

# University of Wollongong - Research Online

## Thesis Collection

Title: On the suitability of RFID anti-collision protocols in energy constrained environments

Author: Dheeraj K Klair

Year: 2009

Repository DOI:

### Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material.

Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

**Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.**

Research Online is the open access repository for the University of Wollongong. For further information contact the UOW Library: [research-pubs@uow.edu.au](mailto:research-pubs@uow.edu.au)

*University of Wollongong Thesis Collections*

*University of Wollongong Thesis Collection*

---

*University of Wollongong*

*Year 2009*

---

On the suitability of RFID anti-collision  
protocols in energy constrained  
environments

Dheeraj K. Klair  
University of Wollongong

Klair, Dheeraj K, On the suitability of RFID anti-collision protocols in energy constrained environments, PhD thesis, School of Electrical, Computing and Telecommunications Engineering, University of Wollongong, 2009. <http://ro.uow.edu.au/theses/809>

This paper is posted at Research Online.

<http://ro.uow.edu.au/theses/809>

## **NOTE**

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

## **UNIVERSITY OF WOLLONGONG**

### **COPYRIGHT WARNING**

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

# On the Suitability of RFID Anti-Collision Protocols in Energy Constrained Environments

A thesis submitted in fulfilment of the  
requirements for the award of the degree

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

Dheeraj K Klair  
Masters of Engineering (Microelectronics),  
Bachelor of Technology (Electronics and Communications)

SCHOOL OF ELECTRICAL, COMPUTER  
AND TELECOMMUNICATIONS ENGINEERING  
2009



In memory of my mother

# Abstract

Radio Frequency Identification (RFID) has revolutionized the asset tracking industry, with applications ranging from automated checkout to monitoring the medication intakes of elderlies. Its wide acceptance and applicability has spurred researchers to create novel RFID applications. One promising development is equipping nodes in a Wireless Sensor Network (WSN) with a RFID reader to create a distributed, self-configuring, ad hoc wireless network for tracking objects with an RFID tag.

A key problem in RFID-enhanced WSNs is the limited battery life of sensor nodes, which imposes a severe energy constraint on communication protocols. To put this into perspective, this thesis shows that the energy consumed by an RFID reader to read a single 96-bit tag is higher than a sensor node transmitting and receiving 96-bits of data. Moreover, in practice, an RFID reader has to read multiple tags in its interrogation zone, all of which may reply simultaneously. As a result, the RFID reader experiences collisions and unnecessary energy wastage that ultimately shortens a WSN's lifetime. For these reasons, it is imperative that a comprehensive study on the energy efficiency of existing RFID tag reading or anti-collision protocols be conducted in order to determine their suitability for use in RFID-enhanced WSNs.

This thesis, therefore, investigates the energy efficiency of Aloha based RFID anti-collision protocols. These protocols have low memory and bandwidth requirements, adaptive to changing tag population, and a small number of reader

to tag commands; thereby, making them easy to implement on sensor nodes. Using analytical and simulation studies, this thesis shows that collisions and idle listening to be the key causes of energy consumption. Idle listening consumes a significant amount of energy, especially when the number of tags is low, but as the number of tags increases, collisions become the main cause of energy expenditure.

Another major finding is that existing anti-collision protocols are unable to monitor tags in an energy efficient manner. Specifically, in order to monitor tags, these protocols must undergo the collision resolution process repeatedly. This problem is particularly acute when tag population changes frequently. Hence, there is a clear need for energy efficient protocols that can determine new and old tags quickly. To this end, this thesis is the first to propose ResMon, an anti-collision protocol that is designed to be energy efficient during identification and monitoring. Extensive simulation studies show ResMon's energy consumption to be significantly lower than state of the art framed Aloha variants; thus, making it ideal for use in RFID-enhanced WSNs.

# Statement of Originality

This is to certify that the work described in this thesis is entirely my own, except where due reference is made in the text.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Dheeraj K Klair

18<sup>th</sup> May, 2009

# Acknowledgments

There are a number of people who have been a consistent support during my doctorate study. First of all, I would like to express my deepest gratitude to my supervisor Dr Kwan-Wu Chin. It has been a great learning experience working with him. Despite being overly busy, he has always shown keenness to my research work that kept me motivated at every phase of my research. I am particularly thankful for his meticulous reviews and prompt feedbacks that have greatly enhanced the quality of my thesis. I also like to thank him for giving useful insights in developing the simulation model used in my paper on *ResMon*.

I like to thank my co-supervisor Dr Raad Raad for his valuable guidance during early stages of my research work. I also appreciate the moral support he has given during the final stages of my thesis.

I like to acknowledge and thank the support of Australian Research Council, who has funded this work under grant number DP0559769. In particular, I appreciate the generous travel support that have allowed me to visit a number of prestigious international conferences.

I like to thank all anonymous reviewers at conferences and journals for their time and efforts they spend in giving valuable feedbacks. Their suggestions have certainly raised the quality of my publications.

To my parents, I am eternally grateful for their constant encouragement and

moral support during my research work. I dedicate this thesis to my mother who passed away on 27<sup>th</sup> October 2008. Her unflagging beliefs in my abilities has always been a great source of motivation for me. I also like to thank my brother and sister for all their love and support.

Heartfelt thanks to my wife for her unswerving love and support during the final stages of my thesis. She has always been a patient listener of my research thoughts. I am indebted to her for the time and efforts she spent in taking care of me while I was finishing my thesis. I also like to acknowledge the time and effort she took to craft an attractive coloring scheme for one of my conference posters.

Finally, I like to thank my friends and colleagues. They include Keni, Jerry, Lu, Quentin, Brad, Jagan, Ranjit, Darek, Ian, Alex, Ricardo, Chao, Jack, Wenbing, Navtej, Dev, Kumar and Sze Yen. They have been a wonderful social support during the lonesome research work. Special thanks goes to Philippe, Matthew, Giang, Sewei, Roland, Ankur and Sean for supporting me during some critical personal issues.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>  | <b>1</b>  |
| 1.1      | RFID . . . . .   | 1         |
| 1.2      | Wireless Sensor Networks . . . . .                           | 3         |
| 1.3      | Motivation . . . . .   | 7         |
| 1.4      | Research Statement . . . . .                                 | 11        |
| 1.5      | Contributions . . . . .                                      | 11        |
| 1.5.1    | Publications . . . . .                                       | 13        |
| 1.6      | Thesis structure . . . . .                                   | 14        |
| <br>     |  |           |
| <b>2</b> | <b>RFID Anti-Collision Protocols</b>                         | <b>16</b> |
| 2.1      | Anti-Collision Protocols for Tag Collision Problem . . . . . | 16        |
| 2.2      | Aloha Based Protocols . . . . .                              | 18        |
| 2.2.1    | Pure Aloha (PA) . . . . .                                    | 18        |
| 2.2.2    | Slotted Aloha (SA) . . . . .                                 | 20        |
| 2.2.3    | Framed Slotted Aloha (FSA) . . . . .                         | 22        |
| 2.2.4    | Discussions . . . . .  | 31        |
| 2.3      | Tree Based Protocols . . . . .                               | 31        |
| 2.3.1    | Tree Splitting . . . . .                                     | 33        |

| <b>CONTENTS</b>   | <b>ix</b> |
|---|-----------|
| 2.3.2 Query Tree Algorithms . . . . .                               | 37        |
| 2.3.3 Binary Search (BS) . . . . .                                  | 44        |
| 2.3.4 Bitwise Arbitration (BTA) Algorithms . . . . .                | 45        |
| 2.3.5 Discussions . . . . .   | 50        |
| 2.4 Hybrid protocols . . . . .                                      | 50        |
| 2.4.1 Tree Slotted Aloha (TSA) . . . . .                            | 50        |
| 2.4.2 Hybrid Query Tree (HQT) Protocol . . . . .                    | 53        |
| 2.4.3 Hybrid Randomized Protocol . . . . .                          | 53        |
| 2.4.4 Hash-tree Protocol . . . . .                                  | 54        |
| 2.4.5 Discussions . . . . .   | 54        |
| 2.5 RFID Anti-Collision Standards . . . . .                         | 55        |
| 2.6 Summary . . . . .   | 58        |
| <b>3 Pure and Slotted Aloha Based RFID Anti-Collision Protocols</b> | <b>59</b> |
| 3.1 System and Analytical Model . . . . .                           | 60        |
| 3.2 Research Methodology . . . . .                                  | 62        |
| 3.3 Delay Equations . . . . .                                       | 63        |
| 3.3.1 Pure Aloha and Slotted Aloha . . . . .                        | 64        |
| 3.3.2 Pure and Slotted Aloha with Muting . . . . .                  | 66        |
| 3.3.3 Pure and Slotted Aloha with Slow Down . . . . .               | 67        |
| 3.3.4 Pure Aloha (Fast Mode) . . . . .                              | 68        |
| 3.3.5 Slotted Aloha Variants with Early End . . . . .               | 71        |
| 3.3.6 Putting It All Together . . . . .                             | 71        |
| 3.4 Results . . . . .   | 72        |
| 3.4.1 Average Energy Consumed to Read One Tag . . . . .             | 73        |
| 3.4.2 Average Energy Wasted in Collisions to Read One Tag . . . . . | 76        |



|          |   |            |
|----------|---|------------|
| 3.4.3    | Average Energy Consumed due to Idle Listening . . . . .         | 80         |
| 3.4.4    | Battery Lifetime . . . . .                                      | 82         |
| 3.4.5    | Battery Wastage . . . . .                                       | 82         |
| 3.5      | Related Works . . . . .   | 85         |
| 3.6      | Conclusions . . . . .   | 85         |
| <b>4</b> | <b>Tag Estimation Functions</b>                                 | <b>88</b>  |
| 4.1      | Motivation . . . . .  | 89         |
| 4.2      | Tag Estimation Functions . . . . .                              | 90         |
| 4.2.1    | Vogt Functions . . . . .  | 91         |
| 4.2.2    | Cha et al. Functions . . . . .                                  | 93         |
| 4.2.3    | Zhen et al. Functions . . . . .                                 | 94         |
| 4.3      | System Model . . . . .  | 95         |
| 4.4      | Research Methodology . . . . .                                  | 95         |
| 4.5      | Results . . . . .   | 97         |
| 4.5.1    | Statistical Analysis . . . . .                                  | 98         |
| 4.5.2    | Comparisons . . . . .   | 104        |
| 4.6      | Conclusion . . . . .  | 108        |
| <b>5</b> | <b>Framed Slotted Aloha Based RFID Anti-Collision Protocols</b> | <b>111</b> |
| 5.1      | FSA . . . . .   | 111        |
| 5.1.1    | Frame Adjustment . . . . .                                      | 113        |
| 5.2      | Research Methodology . . . . .                                  | 114        |
| 5.2.1    | BFSA . . . . .  | 115        |
| 5.2.2    | DFSA . . . . .  | 119        |
| 5.2.3    | EDFSA . . . . .   | 121        |

|  |            |
|--|------------|
| <b>CONTENTS</b>                                  | <b>xi</b>  |
| 5.3 System Model . . . . .                       | 122        |
| 5.4 Results . . . . .                            | 123        |
| 5.4.1 Low tag densities ( $n < 100$ ) . . . . .  | 123        |
| 5.4.2 High tag densities ( $n > 100$ ) . . . . . | 128        |
| 5.5 Conclusion . . . . .                         | 134        |
| <b>6 ResMon</b>                                  | <b>136</b> |
| 6.1 ResMon . . . . .                             | 137        |
| 6.1.1 Frames . . . . .                           | 137        |
| 6.1.2 ResMon Operation . . . . .                 | 143        |
| 6.1.3 New Tags . . . . .                         | 144        |
| 6.1.4 Departing Tags . . . . .                   | 145        |
| 6.2 Simulation Methodology . . . . .             | 147        |
| 6.3 Results . . . . .                            | 149        |
| 6.3.1 Identification Phase . . . . .             | 149        |
| 6.3.2 Monitoring Phase . . . . .                 | 150        |
| 6.3.3 Realistic Scenarios . . . . .              | 151        |
| 6.4 Conclusion . . . . .                         | 154        |
| <b>7 Conclusions</b>                             | <b>157</b> |
| <b>Bibliography</b>                              | <b>160</b> |

# List of Figures

|      |  |    |
|------|--|----|
| 1.1  | Reader and tags interactions. . . . .                              | 2  |
| 1.2  | Wireless sensor network. . . . .                                   | 4  |
| 1.3  | RFID-enhanced wireless sensor network. . . . .                     | 6  |
| 1.4  | Tag collision problem. . . . .                                     | 11 |
| 2.1  | Classification of tag reading or anti-collision protocols. . . . . | 17 |
| 2.2  | Pure Aloha with muting. . . . .                                    | 19 |
| 2.3  | Pure Aloha with slow down. . . . .                                 | 20 |
| 2.4  | Pure Aloha with fast mode. . . . .                                 | 21 |
| 2.5  | Pure Aloha with fast mode and muting. . . . .                      | 22 |
| 2.6  | Pure Aloha with fast mode and slow down. . . . .                   | 23 |
| 2.7  | Slotted Aloha with early end. . . . .                              | 24 |
| 2.8  | Basic Tree splitting (BTS) algorithm. . . . .                      | 34 |
| 2.9  | The QT Algorithm. . . . .  | 39 |
| 2.10 | QT with Shortcutting. . . . .                                      | 40 |
| 2.11 | AQT- new tags 111 and 000. . . . .                                 | 42 |
| 2.12 | AQT-departed tag 111. . . . .                                      | 43 |
| 2.13 | The BS Protocol. . . . .   | 44 |

|   |     |
|---|-----|
| 2.14 ID-BTS. . . . .  | 47  |
| 2.15 Bit Query (BQ). . . . .  | 49  |
| 3.1 Average energy consumed to read one tag successfully for Pure Aloha variants, $\lambda = 20$ , $K = 5$ , PA = Pure Aloha, . . . . .             | 74  |
| 3.2 Average energy consumed to read one tag successfully for Slotted Aloha Variants, $\lambda = 20$ , $K = 5$ , SA = Slotted Aloha. . . . .         | 75  |
| 3.3 Average Arrival Delay . . . . .   | 76  |
| 3.4 Mean Contention Delay for Aloha Variants, PA = Pure Aloha and SA = Slotted Aloha. . . . .   | 77  |
| 3.5 Comparing energy consumed to read one tag successfully for Pure and Slotted Aloha Variants, PA = Slotted Aloha and SA = Slotted Aloha . . . . . | 78  |
| 3.6 System Throughput Versus Number of Tags for Aloha Variants, PA = Pure Aloha and SA = Slotted Aloha. . . . .                                     | 79  |
| 3.7 Average energy wasted in collisions before a successful ID reception, PA = Pure Aloha and SA = Slotted Aloha. . . . .                           | 81  |
| 3.8 Average energy spent in idle listening to read one tag, PA = Pure Aloha and SA = Slotted Aloha. . . . .   | 83  |
| 3.9 Battery Lifetime, PA = Pure Aloha and SA = Slotted Aloha. . . . .   | 84  |
| 3.10 Battery wasted from collisions and idle listening, PA = Pure Aloha and SA = Slotted Aloha. . . . .   | 86  |
| 4.1 System efficiency versus number of tags. . . . .  | 91  |
| 4.2 DFSAV-I error analysis. . . . .   | 99  |
| 4.3 DFSAZ error analysis. . . . .   | 100 |
| 4.4 DFSAC-I error analysis. . . . .   | 102 |
| 4.5 DFSAC-II error analysis. . . . .  | 103 |
| 4.6 DFSAV-II error analysis. . . . .  | 105 |
| 4.7 Mean error versus number of tags. . . . .   | 107 |

|      |  |     |
|------|--|-----|
| 4.8  | Variability versus number of tags. . . . .   | 108 |
| 4.9  | Skew versus number of tags. . . . .  | 109 |
| 4.10 | Kurtosis versus number of tags. . . . .  | 110 |
| 5.1  | Round structure for FSA variants. $S_i$ indicates the slot number. . . . .   | 112 |
| 5.2  | Total energy consumed versus number of tags for FSA variants<br>in low tag density environments. . . . .                   | 125 |
| 5.3  | Total energy wasted in idle listening versus number of tags for<br>FSA variants in low tag density environments. . . . .   | 126 |
| 5.4  | Total energy wasted in collisions versus number of tags for FSA<br>variants in low tag density environments. . . . .       | 127 |
| 5.5  | Battery lifetime. . . . .  | 128 |
| 5.6  | Battery energy wasted to read $n$ tags. . . . .  | 129 |
| 5.7  | Total energy consumed versus number of tags for DFSA variants<br>in high tag density environments. . . . .                 | 130 |
| 5.8  | Total energy wasted in idle listening versus number of tags for<br>DFSA variants in high tag density environments. . . . . | 131 |
| 5.9  | Total energy wasted in collisions versus number of tags for DFSA<br>variants in low tag density environments. . . . .      | 132 |
| 5.10 | Battery lifetime. . . . .  | 133 |
| 5.11 | Battery energy wasted to read $n$ tags. . . . .  | 134 |
| 6.1  | Reservation frame structure. . . . .   | 138 |
| 6.2  | Collision detection using Manchester encoding. . . . .   | 139 |
| 6.3  | Body frame structure. $S_i = BFSC - 1$ when $SO = 0$ . $S_i =$<br>$BFSC - 1 - SO$ when $SO > 0$ . . . . .                  | 141 |
| 6.4  | Monitor frame structure. . . . .   | 144 |
| 6.5  | Energy wasted in collisions during the identification phase. . . . .   | 150 |
| 6.6  | Energy consumed to read $n$ tags during the identification phase. . . . .  | 151 |

|      |   |     |
|------|---|-----|
| 6.7  | Energy consumed to read n tags during the identification phase. | 152 |
| 6.8  | Energy consumed while monitoring a given tag set. . . . .       | 153 |
| 6.9  | Frequency of identification and monitoring round = One Day. .   | 154 |
| 6.10 | Frequency of identification and monitoring round = One Hour. .  | 155 |
| 6.11 | Frequency of identification and monitoring round = Half Hour. . | 156 |

# List of Tables

|      |  |    |
|------|--|----|
| 1.1  | Sensor nodes hardware. . . . .   | 5  |
| 1.2  | RFID readers power consumption . . . . .   | 8  |
| 1.3  | Sensor nodes power consumption [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]  | 10 |
| 2.1  | Optimal frame sizes for a given tag range. . . . .   | 26 |
| 2.2  | EDFSA frame sizes. $n$ denotes the number of tags, $N$ is the frame size, and $M$ is the number of tag groups. . . . . | 30 |
| 2.3  | A comparison of Aloha, Slotted Aloha and Framed Aloha protocols. . . . .   | 32 |
| 2.4  | Tag's counter in the BTS algorithm. . . . .  | 35 |
| 2.5  | Adaptive Binary Splitting (ABS) - TSC, PSC and ACS values. .   | 36 |
| 2.6  | Reader's stack corresponding to Figure 11. . . . .   | 38 |
| 2.7  | Reader stack and tags counter. . . . .   | 46 |
| 2.8  | A comparison of tree protocols. . . . .  | 51 |
| 2.9  | Comparisons between tree and Aloha based algorithms. . . . .   | 52 |
| 2.10 | ISO RFID Standards [11] [12] [13] [14] [15]. . . . .   | 56 |
| 2.11 | EPC RFID Standards [16] [17] [18]. . . . .   | 57 |
| 2.12 | Proprietary Protocols [19] [20] [21]. . . . .  | 58 |

|     |  |     |
|-----|--|-----|
| 3.1 | Energy consumption of Pure/Slotted Aloha. $E_{Success}$ - Average energy consumed to read one tag successfully, $E_{Collisions}$ - Average energy consumed in collision resolution before a tag is read, $E_{Idle}$ - Average energy consumed in idle listening before a successful tag is read, $N_{Battery\_Life}$ - The number of tags a battery can read, $B_{Waste}$ - Battery wastage due to anti-collision. $x = 1$ for Slotted Aloha and $x = 2$ for Pure Aloha, $E_{Collision} = E_{Success} - E_{Idle} - \psi T$ , $\alpha = \frac{K+1}{2}, \beta = \frac{K-1}{2}$ . . . . . | 72  |
| 4.1 | Summary of statistics for error distribution plots . . . . .   | 106 |
| 5.1 | Optimal frame sizes for a given tag range. . . . .   | 113 |
| 5.2 | $N$ and $M$ values for EDFSA . . . . .   | 114 |
| 6.1 | Optimal frame sizes for a given tag range. . . . .   | 139 |
| 6.2 | Removal of collided slots from $B_{Frame}$ . . . . .   | 142 |
| 6.3 | Departing tags example. . . . .  | 146 |
| 6.4 | Simulation parameters. . . . .   | 148 |
| 6.5 | Summary of results involving realistic scenarios. . . . .  | 155 |



# List of Abbreviations

|       |                                       |
|-------|---------------------------------------|
| ABS   | Adaptive Binary Splitting             |
| ACK   | Acknowledgement                       |
| AQT   | Adaptive Query Tree                   |
| ASC   | Allocated Slot Counter                |
| BBT   | Bit-by-Bit                            |
| BF    | Body Frame                            |
| BFSA  | Basic Framed Slotted Aloha            |
| BFSC  | Body Frame Slot Counter               |
| BQ    | Bit Query                             |
| BS    | Binary Search                         |
| BTA   | Bitwise Arbitration                   |
| BTS   | Basic Tree Splitting                  |
| CDMA  | Code Division Multiple Access         |
| CQ    | Candidate Queue                       |
| CRC   | cyclic Redundancy Check               |
| DFSA  | Dynamic Framed Slotted Aloha          |
| DoD   | Department of Defense                 |
| EBBT  | Enhanced Bit by Bit Binary Tree       |
| EBS   | Enhanced Binary Splitting             |
| EBSA  | Enhanced Binary Search Algorithm      |
| EDFSA | Enhanced Dynamic Framed Slotted Aloha |
| EOF   | End-of-Frame                          |
| FDMA  | Frequency Division Multiple Access    |

---

|        |                                    |
|--------|------------------------------------|
| FSA    | Framed Slotted Aloha               |
| HQT    | Hybrid Query Tree Protocol         |
| ID     | Identifier                         |
| ID-BTS | ID-Binary Tree Stack               |
| IQT    | Improved Query Tree                |
| MAS    | Multi-Slotted with Assigned Slots  |
| MBBT   | Modified Bit by Bit Binary Tree    |
| MCU    | Microcontroller Unit               |
| MEMS   | Micro-Electro-Mechanical Systems   |
| MF     | Monitor Frame                      |
| MS     | Multi-Slotted                      |
| MSB    | Most Significant Bit               |
| MSS    | Multi-Slotted with Selective Sleep |
| NACK   | Negative Acknowledgement           |
| PA     | Pure Aloha                         |
| PC     | Personal Computer                  |
| PN     | Pseudo-Random Sequence             |
| PSC    | Progressed Slot Counter            |
| Q      | queue                              |
| QT     | Query Tree                         |
| QT-im  | Query-Tree Incremental-Matching    |
| QT-sl  | Query-Tree Short-Long              |
| QTR    | Query Tree Based Reservation       |
| RF     | Radio Frequency                    |
| RFID   | Radio Frequency IDentification     |
| RH-QT  | Randomized Hashing Query Tree      |
| RN16   | 16-bit Random Number               |
| RTF    | Reader-Talk-First                  |
| SA     | Slotted Aloha                      |
| SBPP   | Scanning Based Pre-Processing      |
| SDMA   | Space Division Multiple Access     |
| SO     | Slot Offset                        |

|      |                               |
|------|-------------------------------|
| SOF  | Start-of-Frame                |
| TDMA | Time Division Multiple Access |
| TS   | Tree Splitting                |
| TSA  | Tree Slotted Aloha            |
| TTF  | Tag-Talk-First                |
| UFSC | Unique Frame Slot Counter     |
| WSN  | Wireless Sensor Network       |

# Chapter 1

## Introduction

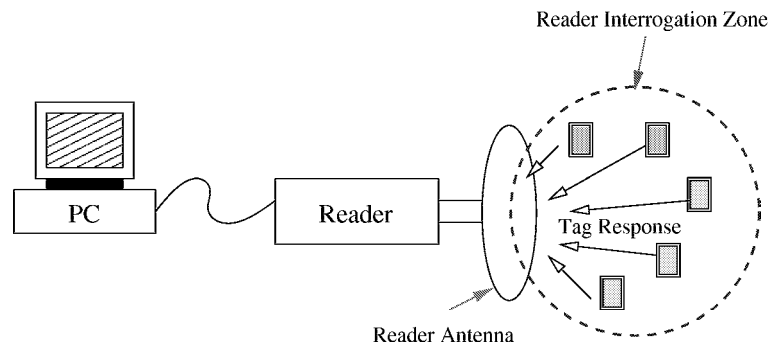
### 1.1 RFID

In recent years, RFID has gained enormous publicity, especially after its adoption by Wal-Mart, Tesco, and the US Department of Defense (DoD) [22] [23] [24]. These organizations maintain that RFID technologies reduce logistical overheads, costs, and product losses. According to IdTech [25], the value of the entire RFID market could rise to 24.5 billion in 2015. Moreover, In-Stat [26] predicts that by 2010, more than 33 billion tags will be produced globally, which is 25 times the number of tags produced in the year 2005.

RFID is an object identification technology that is far more advanced than conventional barcodes. Conceptually, RFID relies on magnetic or electromagnetic field for identifying objects and do not require reader and tags to be in the line of sight. In contrast, barcodes are read optically, and do require line of sight. Another distinctive advantage of RFID is its ability to identify multiple tags. Also, RFID tags are available in a variety of form factors, shapes, and sizes. Moreover, data stored in an RFID tag can be altered by an RFID reader, whereas barcodes do not change once printed. These distinctive RFID features along with falling system implementation costs have sped up the adoption of RFID in various industries [27] [28]. Currently, RFID applications include theft prevention of automobiles and merchandises, automatic toll collection, supply

chain, and elder healthcare to name a few [29] [30] [27].

A typical RFID system consists of an RFID reader, one or more RFID tags and a Personal Computer (PC) for data processing; as shown in Figure 1.1. The RFID reader is a powerful device with ample memory and computational power. It has four main functions: energize tags, carrier signal generation, broadcast messages or commands, and decode a modulated signal. To read tags, the reader energizes them by emitting a time-varying magnetic or electromagnetic field, which also contains a carrier signal to be used by tags to transmit their identifier (ID) back to the reader. In other words, the carrier signal not only energizes tags, but also receives data from tag. The reader then demodulates the received signal to retrieve a tag's ID. A reader can also be fitted with an additional interface to transmit its stored data to a computer or programmable logic controller [31]. Lastly, a reader uses an anti-collision algorithm to arbitrate tag replies.



**Figure 1.1** Reader and tags interactions.

Tags are the basic building block of an RFID system. A tag consists of a small electronic chip that contains a radio receiver, a modulator for sending responses back to the reader, and control logic. RFID tags that do not contain a silicon chip are called chip-less tags, and they promise huge cost savings since they can be printed directly on products [32].

There are three types of tags: passive, active and semi-passive [27] [28]. Passive tags have limited computational capacity, no ability to sense the channel, detect

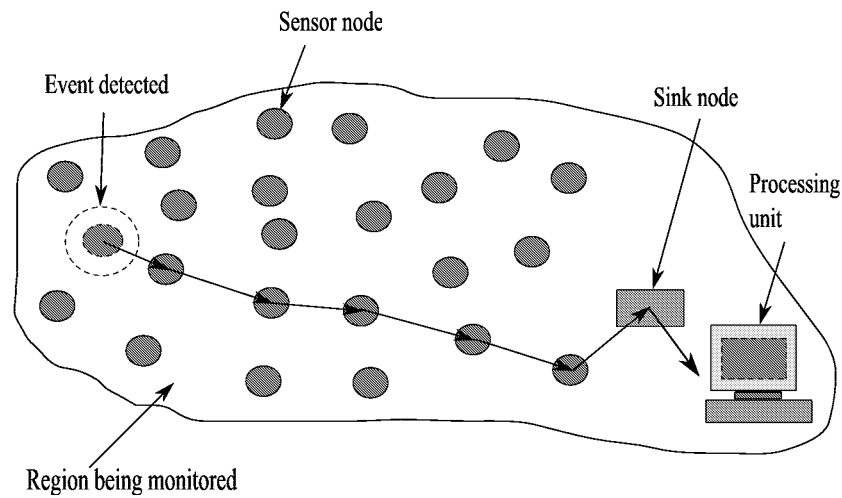
collisions, and communicate with each other. Semi-passive tags behave in a similar manner to passive tags, but have the advantage of an on-board power source that can be used to energize their microchip. In terms of cost, active tags are the most expensive, followed by passive and semi passive tags.

## 1.2 Wireless Sensor Networks

The development of low cost, multifunctional sensor nodes have become popular, spurred by the advances in wireless communications, micro-electro-mechanical systems (MEMS) technologies and electronics over the last few years. These sensors are small in size and are able to sense, process data, and communicate with each other, typically over a radio frequency (RF) channel [33]. More importantly, they can be used to create wireless sensor networks that have applications in habitat monitoring, asset management, intelligent homes, and health care [33] [34].

