Tags (TH2*n)	0.4*n	0.4*n	0.4*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	1555	1750	7535
No of Tags (n)	1000	1000	1000
Remaining Tags (TH2*n)	0.4*n	0.4*n	0.4*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	2985	3430	12685
No of Tags (n)	2000	2000	2000
Remaining Tags (TH2*n)	0.4*n	0.4*n	0.4*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N

Number of slots needed	6000	6840	22040
No of Tags (n)	5000	5000	5000
Remaining Tags (TH2*n)	0.4*n	0.4*n	0.4*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	15450	16850	65850
No of Tags (n)	100	100	100
Remaining Tags (TH2*n)	0.3*n	0.3*n	0.3*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	279	288	497
No of Tags (n)	500	500	500
Remaining Tags (TH2*n)	0.3*n	0.3*n	0.3*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	1488	1470	2663
No of Tags (n)	1000	1000	1000
Remaining Tags (TH2*n)	0.3*n	0.3*n	0.3*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	2910	2980	4615
No of Tags (n)	2000	2000	2000
Remaining Tags (TH2*n)	0.3*n	0.3*n	0.3*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	5970	5940	10040
No of Tags (n)	5000	5000	5000
Remaining Tags (TH2*n)	0.3*n	0.3*n	0.3*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	15175	15050	26225
No of Tags (n)	100	100	100
Remaining Tags (TH2*n)	0.2*n	0.2*n	0.2*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	281	261	309
No of Tags (n)	500	500	500
Remaining Tags (TH2*n)	0.2*n	0.2*n	0.2*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	1425	1375	1470
No of Tags (n)	1000	1000	1000
Remaining Tags (TH2*n)	0.2*n	0.2*n	0.2*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	2925	2755	2925
No of Tags (n)	2000	2000	2000

Remaining Tags (TH2*n)	0.2*n	0.2*n	0.2*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	5860	5640	5800
No of Tags (n)	5000	5000	5000
Remaining Tags (TH2*n)	0.2*n	0.2*n	0.2*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	14925	14200	14575
No of Tags (n)	100	100	100
Remaining Tags (TH2*n)	0.1*n	0.1*n	0.1*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	287	268	279
No of Tags (n)	500	500	500
Remaining Tags (TH2*n)	0.1*n	0.1*n	0.1*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	1470	1395	1410
No of Tags (n)	1000	1000	1000
Remaining Tags (TH2*n)	0.1*n	0.1*n	0.1*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	2965	2790	2815
No of Tags (n)	2000	2000	2000
Remaining Tags (TH2*n)	0.1*n	0.1*n	0.1*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	5960	5640	5700
No of Tags (n)	5000	5000	5000
Remaining Tags (TH2*n)	0.1*n	0.1*n	0.1*n
New frame size (R2*N)	0.3*N	0.2*N	0.1*N
Number of slots needed	15075	14270	14375