Figure 1.2 shows a typical wireless sensor network (WSN) comprising of randomly deployed sensor nodes, a sink node and a processing unit. The placement of these sensors need not be engineered or predetermined, thus allowing deployment in hard to reach terrains [34]. Moreover, sensor nodes are mostly unattended after deployment, permitting neither upgrade of energy sources or troubleshooting. Consequently, nodes are abandoned after they exhaust their power sources [34]. When deployed, sensor nodes self-organize to create a network without relying on any existing infrastructures. Once sensor nodes are inter-connected, the application on each node is activated. Following that, whenever an event is detected, such as a temperature change, one or more sensor nodes transmit the event to the closest sink node, which then transmits it to a processing unit. Conversely, the processing unit can also send queries/commands to register its interest in an event or to collect data stored at sensor nodes [35].

A sensor node consists of a battery, a radio transceiver, microcontroller unit (MCU), memory and transducers or sensors [1] [34]. Table 1.1 summarizes



**Figure 1.2** Wireless sensor network.

the specification of various commercial sensor nodes. BTnode [6], IRIS [8], TELOSB [10], Tmote Sky [2], MICA2 [3], and MICAz [5] use 2AA batteries. On the other hand, Imote2 [9] and MICA2DOT [4] rely on a 3 AAA and 3V coin battery respectively. These sensor nodes either use a Chipcon CC2420 [36] or a Chipcon CC 1000 [37] radio. The key difference is that, CC2420 offers a data rate of 250 kbps and operates at 2400 MHz. The CC 1000 radio, on the other hand, has a data rate of 76.8 kbps and operates in the 300-1000 MHz frequency range. A majority of these sensor nodes use Atmel's Atmega 128L microcontroller. Lastly, sensor nodes are equipped with magnetic, seismic, magnetic, thermal, visual, infrared, acoustic, or radar sensors or transducers. Hence, they can be used to monitor a wide variety of ambient conditions such as, temperature, humidity, soil makeup, and pressure. Moreover, they can be used for continuous sensing, event detection and identification, location sensing, and control of actuators.

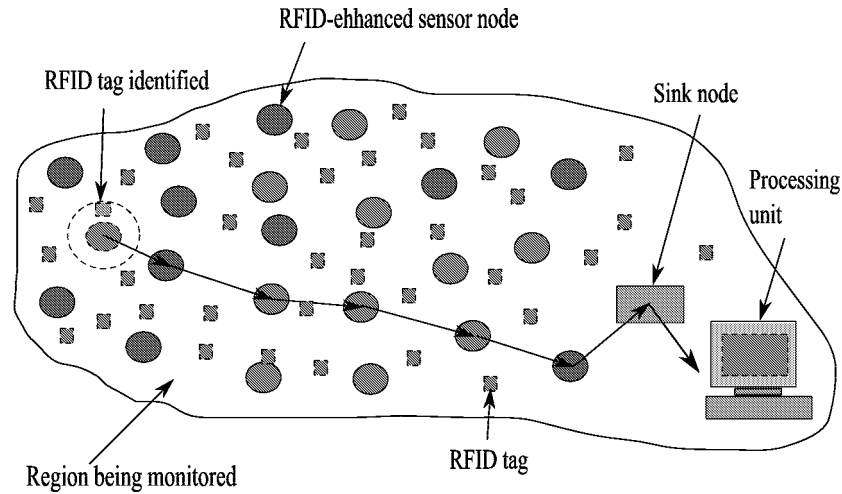
A recent development in the area of WSNs is the coupling of a RFID reader to a wireless sensor node to create RFID enhanced WSNs. For example, using SkyTek's RFID reader [38] with Crossbow's MICA2Dot sensor Motes [4]. The resulting network has many applications that is a marked departure from current RFID systems, and also those in WSNs that have thus far limited to

Table 1.1 Sensor nodes hardware.

| Sensor Node   | Manufacturer | Battery       |                    | Radio           |                    |                     |            | Microcontroller          | RAM<br>(KB) | ROM<br>(KB) |
|---------------|--------------|---------------|--------------------|-----------------|--------------------|---------------------|------------|--------------------------|-------------|-------------|
|               |              | Type          | Voltage<br>(Volts) | Type            | Frequency<br>(MHz) | Data rate<br>(Kbps) | Modulation |                          |             |             |
| MICA2 [3]     | Crossbow     | 2 AA          | 2.2 - 3.3          | Chipcon CC 1000 | 300 - 1000         | 76.8                | FSK        | Atmega 128L              | 4           | 128         |
| MICA2DOT [4]  | Crossbow     | 3 V Coin cell | 2.7 - 3.3          | Chipcon CC 1000 | 300 - 1000         | 76.8                | FSK        | Atmel Atmega 128L        | 4           | 128         |
| MICAz [5]     | Crossbow     | 2 AA          | 2.7 - 3.3          | Chipcon CC2420  | 2400               | 250                 | DSSS       | Atmel Atmega 128L        | 4           | 128         |
| IRIS [8]      | Crossbow     | 2 AA          | 2.7-3.3            | Chipcon CC2420  | 2400               | 250                 | DSSS       | Atmel Atmega 128L        | 8           | 128         |
| TELOSb [10]   | Crossbow     | 2 AA          | -                  | Chipcon CC2420  | 2400               | 250                 | DSSS       | Texas Instruments MSP430 | 10          | 48          |
| Imote2 [9]    | Crossbow     | 3 AAA         | 3.2 - 4.5          | Chipcon CC2420  | 2400               | 250                 | DSSS       | Intel PXA271 CPU         | 256         | 32MB        |
| Tmote Sky [2] | Sentilla     | 2 AA          | 2.1 - 3.6          | Chipcon CC2420  | 2400               | 250                 | DSSS       | Texas Instruments MSP430 | 2           | 60          |
| Btnode [6]    | ETH Zurich   | 2 AA          | 2-3                | Chipcon CC1000  | 300 - 1000         | 76.8                | FSK        | Atmel Atmega 128L        | 64+180      | 128         |



using conventional sensing elements mentioned previously. Figure 1.3 shows an example RFID-enhanced WSN that can be deployed either indoors or outdoors. In the former, applications can range from crowd control, book tracking in libraries, smart homes, and smart offices. For example, in a smart home, items have an RFID tag that can be tracked using randomly deployed sensor nodes, thereby allowing users to manage their assets efficiently [39] [40] [41]. On the other hand, outdoor applications include disaster or habitat monitoring [42] [43] [44]. For example, in habitat monitoring, RFID-enhanced sensor nodes can be deployed on a farm to monitor tagged animals.



**Figure 1.3** RFID-enhanced wireless sensor network.

A number of organizations and researchers have also started investigating RFID-enhanced WSNs. In [30], Ho et al. outline a project to build an in-home elder healthcare system that monitors patients' medication. In a similar work, Intel [45] has developed a system called "Caregiver's Assistant" to detect an elder's activities without requiring direct observation. This is achieved by fastening RFID tags on household objects, and tracking when these objects are touched by an elder. In [46], NASA/JPL outlines a project called Sensor Webs. The goal is to develop a web of sensors for monitoring environment changes such as humidity, and take actions when events happen. Finally, BP Oil [47] is using RFID for location tracking and to sense a machine's working condition by monitoring its vibrations.

### 1.3 Motivation

A key problem in WSNs is the limited battery life of sensor nodes. This is in addition to their low memory and computational capabilities. Therefore, integrating an RFID reader to sensor nodes presents an additional load, which in turn exacerbates these sensor nodes' energy expenditure. To gain insight into this energy expenditure, the following analysis investigates the energy consumption of an RFID reader.

Table 1.2 summarizes the current and power consumptions of commercial RFID readers. Note, the power consumptions evaluation to follow assumes the reader has a 3V supply. From the table, a RFID reader can operate in three modes, i) scan, ii) idle, and iii) sleep. Out of the RFID readers listed in Table 1.2, only SkyeTek's M1-Mini [48] and M1 [49] RFID readers are designed to mate with the MICA2DOT [4] sensor node. We see that the M1-Mini and M1 readers consume 180 and 330 milli-watts respectively during scanning. In idle mode, they consume 30 and 45 milli-watts, whilst in sleep mode they consume 150 and 180 micro-watts respectively. Thus, the scanning process consumes the most power. This is true for all RFID readers in Table 1.2.

Now consider the energy consumed by an RFID reader to read one tag. The energy consumed is proportional to a reader's scanning duration, and is dependent on the time it takes a tag to transmit its ID. In other words,

$$T = \frac{ID}{data\_rate} \quad (1.1)$$

where ID is the identification code in bits and *data\_rate* is the tag's data rate in bits per second. Taking the SkyeModule M1-Mini [48] as an example, which is capable of transmitting at 26 kbps and 106 kbps, the scanning duration to read one tag is 3.6 and 0.9 milli-seconds respectively.

Table 1.2 RFID readers power consumption

| HF RFID Systems                       |            |                     |                |                |                    |                   |                   |
|---------------------------------------|------------|---------------------|----------------|----------------|--------------------|-------------------|-------------------|
| RFID Reader                           | Supply (V) | Current Consumption |                |                | Power Consumption  |                   |                   |
|                                       |            | Sleep Mode (uA)     | Idle Mode (mA) | Scan Mode (mA) | Sleep Mode (uwatt) | Idle Mode (mwatt) | Scan Mode (mwatt) |
| Skye Module M1-Mini [48]              | 3.2 – 6    | 50                  | 15             | 60             | 150                | 45                | 180               |
| Skye Module M1 [49]                   | 1.8 – 5    | 60                  | 10             | 110            | 180                | 30                | 330               |
| DR1000 Dual RFID/Bar Code Reader [50] | 5          | –                   | 100            | 400            | –                  | 500               | 2000              |
| Socket RFID Reader Card [51]          | 5          | –                   | 11             | 52             | –                  | 55                | 260               |
| Micro-1356 Multi-Protocol Reader [52] | 3 – 5      | –                   | < 1            | 80             | –                  | < 3               | 240               |
| ZU-9A Series [53]                     | 4.75 – 5.5 | –                   | –              | 230            | –                  | –                 | 1150              |
| Compact Flash RFID Reader [54]        | 3.3        | –                   | 15             | 55             | –                  | 45                | 165               |
| IO RFID Reader [55]                   | 2.7        | < 1                 | –              | 45             | < 2.7              | –                 | 121.5             |
| CFCard Transponder Reader [56]        | 3.3 or 5   | –                   | 40             | 100            | –                  | 200               | 500               |
| UHF RFID Systems                      |            |                     |                |                |                    |                   |                   |
| Skye Module M8 [57]                   | 5          | < 100               | 50             | 250 – 650      | < 500              | < 250             | 2250              |
| NANO-UHF RFID Reader [58]             | 5          | –                   | < 2            | 100            | –                  | < 10              | 500               |

Power calculated at 3V for HF and 5V for UHF RFID Systems

Then using  $T$ , the total energy consumed by a reader is,

$$E = P \times D \quad (1.2)$$

where  $P = VI$  (Watts) is the power consumed by the reader during scanning,  $V$  (Volts) is the supply,  $I$  (Amperes) is the current consumed during scanning, and  $D$  (seconds) is the scanning duration. Note,  $D = T$  for a single tag.

Applying Eq. 1.2, the energy consumed by the SkyeModule M1-Mini to receive 96 bits of ID is around 648 micro joules at 26 kbps, and 162 micro joules at 106 kbps. Similarly, for the SkyeModule M1, the energy consumed when scanning a single tag is around 1188 micro-joules at 26 kbps and 297 micro-joules at 106 kbps.

Now, let us consider the amount of energy consumed by a sensor node to transmit/receive data. Table 1.3 presents the power consumption of existing sensor nodes. In order to compare the energy consumed by a sensor node and an RFID reader, the following analysis considers the energy consumed by a MICA2DOT mote when transmitting and receiving 96 bits of data in a noise free channel. The power consumed by a MICA2DOT in reception and transmission is 27 and 52 milli-watts respectively. Its transceiver is capable of transmitting at 38.4 kbps. Therefore, using Eq. 1.1 and 1.2, the energy consumed to receive 96 bits of data is 67.5 micro-joules, and to transmit the same amount of data takes 130 micro-joules. On the other hand, the SkyeTek RFID readers' energy consumption range from 162 to 1188 micro-joules when scanning for a single 96-bits tag. Therefore, the energy expended by an RFID reader in scan mode is higher compared to the MICADOT during transmission and reception.

The analysis above shows that reading just one tag consumes a significant amount of energy. In practice, there are multiple tags in a reader's interrogation zone and tag responding simultaneously cause collisions at the reader. To mitigate these collisions, a reader must use a tag reading or an anti-collision protocol. Moreover, in RFID-enhanced WSNs, coordinating the transmissions

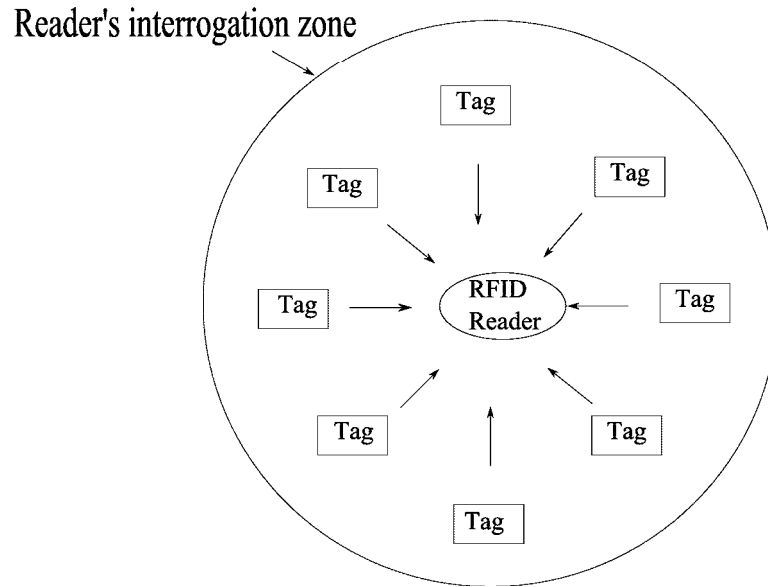
Table 1.3 Sensor nodes power consumption [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]

| Sensor Node          | –         | Radio Rx             |              |               | Radio TX     |               | MCU Idle and Radio Off |               | Data rate (kbps) |
|----------------------|-----------|----------------------|--------------|---------------|--------------|---------------|------------------------|---------------|------------------|
|                      |           | Supply (min-max) (V) | Current (mA) | Power (mwatt) | Current (mA) | Power (mwatt) | Current (uA)           | Power (uwatt) |                  |
| tmote sky (or telos) | 2.1 – 3.6 | 19.7                 | 59.1         |               | 17.4         | 52.2          | 54.5                   | 163.5         | 250              |
| mica2                | 2.2 – 3.3 | 8 – 10               | 27           |               | 25 – 27      | 52            | < 15.1                 | < 45.3        | 38.4             |
| Micat2Dot            | 2.7 – 3.3 | 8 – 10               | 27           |               | 25 – 27      | 52            | < 15.1                 | < 45.3        | 38.4             |
| micaz                | 2.7 – 3.3 | 19.7                 | 59.1         |               | 11 – 17.4    | 42.6          | < 15.1                 | < 45.3        | 250              |

of multiple sensor nodes further adds to energy expenditure.

## 1.4 Research Statement

This thesis, therefore, aims to analyze the energy efficiency of RFID tag reading protocols. Specifically, it studies protocols that minimize tag collisions [27]. As shown in Figure 1.4, a collision results when more than one tag responds to the reader at the same time, thereby causing extended identification delays. Moreover, it causes energy and bandwidth wastage.



**Figure 1.4** Tag collision problem.

## 1.5 Contributions

This thesis contributes to the state-of-art as follows:

1. *A survey of RFID tag reading protocols.* There has been no works that provide a comprehensive survey of RFID tag reading techniques. Although brief surveys are presented in [59] and [12], none of them provide an

in-depth discussions of works published after the year 2004. The methodology used in this thesis is based on firstly identifying the pros and cons of each protocol and then presenting in-depth comparisons among different protocol classes. Lastly, due to the growing interests in ad-hoc deployments of RFID systems [30] [45] [46] [47], this thesis identifies promising tag reading methods that are suitable for energy constrained systems.

2. *A quantitative model to evaluate the energy consumption of Pure and Slotted Aloha tag reading protocols.* To date, no work has quantified the energy consumption of Aloha based tag reading protocols. This thesis presents an analytical model to quantify the energy consumption of twelve Aloha variants in three phases: namely i) success, ii) collision, and iii) idle listening. The key finding is that Pure Aloha with fast mode consumes the lowest energy and is most suitable for tag identification in RFID-enhanced WSNs. However, no variants promise energy efficient monitoring of identified tags. In other words, the reader is required to re-read all tags every time to sense their presence; a process that consumes a significant amount of energy.
3. *A quantitative model to analyze the accuracy of tag estimation functions.* Dynamic framed slotted Aloha (DFSA) protocols rely on a tag estimation function to set their frame size. To date, very little works have conducted a comprehensive study on the accuracy of current tag estimation functions. Moreover, existing works have not compared these functions using a consistent set of metrics. To this end, this thesis studies tag estimation functions via an error evaluation model that uses descriptive statistics. The model relies on making an iterative estimate of a given set of tags to evaluate a function's mean error, variability, skew and Kurtosis. Lastly, using Monte Carlo simulations, the most energy efficient tag estimation function is identified.
4. *A framework to quantify energy consumption of framed slotted Aloha (FSA) variants.* No work has conducted an in-depth investigation on the energy efficiency of FSA protocols. To this end, Chapter 5 analyzes the energy

consumption of twelve FSA variants. The main findings are that FSA variants that adjust their frame size in accordance with tag population, and also those that incorporate the muting and early-end features have the lowest energy consumption. However, none of them promise energy efficient monitoring in RFID-enhanced WSNs.

5. *ResMon*. Based on the findings in Chapters 2 to 5, Chapter 6 proposes ResMon, an energy efficient protocol that allows for both reading and monitoring of tags. ResMon uses three different frames: 1) reservation, 2) body, and 3) monitor. Reservation and body frames are used for tag identification, and the monitor frame is used to track changes in tag population over time. Results show that, in terms of energy consumption, ResMon achieves 11% and 90% improvements over current framed Aloha variants during identification and monitoring respectively.

### 1.5.1 Publications

The results from the aforementioned chapters have been published in:

1. D. K. Klair, K-W Chin and R. Raad. An Investigation into the Energy Efficiency of Pure and Slotted Aloha Based RFID Anti-Collision Protocols. In IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM'07), June 18-21, Helsinki, Finland, 2007.
2. D. K. Klair, K-W Chin and R. Raad. On the Suitability of Framed Aloha Based RFID Anti-Collision Protocols for RFID-Enhanced WSNs. In The 16th IEEE International Conference on Computer Communications and Networks (IEEE ICCCN'2007), August 13-16, Honolulu, Hawaii, USA, 2007.
3. D. K. Klair, K-W Chin and R. Raad. On the Accuracy of RFID tag Estimation Functions. In The 7th IEEE International Symposium on Communications and Information Technologies (IEEE ISCIT'07), Oct 16-18, Sydney, Australia, 2007.



4. D. K. Klair, K-W Chin, R. Raad and Darryn Lowe. A Spatially Aware RFID-Enhanced Wireless Sensor Network. The Sixth Annual IEEE International Conference on Pervasive Computing and Communications, (IEEE PerCom: Google Ph.D. Forum), March 17-21, 2008, Hong Kong.
5. D. K. Klair and K-W Chin. A Novel Anti-Collision Protocol for Energy Efficient Identification and Monitoring in RFID-Enhanced WSNs. The 17th IEEE International Conference on Computer Communications and Networks (IEEE ICCCN'2008), August 3-7, St Thomas, US Virgin Islands, USA, 2008. [**Best Paper Candidate**]
6. A. R. Rivera, D.K. Klair and K-W Chin. A Simulation Study on the Energy Efficiency of Pure and Slotted Aloha based Tag Reading Protocols. The 6th IEEE Consumer Communications and Networking Conference (IEEE CCNC'09), Las Vegas, CA, USA, Jan 10-15, 2009.
7. D. K. Klair, K-W Chin and R. Raad. On the Energy Consumption of Pure and Slotted Aloha Anti-Collision Protocols in RFID-Enhanced Wireless Sensor Nodes. Elsevier Computer Communications, 32(5), pg 961-973, March, 2009.
8. D. K. Klair, K-W Chin and R. Raad. A Survey and Tutorial of RFID Anti-Collision Protocols. IEEE Communications Surveys and Tutorials Magazine, 2009. To Appear

## 1.6 Thesis structure

The remainder of the thesis is organized as follows:

1. *Chapter 2.* This chapter presents RFID multiple access protocols. Specifically, this chapter surveys RFID anti-collision protocols.
2. *Chapter 3.* This chapter presents an analysis of the energy consumption of twelve framed Aloha protocols. In addition, it highlights a key shortcoming that makes them unsuitable for use in RFID-enhanced WSNs.

3. *Chapter 4.* This chapter analyzes the accuracy of five tag estimation functions. The objective is to find the most suitable tag estimation function for use in RFID-enhanced WSNs.
4. *Chapter 5.* In this chapter, a quantitative comparison of the energy consumption of twelve FSA variants is presented. Moreover, it details their key advantages and shortcomings.
5. *Chapter 6.* Based on the findings of Chapters 3, 4, and 5, this chapter presents a novel anti-collision protocol that is capable of energy efficient identification and monitoring in RFID-enhanced WSNs.
6. *Chapter 7.* This chapter concludes the thesis, and provides a summary of research outcomes and future research directions.

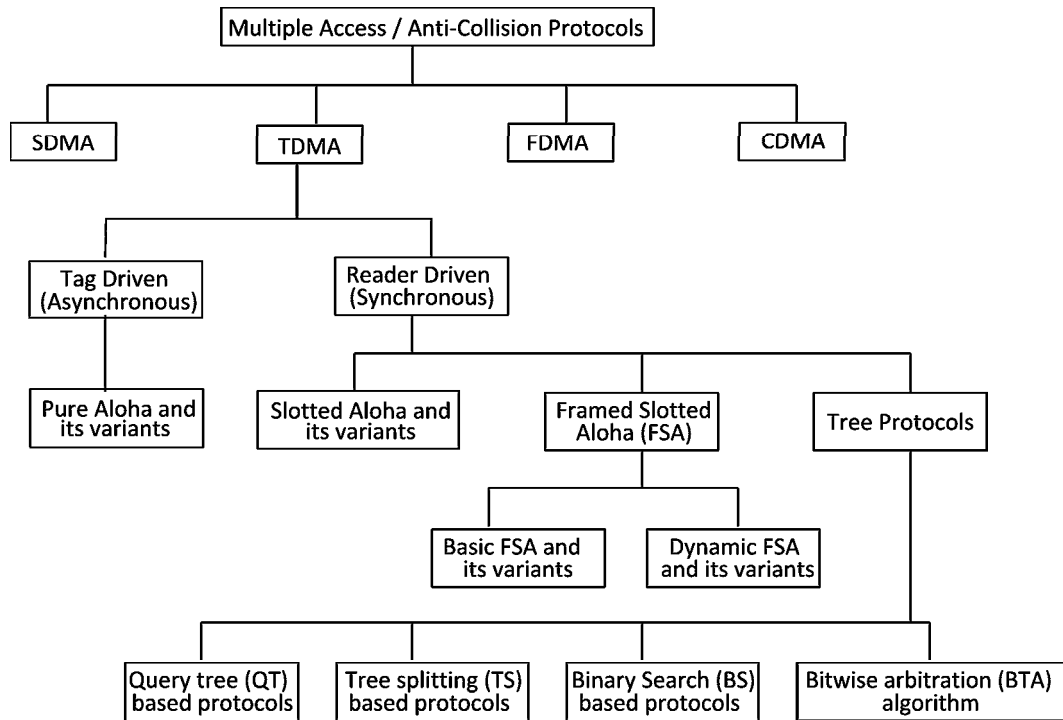
## RFID Anti-Collision Protocols

Collision due to simultaneous tag responses is one of the key issues in RFID systems. It results in wastage of bandwidth, energy, and increases identification delays. To minimize collisions, RFID readers must use an anti-collision protocol. To this end, this chapter reviews state-of-the-art tag reading or anti-collision protocols, and provides a detailed comparison of the different approaches used to minimize collisions, and hence help reduce identification delays.

The remainder of this chapter is organized as follows. Section 2.2 and 2.3 present a comprehensive survey and comparison of Aloha and tree based protocols respectively. Next, Section 2.4 surveys five hybrid tag reading protocols. This is followed by a review of current RFID standards in Section 2.5. Lastly, Section 2.6 concludes the paper.

### 2.1 Anti-Collision Protocols for Tag Collision Problem

Figure 2.1 classifies various anti-collision protocols in existent [27] [59]. Broadly, they can be categorized into, space division multiple access (SDMA), frequency division multiple access (FDMA), code division multiple access (CDMA), and time division multiple access (TDMA).



**Figure 2.1** Classification of tag reading or anti-collision protocols.

Briefly, SDMA protocols [27] [60] spatially separate the channel using directional antennas or multiple readers to identify tags. They, however, are expensive and require intricate antenna designs. On the other hand, FDMA [27] protocols involve tags transmitting in one of several predefined frequency channels; thus, requiring a complex receiver at the reader. Lastly, CDMA [27] [60] based systems involve multiplying tag ID with a pseudo-random sequence (PN) before transmission. Unfortunately, CDMA based systems are expensive and power hungry.

TDMA protocols constitute the largest group of anti-collision protocols [27], and hence is the focus of this thesis. These protocols can be classified as Reader Driven, and Tag Driven. The former and latter are also called Reader-talk-first (RTF) and Tag-talk-first (TTF) respectively. Most applications use RTF protocols, which can be further classified into Aloha and tree based protocols/algorithms. Note, there is also a hybrid class, which combines Aloha and tree protocols. The basic idea behind RTF is that tags remain quiet until specif-

ically addressed or commanded by a reader. On the other hand, TTF protocols function asynchronously. This means a TTF tag announces itself to the reader by transmitting its ID in the presence of a reader. Tags driven procedures are slow compared to RTF protocols [61].

## 2.2 Aloha Based Protocols

First a review of Aloha based tag reading protocols is presented before discussing tree protocols in Section 2.3. The following are Aloha variants in existent:

1. Pure Aloha (PA).
2. Slotted Aloha (SA).
3. Framed Slotted Aloha (FSA).
  - (a) Basic framed slotted Aloha (BFSA).
  - (b) Dynamic framed slotted Aloha (DFSA).
  - (c) Enhanced Dynamic framed slotted Aloha (EDFSA).

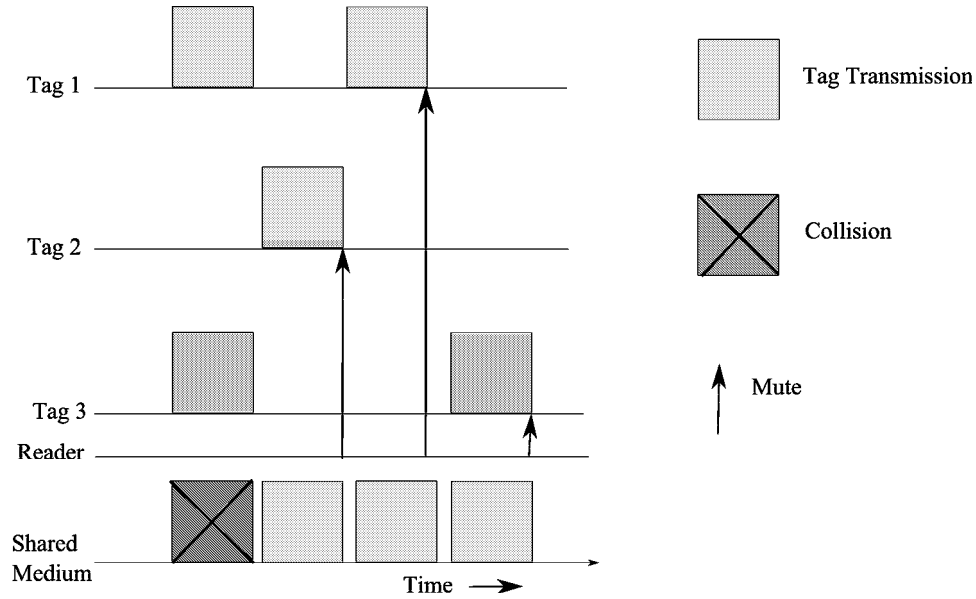
### 2.2.1 Pure Aloha (PA)

In PA based RFID systems, a tag responds with its ID randomly after being energized by a reader. It then waits for the reader to reply with, i) a positive acknowledgement (*ACK*), indicating its ID has been received correctly, or ii) a negative acknowledgement (*NACK*), meaning a collision has occurred. If two or more tags transmit, a complete or partial collision occurs [12], which tags then resolve by backing off randomly before retransmitting their ID.

Pure Aloha based systems have several variants [62] [12] [63]:

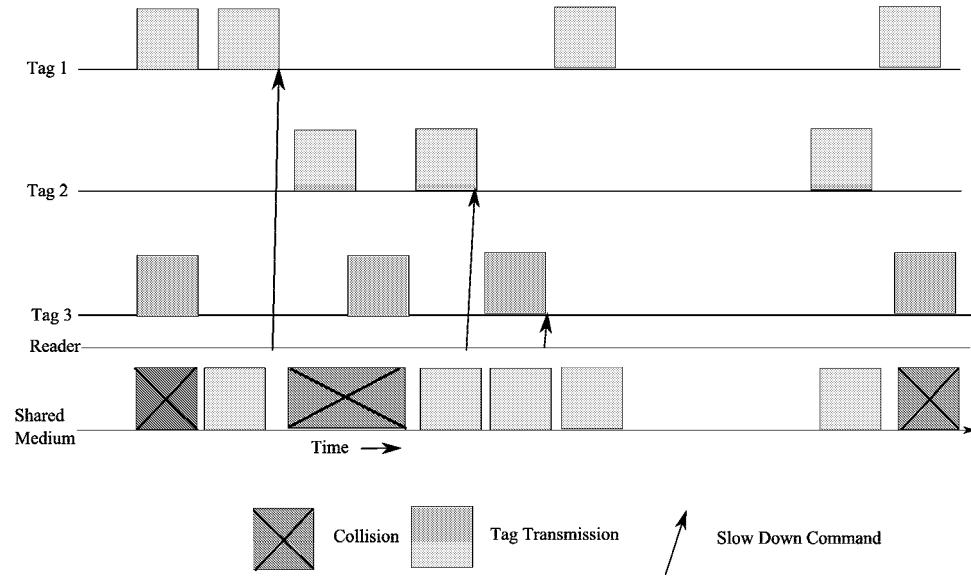
1. *PA with Muting*. When muting is used, the number of tags in a reader's interrogation zone is reduced after each successful tag response, meaning the offered load to the reader is reduced after a tag is identified. Figure

2.2 shows the behavior of PA with muting. Initially, tags 1 and 3's transmission collides, causing them to wait a random amount of time before retransmitting again. After identification, the reader silences read tags using the "mute" command.



**Figure 2.2** Pure Aloha with muting.

2. *PA with Slow Down.* Instead of being muted, a tag can be instructed using a "slow down" command to reduce its rate of transmission, hence decreasing the probability of collision. Figure 2.3 shows how the reader slows tag 1 down after identification, resulting in tag 1 adapting its random back-off counter to reduce its transmission rate.
3. *PA with Fast Mode.* A "silence" command is sent by the reader once it has detected the start of a tag transmission. This command has the effect of stopping other tags from transmitting. Tags are allowed to transmit again after the reader has sent an ACK command or until their waiting timer expires. Figure 2.4 shows PA with fast mode. Once the reader detects a transmission from tag 2, tag 1 and tag 3 are silenced and reactivated only after tag 2 has finished transmitting.
4. *Other Variants.* Lastly, we can create two more variants, namely PA



**Figure 2.3** Pure Aloha with slow down.

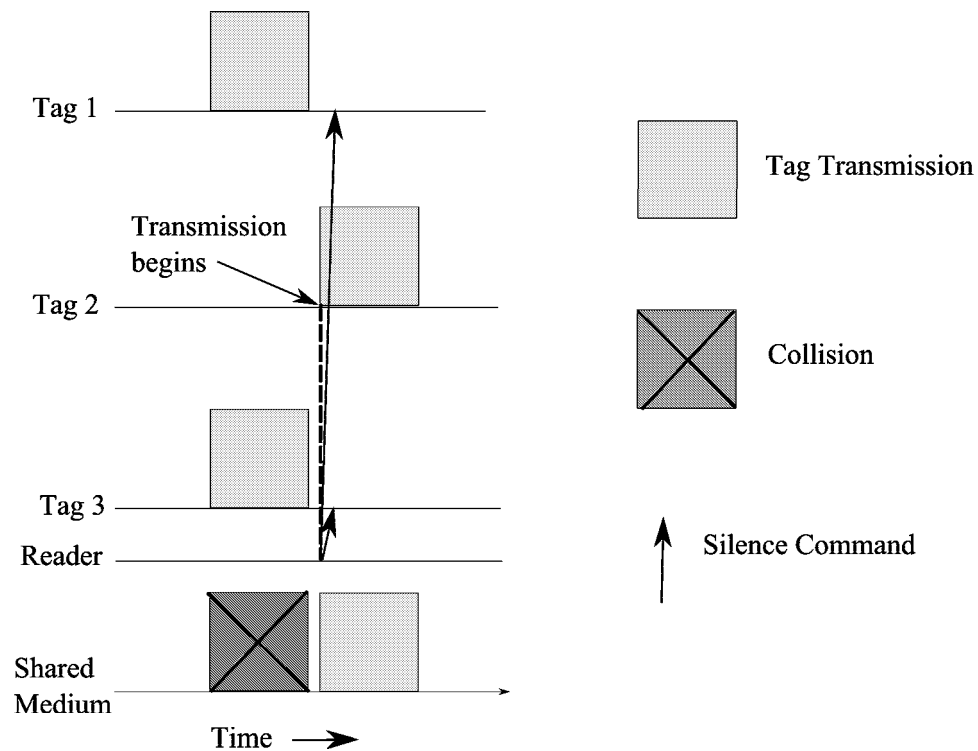
with fast mode and muting, and PA with fast mode and slow down by combining the respective features. These variants are shown in Figures 2.5 and 2.6 respectively. In Figure 2.5, tags 1 and 3 are silenced when tag 2 starts transmitting. After tag 2 is identified, it is muted. Similarly, in Figure 2.6, after tag 2 is identified using fast mode, it is slowed down to allow other tags to transmit.

### 2.2.2 Slotted Aloha (SA)

In Slotted Aloha (SA) based RFID systems, tags transmit their ID in synchronous time slots. If there is a collision, tags retransmit after a random delay. The collision occurs at slots boundary only, hence there are no partial collisions [64].

Slotted Aloha also has numerous variants [62] [12] [63]:

1. *SA with Muting/Slow Down.* The principle operation is similar to PA with muting/slow down, but operates in a slotted manner.
2. *SA with Early End.* If no transmission is detected at the beginning of



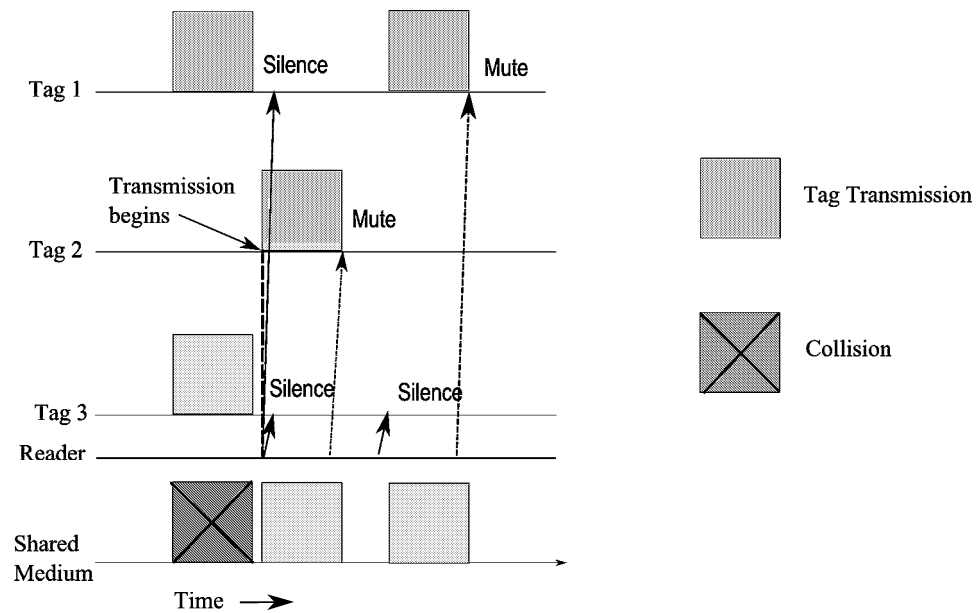
**Figure 2.4** Pure Aloha with fast mode.

a slot, the reader closes the slot early. Two commands are used: start-of-frame (SOF) and end-of-frame (EOF). The former is used to start a reading cycle, and the later is used by the reader to close an idle slot early. Figure 2.7 depicts how early end is used to terminate idle slots.

3. *SA with Early End and Muting.* When tags have been identified, the reader sends a muting command, thereby reducing the number of responding tags. When no replies are detected after a small period of time, the reader closes the slot early using the EOF command.
4. *SA with Slow Down and Early End:* This combines slow down with the early end feature.

In summary, there are four key features being used to increase the performance of Pure and Slotted Aloha based tag reading protocols: i) muting, ii) slow down, iii) early-end, and iv) fast mode. To recap, fast mode is only used in conjunction





**Figure 2.5** Pure Aloha with fast mode and muting.

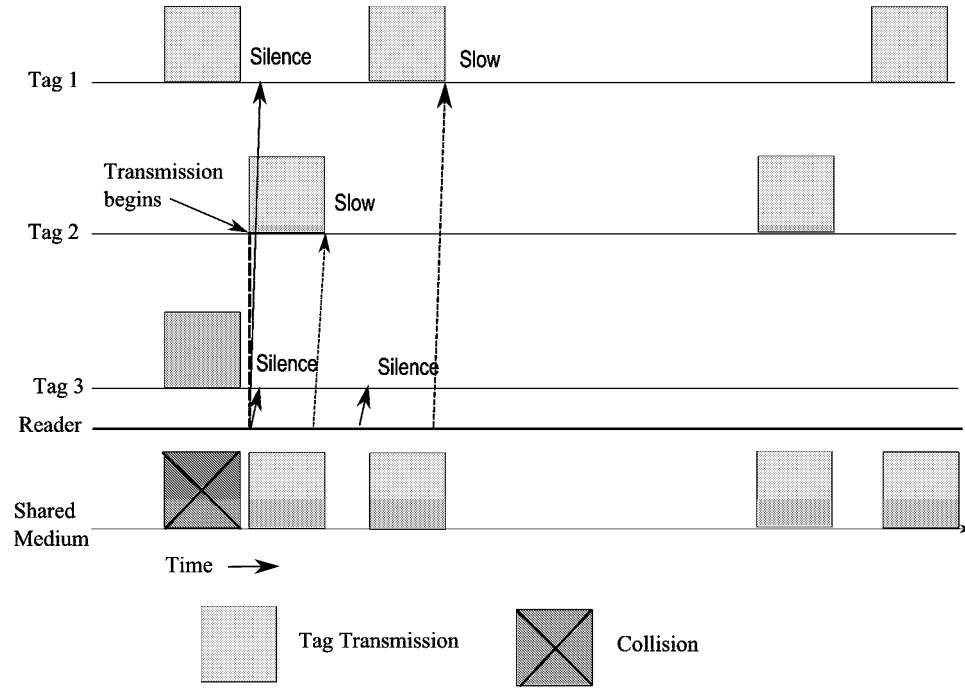
with Pure Aloha variants to reduce their vulnerability period. Early end is used by slotted Aloha variants to reduce idle listening where idle slots are terminated early. Lastly, muting and slow down have the effect of reducing the offered load to the reader.

### 2.2.3 Framed Slotted Aloha (FSA)

In PA and SA based systems, a tag with a high response rate will frequently collide with potentially valid responses from other tags. Therefore, FSA protocols mandates that each tag responds only once per frame. The following sections describe various FSA variants.

#### 2.2.3.1 Basic Frame Slotted Aloha (BFSA)

BFSA has four variants. They are, 1) BFSA-Non Muting, 2) BFSA-Muting, 3) BFSA-Non-muting-early-end, and 4) BFSA-Muting-early end. Note, the term “basic” refers to the frame size being fixed throughout the reading process. In BFSA-Non muting, a tag is required to transmit its ID in each read round. In non-muting variants, the reading delay is dependent on the confidence level



**Figure 2.6** Pure Aloha with fast mode and slow down.

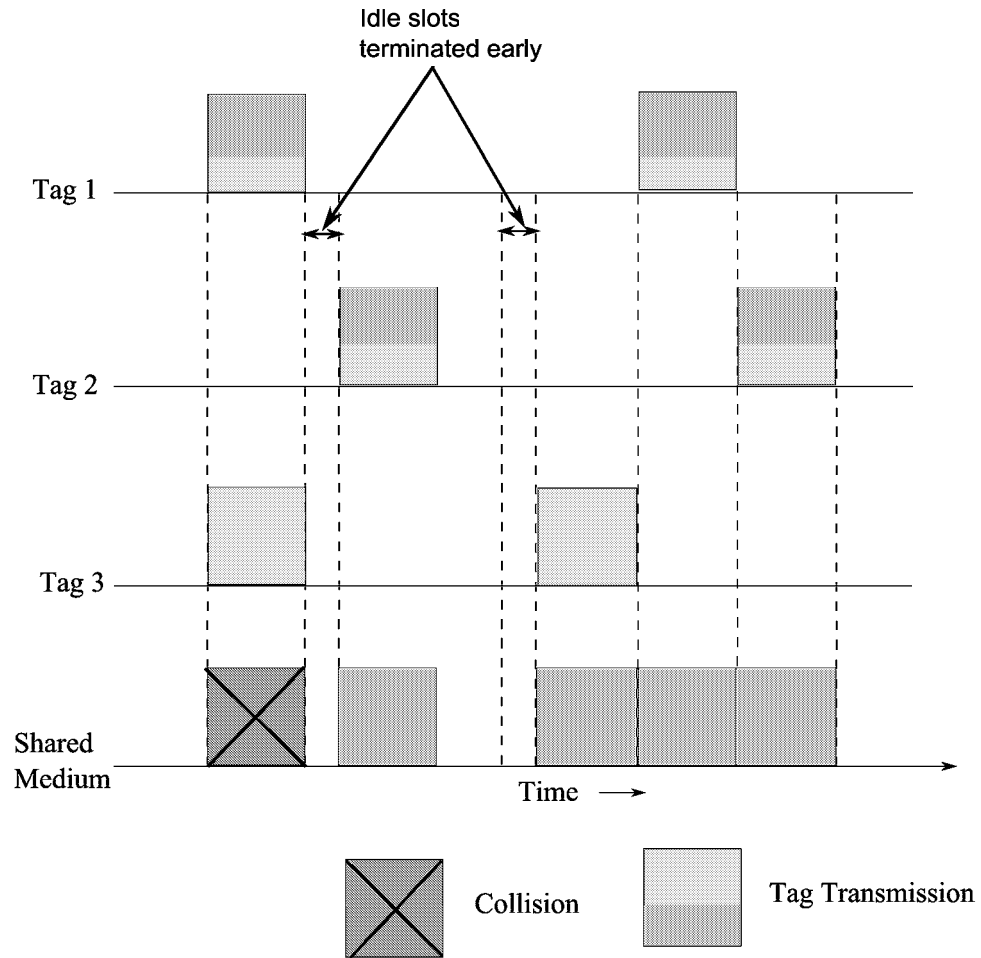
$\alpha$ , where  $\alpha=0.99$  indicates 99% of the tags have been read successfully. The number of read cycles  $R$  needed to read a tag set with  $\alpha$  confidence level is given by [65],

$$R \geq \left\lceil \frac{\log(1 - \alpha)}{\log\left(1 - \frac{Np_1}{n}\right)} \right\rceil \quad (2.1)$$

where  $N$  is the frame size,  $n$  is the number of tags, and the probability of having a successful transmission is  $p_1 = \left(1 - \frac{1}{N}\right)^{n-1}$ . To obtain an integral value, and avoid conservative delay values, Equ. 2.1 uses the ceil function.

For BFSA-Muting, the number of tags reduces after each read round, since tags are silenced after identification. When a read round is collision free, the reader concludes that all tags have been identified successfully.

BFSA-Non-muting-early-end and BFSA-Muting-early-end variants incorporate the early-end feature. Specifically, the reader transmits a *close slot* command when no response is received in a particular slot.



**Figure 2.7** Slotted Aloha with early end.

BFSA non-muting suffers from an exponential increase in identification delay when the number of tags is very high compared to the frame size [66]. To address this problem, Hwang et al. [67] present a BFSA variant that limits the number of responding tags. At initialization, the reader requests tags to compare a part of their ID with the bit string it is transmitting. and those with smaller bit values respond. A key observation is that when the number of tags is much smaller than the frame size, restricting tag responses increases identification delays. Therefore, the authors define a threshold based on the ratio of collision slots and the frame size to decide if restricting tag responses is necessary.

A new approach, called detection and jump, is presented by Wang et al. [68],

where the reader first transmits a detection frame with 4-bit sized slots. Tags then respond with a 4-bit random sequence in the detection frame. Tags that respond successfully in the detection frame transmit their complete ID in the jump frame. The number of slots in the jump frame is equal to the number of slots with a single reply in the detection frame.

### 2.2.3.2 Dynamic Frame Slotted Aloha (DFSA)

FSA protocols with variable frame sizes are called dynamic framed slotted Aloha (DFSA) [27]. Similar to BFSA, DFSA operates in multiple rounds, and it can also incorporate the early-end feature. The key difference, however, is that in each read round, the reader uses a tag estimation function to vary its frame size [66].

A tag estimation function calculates the number of tags based on feedback from a reader's frame, in particular the number of slots filled with zero ( $c_0$ ), one ( $c_1$ ) and multiple tag responses ( $c_k$ ). This information is then used by the function to obtain a tag estimate, and hence the optimal frame size  $N$  for a given round. Here, the optimal frame size is one which promises the maximum system efficiency and minimum identification delay. Theoretically, the optimal frame size is equal to the number of tags [66].

Following sections review various tag estimation functions, each of which defines a new DFSA variant.

**Vogt [69] [70].** The author presents two tag estimation functions, denoted as Vogt-I and Vogt-II. Vogt-I is based on the principle that during collisions, at least two tags are involved, hence the tag estimate is  $c_1 + 2c_k$ . On the other hand, Vogt-II is based on Chebyshevs inequality and aims to minimize the distance  $\varepsilon_{vd}$  between an actual read result vector  $\langle c_0, c_1, c_k \rangle$  and the theoretically

computed result  $\langle a_0^{N,n}, a_1^{N,n}, a_k^{N,n} \rangle$  obtained by evaluating Equ. 2.2.

$$\varepsilon_{vd}(N, c_0, c_1, c_k) = \min_t \left\| \begin{pmatrix} a_0^{N,t} \\ a_1^{N,t} \\ a_k^{N,t} \end{pmatrix} - \begin{pmatrix} c_0 \\ c_1 \\ c_k \end{pmatrix} \right\| \quad (2.2)$$

In Equ. 2.2, the elements of the vector  $\langle a_0^{N,t}, a_1^{N,t}, a_k^{N,t} \rangle$  correspond to the expected number of empty slots, slots filled with one tag, and slots with collisions, respectively. With a frame size of  $N$ , and the number of tags  $t$ , the expected number of slots filled with  $r$  responding tags is given by,

$$a_r^{N,t} = N \times \binom{t}{r} \left( \frac{1}{N} \right)^r \left( 1 - \frac{1}{N} \right)^{t-r} \quad (2.3)$$

Vogt also proposed a set of frame sizes promising lower identification delays for a given tag range. They are shown in Table 2.1. For example, a frame size of sixteen is considered optimal when there are one to nine tags.

**Table 2.1** Optimal frame sizes for a given tag range.

**Zhen et al. [65].** This function is based on computing the *a posteriori* expected value of collided slots. According to Zhen et al., this value is 2.39, corresponding to an average of 2.39 collisions per-slot. Thus, the number of estimated tags is  $c_1 + 2.39c_k$ . In addition, Zhen et al. proposed to overestimate the tag set, since doing so lowers identification delays. Based on their experimentations, they proposed  $1.4 \times (c_1 + 2.39c_k)$  as a tag estimate. On the other hand, for muting environments, they proposed  $0.65 \times (c_1 + 2.39c_k)$ .

**Cha et al. [66].** The authors present two tag estimation functions for muting based RFID environments. Cha-I estimates tags by computing the ratio of the number of slots with collisions and the frame size, and is given by,

$$C_{ratio} = 1 - \left(1 - \frac{1}{N}\right)^n \left(1 + \frac{n}{N-1}\right) \quad (2.4)$$

where  $n$  is the tags to be estimated.  $C_{ratio}$  is computed after a read round as  $C_{ratio} = \frac{c_k}{N}$ . In Cha-II, a tag estimate is simply  $2.39c_k$ .

**Khandelwal et al. [71].** The authors propose to estimate the number of tags using,

$$n = \frac{\log\left(\frac{c_0}{N}\right)}{\log\left(1 - \frac{1}{N}\right)} \quad (2.5)$$

Here,  $N$  is the current frame size. Note, Equ 2.5 cannot be applied when  $c_0 = 0$ . When this happens, the tag estimate is  $n = c_1 + 2c_k$ . Lastly, Khandelwal et al. propose that the frame size should be  $1.943 \times n$  times the estimated number of tags.

**Floerkemeier [72] [73].** Two functions are proposed by authors. Namely Floerkemeier-I and Floerkemeier-II. These functions estimate tags based on Bayesian broadcast strategies. In Floerkemeier-I, a reader not only considers read results in the current read round, but also records those in the last frame to determine the frame size of the next read round. On the other hand, Floerkemeier-II updates the frame size as the frame progresses, i.e., slot-by-slot according to read results in the last and current slot. It restarts the current frame if it is non-optimal.

**Kodialam et al. [74].** The authors proposed an estimation function that computes the expected number of idle and single response slots by inserting

$r = 0$  and  $r = 1$  in Equ. 2.3. The resulting equations are then used to derive two estimators, called zero estimator (ZE) and collision estimator (CE),

$$ZE = e^{-(n_0/N)} = \frac{c_0}{N} \quad (2.6)$$

$$CE = 1 - \left(1 + \frac{n_k}{N}\right) e^{-(n_k/N)} = \frac{c_k}{N} \quad (2.7)$$

In Equ. 2.6 and 2.7,  $n_0$  is the tag estimate obtained from ZE, and  $n_k$  is the tag estimate computed from CE, respectively. The values of  $c_0$  and  $c_1$  are obtained by observing the number of idle slots and slots with single response. They are then used to solve ZE for  $n_0$  and CE for  $n_k$ . If  $n_0 < n_k$ , then the tag estimate is  $n_0$ , otherwise it is  $n_k$ . The authors assume that the estimation phase is separate and precedes the identification phase. In addition, slots in the estimation phase are only 10-bits long.

**Chen et al. [75].** Two estimation functions are introduced by Chen et al., called Chen-I and Chen-II. In the former, the authors compute the probability of exactly  $k$  tags in  $m$  slots as [76],

$$p(k, m) = \frac{(-1)^m N! n!}{m! N^n} \times \sum_{j=m}^{\min(N, \lceil n/k \rceil)} (-1)^j \frac{(N-j)^{n-jk}}{(j-m)! (N-j)! (n-jk)! (k!)^j} \quad (2.8)$$

Using Equ. 2.8, the authors calculate the probability of exactly  $m$  slots with zero tag responses, i.e.,  $k = 0$ . The actual value of  $m$  is  $c_0$ , which is obtained from the reader's feedback. Equ. 2.8 is then solved for the value of  $n$ , which is the tag estimate.

On the other hand, Chen-II computes the expected number of slots filled with zero and a single tag using Equ. 2.3. The results, denoted as  $E$  and  $S$ , are then

fed into the following equation,

$$n = (N - E - 1) \frac{S}{E} \quad (2.9)$$

where  $N$  is the frame size. Equ. 2.9 is then solved for the tag estimate  $n$ .

***Q protocol [16].*** The proposed tag estimation function requires the reader to increment and decrement the frame size with a constant. A reader initially broadcasts a query command that contains a slot counter  $Q$  and a frame of size  $2^Q$ .  $Q$  is an integer between zero and eight. Tags choose a slot randomly from 0 to  $2^Q - 1$ . The reader then increments or decrements  $Q$  by a constant  $c$ , where  $0.1 \leq c \leq 0.5$ , for each collision or idle slot respectively. Slots with a single response do not change  $Q$ . The resulting value of  $Q$  is then used to determine the frame size of the next round [68] [72] [73].

***Comparisons and Discussions.*** In general, two methodologies are used for tag estimations. The first is based on computing a tag estimate using a fixed multiplier. This is called static estimation. The functions Cha-I, Zhen, Q-protocol and Vogt-I belong to this methodology. On the other hand, functions which derive tag estimates using probabilistic or statistical methods are called dynamic estimation. Chen-I, Chen-II, Cha-II, Vogt-II, Khandelwal, Kodialam, FloerkemeierI, FloerkemeierII are examples of this methodology.

In static estimation methods, tag estimate error increases when the number of tags exceeds the frame size [69]. On the other hand, dynamic estimation schemes promise low errors even when the number of tags is higher than the frame size.

Estimation functions can also be classified according to their consideration for the muting feature. Among those studied, Cha-I, Cha-II, Chen-I, Chen-II, Floerkemeier-I, and Floerkemeier-II consider muting while the rest do not.

The computational requirements of tag estimation functions vary for each method-



ology. Static estimation techniques are simpler to implement and have low computational requirements. The computation only involves simple additions and multiplications. On the other hand, dynamic estimation techniques have higher computational requirements since they need to evaluate theoretical values and compare them to read values.

Vogt-I, Cha-II, Q-Protocol, and Zhen estimate tags using simple calculations involving additions and multiplications. Relatively higher computations are required for Chen-II, Khandelwal, Kodialam, Floerkemeier- I, and Floerkemeier-II because these functions involve the calculation of factorials and fractions. Lastly, Vogt- II, Cha-I and Chen-I have the highest computational requirements since they involve recursions.

### 2.2.3.3 Enhanced Dynamic Framed Slotted Aloha (EDFSA)

A limitation of DFSA variants is that the frame size is bounded to a maximum value of 256 [69] or 512 [17]. If the number of tags exceeds this value, achieving an optimal frame size becomes an issue. To this end, Lee et al. [77] propose enhanced-DFSA or EDFSA, where tags are divided into  $M$  groups if the tag population is larger than the maximum frame size available. Table 2.2 shows the value of  $M$  for a given tag range.

**Table 2.2**

EDFSA frame sizes.  $n$  denotes the number of tags,  $N$  is the frame size, and  $M$  is the number of tag groups.

In Table 2.2, Lee et al. also propose frame sizes for varying tag ranges to achieve maximum system efficiency. The value of  $M$  is one when the number of tags is lower than 354. However, when the number of tags increases, the modulo

operation comes into effect, which divides responding tags into  $M$  groups. The reader then read tags on a group-by-group basis.

Lastly, similar to BFSA and DFSA, EDFSA can also incorporate the early end and muting feature [78].

### 2.2.4 Discussions

Table 2.3 summarizes key observations pertaining to Aloha based protocols. The performance of Aloha based protocols increases as we move from PA to DFSA variants. However, this performance improvement is at the expense of increased system cost and complexity.

The most suitable protocol depends largely upon the application in question. If low cost and complexity is desired, then PA variants are suitable. On the other hand, DFSA variants are ideal if high speed, accuracy, and efficiency are of concern. Overall, DFSA variants are the most popular due to their adaptability to varying loads and high system efficiency.

## 2.3 Tree Based Protocols

Tree based protocols were originally developed for multiple access arbitration in wireless systems [80]. These protocols are able to single out and read every tag, provided each tag has a unique ID. All tree based protocols require tags to have muting capability, as tags are silenced after identification. Tree based algorithms can be classified into the following categories:

1. Tree splitting (TS).
2. Query tree (QT).
3. Binary search (BS).
4. Bitwise arbitration (BTA).

Table 2.3 A comparison of Aloha, Slotted Aloha and Framed Aloha protocols.

| Criterion   | Pure Aloha (PA)  | Slotted Aloha (SA)   | Basic Framed Slotted Aloha (BFSA)  | Dynamic Framed Slotted Aloha (DFSA)  |
|---|--|--|--|--|
| Protocol feature  | A tag transmits its ID after a random time to the reader. In the event of a collision, a tag will retransmit after a random delay. | Tags transmit their ID in synchronized slots. If there is a collision, a tag responds after a random number of slots.                              | A tag is permitted to transmit at most once in a fixed frame (1).  | A tag transmits once per frame, and the frame size varies according to tag population (2). |
| Tag requirements  | Timer  | Random number generator, timer, and synchronization circuits.  | Random number generator, Some tags in DFSA based variants also need to generate short pseudo IDs for identification or tag estimation. |  |
| Throughput (3)  | 18.4% [27]   | 36.8%  |  | 42.6% [79]   |
| Disadvantages   | If the offered load is increased, the number of collisions increases exponentially.  | If the offered load is increased, the number of collisions increases exponentially. Also, it requires synchronization between the reader and tags. | Tags need to know the frame size in use, and they also require synchronization circuits.   | Monitoring slots with single, zero or no responses, and requires a sophisticated receiver. |
| RTF / TTF   | RTF  |  |  |  |
| Tag cost  | TTF  |  |  |  |
|   | Least  |  |  | → Most   |
| Protocol complexity   | Least  |  |  | → Most   |
| System cost   | Least  |  |  | → Most   |
| (1) In Kodialam et al. [74], a tag can skip transmission in a particular frame.             |  |  |  |  |
| (2) EDFSA performs better than DFSA for high tag densities in non-muting environments [78]. |  |  |  |  |
| (3) Normalized to offered load  |  |  |  |  |

### 2.3.1 Tree Splitting

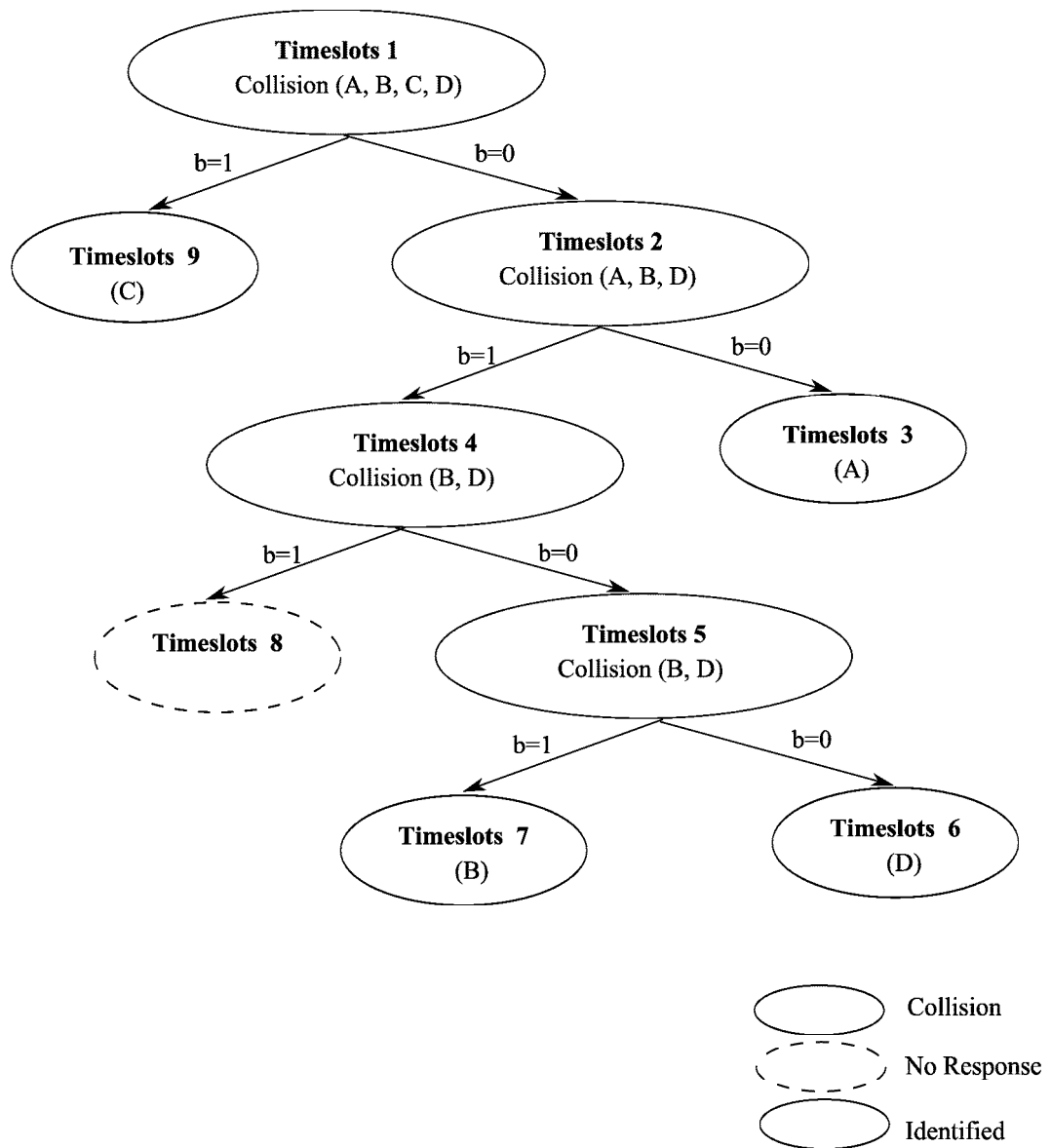
TS protocols operate by splitting responding tags into multiple subsets using a random number generator. The following sections present two algorithms in this category.

#### 2.3.1.1 Basic Tree Splitting (BTS)

Hush et al. [81] present BTS, an algorithm that performs collision resolution by splitting collided tags into  $b$  disjoint subsets. These subsets become increasingly smaller until they contain one tag. Identification is achieved in a sequence of timeslots. Each tag has a random binary number generator  $b$ . In addition, each tag maintains a counter to record its position in the resulting tree. Tags with a counter value of zero are considered to be in the transmit state, otherwise tags are in the wait or sleep state. After each timeslot, the reader informs tags whether the last timeslot resulted in a collision, single, or no response. If there was a collision, each tag in the transmit state generates a random binary number and adds the number to its current counter value. On the other hand, tags in the wait state increment their counter by one. In the case of idle or single response, tags in the wait state decrement their counter by one. After identification, tags enter the sleep state.

As an example, let's say there are four tags: A=010, B=011, C=100, and D=110. Figure 2.8 depicts the identification process, and Table 2.4 shows each tag's counter value at a given timeslot. In timeslot 1, each tag's counter is initialized to zero, meaning all tags are allowed to transmit, thus causing a collision. The reader then informs tags of the collision and tags in the transmit state split into two subsets by generating a random binary number. Tags A, B, and D have selected binary zero and therefore are allowed to transmit again, which unfortunately causes a collision in timeslot 2. At timeslot 3, only tag A has a counter value of zero, whilst the rest of the tags are in the wait state. Since tag A is the only one in the transmit state, it is identified successfully. The reader informs tags in the wait state of the single response, causing them to decrement their counter by one. In timeslot 4, tags B and D have a counter

value of zero, meaning their transmission causes another collision. Tags B and D then update their counter, but experience a collision in timeslot 5. They are not identified until timeslots 6 and 7 respectively. Finally, after an idle timeslot, tag C is identified in timeslot 9. Overall, the reader uses nine timeslots to identify all four tags.



**Figure 2.8** Basic Tree splitting (BTS) algorithm.

**Table 2.4** Tag's counter in the BTS algorithm.

| Time slots | Feedback   | Tag Counter  |              |                 |              |
|------------|------------|--------------|--------------|-----------------|--------------|
|            |            | Tag A        | Tag B        | Tag C           | Tag D        |
| 1          | Collision  | 0 (Transmit) | 0 (Transmit) | 0 (Transmit)    | 0 (Transmit) |
| 2          | Collision  | 0 (Transmit) | 0 (Transmit) | 1 (Wait)        | 0 (Transmit) |
| 3          | Identified | 0 (Transmit) | 1 (Wait)     | 2 (Wait)*       | 1 (Wait)     |
| 4          | Collision  | —            | 0 (Transmit) | 1 (Wait)**      | 0 (Transmit) |
| 5          | Collision  | —            | 0 (Transmit) | 2 (Wait)        | 0 (Transmit) |
| 6          | Identified | —            | 1 (Wait)     | 3 (Wait)        | 0 (Transmit) |
| 7          | Identified | —            | 0 (Transmit) | 2 (Wait)        | —            |
| 8          | Idle       | —            | —            | 1 (Wait)        | —            |
| 9          | Identified | —            | —            | 0 (Transmit)*** | —            |

\*Tags in the wait state increment their counter by one because of collision.  
 \*\* Tags in the wait state decrement their counter by one because of identified tag.  
 \*\*\* Tags in the wait state decrement their counter by one because of idle response.

### 2.3.1.2 Adaptive Binary Splitting (ABS)

Myung et al. [82] [83] propose Adaptive Binary Splitting (ABS), an advancement to Hush et al. [81]'s BTS algorithm. ABS achieves fast identification by reducing not only collisions but also unnecessary idle slots. Similar to the BTS algorithm, tags can either be in the transmit or wait state. However, unlike BTS, tags have two counters, progressed slot counter (PSC) and Allocated-slot counter (ASC). The PSC of each tag is incremented by one whenever the reader successfully identifies a tag, and ASC specifies a tag's transmitting timeslot. A tag is allowed to transmit when its ASC and PSC are equal. Moreover, identified tags have a smaller ASC compared to their PSC. As in BTS, the reader informs tags of the read result of the last timeslot. If there was a collision, tags in the transmit state or collided tags select a random binary number and add it to their current ASC. For no response or idle slots, tags in the wait state decrement their ASC by one. Lastly, if there was only a single response, tags in the wait state increment their PSC by one.

The operation of ABS can be illustrated using the tags set presented earlier. Note, ABS and TS shares the same tree. Table 2.5 shows the counter value of each tag at a given tree node. Initially, the ASC and PSC value of each tag is initialized to zero. This results in a collision at timeslot 1. The tags then generate a binary random number and add the result to their ASC. In timeslot 2, tags A, B and D have equal ASC and PSC value, which causes them to

enter the transmit state. As a result, their transmission collides. In timeslot 3, only tag A has equal ASC and PSC value, hence it is identified successfully. The reader then informs tags in the wait state of the successful identification in timeslot 3. Upon receiving the feedback, tags increment their PSC by one. In timeslot 4, tags B and D have equal ASC and PSC value, meaning they are allowed to transmit. Unfortunately, their transmission results in a collision. In timeslot 5, both tags B and D have a random number outcome of zero, which leaves their ASC and PSC unchanged, thus causing a collision in timeslot 5. However, in timeslot 6, only tag D has equal ASC and PSC value, which allows it to be identified successfully. Finally, tags B and C are read in timeslots 7 and 9 respectively.

**Table 2.5** Adaptive Binary Splitting (ABS) - TSC, PSC and ACS values.

| Time slot | Feedback   | PSC | ASC          |               |                 |              | TSC |
|-----------|------------|-----|--------------|---------------|-----------------|--------------|-----|
|           |            |     | Tag A        | Tag B         | Tag C           | Tag D        |     |
| 1         | Collision  | 0   | 0 (Transmit) | 0 (Transmit)  | 0 (Transmit)    | 0 (Transmit) | 0   |
| 2         | Collision  | 0   | 0 (Transmit) | 0 (Transmit)  | 1 (Wait)        | 0 (Transmit) | 1   |
| 3         | Identified | 0   | 0 (Transmit) | 1 (Wait)      | 2 (Wait)*       | 1 (Wait)     | 2   |
| 4         | Collision  | 1   | —            | 1 (Transmit)  | 2 (Wait)**      | 1 (Transmit) | 3   |
| 5         | Collision  | 1   | —            | 1 (Transmit)  | 3 (Wait)*       | 1 (Transmit) | 3   |
| 6         | Identified | 1   | —            | 2 (Wait)      | 4 (Wait)*       | 1 (Transmit) | 4   |
| 7         | Identified | 2   | —            | 2(Transmit)** | 4 (Wait)**      | —            | 5   |
| 8         | Idle       | 3   | —            | —             | 4 (Wait)        | —            | 4   |
| 9         | Identified | 3   | —            | —             | 3 (Transmit)*** | —            | 3   |

\* Tags in the wait state increment their ASC by one because of collision.  
 \*\* ASC remains unchanged and PSC is incremented by one.  
 \*\*\* Tags in the wait state decrement their ASC by one because of no response.

Once all tags are identified, the reader ends the reading process using a terminating slot counter (TSC). The value of TSC is updated after each timeslot as follows: 1) if there was a collision, the reader increments TSC by one, 2) for an idle slot, the reader reduces TSC by one, and 3) for a slot with single response, TSC is left unchanged. As soon as PSC becomes greater than TSC, the reader terminates the reading process. In Table 2.5, after timeslot 9, the PSC is incremented to four, which is greater than TSC, hence terminating the read process in timeslot 9.

After all tags are identified, the reader and tags preserve their TSC and ASC value. From Table 2.5, the ASC value of tag A is zero, tag B is two, tag C is

three, tag D is one and the reader's TSC is three. Using these TSC and ASC values, re-identification of tags can be carried out in four consecutive timeslots. This is achieved as follows. The reader first initializes PSC to zero. In the first timeslot, since the ASC for tag A is also zero, tag A enters the transmit state and is identified in the first timeslot. The PSC is then incremented by one, which equals tag D's ASC. As a result, tag D is identified in the second timeslot. Similarly, tags B and C are identified in timeslots three and four respectively.

If a new tag E is added to the tag set, it is allowed to choose an ASC value ranging from zero to TSC. If tag E selects an ASC value of two, then there will be a collision in timeslot 3. This is because both tag E and B have the same ASC value. These two tags then split into two subsets by generating a unique random binary number and are identified in either timeslots four or five depending upon their binary outcome. On the other hand, if a tag departs from the reader's interrogation zone, tags in the wait state decrement their ASC and TSC by one to eliminate idle slots.

Chen et al. [84] present a variant of the ABS algorithm called enhanced binary splitting (EBS). EBS uses Manchester coding to identify the location of collided bits. If a collided bit is detected, the reader stops tags from transmitting their remaining ID bits. Each tag maintains a pointer that stores the location of the first collided bit. If the pointer has a value  $k$ , it means the  $k^{th}$  bit suffered a collision. In other words, all bits prior to the  $k^{th}$  bits have been received correctly. Thus, in future read requests, tags only need to transmit those bits from their ID that occur after the  $k$ th bit. These bits are then identified using ABS.

### 2.3.2 Query Tree Algorithms

In TS variants, tags require a random number generator and a counter to track their tree position, thus making them costly and computationally complex. Query tree algorithms overcome these problems by storing tree construction information at the reader, and tags only need to have a prefix matching circuitry. Numerous variants of query tree algorithms exist. They are discussed in the



following sections.

### 2.3.2.1 Query Tree

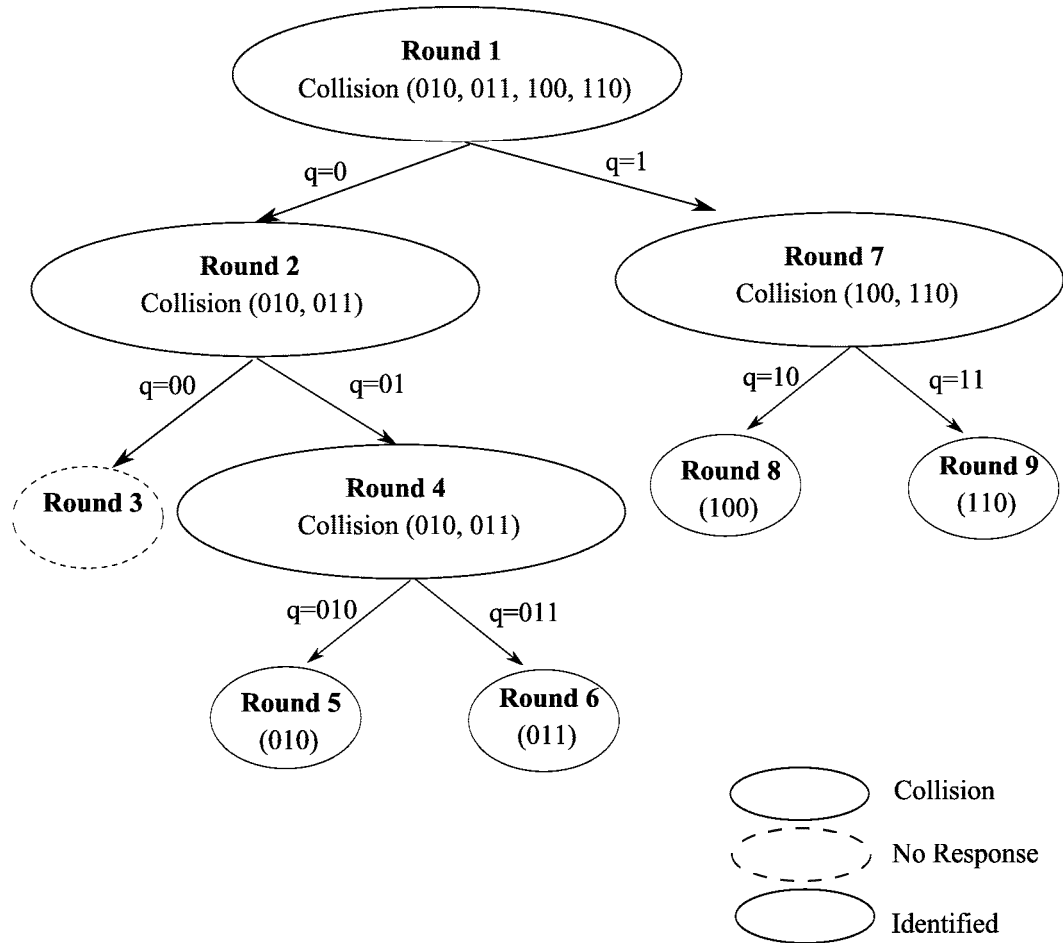
Law et al. [85] propose query tree (QT). Each tag has a prefix matching circuitry. The reader transmits a query  $q$ , and tags with a matching prefix reply to the reader. Collision occurs when multiple tags have the same prefix. In this case, the reader forms a new query by appending  $q$  with a binary 0 or 1. The reader then repeats the reading processing using the augmented query.

Figure 2.9 shows the QT protocol being used to read the tags set presented earlier. Table 2.6 shows the content of the reader's stack, which stores pending queries. The reader starts with a null string. Since this causes a collision, the reader pushes queries 0 and 1 onto the stack, i.e.,  $q = 0$  and  $q = 1$ . In round 2, the reader pops and transmits query 0. In our example, tags 010 and 011 have prefix 0, which causes them to transmit and collide. The reader then pushes queries 01 and 00 onto the stack. In round 3, the reader pops and transmits query 00. This query solicits no reply since there are no tags with the prefix 00. In round 4, the reader experiences a collision, since tags 010 and 011 responded to the query 01. As a result, queries 010 and 011 are pushed onto the stack. The reader then transmits query 010 in round 4, which matches tag 010. In round 5, query 011 identifies tag 011. Similarly, tags 100 and 111 are identified after the reader sent queries 10 and 11 in round 8 and 9 respectively. Overall, the reader uses nine rounds to read four tags.

**Table 2.6** Reader's stack corresponding to Figure 11.

| Round | Query $q$ | Response   | Reader's Stack |
|-------|-----------|------------|----------------|
| 1     | Empty     | Collision  | (0, 1)         |
| 2     | 0         | Collision  | (00, 01, 1)    |
| 3     | 00        | Idle       | (01, 1)        |
| 4     | 01        | Collision  | (010, 011, 1)  |
| 5     | 010       | Identified | (011, 1)       |
| 6     | 011       | Identified | (1)            |
| 7     | 1         | Collision  | (10, 11)       |
| 8     | 10        | Identified | (11)           |
| 9     | 11        | Identified | Empty          |

Law et al. [85] also propose numerous extensions to the QT protocol. They are

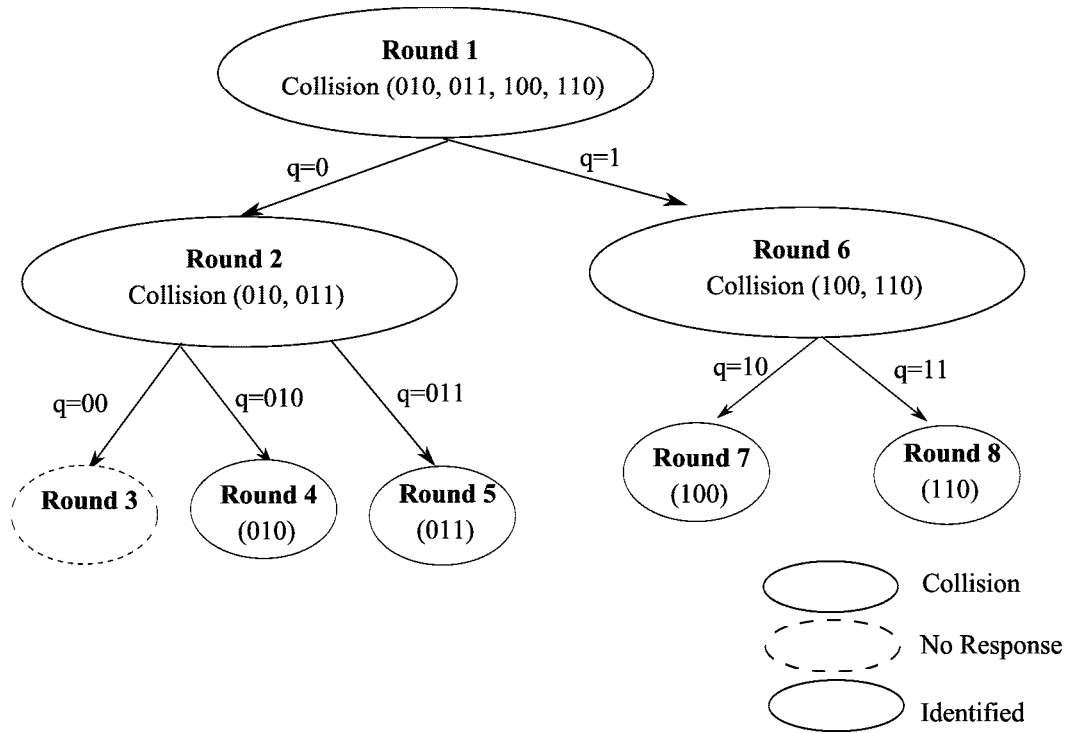


**Figure 2.9** The QT Algorithm.

summarized below [85]:

*Shortcutting:* This extension reduces QT's identification delay by removing redundant queries. It works as follows. The reader transmits a query  $q$ , and if there was a collision, the reader appends  $q$  with 0 and 1, and pushes  $q0$  and  $q1$  onto the stack. The reader first transmits the query  $q0$ . If there was no response, the reader infers that at least two tags have the prefix  $q1$ . Thus, if the reader transmits  $q1$ , a collision will occur. Therefore, the reader removes the query  $q1$  from the stack and pushes  $q10$  and  $q11$  onto the stack instead. Figure 2.10 shows the shortcutting procedure using the example shown in Figure 2.9. In round 2, a collision occurs for query 0. In round 3, the reader transmits query 00 but received no response. The reader then skips the transmission of

query 01, and pushes queries 010 and 011 onto the stack. Tags 010 and 011 are then identified in round 4 and 5 respectively. Notice that in Figure 2.9 there is a collision in round 4, which does not exist when using the shortcutting extension.



**Figure 2.10** QT with Shortcutting.

*Aggressive enhancement.* In this extension, queries are appended with multiple bits, instead of a single bit. For example, if query  $q$  causes a collision, the reader proceeds with queries  $q00$ ,  $q01$ ,  $q10$  and  $q11$  directly. This approach requires more queries compared to the original QT protocol, and is suitable for high tags density RFID environments.

*Categorization.* In this QT enhancement, the reader has prior knowledge of tag IDs, thereby allowing the reader to group tags according to predefined prefixes.

*QT-sl (Query-tree short-long) protocol.* Here, the reader separates tag responses into short and long queries. Short queries solicit 1-bit response from tags, while long queries cause tags to send all bits of their ID. Long queries are sent when

the reader knows there is only going to be one matching tag.

*QT-im (Query-tree incremental-matching) protocol.* QT-im reduces the number of query bits transmitted by requiring tags to remember the last query sent by the reader. For example, if the query transmitted by the reader in the last read round is  $q$ , then in the next read round, instead of sending query  $q0$  or  $q1$ , the reader transmits 0 or 1.

Lastly, Choi et al. [86] propose a scanning based pre-processing (SBPP) technique that uses Manchester coding to locate collided bits in tag responses. The reader notifies tags the whereabouts of these collided bits, and uses a QT algorithm to identify them.

### 2.3.2.2 Adaptive QT (AQT)

In [87] [88], Myung et al. propose a protocol, called the adaptive query tree (AQT), that requires the reader to maintain a queue  $Q$ , which operates similarly to the stack in the QT algorithm. In addition, the reader is required to maintain a candidate queue (CQ) for storing queries sent in past identification rounds.

Using AQT, the earlier tags set can be identified as follows. Initially, with no past information, the tree construction of AQT is similar to the QT protocol; see Figure 2.9. Once the tree is formed, leaf nodes 00, 010, 011, 10 and 11 are stored in CQ. The leaf nodes comprise of no response queries and those with a single tag response. To re-identify the same set of tags again, the reader uses the queries stored in CQ; i.e., 010, 011, 10 and 11.

To identify new tags, the reader relies on CQ. Consider two new tags, 111 and 000. The reader begins with query 00, which matches tag 000. Tags 010, 011, and 100 are identified using queries 010, 011 and 10 from the CQ respectively. Query 11 results in a collision between tags 110 and 111. Thus, the reader pushes queries 110 and 111 onto the stack. These two queries are then used to identify tags 110 and 111. Figure 2.11 shows the updated tree with tags 000 and 111. Lastly, CQ is updated to store the new leaf nodes.

On the other hand, if a tag, say 111, departs, there will be no response for the query 111. This means for query 11, only tag 110 replies. The reader then replaces queries 111 and 110 with the query 11. The resulting tree only have a single response node for query 11, as shown in Figure 2.12.

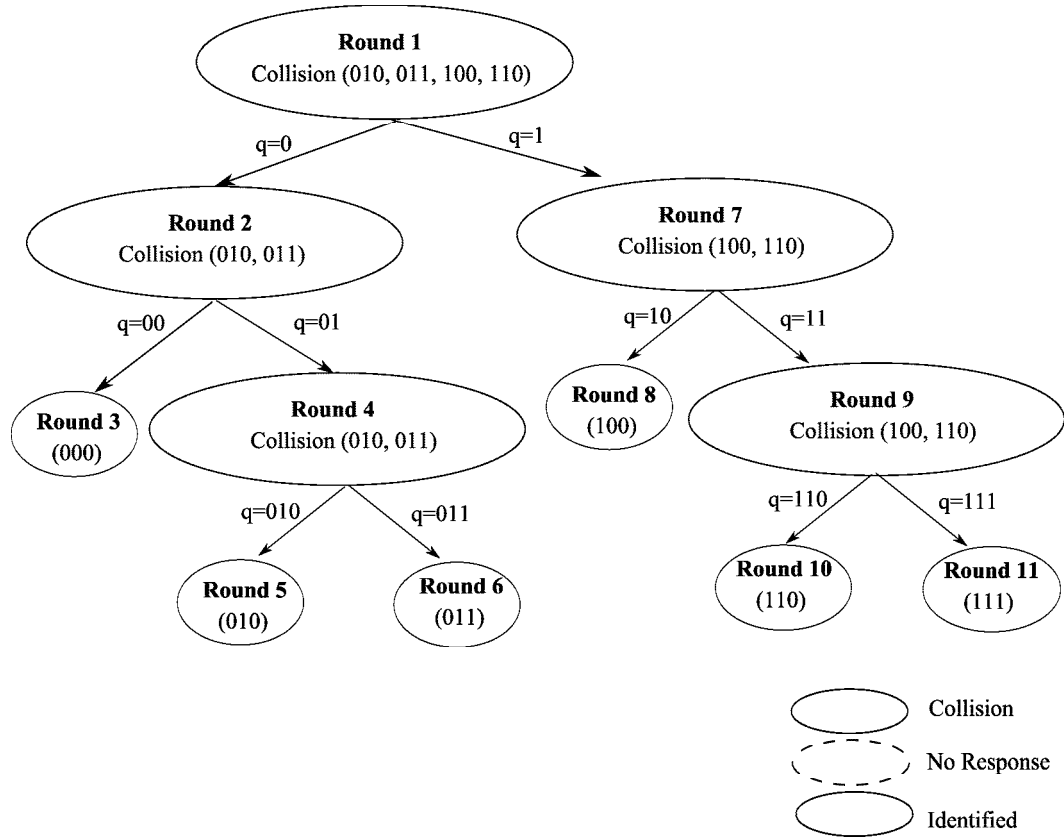
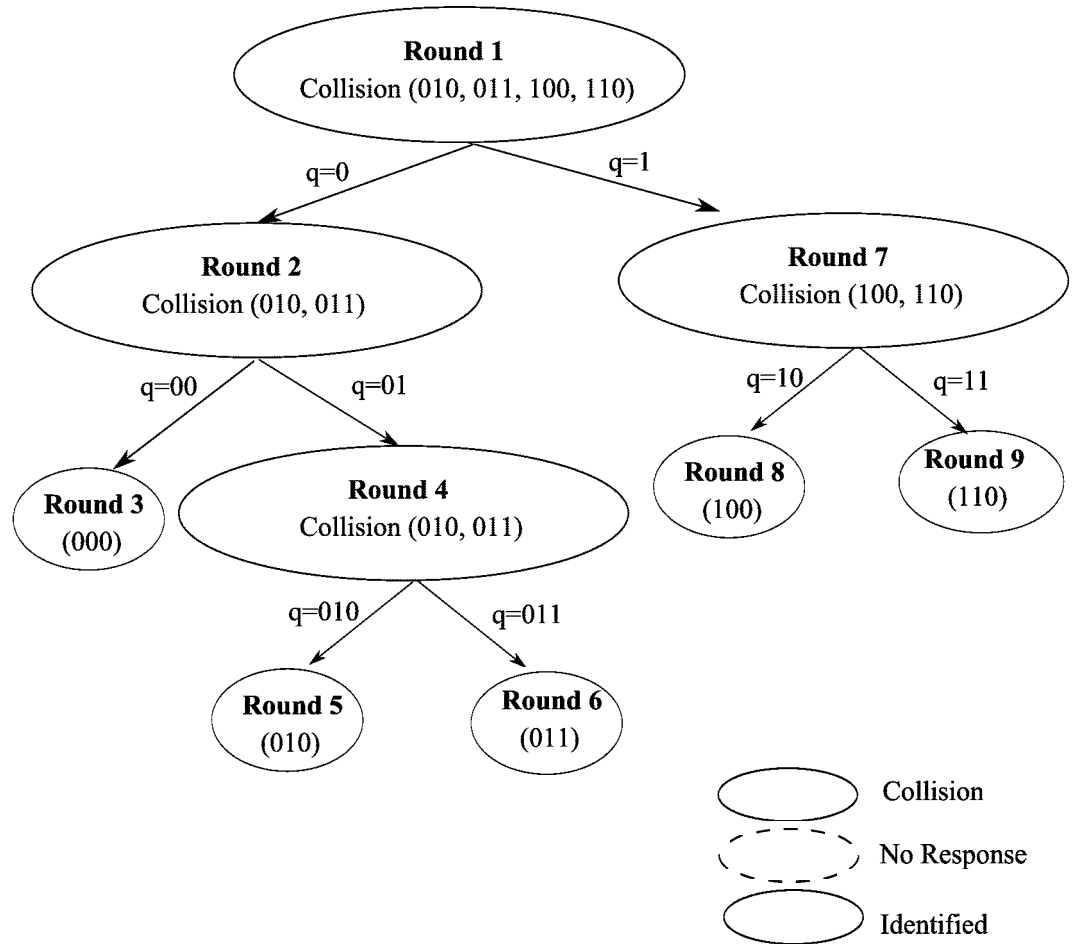


Figure 2.11 AQT- new tags 111 and 000.

### 2.3.2.3 Improved QT

Zhou et al. [89] improve the QT algorithm, referred to as IQT, by reducing the number of bits transmitted from tags to the reader when a collision occurs. The key feature of IQT is that the reader monitors tag responses in a bit by bit manner. If a collision occurs at a particular bit, the reader signals tags to stop transmitting.



**Figure 2.12** AQT-departed tag 111.

#### 2.3.2.4 QT Based Reservation (QTR)

Choi et al. [90] propose a QTR algorithm. The key difference to QT protocol is that tags use a 16-bit random number (RN16) during the identification process. After the RN16 is identified, the reader requests tags to respond with their complete ID.

#### 2.3.2.5 Randomized Hashing Query Tree (RH-QT)

Bonuccelli et al. [91] introduce a randomized hashing based QT approach. Each tag generates a random number from a predefined hash function using parameters sent by the reader. The reader has prior knowledge of all possible random

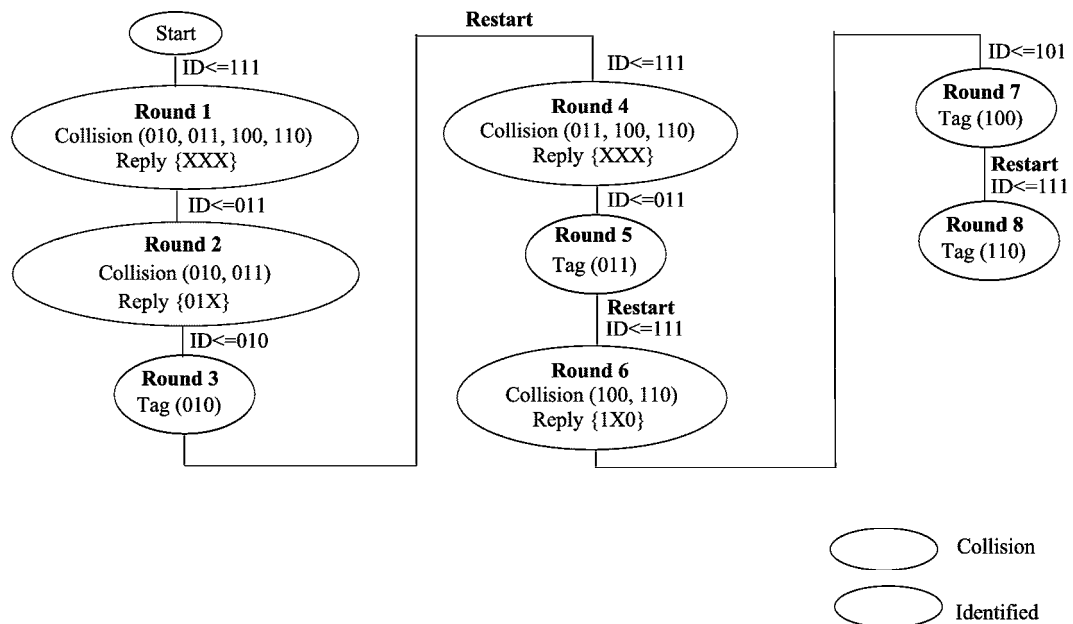
numbers that can be generated from the hash function. The reader then uses these numbers to query tags. A tag replies if it finds that the number sent by the reader matches its own number. If multiple tags have the same random number, collisions occur. Hence, these tags will have to select a new random number, and the reader then repeats the process to identify the collided tags.

### 2.3.2.6 Intelligent Query Tree (IQT)

IQT [92] exploits tags' prefix patterns, e.g., common vendor or product ID. This means a reader using IQT will first identify common prefix bits, and skips these bits in subsequent read rounds.

### 2.3.3 Binary Search (BS)

BS protocols [27] involve the reader transmitting a serial number to tags, which they then compare against their ID. Those tags with ID equal to or lower than the serial number respond. The reader then monitors tags reply bit by bit using Manchester coding, and once a collision occurs the reader splits tags into subsets based on collided bits.



**Figure 2.13** The BS Protocol.

Figure 2.13 depicts a reader using BS to read the tags set presented earlier. Initially, the reader starts reading with the maximum possible tag ID value, i.e., 111. Tags with an ID value less than 111 respond, resulting in the reply XXX. This indicates all three bits have experienced a collision. The reader then transmits another query by replacing the most significant collided bit with 0, and sets the other bits to 1, i.e., the new query becomes 011. This subsequent query solicits the response 01X. The reader then sends the query 011. Only tag 010 have ID lower than 011 and therefore it is identified successfully. After that, the reader restarts the reading with query 111.

Yu et al. [93] present a variant of BS called enhanced-BS (EBSA). The key difference to BS is that EBSA does not restart the reading process after a tag is identified. Moreover, during initialization, the reader transmits a '1' instead of sending a serial number consisting of all ones. Liu et al. [94] improved EBSA further by identifying two tags simultaneously when there is only a single collided bit.

Another enhancement to the BS protocol is called the dynamic BS algorithm (DBSA) [27]. In DIDS, the reader and tags do not use the entire length of serial number and tags ID during the identification process. For example, if a reader receives the response 01X, tags only need to transmit the remaining part of their ID since the reader has identified the prefix 01. The enhancement halves the amount of data sent by the reader to tags.

### 2.3.4 Bitwise Arbitration (BTA) Algorithms

Researchers have proposed various BTA algorithms. Unlike TS, QT, and IDS protocols, BTA algorithms operate by requesting tags to respond bit by bit from the most significant bit (MSB) to the least significant bit (LSB) of their ID. The key feature of BTA algorithms is that bit replies are synchronized, meaning multiple tags responses of the same bit value result in no collision. A collision is observed only if two tags respond with different bit values. Moreover, the reader has to specify the bit position it wants to read.



### 2.3.4.1 ID-Binary Tree Stack

The ID binary tree stack (ID BTS) [95] works by constructing a binary tree that has a height  $k$ , corresponding to the maximum tag ID with length  $k$ . Every branch corresponds to the bit of the tag ID. For any node  $x$  in the ID-binary tree, the left and right branch is labeled with binary zero and one respectively. A path from the root to an internal node represents a tag prefix, and a path from the root to a leaf node leaf node defines a unique tag ID.

The reader uses a stack to store tags' position on the tree, while a tag has a counter to record the depth of the reader's stack. Based on this counter value, a tag determines whether it is in the transmit or wait state. In other words, a counter value of zero moves a tag into the transmit state. Otherwise, the tag enters the wait state. Once a tag is identified, it enters the sleep state.

**Table 2.7** Reader stack and tags counter.

| Round | Response | Tag Counter    |                |                |                | Reader's Stack |
|-------|----------|----------------|----------------|----------------|----------------|----------------|
|       |          | Tag A          | Tag B          | Tag C          | Tag D          |                |
| 1     | X        | 0 (Transmit)   | 0 (Transmit)   | 0 (Transmit)   | 0 (Transmit)   | Empty          |
| 2     | 1        | 0 (Transmit)   | 0 (Transmit)   | 1 (Wait)       | 1 (Wait)       | (1)            |
| 3     | X        | 0 (Identified) | 0 (Identified) | 1 (Wait)       | 1 (Wait)       | (1)            |
| 4     | X        | —              | —              | 0 (Transmit)   | 0 (Transmit)   | Empty          |
| 5     | 0        | —              | —              | 0 (Transmit)   | 1 (Wait)       | (11)           |
| 6     | 0        | —              | —              | 0 (Identified) | 1 (Wait)       | (11)           |
| 7     | 1        | —              | —              | —              | 0 (Transmit)   | Empty          |
| 8     | 0        | —              | —              | —              | 0 (Identified) | Empty          |

Figure 2.14 shows the construction of an ID binary tree for the example tags set A=010, B=011, C=100, and D=110. Table 10 shows the reader stack and tags counter. In round 1, the reader commands tags to respond with their first or MSB, which results in a collision. The reader then transmits a control bit to silence tags that responded with a binary one. After that, the reader pushes a binary one into the stack, and silenced tags increment their counter by one to record their stack position. The reader then proceeds to read tags that responded with a binary zero in round 2. The reader requests the second MSB from tags A and B, which is received correctly as both tags transmitted a bit value of one. In round 3, tags respond with their third ID bit, causing a

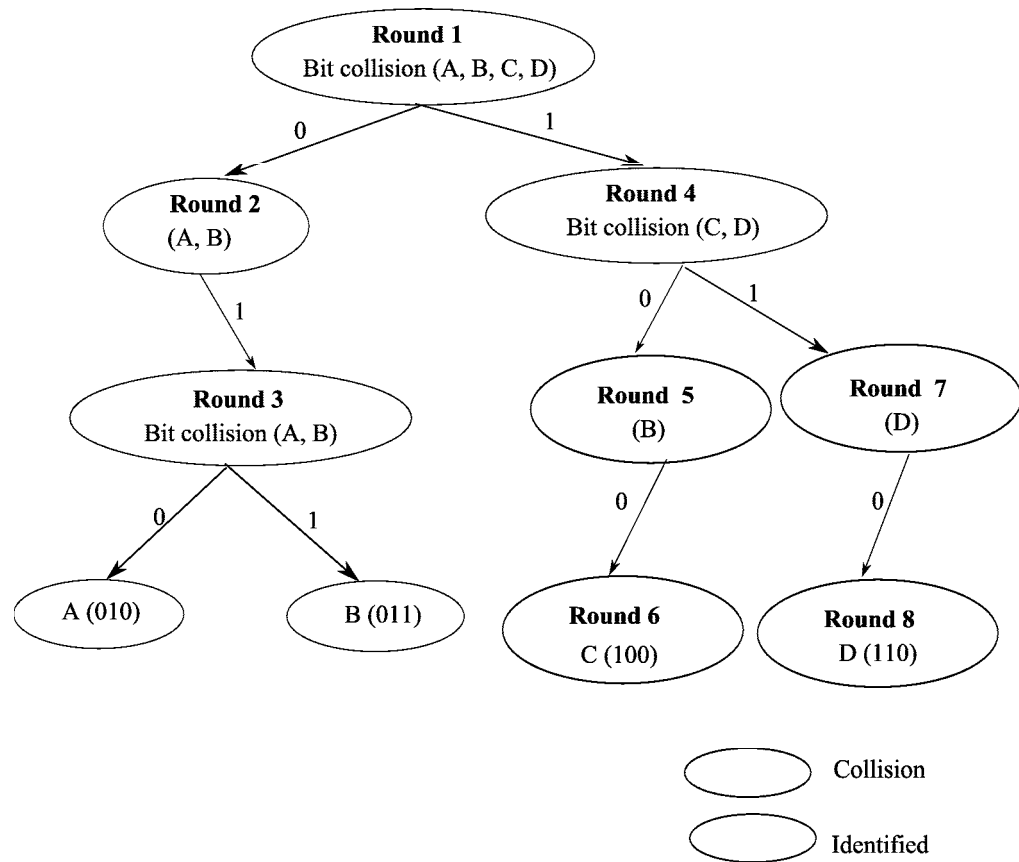


Figure 2.14 ID-BTS.

collision. Since the tag ID in this example is three bits in length, a collision in the third bit indicates two responding tags have a third bit value of zero and one. The reader thus appends zero and one to the first two received bits, thereby identifying tags A and B successfully. After that, the reader pops binary one from the stack, which is the first bit of the silenced tags or in other words, the tree position of the silenced tags C and D. Also, tags C and D decrement their counter by one. In round 4, the reader requests the second bit from tags C and D, which ends in a collision. Similarly, the reader pushes binary one onto the stack. In round 5 and 6, the second and third ID bits of tag C are identified respectively. Finally, tag D is identified in round 8.

#### 2.3.4.2 Bit-by-bit (BBT)

Jacomet et al. [96] present a BBT arbitration method where a separate channel is used for binary zero and one. When requested, each tag transmits the specified bit in one of these channels. If the reader receives a different response from both channels, it sends a control bit silencing the subset of tags that replied with 0 (or 1). On the other hand, if the reader receives a response in only one of the two channels, a bit is identified successfully. Similar to ID-BTS, the reader has a stack and each tag has a counter to store its tree position.

#### 2.3.4.3 Modified bit by bit binary tree (MBBT)

Choi et al. [86] propose the MBBT algorithm, which operates in a similar way to the BBT algorithm. The key difference is that MBBT does not use multiple timeslots to receive binary 0s and 1s.

#### 2.3.4.4 Enhanced bit by bit binary tree (EBBT)

Choi et al. [86] also proposed the EBBT algorithm. In EBBT, a reader first requests tags to respond with their complete ID. The assumption here is that tags responses are synchronized. From these responses, the reader identifies collided and collision-free ID bits. For example, let's say there are three tags 010, 100, and 110. Initially, the reader requests tags to respond with their entire ID, which resulted in the response XX0, indicating the first two bits have experienced a collision. The reader then uses MBBT to identify the collided bits.

#### 2.3.4.5 Bit query (BQ)

Kim et al. [97] [98] propose a bit query (BQ) algorithm. A reader transmits a bit query  $q$  to tags. Tags with their prefix matching the query  $q$  respond with the bit that is adjacent to the requested prefix. Other tags inactive themselves. If the reader receives a tag's bit response successfully, that bit is sent as the next query. However, if there is a collision, the reader uses bit zero as the next query.

Let's demonstrate the operation of BQ using the tags set earlier. Similar to the QT protocol, the reader maintains a stack and each tag has a counter. Figure 2.15 demonstrates the identification process for BQ. Initially, the reader transmits a bit query  $q = 0$ , and stores  $q = 1$ . This query solicits a 1 response from tags 010 and 011, which is the bit consecutive to the requested prefix. Since there is no collision, the reader uses  $q = 1$  as next query in round 2. This results in a collision due to differing bit responses received by the reader. The reader then uses  $q = 0$  as the next query, and stores  $q = 1$ . In round 3, tag 010 is the only tag with its last bit matching the requested bit, and therefore it is identified. The reader then retrieves query  $q = 1$  in round 4, which identifies the last bit of tag 011. After round 4, the reader transmits  $q = 1$  and tags 100 and 110 respond with 0 and 1, thereby resulting in a collision. Similarly, due to collision, the reader transmits  $q = 0$  in round 5 and stores  $q = 1$ . Tag 100 is identified in round 6. After that, the reader transmits the last prefix query  $q = 1$ , which identifies the last bit of tag 110 in round 7.

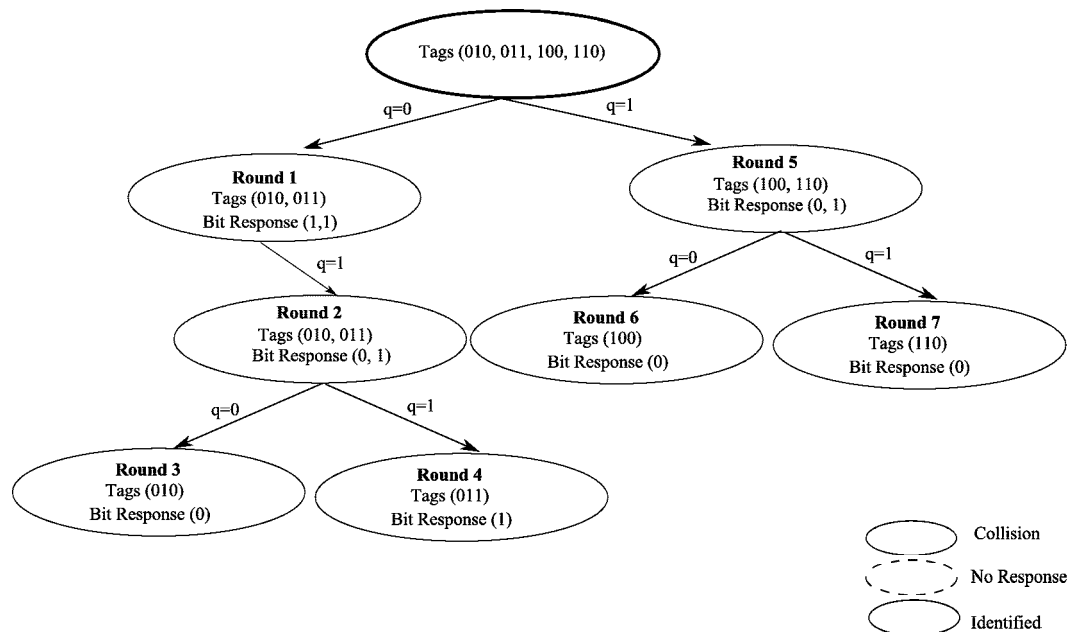


Figure 2.15 Bit Query (BQ).

### 2.3.5 Discussions

Table 2.8 compares tree protocols. Those using BTA require tags to respond bit by bit, hence are the most complex in terms of reader and tag hardware requirements when compared to QT, TS and BS protocols. Among all tree protocols, QT protocols promise the simplest tag design.

Tree algorithms provide a deterministic approach to identify tags. On the other hand, Aloha based approaches are probabilistic in nature, simple, and promise dynamic adaptability to varying loads; unlike tree protocols which must restart their reading process if a new tag enters a reader's interrogation zone while tags are being read. For these reasons, this thesis will focus solely on Aloha based protocols. Table 2.9 shows a comparison between Aloha and tree based algorithms.

## 2.4 Hybrid protocols

Hybrid protocols are a new branch of tag reading protocols that combine the advantages of tree and Aloha protocols. A number of protocols have been proposed under this category.

### 2.4.1 Tree Slotted Aloha (TSA)

TSA [101], an enhanced FSA protocol, uses a tree structure during the identification process. The root node of the tree denotes a frame to be transmitted in the first read round. Each tag remembers the slot number used to transmit. At the end of a read round, if there were collisions, the reader starts a new reading cycle for each collided slot. This corresponds to adding new nodes to the tree. Each tag has a counter to remember its position in the tree. Each time a collision occurs, a new node is inserted onto the tree, and another reading cycle is initiated. The whole process is repeated until a cycle is collision free.

Table 2.8 A comparison of tree protocols.

| Criterion   | Query Tree (QT)  | Tree Splitting (TS)   | ID Search (IDS)  | Bitwise Arbitration (BTA)  |
|---|--|---|--|--|
| <b>Protocol feature</b>   | The reader transmits a query, and tags with prefix matching the query respond. | The algorithm performs collision resolution by splitting collided tags into $b$ disjoint subsets. | The reader sends a serial number to tags, and those with values less than or equal to the serial number reply. | Each tag responds in a bit by bit manner.                            |
| <b>Tag Requirements</b>   | Prefix matching and synchronization circuitry.                                 | Random number generator, synchronization circuitry, and counters to store state information.      | Manchester coding scheme, synchronization circuits.  | Synchronization circuits, ability to respond in a bit-by-bit manner. |
| <b>RTF/ TTF</b>   |  |   | RTF  |  |
| <b>Time Complexity (1)(3)</b>   | $O(n)$   | $O(n)$  | $O(\log n)$  | $O(2^k)$   |
| <b>Message Complexity (2)(3)</b>  | $2.21k \log n + 4.19k$   | $n \log n$  | Not specified  | $O(n(k+1))$  |
| <b>Tag cost</b>   | Least  | ←   |  | → Most   |
| <b>System Cost</b>  | Least  | ←   |  | → Most   |
| (1) Time required to identify all tags.   |  |   |  |  |
| (2) Number of messages tags need to transmit before they are identified successfully. |  |   |  |  |
| (3) $n$ denotes the number of tags and $k$ denotes the length of tag's ID.            |  |   |  |  |

Table 2.9 Comparisons between tree and Aloha based algorithms.

| Criterion  | Tree protocols   | Aloha protocols  |
|--|--|--|
| <b>Protocol Feature</b>  | Tree protocol operates by grouping responding tags into subsets and then identifying tags in each subset sequentially. | Aloha based protocols require tags to respond randomly in an asynchronous manner or in synchronized slots or frames. |
| <b>Newly arriving tags</b> [99] [100]  | New tags cause the re-construction of an existing tree (2).  | New tags participate in collision resolution upon arrival.   |
| <b>Departing tags</b>  | A departing tag causes tree reconstruction (2).  | Departing tags do not affect reading.  |
| <b>Usage</b>   | Tree protocols are mainly used in UHF and microwave RFID systems.  | Aloha protocols are mainly used in LF and HF RFID systems.   |
| <b>Number of reader to tag commands</b>  | High   | Low  |
| <b>Tag Starvation</b>  | No   | Yes (1)  |
| <b>Delay versus tag density</b> [75]   | Low identification delays in high tag density environments.  | Low identification delays achievable only when tag density is low.   |
| <b>Method</b>  | Deterministic Deterministic  | Probabilistic  |
| <b>Optimum Channel Utilization</b>   | 43% [79]   | 18.4% (Pure Aloha), 36.8% [27] (BFSA) [27], 42.6% (DFSA) [79]  |
| (1) Tag starvation is largely mitigated by features such as muting and fast mode.                                      |  |  |
| (2) Recently proposed tree protocols such as ABS and AQT do not require tree re-construction for new or departed tags. |  |  |

## 2.4.2 Hybrid Query Tree (HQT) Protocol

Ryu et al. [102] combine the QT protocol with a slotted random back-off mechanism. The identification proceeds as follows. A reader transmits a two bits query to tags, and tags with a matching prefix respond after a back-off delay. The duration of the back-off timer is determined as follows. Let's say there are three tags 0100, 0101, and 0110. If the reader sends query 01, then the two bits following the prefix queried for each tag are 00, 01, and 10. These tags then set their backoff timer as zero, one and two slots respectively. Ryu et al. also proposed an enhanced HQT protocol, which uses the slotted back-off with the AQT protocol [87] [88].

### 2.4.2.1 HQT variants

Shin et al. [103] propose two algorithms that use a combination of QT and Framed Aloha protocols: Framed Query Tree Algorithm and Query Tree ALOHA Algorithm. In the former, the readers transmit a frame to tags, and tags choose a slot randomly. Within each slot, QT is used to identify tags. On the other hand, in the later algorithm, the reader transmits a prefix and frame size, and tags with a matching prefix choose a slot randomly in the frame. In other words, tags with a matching prefix are identified using framed Aloha protocol.

## 2.4.3 Hybrid Randomized Protocol

Namboodiri et al. [104] introduce three anti-collision protocols that combine the QT protocol with DFSA. The first of these, called multi-slotted (MS) scheme, relies on using multiple-slots per query to reduce the chances of collisions. The second, called Multi-Slotted with Selective Sleep (MSS), uses the muting feature to silence identified tags. The third scheme, called the Multi-Slotted with Assigned Slots (MAS) scheme, assigns tags a specific slot in a query frame. All three protocols are capable of adjusting their frame size after each query.



### 2.4.4 Hash-tree Protocol

Zhang et al. [105] present an advanced FSA algorithm which uses a hash function for slot selection in a reader's frame. The function is given in Equ. 2.10.

$$Hash(ID) = \frac{ID}{w} \% N \quad (2.10)$$

Where ID is the identification code of the tag,  $w$  is a positive integer provided by the reader, and  $N$  is the frame size. The reader starts reading with a frame size  $L$ . The maximum possible frame size is  $L_{max}$ . The reader then estimates the number of tags as  $2.39c_k$ , where  $c_k$  is number of collided slots. If the number of estimated tags is less than the current frame size, the collided tags are identified in sub-frames using an approach similar to MS algorithm. Otherwise, the reader expands the current frame size to 2.39 times the original frame size

### 2.4.5 Discussions

From the aforementioned works, it is clear that most hybrid protocols combine the QT protocol with a Aloha variant. This is because QT helps a reader separates tags into smaller groups, thereby reducing contention. Each group can then be read using a tree or an Aloha variant.

The results in [104] show that hybrid randomized protocols consume lower energy than the QT protocol. Specifically, MSS saves more energy than MS. Overall, MAS achieves the highest energy savings since it experiences less collisions. On the other hand, HQT [102] and its variants [103] outperform the QT protocol in terms of identification delays. Similarly, the number of collisions is lower in HQT and its variants compared to the QT protocol. In [101], the authors show that TSA achieves a higher system efficiency compared to DFSA, EDFSA, QT, and QT with an aggressive enhancement when the number of tags is more than 60. On the other hand, when the number of tags is lower than 50, QT with the aggressive enhancement has the highest system efficiency.

## 2.5 RFID Anti-Collision Standards

Two bodies are responsible for RFID air interface standards: EPCglobal [18] and international organization for standardization (ISO) [14]. EPCglobal develops industry-driven standards for international supply chain networks. ISO, on the other hand, specifies air interface specifications for tracking cattle, payment systems, contact less smart cards, and vicinity cards.

Table 2.10 summarizes ISO standards. ISO 18000-3 “MODE 1” has two extensions. The first uses Pure Aloha, and the other relies on DFSA. ISO 18000-3 “MODE 2”, on the other hand, uses a combination of frequency and time division multiple access. ISO 14443-3 Type-A and Type B use Dynamic BS algorithm and DFSA protocol respectively. ISO-18000-6A uses Framed slotted Aloha with muting and early-end, whereas ISO-18000-6B uses ID-BTS.

Table 2.11 presents the standards proposed by EPCglobal. Class 0 and 1, which are developed for UHF RFID systems, use a variant of ID-BTS. Specifically, Class 0 relies on ID-BTS, whereas Class 1 uses an advanced ID-BTS, where a tag transmits eight consecutive bits to the reader, which are then identified by the reader sequentially. The class 1 HF standard, on the other hand, uses BFSA with early-end. In addition, the protocol uses partial IDs during contention. The second generation of Class 1 uses the Q protocol [16].

Lastly, Table 2.12 shows propriety RFID specifications from Philips. I-Code and U- Code are developed for HF and UHF RFID systems respectively. I-Code uses DFSA for collision resolution and U- Code relies on the Q protocol [16]. Philips also proposed another HF standard, called *Mifare*, that uses the Dynamic BS algorithm.

From Table 2.10, 2.11 and 2.12, we can see that most HF RFID standards use an Aloha variant, whereas RFID standards for UHF use both Aloha and tree protocols. In general, standards with a low bandwidth air interface rely on Aloha variant. Otherwise, systems have the flexibility to choose either tree or Aloha variants.

Table 2.10 ISO RFID Standards [11] [12] [13] [14] [15].

| Standard             | Frequency | Protocol Used  |
|----------------------|-----------|--|
| ISO 18000-3 “MODE 1” | HF        | There are two extensions: Pure Aloha and Dynamic Framed Slotted Aloha.   |
| ISO 18000-3 “MODE 2” | HF        | This protocol is a combination of both frequency and time division multiple access (FTDMA). A tag has a choice of eight reply channels. After selecting a channel, the node uses slotted Aloha to access the channel. An extension here is to combine slotted Aloha with muting and slow down. |
| ISO 14443-3 Type-A   | HF        | Dynaic BS Algorithm (DBSA).  |
| ISO 14443-3 Type-B   | HF        | Dynamic Framed Slotted Aloha.  |
| ISO-18000-6A         | UHF       | Framed slotted Aloha with muting and early end.  |
| ISO-18000-6B         | UHF       | ID-BTS.  |

Table 2.11 EPC RFID Standards [16] [17] [18] .

| Standard                | Frequency | Protocol Used  |
|-------------------------|-----------|--|
| EPCglobal Class 0       | UHF       | ID-Binary Tree Stack (ID-BTS).   |
| EPCglobal Class 1       | UHF       | Advanced ID-BTS. The protocol is an advancement of ID-BTS, where a tag responds to the reader with eight consecutive bits, which the reader then reads sequentially. |
| EPCglobal Class 1 Gen 2 | UHF       | Q Protocol   |
| EPCglobal Class 1       | HF        | Basic Framed Slotted Aloha with early end. In addition, the protocol uses partial IDs during identification.   |

**Table 2.12** Proprietary Protocols [19] [20] [21].

| Standard       | Frequency | Protocol Used                 |
|----------------|-----------|-------------------------------|
| Philips I Code | HF        | Dynamic Framed Slotted Aloha. |
| Philips U Code | UHF       | Q Protocol.                   |
| Philips Mifare | HF        | Dynaic BS Algorithm (DBSA).   |

## 2.6 Summary

This chapter has presented a comprehensive survey and classification of anti-collision protocols. In general, two methods are used to identify tags: tree and Aloha. Tree protocols promise deterministic identifications but are complex, incur significant memory overheads, and require complex hardware [27] [95] [81]. This is in contrast to Aloha protocols, which have simpler reader designs, lower protocol complexity and bandwidth requirements, smaller number of reader to tag commands, and are able to dynamically adapt to varying number of tags. In addition, the use of muting in Aloha based protocols promises 100% read rate. All these properties argue in favor of Aloha based protocols being used in RFID-enhanced WSNs. Thus, Chapter 3 and 5 will focus solely on the energy efficiency of Aloha based anti-collision protocols.

## Pure and Slotted Aloha Based RFID Anti-Collision Protocols

This chapter aims to identify the most energy efficient variant amongst twelve Pure and Slotted Aloha based RFID anti-collision protocols. This is achieved using an analytical methodology that evaluates the energy consumed in the following phases: i) success, ii) collision, and iii) idle listening. First, the delay in each phase is computed and then it is used to formulate the energy consumption, battery lifetime, and battery wastage of all variants. Results show that Pure Aloha with fast mode consumes the lowest energy and is suitable for tag identification. However, no protocol promises energy efficient monitoring of identified tags. In other words, to monitor tags, the reader is required to re-read all tags; a process that consumes a significant amount of energy.

The rest of the chapter is organized as follows. Section 3.1 presents the system model and assumptions. This is then followed by the research methodology used to evaluate Aloha variants in Section 3.2. Delay equations for each of the aforementioned phases are derived in Section 3.3, which are then used in Section 3.4 to analyze each protocol's energy efficiency. Next, Section 3.5 presents related work and finally, Section 3.6 concludes the chapter.

### 3.1 System and Analytical Model

The system consists of an RFID reader and  $n$  tags in its interrogation zone. The reader model is based on the design features of SkyeTek's M1-Mini RFID reader [48]. This device is chosen because it is specifically designed for sensor networks and has very low energy consumption during scanning compared to other RFID readers.

The reader operates from a Lithium rechargeable battery ( $B$ ) which has 0.48 Kilo-joules of energy. The tag's data rate is 26 kbps (ISO 15693). The power consumed during scanning ( $\psi$ ) is 180 milli-watts.

The communications from a tag to the reader is modeled as a Poisson process [64]. Each tag responds on average  $\lambda$  times per second. The model requires independence among tag transmissions, which is supported by the lack of tag-to-tag communication capabilities. An RFID reader is assumed to transmit energy until all tags are read. Note, although tags are energized at the same time, our analysis quantifies the energy consumption after the reading process has started, and hence tag responses are independent of the reader. This means, the number of tags that replies in a time interval depends only on their transmission time and are independent of any transmissions that occur in other time intervals.

In a Poisson process, the mean arrival time between tag responses is  $\frac{1}{\lambda}$  [64], where  $\lambda$  is the average duty cycle of tags. To understand the impact of  $\lambda$ , the analysis to follow studies two  $\lambda$  values: 20 and 50. These choices are representative values that are sufficiently far apart. Other values exhibit similar trends and do not affect our comparisons between tag reading protocols. Therefore, results for other values of  $\lambda$  are omitted.

The analysis assumes the delay associated with energizing tags, and propagation and processing delays are negligible. Further, it assumes a noise free channel, i.e., packet losses are due to collisions only. The reader detects collisions when the cyclic redundancy check (CRC) fails, and it transmits an ACK only when an ID is received correctly. Finally, tag ID is 112 bits in size, which includes

16-bit  $s$  of CRC.

A tag waits  $t_{ACK}$  seconds for an ACK after transmitting its ID. Thus, the delay incurred by a tag to transmit its ID and receive an acknowledgment is  $T = T_{ID} + t_{ACK}$ . From [62], the size of  $t_{ACK}$  is six bits, hence  $t_{ACK} = 0.23\text{ms}$  when transmitted at 26 kbps.

Tags are assumed to be passive, with no power source, static, and are used in read-only mode. Further, tags' antenna is never at 90 degrees with respect to the reader. Otherwise, tags become unreadable, and hence they do not contribute to the offered load. In other words, a reader is unaware of tags that are displaced by 90 degrees since they are not energized to participate in any communications [27] [106].

Retransmission delays are bounded to  $K$  random slots. According to [64] and [107],  $K = 5$  is an optimum value for analysis. Increasing  $K$  after protocols have reached maximum throughput results in higher transmission delays [108].

For Pure Aloha with fast mode, a reader sends a *quiet* command to silence tags temporarily. The *quiet* command is sent asynchronously, in a separate channel, as soon as an ID transmission is detected [62]. The duration and length of the *quiet* command,  $t_{quiet}$ , is 0.19ms and 5-bits respectively. A tag's ID is validated by verifying its preamble, which consists of a predefined sequence of bits followed by synchronization bits. As per [62], on average, the reader sends a quiet command after a duration  $T_f = T_{ID}/4$ .

For Pure and Slotted Aloha with slow down mode,  $\lambda'$  denotes a tag's new response duty cycle after identification, where  $\lambda' = \lambda/r$ . Here,  $r$  is the magnitude by which a tag's duty cycle is reduced and it is assumed to have the value three i.e.,  $r = 3$  [62].

For protocols involving the early-end feature, an idle slot is only active for  $t$  time. According to [17],  $t$  can be approximated as  $\frac{T_{ID}}{10}$ , which includes the time required to sense for responses, and transmit SOF and EOF commands.



## 3.2 Research Methodology

In order to determine the scanning duration or average delay incurred by a reader to read a tag, we first compute the delay incurred in the following phases, i) success, when tags are read successfully, ii) collision, due to simultaneous tags responses, and iii) idle listening, where the reader receives no responses. The delay in each phase is then multiplied by the scanning power to derive the energy consumption, and the battery lifetime of an RFID-enhanced sensor node.

Let us now calculate a nodes battery lifetime i.e., the average number of tags a given battery can read in its life. This is determined by dividing the available battery energy with the average energy consumed to read a tag,

$$N_p = \frac{B}{E} \quad (3.1)$$

where  $B$  (Joules) is the capacity of a given battery,  $E = \psi D$  (Joules) is the average energy consumed to read a tag,  $\psi$  (Watts) is the power used during scanning, and  $D$  (seconds) is the average delay to read a tag from  $n$  tags. Another important evaluation is the amount of energy wasted while reading  $n$  tags. We first compute number of tags a battery can read and then use the result to derive the battery wastage during tag reading. Under ideal conditions, i.e., instantaneous tag response and no collisions, a battery can read  $N_{Ideal}$  number of tags in its lifetime. Consequently,

$$N_{Ideal} = \frac{B}{E_{single\_tag}} \quad (3.2)$$

where  $E_{single\_tag} = \psi T$  (Joules) is the energy consumed to read a tag. Subtracting Eq. 3.1 from 3.2 gives us the total number of transmissions which resulted in collisions and idle listening, so called “wasted” transmission opportunities,

$$N_{waste} = N_{Ideal} - N_p \quad (3.3)$$

Inserting Eq. 3.1 and 3.2 into 3.3, simplifying, and multiplying both sides by

$T$ , we get,

$$E_{single\_tag} N_{waste} = B \left( 1 - \frac{T}{D} \right) \quad (3.4)$$

The LHS of Eq. 3.4 gives us the average battery energy wasted  $B_{waste}$  (Joules) during tag identification. Eq. 3.4 can be rewritten to obtain,

$$B_{waste} = B \left( 1 - \frac{T}{D} \right) \quad (3.5)$$

Section 3.3 derives  $D$  for 1) Pure and Slotted Aloha with and without muting, 2) Pure and Slotted Aloha with slow down mode, 3) Pure Aloha with fast mode; with and without muting, 4) Slotted Aloha with early-end; with and without muting, and 5) Slotted Aloha with early-end; with and without slow down.

Once  $D$  is calculated, the energy consumption is obtained by multiplying it with a reader's power consumption. Next, Eq. 3.1 and 3.5 are used to compute a reader's battery lifetime and wastage respectively. After that, a comparison of all protocols is conducted according to the following metrics, (i) average energy consumption in each phase of the reading process, (ii) the total energy used to read a set of tags, (iii) estimated battery lifetime, and (iv) the average energy wastage.

### 3.3 Delay Equations

The average delay incurred by a reader to read a tag consists of two components:

1. *Arrival delay.* When reading starts, each tag transmits after a random delay. Since each tag's transmission is Poisson distributed, there is a mean delay of  $1/\lambda$  between consecutive transmissions. This is referred to as the arrival delay [64]. If there are  $n$  tags, then on average each tag takes  $\frac{1}{n\lambda}$  time to transmit its ID for the first time. Thus, for each tag, before it enters into contention, a reader experiences an average arrival delay of  $\frac{1}{n\lambda}$ , denoted by  $D_{Arrival}$ . Note, as muting, slow down, and fast

mode features are only activated after contention starts, the arrival delay is the same for all protocols, except for those using the early-end feature, where idle slots are closed early.

2. *Contention delay.* This is the average contention delay incurred by a tag before it is successfully identified.

### 3.3.1 Pure Aloha and Slotted Aloha

The following section first presents the average delay to read a tag successfully from  $n$  tags. Then Section 3.3.1.2 and 3.3.1.3 derive a reader's idle listening and collision delay.

#### 3.3.1.1 Average delay to read a tag successfully

In [64], the authors presented a Pure and Slotted Aloha model to evaluate the average delay taken by a node to transmit a packet successfully given  $n$  competing users. In RFID systems, tags and the reader are analogous to competing nodes and destination node respectively. Therefore, as per [64], the average delay to transmit a tag ID successfully to a reader is,

$$D_{Success} = T (1 + (e^{xG} - 1) \alpha) \quad (3.6)$$

where  $x = 2$  for Pure Aloha, and  $x = 1$  for Slotted Aloha,  $G$  is the offered load,  $T$  is the message transmission time, and  $K$  is the number of retransmission intervals of duration  $T$  and  $\alpha = (\frac{K+1}{2})$ . For Pure/Slotted Aloha,  $G = G_A = n\lambda T$  [64]. Thus, the average delay to successfully transmit a tag ID is,

$$D_{Success\_Tag} = \{D_{Success}\}_{G=G_A} \quad (3.7)$$

Equ 3.7 is derived as follows. For Aloha based protocols, during collisions, tags retransmit after a random time. In [64] and [108]'s analysis, the retransmission time is divided into  $K$  slots of duration  $T$ , which is assumed to be uniformly distributed. Each tag retransmits at random during one of the next  $K$  random slots with probability  $1/K$ . This means tags will retransmit within a period of

$K \times T$  after experiencing a collision. On average, a tag will retransmit after a duration of  $((K + 1)/2)T = \alpha$  slots.

The number of collisions before a tag successfully response is  $e^{xG_A} - 1$ , where  $e^{xG_A}$  denotes the average transmission attempts made before a successful identification. Since each collision is followed by a retransmission, the average delay before a successful tag response is calculated as  $(e^{xG_A} - 1)\alpha$ , followed by a single successful transmission of duration  $T$ . In total, the average delay a tag takes to transmit its ID successfully to the reader is  $(e^{xG_A} - 1)\alpha T + T = D_{Success\_Tag}$ .

With the contention and arrival delay in hand, the average delay a reader takes to read a tag successfully is,

$$D_{Success\_PS} = D_{Success\_Tag} + D_{Arrival} \quad (3.8)$$

$$= T [1 + (e^{xG_A} - 1) \alpha] + \frac{1}{n\lambda} \quad (3.9)$$

### 3.3.1.2 Average delay in idle listening

The idle listening delay is the sum of two values:  $D_{Arrival}$ , and the average idle delay experienced by the reader during contention  $D_{Idle\_Reader}$ .

To derive  $D_{Idle\_Reader}$ , the following methodology is used. For Pure and Slotted Aloha, if the offered load is  $G_A$ , the mean number of successful transmissions out of  $G_A$  is  $G_A e^{-xG_A}$ , which corresponds to the throughput of Pure and Slotted Aloha [64].  $G_A$  in terms of slots is computed as  $G_{Slots} = G_A/T$ . Therefore, if the offered load is  $G_{Slots} = G_A/T$ , the reader observes  $G_{slots} e^{-xG_{slots}}$  collision-free slots. Thus, there are  $N_{Idle\_Tag}$  slots for which a tag is waiting to transmit, a reader is idle only for  $N_{Idle\_Tag} e^{-xN_{Idle\_Tag}}$  slots. The following paragraphs show how to compute  $N_{Idle\_Tag}$  and  $D_{Idle\_Reader}$ .

In Eq. 3.9, the term  $e^{xG_A}$  denotes the average number of attempts made before a tag is identified. The corresponding average delay for these attempts is,

$$D_{Attempts\_Tag} = T e^{xG_A} \quad (3.10)$$

The average delay incurred by a tag while waiting to transmit is determined by subtracting Eq. 3.10 from Eq. 3.7. In other words,  $D_{Success\_PS} - D_{Attempts\_Tag}$ . In terms of slots, we have,

$$N_{Idle\_Tag} = \frac{D_{Success\_Tag} - D_{Attempts\_Tag}}{T} \quad (3.11)$$

$$= (e^{xG_A} - 1)\beta \quad (3.12)$$

where  $\beta = \frac{K-1}{2}$ . Eq. 3.12 can also be interpreted as the number of slots where a reader observes no response, so called idle slots. Therefore, we have,

$$D_{Idle\_Reader} = TN_{Idle\_Tag}e^{-xN_{Idle\_Tag}} \quad (3.13)$$

Finally, the average delay due to idle listening is,

$$D_{Idle\_PS} = D_{Idle\_Reader} + D_{Arrival} \quad (3.14)$$

$$= T \left[ (e^{xG_A} - 1)\beta e^{-x(e^{xG_A}-1)\beta} \right] + \frac{1}{n\lambda} \quad (3.15)$$

Note, the same methodology is used to derive the idle listening delay of all protocols of interest.

### 3.3.1.3 Average delay in collisions

The average delay experienced by a reader due to collisions is,

$$D_{Collisions\_PS} = D_{Success\_PS} - D_{Idle\_PS} - T \quad (3.16)$$

For each of the subsequent protocols, the average delay due to collisions is evaluated using Eq. 3.16, hence are omitted from discussion.

## 3.3.2 Pure and Slotted Aloha with Muting

When muting is used, the number of tags in a reader's interrogation zone becomes smaller after each successive identification, and in turn reduces the offered load. If  $i$  out of  $n$  is the number of tags which are identified and then muted by the reader, then the offered load due to the remaining tags is given by,

$$G_A(n - i) = (n - i)\lambda T \quad (3.17)$$

As the number of muted tags increases from  $i = 0, 1, 2, 3, \dots, n$ , the offered load reduces to  $G_A(n), G_A(n-1), G_A(n-2), G_A(n-3), \dots, G_A(0)$ .

Therefore, the average offered load to the reader for Pure/Slotted Aloha with muting,  $G_B$ , is given by,

$$G_B = \lambda T \frac{\sum_{i=0}^{n-1} (n-i)}{n} \quad (3.18)$$

Inserting  $G_B$  from Eq. 3.18 into Eq. 3.9, the average delay to read a tag using Pure/Slotted Aloha with muting is,

$$D_{Success\_PS\_Mute} = \{D_{Success\_PS}\}_{G_A=G_B} \quad (3.19)$$

### 3.3.3 Pure and Slotted Aloha with Slow Down

When slow down is used, the duty cycle or mean transmission by a tag is reduced to  $\lambda' < \lambda$ . Since  $\lambda$  is directly proportional to the offered load, slowed down tags contribute with a lower load compared to the remaining tags. Thus, in order to formulate the average delay of a tag with slow down mode, we need to calculate the offered load experienced by a reader with varying duty cycles. If there are  $j$  identified or slowed down tags, then the offered load due to these tags is,

$$G_1 = j\lambda'T \quad (3.20)$$

The remaining  $(n-j)$  tags will have an offered load equal to,

$$G_2 = (n-j)\lambda T \quad (3.21)$$

The combined offered load to the reader is then  $G_{Tot} = G_1 + G_2$ . In other words,

$$G_{Tot} = j\lambda'T + (n-j)\lambda T \quad (3.22)$$

$$= T(j\lambda' + (n-j)\lambda) \quad (3.23)$$

With each identified tag, the offered load reduces. This is represented in Eq.

3.25.

$$G_C = \frac{\sum_{j=0}^{j=n-1} G_{Tot}}{n} \quad (3.24)$$

$$= T \frac{\sum_{j=0}^{j=n-1} ((n-j)\lambda + j\lambda')}{n} \quad (3.25)$$

In Eq. 3.25,  $\lambda' = \lambda$  indicates there is no change in a tag's duty cycle after identification. In that case, Eq. 3.25 reduces to  $G_C = n\lambda T$ , which is the same as  $G_A$  for Pure and Slotted Aloha without slow down. On the other hand, if  $\lambda' = 0$  then Eq. 3.25 reduces to  $G_C = \lambda T \frac{\sum_{j=0}^{j=n-1} (n-j)}{n+1}$ , which has the same form as  $G_B$  for Pure and Slotted Aloha with muting; as in Eq. 3.18 with  $j = i$ . Therefore, conventional Pure and Slotted Aloha is a special form of Pure and Slotted Aloha with slow down when tags' duty cycle is unchanged. Similarly, muting is a special form of slow down where tags' duty cycle is zero.

Lastly, to simplify Eq. 3.25,  $\lambda'$  can be made an integral fraction of  $\lambda$ ,  $\lambda' = \lambda/r$ , thereby reducing a tag's duty cycle by a factor of  $r$ . In this case, Eq. 3.25 becomes,

$$G_C = T\lambda \frac{\sum_{j=0}^{j=n-1} (n-j(1 - \frac{1}{r}))}{n} \quad (3.26)$$

By inserting  $G_A = G_C$  in Eq. 3.9, we can evaluate the average delay to read a tag successfully using Pure Aloha with slow down mode. Similarly, Pure Aloha and Slotted Aloha with the slow down and muting features can be analyzed using the methodology presented in Section 3.3.2.

Note that an iterative approach is used to compute the offered load in order to maintain independence among tag transmissions. This is because in practice tags rely on ACK sent by the reader to slow down, and thereby nullify the independence assumption required by the Poisson model.

### 3.3.4 Pure Aloha (Fast Mode)

When fast mode is used, the reader inhibits other tags from transmitting by sending a *quiet* command in a separate channel after receiving a tag's preamble successfully. Specifically, the reader spends  $T_f$  time determining whether the

preamble is valid before transmitting a *quiet* command to other tags to stop them from transmitting.

In order to receive a tag ID successfully, a reader must not experience any collisions for a duration of  $T_f$  when receiving. Thus, the probability that no tag transmits in  $T_f$  is<sup>1</sup>,

$$P_s[0, T_f] = \frac{[G_D]^0 e^{-G_D}}{0!} \quad (3.27)$$

$$= e^{-G_D} \quad (3.28)$$

Tag responses during  $T_f$  constitute the offered load to the reader. Therefore, the average offered load to the reader for Pure Aloha with fast mode is,

$$G_D = n\lambda T_f \quad (3.29)$$

In order to determine the average delay incurred to read a tag, we also need to determine the average retransmissions made by a tag before it is identified.

Using Eq. 3.28, the throughput of Pure Aloha with slow down is calculated as,

$$S = G_D P_s \quad (3.30)$$

$$= G_D e^{-G_D} \quad (3.31)$$

The mean attempts taken by a tag to transmit its ID successfully is  $G_D/S$  or  $e^{G_D}$ . From this, the average delay due to unsuccessful attempts is,

$$D_{Attempts\_Failed} = T (e^{G_D} - 1) \alpha \quad (3.32)$$

---

<sup>1</sup>Assuming a Poissonian probability distribution.



$D_{Attempts\_Failed}$  corresponds to the number of attempts made before a tag manages to transmit its preamble successfully. The delay experienced by a reader to receive preamble correctly is therefore,

$$D_{Preamble} = T_f + D_{Attempts\_Failed} \quad (3.33)$$

$$= T_f + T(e^{G_D} - 1)\alpha \quad (3.34)$$

After receiving a tag's preamble, the reader silences the remaining tags and proceeds to receive the rest of the tag's ID. The average delay to read a complete tag ID successfully is therefore,

$$D_{Success\_fast} = D_{Preamble} + (T - T_f) + t_{quiet} \quad (3.35)$$

$$= T[1 + (e^{G_D} - 1)\alpha] + t_{quiet} \quad (3.36)$$

Finally, adding the mean arrival delay experienced by the reader in Eq. 3.36, we have the average delay to receive a tag ID successfully for Pure Aloha with fast mode,

$$D_{Success\_fast} = T[1 + (e^{G_D} - 1)\alpha] + \frac{1}{n\lambda} + t_{quiet} \quad (3.37)$$

Idle listening delay can be obtained using a similar methodology to Pure/Slotted Aloha in Section 3.3.1 and the final expression is,

$$D_{Idle\_fast} = T[(e^{G_D} - 1)\beta e^{-(e^{G_D} - 1)\beta}] + \frac{1}{n\lambda} \quad (3.38)$$

Finally, the delay due to collisions is computed as

$$D_{Collisions\_PS} = D_{Success\_fast} - D_{Idle\_fast} - (T + t_{quiet}) \quad (3.39)$$

Pure Aloha with fast mode can also include muting. In that case, the methodology presented in Section 3.3.2 is used to analyze its performance.

### 3.3.5 Slotted Aloha Variants with Early End

The key consideration when the early-end feature is used is the reduction in the average delay to read a tag due to the early closure of idle slots. Let  $t < T$  be the duration after which a reader closes a slot if no responses are detected. Then, the average delay to read a tag in Slotted Aloha variants with early-end is,

$$D_{Success\_E} = \{D_{Success\_PS}\}_{G=G_Y} - (T - t)\{N_{Idle\_PS}\}_{G=G_Y} \quad (3.40)$$

where  $D_{Success\_PS}$  is given by Eq. 3.9,  $N_{Idle\_PS}$  is equal to Eq. 3.15 divided by  $T$ , and  $G_Y = G_A$  for Slotted Aloha with early-end,  $G_B$  for Slotted Aloha with muting and early-end, and  $G_C$  for Slotted Aloha with slow down and early-end.

### 3.3.6 Putting It All Together

Let us now use the delay equations of Pure and Slotted Aloha without muting to quantify energy consumption, battery lifetime and battery wastage. To evaluate the energy consumption,  $D$  is multiplied with readers power consumption. After that, Eq. 3.1 and 3.6 is used to derive battery lifetime and wastage respectively.

For Pure and Slotted Aloha,  $D_{Success\_PS}$  is the average energy consumed to read one tag successfully,  $D_{Collisions\_PS}$  is the average delay due to collisions, and  $D_{Idle\_PS}$  is the average idle listening delay. The energy consumption of each phase is,

$$E_{Success} = \psi \times D_{Success\_PS} \quad (3.41)$$

$$E_{Collisions} = \psi \times D_{Collisions\_PS} \quad (3.42)$$

$$E_{Idle} = \psi \times D_{Idle\_PS} \quad (3.43)$$

Adding Eq. 3.42 and 3.43, the average energy wasted due to collisions and idle listening is,

$$E_{Waste} = \psi \times (D_{Collisions\_PS} + D_{Idle\_PS}) \quad (3.44)$$

To evaluate a reader's battery lifetime,  $D_{Success\_PS}$  is inserted into Eq. 3.1 and 3.6 to obtain,

$$N_{Lifetime\_Pure\_Aloha} = \frac{B}{\psi D_{Success\_PS}} \quad (3.45)$$

$$B_{Waste\_Pure\_Aloha} = B \left( 1 - \frac{T}{D_{Success\_PS}} \right) \quad (3.46)$$

For each protocol of interest, the above procedure is repeated to obtain their respective energy consumption equations. The resulting equations are presented in Table 3.1.

**Table 3.1**

Energy consumption of Pure/Slotted Aloha.  $E_{Success}$  - Average energy consumed to read one tag successfully,  $E_{Collisions}$  - Average energy consumed in collision resolution before a tag is read,  $E_{Idle}$  - Average energy consumed in idle listening before a successful tag is read,  $N_{Battery\_Life}$  - The number of tags a battery can read,  $B_{Waste}$  - Battery wastage due to anti-collision.  $x = 1$  for Slotted Aloha and  $x = 2$  for Pure Aloha,  $E_{Collision} = E_{Success} - E_{Idle} - \psi T$ ,  $\alpha = \frac{K+1}{2}$ ,  $\beta = \frac{K-1}{2}$

|  |  |
|--|--|
| Protocols  | i) Pure/Slotted Aloha Non-Muting, $G = G_A$<br>ii) Pure/Slotted Aloha Muting, $G = G_B$<br>iii) Pure/Slotted Aloha Slow-Down, $G = G_C$                                |
| $E_{Success}$  | $\psi \left[ T \left\{ 1 + (e^{xG} - 1) \alpha \right\} + \frac{1}{n\lambda} \right]$  |
| $E_{Idle}$   | $\psi \left[ T \left\{ (e^{xG} - 1) \beta e^{-x(e^{xG}-1)\beta} \right\} + \frac{1}{n\lambda} \right]$   |
| Protocols  | i) Pure Aloha-Fast mode, $G = G_D$<br>ii) Pure Aloha with Fast Mode and Muting, $G = G_E$<br>iii) Pure Aloha with Fast Mode and Slow down, $G = G_F$                   |
| $E_{Success}$  | $\psi \left[ T \left\{ 1 + (e^G - 1) \alpha \right\} + \frac{1}{n\lambda} + t_{quiet} \right]$   |
| $E_{Idle}$   | $\psi \left[ T \left\{ (e^G - 1) \beta e^{-(e^G-1)\beta} \right\} + \frac{1}{n\lambda} \right]$  |
| Protocols  | i) Slotted Aloha with Early End, $G = G_A$<br>ii) Slotted Aloha with Early End and Muting, $G = G_B$<br>iii) Slotted Aloha with Early End and Slow Down, $G = G_C$     |
| $E_{Success}$  | $\psi \left[ T \left\{ 1 + (e^G - 1) \alpha + \frac{1}{n\lambda} \right\} - (T - t) \left\{ (e^G - 1) \beta e^{-(e^G-1)\beta} + \frac{1}{n\lambda T} \right\} \right]$ |
| $E_{Idle}$   | $t\psi \left[ (e^G - 1) \beta e^{-(e^G-1)\beta} + \frac{1}{n\lambda T} \right]$  |
| Note: Dividing the above equations by $\psi$ give us the average delay in different anti-collision phases, which can then be used to evaluate battery lifetime $N_{Battery\_Life}$ and wastage $B_{Waste}$ |  |

### 3.4 Results

In the following sections, equations from 3.1 are used to compare, i) the average energy consumed to read one tag successfully, ii) the average energy wasted in collisions, iii) the average energy spent in idle listening, iv) battery efficiency, i.e., the number of tags a battery is able to read, and v) battery wastage.

### 3.4.1 Average Energy Consumed to Read One Tag

This section compares and contrasts the energy consumption of Pure and Slotted Aloha variants.

Figure 3.1 compares Pure Aloha variants. The energy consumption of conventional Pure Aloha is the highest because collisions consume the most energy due to its well known vulnerability period of  $2T$  [27].

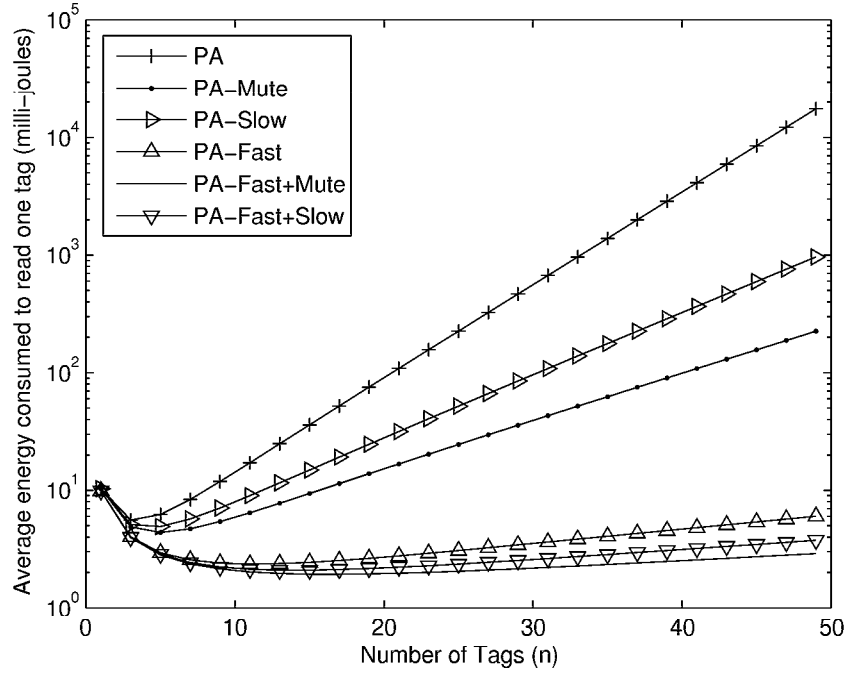
Pure with slow down has a lower energy consumption than conventional Pure Aloha. The reason being slow down reduces the offered load to the reader, thereby reducing collisions.

Pure Aloha with muting further reduces the energy consumption of conventional Pure Aloha as muting silences a tag after identification, thereby removing the tag from contributing to the offered load of the system. As a result, muting variants have a lower energy wastage from collisions compared to slow down variants. Further, saturation is largely reduced.

Finally, Pure Aloha variants which employ fast mode achieve the highest energy savings compared to other protocols. This is due to the low energy wastage from collisions since a reader's vulnerability period is reduced to  $T_f$ , which increases system throughput (see Figure 3.6) and lowers energy consumption.

Figure 3.2 plots the results for Slotted Aloha variants. Slotted Aloha reduces the vulnerability period to  $T$  [27]. Similar to Pure Aloha, conventional Slotted Aloha shows an exponential rise in energy consumption with increasing number of tags.

Slotted Aloha variants with early-end have significantly lower energy consumption compared to other variants, notably when the number of tags is low. This is because less energy are consumed by idle listening; see Section 3.4.3 for details. However, with increasing number of tags, the energy consumption of Slotted Aloha variants with early end gradually approaches variants without early-end support. This is because the probability of idle slots reduces with increasing

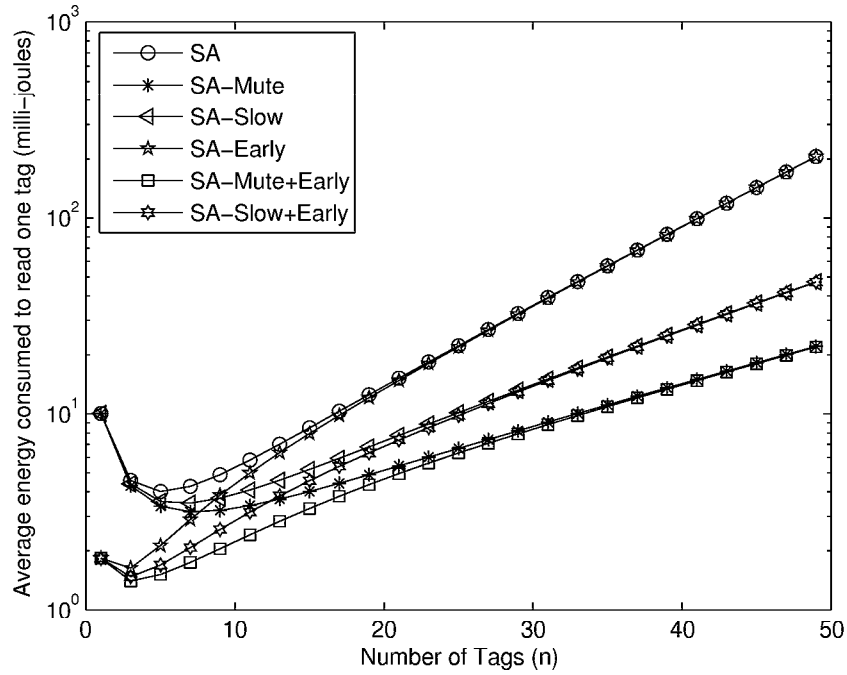


**Figure 3.1** Average energy consumed to read one tag successfully for Pure Aloha variants,  $\lambda = 20$ ,  $K = 5$ , PA = Pure Aloha,

tags, and hence diminishes the advantages provided by the early-end feature.

Let us now compare and contrast Pure and Slotted Aloha variants. The energy consumption of both Pure and Slotted Aloha variants, as depicted in Figure 3.1 and 3.2, is high when the number of tags is low, but reduces to a given point with increasing number of tags. After that, the energy consumed starts increasing as the number of tags grows.

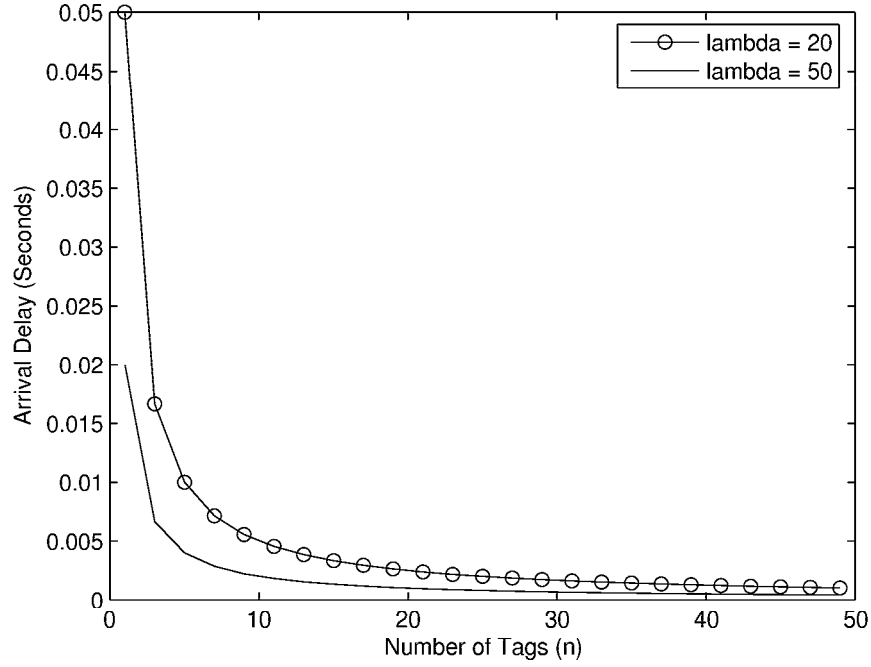
The above behavior can be explained by comparing the contention delay shown in Figure 3.4 with the arrival delay in Figure 3.3. The contention delay rises with increasing number of tags. On the other hand, arrival delay reduces with increasing number of tags. However, when the number of tags is low, the reader incurs a significantly higher arrival delay. Therefore, for low tag numbers, arrival delay constitutes a significant portion of the average delay incurred by a reader. On the other hand, when the number of tags is high, contention delay dominates.



**Figure 3.2** Average energy consumed to read one tag successfully for Slotted Aloha Variants,  $\lambda = 20$ ,  $K = 5$ , SA = Slotted Aloha.

Figures 3.5(a) and 3.5(b) compare Pure and Slotted Aloha protocols with  $\lambda$  values of 20 and 50. From the figures, Pure Aloha consumes the most energy because of its vulnerability period. On the other hand, fast mode variants of Pure Aloha conserve a significant amount of energy because of its low vulnerability period. Specifically, Pure Aloha with fast mode and muting has the lowest energy consumption as it further mitigates collisions using muting.

Lastly, Figure 3.6 compares the throughput characteristics of Pure and Slotted Aloha variants. It can be observed that the system throughput for Pure Aloha in fast mode is as high as Slotted Aloha variants. In addition, it can be observed that Pure Aloha protocols using the fast mode feature approach the maximum throughput when there is a large number of tags. Therefore, when protocols are operating at their maximum system efficiency, fast mode variants of Pure Aloha operate below their maximum system throughput. This means, given the same number of tags, Pure Aloha variants using fast mode experience more



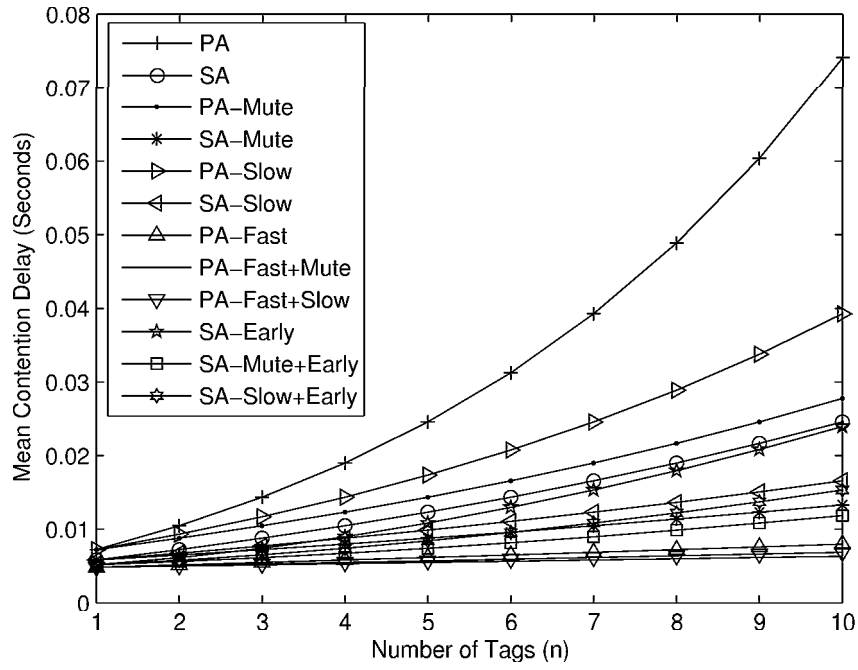
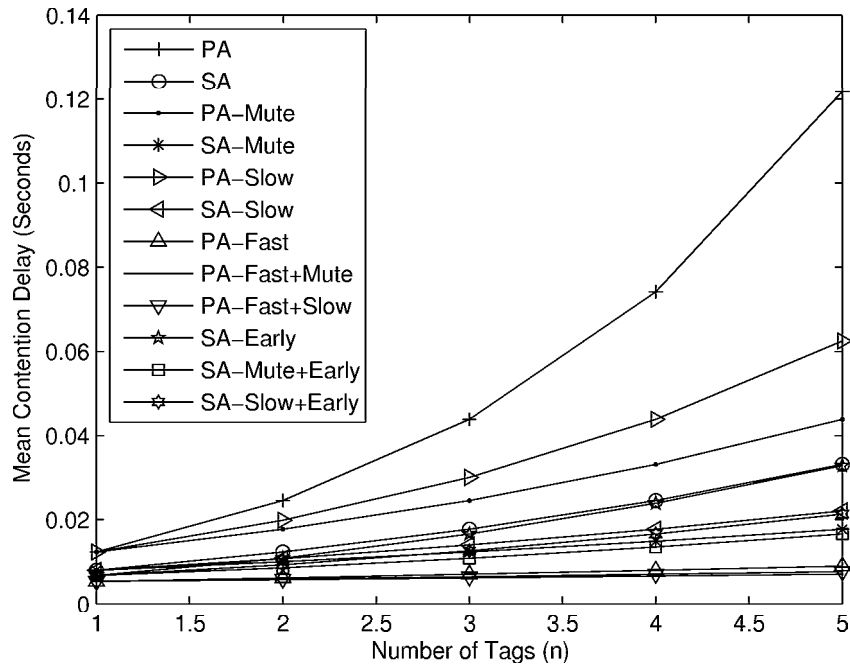
**Figure 3.3** Average Arrival Delay

free channel times compared to other protocols, hence tags' responses are less likely to collide.

### 3.4.2 Average Energy Wasted in Collisions to Read One Tag

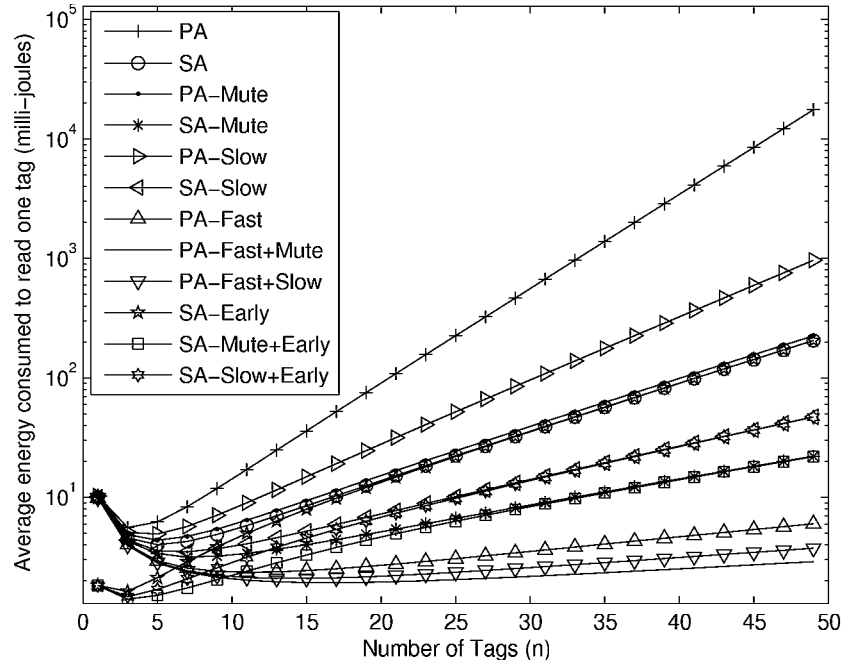
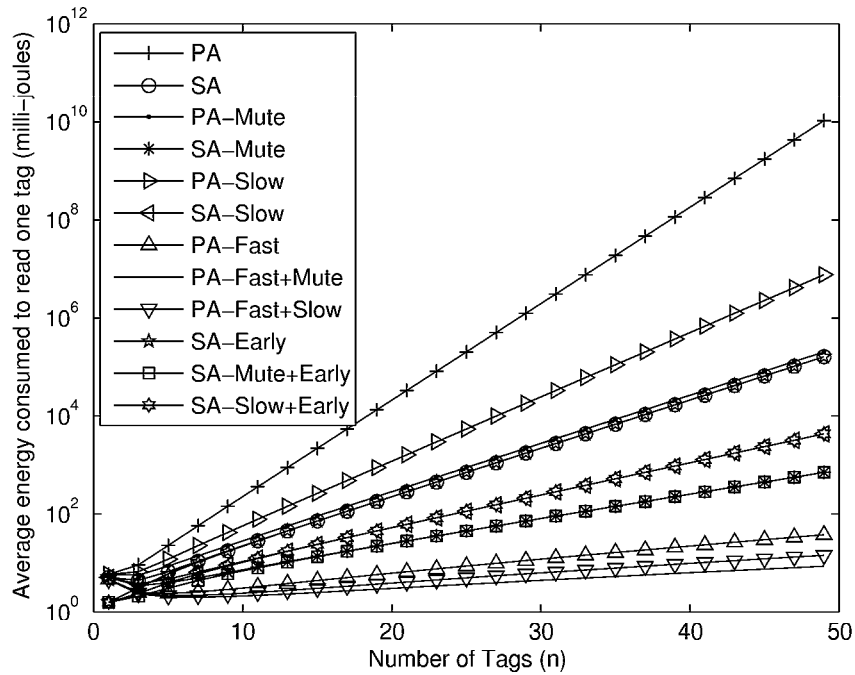
Figure 3.7(a) and 3.7(b) plot the average energy wasted due to collisions before a tag is read successfully. As can be seen, reducing  $\lambda$  significantly lowers energy consumption, especially when the number of tags is high. This is because, a lower value of  $\lambda$  reduces the offered load to the reader. However, when the number of tags is low, a higher energy consumption is observed for low  $\lambda$  values because of higher arrival delays.

Conventional Pure Aloha consumes the highest energy in collisions because of its high vulnerability period. However, Pure Aloha with fast mode and muting has the lowest energy wastage. This is because muting and fast mode both reduce collisions and therefore minimize energy consumption.

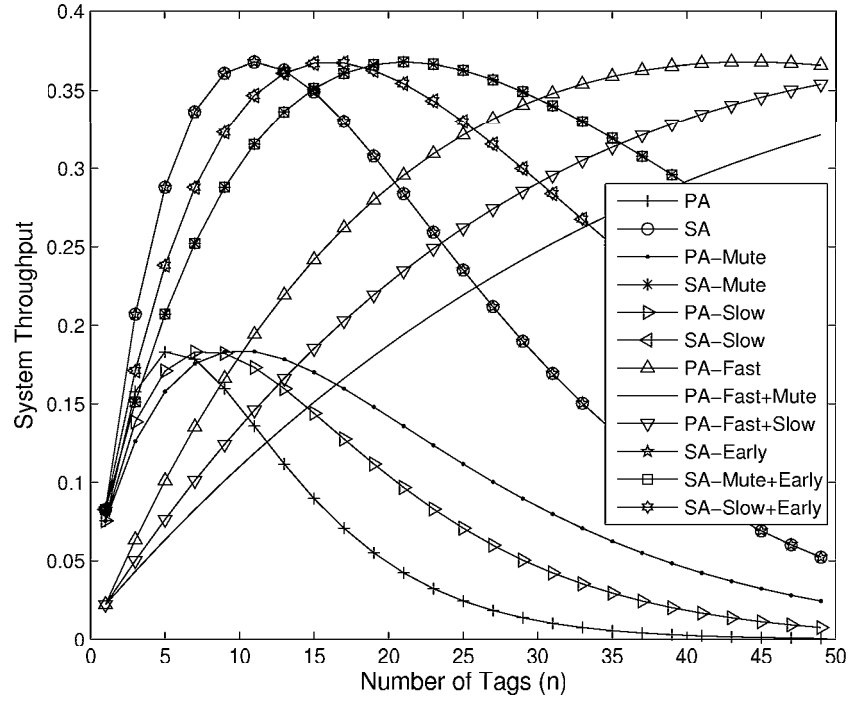
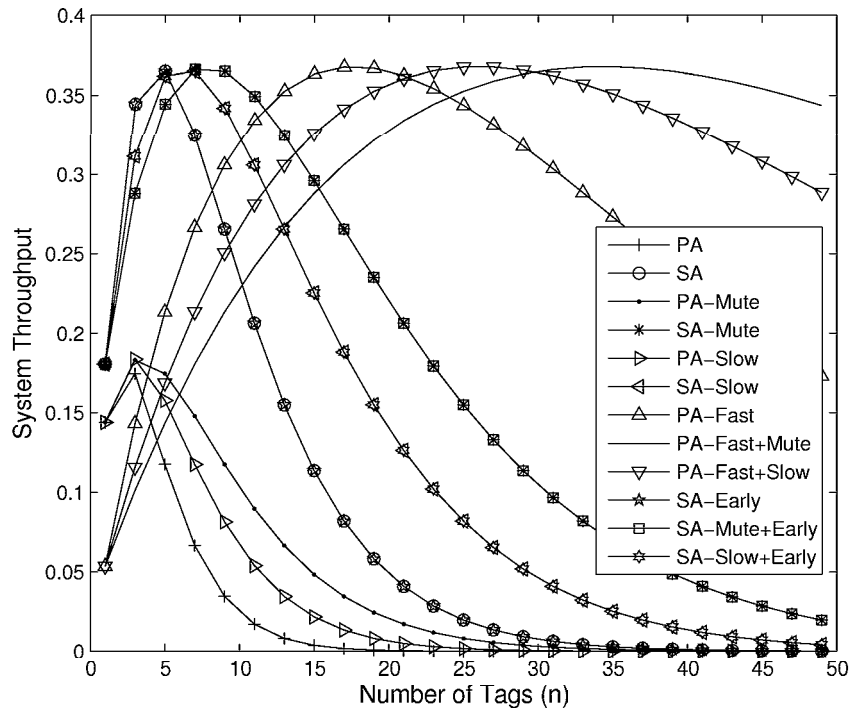
(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ 

**Figure 3.4** Mean Contention Delay for Aloha Variants, PA = Pure Aloha and SA = Slotted Aloha.



(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ 

**Figure 3.5** Comparing energy consumed to read one tag successfully for Pure and Slotted Aloha Variants, PA = Slotted Aloha and SA = Slotted Aloha

(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ 

**Figure 3.6** System Throughput Versus Number of Tags for Aloha Variants, PA = Pure Aloha and SA = Slotted Aloha.

For Slotted Aloha variants, it can be observed that early end has no effect on energy wastage from collisions. This is because early ending slots does not reduce collisions. On the other hand, the muting and slow down features do reduce collisions, hence tag reading protocols employing both features achieve better energy efficiency.

Overall, conventional Pure Aloha and Slotted Aloha have the highest energy wastage from collisions, but those with fast mode have the best energy efficiency.

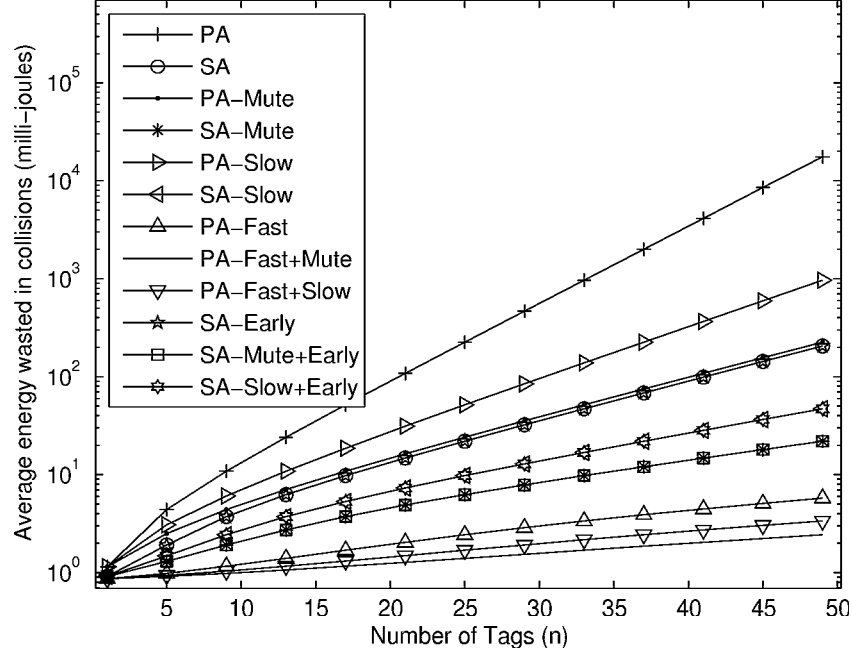
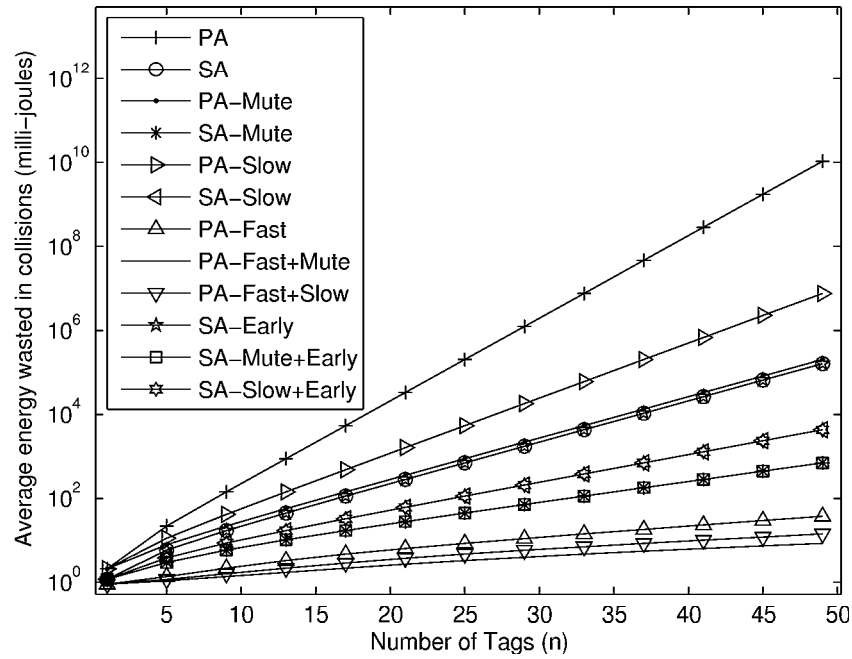
### 3.4.3 Average Energy Consumed due to Idle Listening

Figures 3.8(a) and 3.8(b) show the average energy consumed from idle listening. The early end feature reduces energy consumption in idle listening significantly due to the early closure of idle slots.

Among Pure Aloha variants, conventional Pure Aloha consumes the lowest energy in idle listening. This is because there are the lowest number of idle slots due to a significant load offered by competing tags. On the other hand, Pure Aloha with fast mode consumes the highest energy. It is because, using fast mode, the reader inhibits competing tags to stop transmitting once a successful transmission is detected. This results in significant reduction in vulnerability period as compared to Pure Aloha. Due to this, the offered load to the reader is significant reduced resulting in a larger delay due to idle listening during the collision resolution process.

Similar to fast mode, muting and slow down variants demonstrate an increase in the energy consumption in idle listening. This is because both these features lead to reduction in offered load to the reader, resulting in lowering collisions and increasing the idle time between successive receptions by the reader.

In summary, muting, slow down, and fast mode increase energy wastage from idle listening, with fast mode resulting in the highest wastage. This is because the said features reduce the offered load and increase the probability of having idle slots. Hence, the reader is able to save a significant amount of energy due to less collisions albeit with a comparatively small energy wastage from idle

(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ 

**Figure 3.7** Average energy wasted in collisions before a successful ID reception, PA = Pure Aloha and SA = Slotted Aloha.

listening.

### 3.4.4 Battery Lifetime

Figures 3.9(a) and 3.9(b) show the battery lifetime of each protocol. Recall that the lifetime of a battery is the number of tags it can read when the same set of tags is read repeatedly. Among Pure Aloha variants, those with fast mode have the highest lifetime. On the other hand, conventional Pure Aloha has the worst battery life. The main reason for the performance discrepancy is that the use of fast mode reduces collisions significantly, thereby increasing the “goodput” of the system.

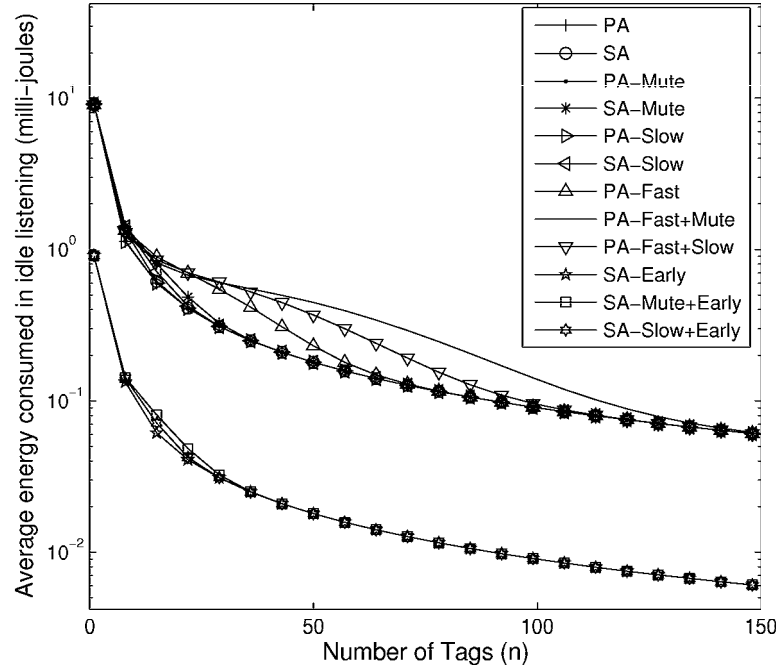
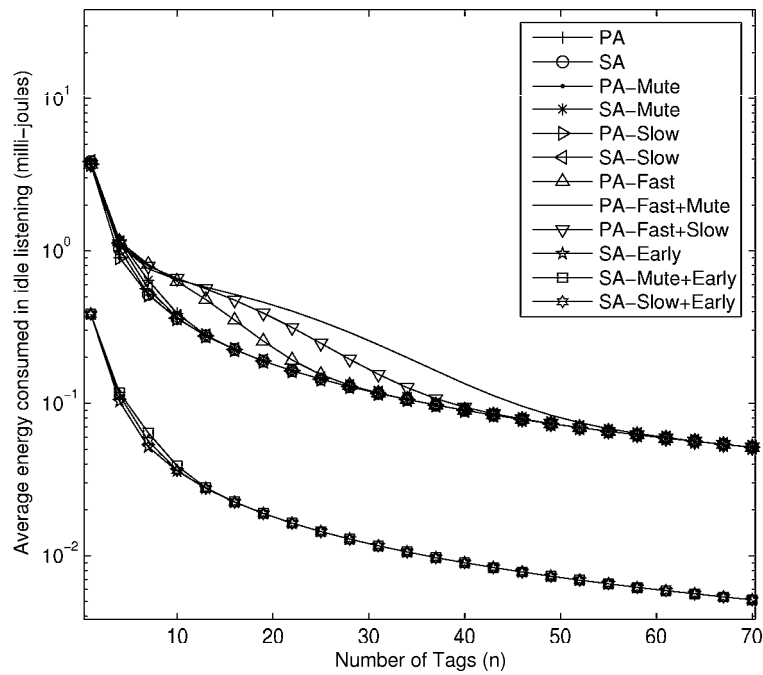
Among Slotted Aloha variants, those with muting and early end have the highest battery lifetime and conventional Slotted Aloha has the lowest lifetime.

Overall, Pure Aloha variants with the fast mode feature have long battery lifetime, in particular the variant incorporating both fast mode and muting. On the other hand, conventional Pure and Slotted Aloha have the least battery lifetime.

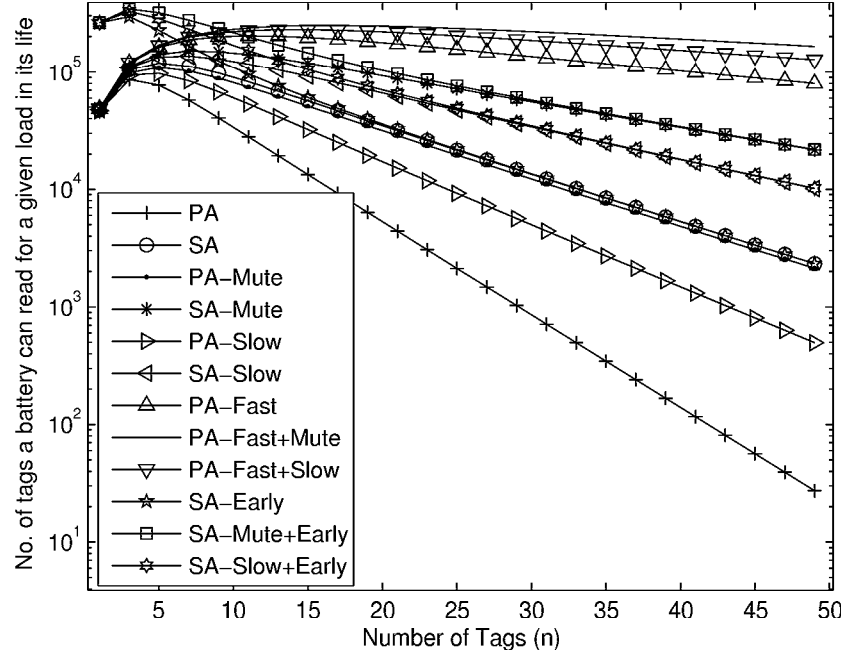
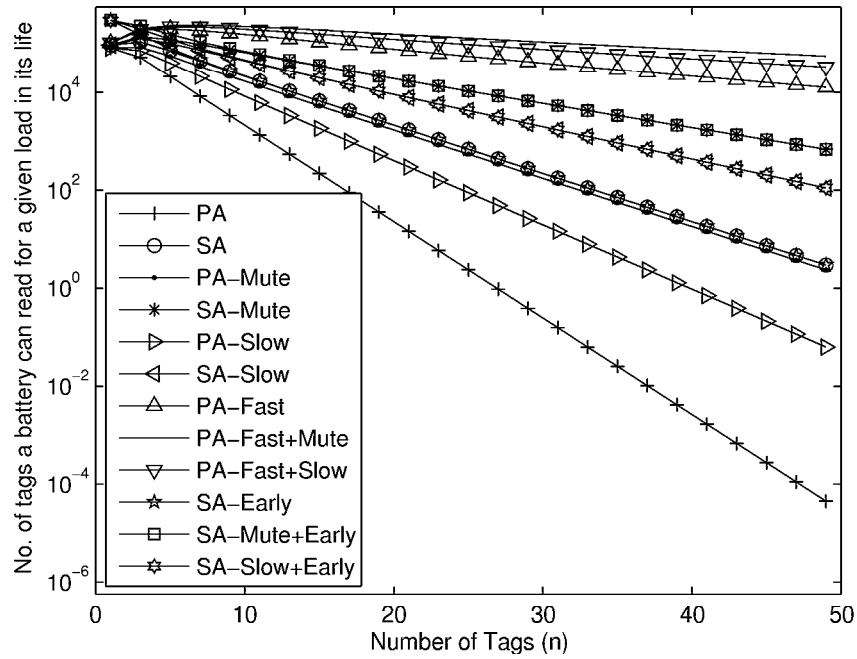
### 3.4.5 Battery Wastage

Figure 3.10(a) and 3.10(b) plot the energy wasted to read a tag successfully. Among Pure Aloha variants, those with fast mode and muting have the lowest battery wastage. For Slotted Aloha variants without early-end, Slotted Aloha with muting has the lowest energy wastage, and conventional Slotted Aloha has the highest wastage from idle listening. When early-end is used, the battery wastage for early-end variants of Slotted Aloha, Slotted Aloha with slow down, and Slotted Aloha with muting start to converge to variants without early-end. Among all Slotted Aloha variants, those incorporating muting and early-end have the lowest battery wastage.

Overall, muting, slow down, early-end and fast mode reduce battery wastage, with early-end having the highest impact when the number of tags is low. However, as the number of tags increases, those with fast mode have the lowest

(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ 

**Figure 3.8** Average energy spent in idle listening to read one tag, PA = Pure Aloha and SA = Slotted Aloha.

(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ **Figure 3.9** Battery Lifetime, PA = Pure Aloha and SA = Slotted Aloha.

wastage.

### 3.5 Related Works

To date, only EM Microelectronic [62] has analyzed and compared the reading delay of six Pure Aloha variants. This chapter extends upon theirs in the following manner. First, it provide a detailed comparison that includes Slotted Aloha variants. Second, the chapter analyze the energy consumption in every contention phase. Lastly, it provide a theoretical analysis of both Pure and Slotted Aloha based tag reading variants.

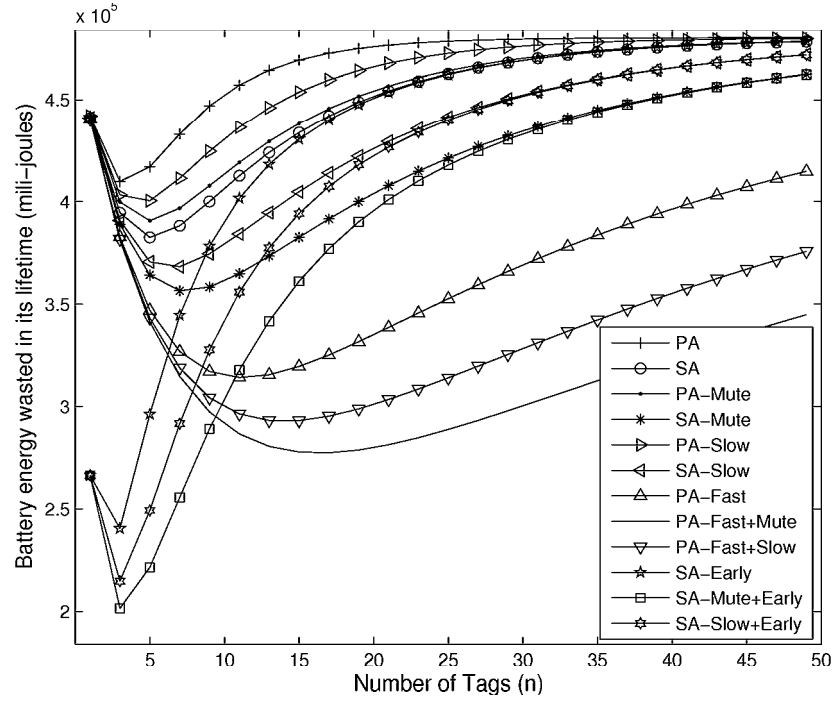
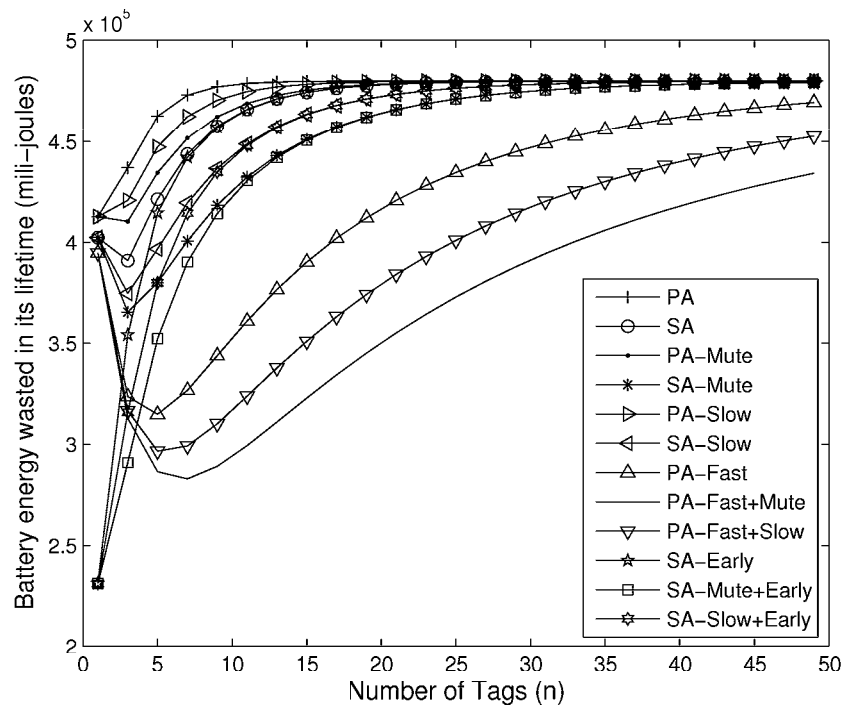
Apart from [62], none have considered Pure and Slotted Aloha based tag reading protocols, especially the variants analyzed in this paper. Namboodiri et al. [109] compared the energy efficiency of tree based protocols with their proposed framed Aloha based protocol. Similarly, Fernandes [110] compared the energy efficiency of tree search algorithms with Framed Aloha variants that use muting and varying frame sizes. The authors of [111] and [97] studied the energy consumed during collision resolution.

Many works, e.g., [112] [113] [114] [115] [27] [64] [116], have analyzed the performance of Pure and Slotted Aloha protocols in conventional wireless networks. These works however are not focused on energy efficiency. Moreover, they only investigate the delay incurred by competing nodes rather than the delay incurred by the destination/reader to receive/read responses. Lastly, these works do not consider the variants studied in this chapter.

### 3.6 Conclusions

This chapter analyzes the energy energy consumption of twelve Aloha variants and determines their suitability for use in RFID-enhanced WSNs. Based on the results, Pure Aloha with fast mode and muting consumes the lowest energy to identify a given number of tags. However, all the protocols studied are



(a)  $\lambda = 20, K = 5$ (b)  $\lambda = 50, K = 5$ 

**Figure 3.10** Battery wasted from collisions and idle listening, PA = Pure Aloha and SA = Slotted Aloha.

incapable of efficient monitoring, thereby making them unsuitable for use in RFID-enhanced WSNs. Lastly, the analytical model presented in this chapter has been verified by Rivera et al. [117].

## Tag Estimation Functions

The Dynamic Framed Slotted Aloha protocols to be presented in Chapter 5 rely on a tag estimation function for the best frame size to use for a given tag set. An inaccurate estimate results in a non-optimal frame size that causes high identification delays and energy consumption. This is particularly serious when DFSA based tag reading protocols are used in a RFID-enhanced WSN, where nodes are battery constrained. Therefore, there needs to be a study on the accuracy of tag estimation functions.

To date, very little works have conducted a comprehensive study on the accuracy of current tag estimation functions. Vogt [69] [70] proposes two tag estimation functions and analyzes their accuracy using weighted error and variance. On the other hand, Kodialam [74] et al. analyze the accuracy of their tag estimation functions using only the variance of tag estimates. Both works are limited to their respective tag estimation functions. Floerkemeier [72] [73] compares his tag estimation function using two approaches. In the first approach, he compares his tag estimation functions with the Slot-count (Q) algorithm [16] and Schoute's algorithm [79] by evaluating the difference between simulated and theoretical estimate. In the second approach, he compares one of Vogt's [69] function to Schoute's [79] tag estimation functions with his Bayesian approach using a test-bed comprising of a field programmable RFID reader and 64 HF Philips I Code RFID tags [73]. Floerkemeier, however, did not evaluate the

functions proposed by Zhen et al. [65], and Cha et al. [118] [66]. Zhen et al. [65], Cha et al. [118] [66] and Khandelwal et al. [71] analyze the identification delays of their proposed estimation functions instead of accuracy.

As we can see, amongst the aforementioned tag estimation functions, it is unclear which is the best or most accurate. Moreover, the aforementioned works have not used a consistent set of metrics in their studies. Given these limitations and the impact of tag estimation functions on the performance of DFSA, it is critical that a comprehensive study is carried out to identify the best function to use.

To this end, this chapter presents qualitative and quantitative analysis of five tag estimation functions using Monte Carlo simulations. A given tag set is estimated iteratively to determine a function's error distribution, which can then be used to evaluate the estimate's mean error, variability, skew and Kurtosis of the error distribution. Lastly, a comparison is carried out to identify the most efficient tag estimation function that is suitable for RFID-enhanced WSNs is presented.

The chapter is organized as follows. Section 4.1 describes importance of tag estimation functions in DFSA protocols. Following that, Section 4.2 presents a qualitative analysis of tag estimation functions is presented. Next, Section 4.3 and 4.4 present the system model and research methodology used to evaluate tag estimation functions respectively. Section 4.5 presents results and Section 4.6 concludes the chapter.

## 4.1 Motivation

In Frame Slotted ALOHA (FSA) based protocols, a tag selects a slot randomly, and waits for a random time period before transmitting. Unlike conventional Pure and Slotted Aloha, a tag is only permitted to transmit its ID at most once per frame. FSA protocols have a fixed or varying frame size. The former is referred as basic FSA (BFSA) and the later as dynamic (DFSA) respectively.

FSA based protocols are often measured according to their system efficiency;

defined as the ratio of the number of slots filled with one tag with respect to the current frame size. The system efficiency,  $a_1^{N,n}$ , of BFSA protocols is given as [77] [79],

$$a_1^{N,n} = \frac{n}{N} \left(1 - \frac{1}{N}\right)^{n-1} \quad (4.1)$$

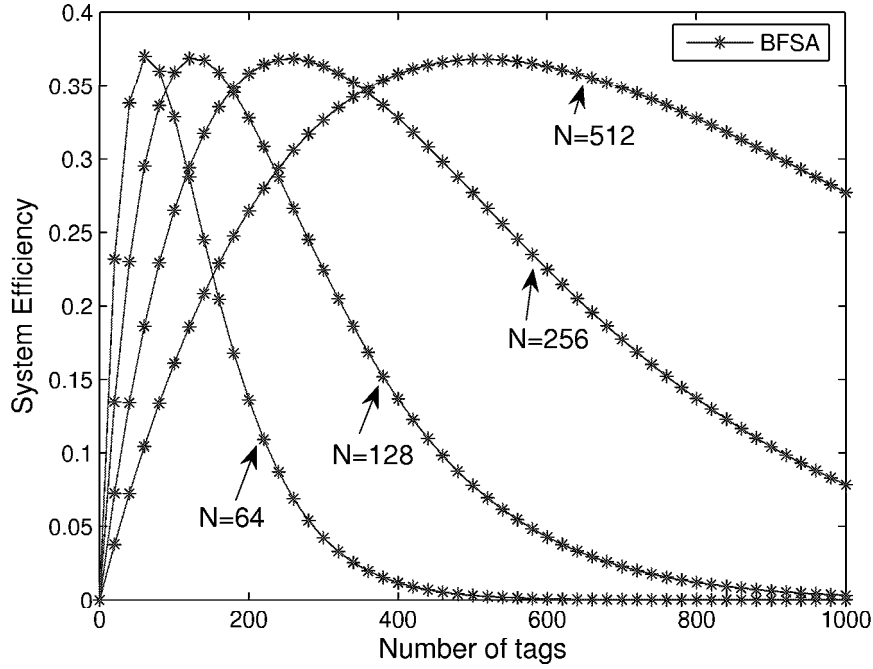
where  $n$  and  $N$  denote the number of tags and frame size respectively. The optimal throughput for BFSA can be evaluated from Equ. 4.1 and is equal to  $e^{-1}$  for  $N = n$  [73].

Figure 4.1 plots Equ. 4.1 with varying number of tags and frame sizes;  $N=64$ , 128, 256 and 512. From Figure 4.1, it is clear that system efficiency starts to fall as the number of tags increases beyond the number of slots within a frame, i.e.,  $N > n$ , and drops nearly to zero as the number of tags becomes very large. The system efficiency is at its highest when the number of tags is equal to the number of slots in a frame [77]. Therefore, if the frame size can be adjusted depending upon tag population, a reader will operate at the highest system efficiency.

DFSA protocols have the ability to adjust their frame size according to tag estimates calculated by a tag estimation function. In order to achieve minimal delay and high system efficiency, DFSA protocols must adjust their frame size to match the number of tags [79] [66]. Thus, DFSA protocols' performance largely depend upon the tag estimation function used. Therefore, it is important that to conduct a comprehensive study on the accuracy of tag estimation functions.

## 4.2 Tag Estimation Functions

A tag estimation function calculate the number of tags based on feedback obtained in reader's frame when a reader transmits a frame of size  $N$  to tags. Here, the feedback refers to slots with no response ( $c_0$ ), one tag reply ( $c_1$ ), and multiple tag replies ( $c_k$ ). Based on this feedback, researchers have developed



**Figure 4.1** System efficiency versus number of tags.

several estimation functions.

#### 4.2.1 Vogt Functions

Vogt [69] [70] present two tag estimation functions; referred to as DFSAV-I and DFSAV-II. In both functions, Vogt assume tags reply in each read round and omitted the muting feature, where tags are stopped from replying after they are identified.

DFSAV-I is given by Equ. 4.2, and estimated tags are denoted as  $\varepsilon_{lb}$ . The principle behind Equ. 4.2 is that during collisions, there will be at least two tags involved, which defines the lower bound of a tag estimate. The main downside of DFSAV-I is that it becomes erroneous as the number of tags increases. This is because the term  $2c_k$  in Equ. 4.2 assumes two tags are involved in a collision. However, when the number of tags increases, more than two tags may collide

within a single slot.

$$\varepsilon_{lb}(N, c_0, c_1, c_k) = c_1 + 2c_k \quad (4.2)$$

DFSAV-II takes a different approach and is based on Chebyshev's inequality. The algorithm determines the distance between an actual read result vector  $\langle c_0, c_1, c_k \rangle$  and the theoretical expected result vector  $\langle a_0^{N,n}, a_1^{N,n}, a_k^{N,n} \rangle$  of a reading cycle, see Equ 4.3. The value of  $t$  for which the  $\varepsilon_{vd}$  becomes minimum is the estimated tags.

$$\varepsilon_{vd}(N, c_0, c_1, c_k) = \min_t \left\| \begin{pmatrix} a_0^{N,t} \\ a_1^{N,t} \\ a_k^{N,t} \end{pmatrix} - \begin{pmatrix} c_0 \\ c_1 \\ c_k \end{pmatrix} \right\| \quad (4.3)$$

In Equ. 4.3, the elements of the vector  $\langle a_0^{N,t}, a_1^{N,t}, a_k^{N,t} \rangle$  correspond to the expected number of empty slots, slots filled with one tag, and slots with collisions, respectively. With a frame size of  $N$ , and the number of tags  $t$ , the expected number of slots filled with  $r$  responding tags is given by,

$$a_r^{N,t} = N \times \binom{t}{r} \left( \frac{1}{N} \right)^r \left( 1 - \frac{1}{N} \right)^{t-r} \quad (4.4)$$

From Equ. 4.4, each element of the vector  $\langle a_0^{N,t}, a_1^{N,t}, a_k^{N,t} \rangle$  is calculated as follows [70]:

$$a_0^{N,t} = N \times \left( 1 - \frac{1}{N} \right)^t \quad (4.5)$$

$$a_1^{N,t} = t \times \left( 1 - \frac{1}{N} \right)^{t-1} \quad (4.6)$$

$$a_k^{N,t} = N - a_0^{N,t} - a_1^{N,t} \quad (4.7)$$

According to [69], DFSAV-II is more applicable to scenarios where tag densities are high. However, DFSAV-II is computationally more complex than DFSAV-I.

#### 4.2.2 Cha et al. Functions

Cha et al. [118] [66] present two tag estimation functions; referred to as DFSAC-I and DFSAC-II.

DFSAC-I computes a tag estimate from the collision rate that is derived using the "maximum throughput condition". The collision rate and maximum throughput condition are defined as follows.

According to Cha et al., the throughput  $S$  of FSA is defined as,

$$S = \frac{P_{succ}}{P_{succ} + P_{coll} + P_{idle}} \quad (4.8)$$

where  $P_{coll} = 1 - P_{idle} - P_{succ}$  and  $P_{idle} = (1 - p)^n$  and  $P_{succ} = np(1 - p)^{n-1}$  respectively;  $p = \frac{1}{N}$  is the probability that one tag transmits at a particular slot in a frame. The maximum throughput happens when,

$$\frac{dS}{dp} = 0 \quad (4.9)$$

Solving Equ. 4.9 gives,

$$p = \frac{1}{n} \quad (4.10)$$

Using Equ. 4.10, the collision rate is calculated as [66],

$$C_{rate} = \lim_{n \rightarrow \infty} \frac{P_{coll}}{1 - P_{succ}} \quad (4.11)$$

Inserting  $P_{coll}$ ,  $P_{succ}$  and Equ. 4.10 into Equ. 4.11, we get  $C_{rate} = 0.4180$ , which Cha et al. refer to as the "maximum throughput condition". Using  $C_{rate}$ , the



number of colliding tags in a slot is then calculated as,

$$C_{tags} = \frac{1}{C_{rate}} = 2.3992 \quad (4.12)$$

From Equ. 4.12, if the number of colliding slots is  $c_k$ , the number of tags is estimated as  $2.3992c_k$

DFSAC-II, however, takes a different approach. It estimates a tag set based on a collision ratio. Cha et al. define collision ratio to be the ratio of the number of slots with collisions to the frame size, and is computed as [66],

$$C_{ratio} = 1 - \left(1 - \frac{1}{N}\right)^n \left(1 + \frac{n}{N-1}\right) \quad (4.13)$$

where  $n$  is the estimated number of tags, and is computed after a read round from the number of colliding slots as,

$$C_{ratio} = \frac{c_k}{N} \quad (4.14)$$

Using Equ. 4.14, Cha et al. solves for  $n$  iteratively by equating Equ. 4.13 to Equ. 4.14.

DFSAC-II is computationally more complex than DFSAC-I. Note, both these functions are developed for scenarios where tags are muted once identified.

### 4.2.3 Zhen et al. Functions

Zhen et al [65]'s function, referred to as DFSAC, works as follows. For an observed slot, the *a posteriori* probability distribution of  $k$  tags choosing a slot is,

$$p_k^0(i) = \begin{cases} 0 & \text{if } k = 0, 1 \\ \frac{p_k(i)}{1-p_0(i)-p_1(i)} & \text{if } k \geq 2 \end{cases} \quad (4.15)$$

From Equ. 4.15, the *a posteriori* expected value of a garbled slot is,  $\lim_{N \rightarrow \infty} \sum_{k=2}^N k p_k^0(i) =$

$2.39 = K$ . This indicates, that on average 2.39 tags respond in a collided slot. Thus, according to Zhen et al. [65], the estimated tags will be  $c_1 + 2.39c_k$ .

The estimation function is devised for passive tag environments and muting of tags is not taken into consideration.

### 4.3 System Model

The system consists of an RFID reader and  $n$  tags in its interrogation zone. Tags are assumed to be passive and used in read-only mode. Further, tags are static and can be read regardless of their orientation<sup>1</sup>.

A reader starts collision resolution with an arbitrary frame size  $N$  and observes tag responses in each slot. These responses are refereed as slots with no response ( $c_0$ ), slots with one tag reply ( $c_1$ ), and slots with multiple tag replies ( $c_k$ ).

### 4.4 Research Methodology

Descriptive statistics are used to quantify the error distribution of tag estimation functions. For each corresponding error distribution, its mean error, standard deviation, variance, skewness and Kurtosis is evaluated [119].

The error in the  $u^{th}$  read round can be evaluated as,

$$e(u) = n(u) - n' \quad (4.16)$$

where  $e(u)$  is the error estimated in the  $u^{th}$  read round,  $n(u)$  is the number of tags estimated using a tag estimation function in  $u^{th}$  the read round, and  $n'$  is the actual number of tags in a reader's interrogation zone, respectively. If  $e(u)$  is negative, it indicates the estimated number of tags is less than the actual tag count, and vice versa.

<sup>1</sup>The placement of tags with respect to the polarization of the reader's field can have a negative affect on the communication distance of RFID tags, in particular, reduced operating range and in the case of the tag being displaced by 90 degrees, a tag becomes unreadable.

In Equ. 4.16,  $e(u)$  and  $n(u)$  are random variables and  $n'$  is a constant.  $e(u)$  and  $n(u)$  are evaluated at each read cycle and  $e(u)$  is dependent upon the estimated number of tags  $n(u)$  in a particular read round.

$R$  read cycles are performed on a tag set with  $n'$  tags and then calculate the mean of the distribution  $e(u)$  as,

$$\mu(n') = \frac{1}{R} \sum_{u=1}^R (e(u)) \quad (4.17)$$

$$= \frac{1}{R} \sum_{u=1}^R (n(u) - n') \quad (4.18)$$

where  $\mu(n')$  denotes the mean of the error computed in  $R$  read rounds.

The mean percentage of error of each tag estimate is evaluated as,

$$\mu_{per}(n') = \frac{1}{R} \sum_{u=1}^R \frac{e(u)}{n'} \times 100 \quad (4.19)$$

In order to evaluate the variability or spread of the distribution  $e(u)$ , its standard deviation is evaluated as,

$$s = \sqrt{\frac{\sum_{u=1}^R (e(u) - \mu)^2}{R - 1}} \quad (4.20)$$

Squaring Equ. 4.20 gives us the variance  $\sigma(n')$ .

The skewness, i.e., lack of symmetry of  $e(u)$ , can be evaluated as

$$sk(n') = \frac{3(\mu(n') - m(n'))}{s(n')} \quad (4.21)$$

where  $m(n')$  is the median of  $e(u)$ .

Finally, the flatness or peakness of a distribution is measured, i.e., Kurtosis of  $e(u)$ , as,

$$kurt = \frac{\sum_{u=1}^R (e(u) - \mu(n'))^4}{Rs^4} \quad (4.22)$$

If the distribution is relatively flat compared to the normal or bell-shaped distribution,  $e(u)$  is platykurtic. Otherwise, it is called leptokurtic.

```

N=32;
t=100;
R=10000;
for n = 1 : t do
    for u = 1 : R do
        c = perform_read_cycle(N,c);
        n(u)=estimate_tags(N,c);
        e(u)=n(u)-n';
    end
     $\mu(n') = \text{mean}(e(u), R);$ 
     $\mu\_per(n') = \text{percent\_error}(e(u), n');$ 
     $s(n') = \text{standard\_deviation}(e(u), R);$ 
     $\sigma(n') = (s(n'))^2;$ 
     $sk(n') = \text{skewness}(e(u));$ 
     $kurt(n') = \text{kurtosis}(e(u));$ 
end

```

**Algorithm 1:** Procedure for evaluating tag estimation functions.

Algorithm 1 shows the pseudo-code of our implementation. The function  $\text{perform\_read\_cycle}(N, n)$  returns the vector  $\langle c_0, c_1, c_k \rangle$  after being given the following parameters; frame size and actual number of tags in the current read cycle. The result is then used by  $\text{estimate\_tags}(N, c)$ , which returns the estimated tags obtained by the tag estimation function being investigated. The rest of the pseudo-code implement equations 4.16, 4.18, 4.20, 4.21, and 4.22.

## 4.5 Results

This section starts by presenting results from a statistical analysis of each tag estimation function using a frame size  $N = 32$ , and a low tag density, i.e.,  $n' = 32$  and a high tag density i.e.,  $n' = 100$ . After that, results on the number of read cycles or frequency versus estimate error are presented. Lastly, the tag estimation functions are compared according to their mean error, variance, skew and Kurtosis.

## 4.5.1 Statistical Analysis

### 4.5.1.1 DFSAV-I

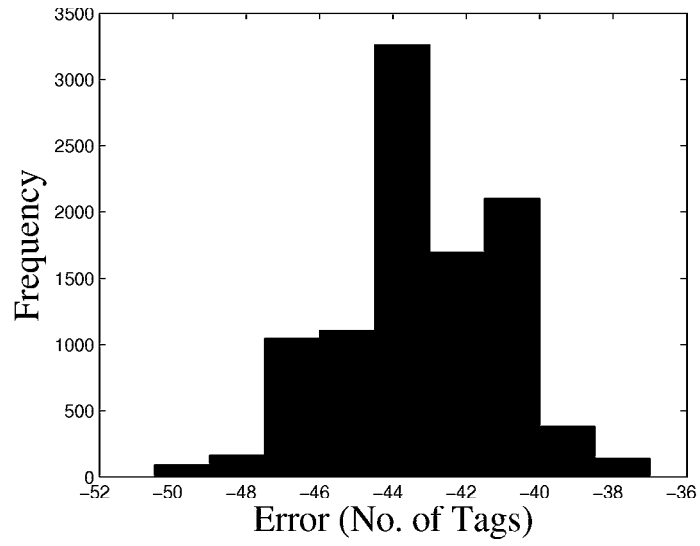
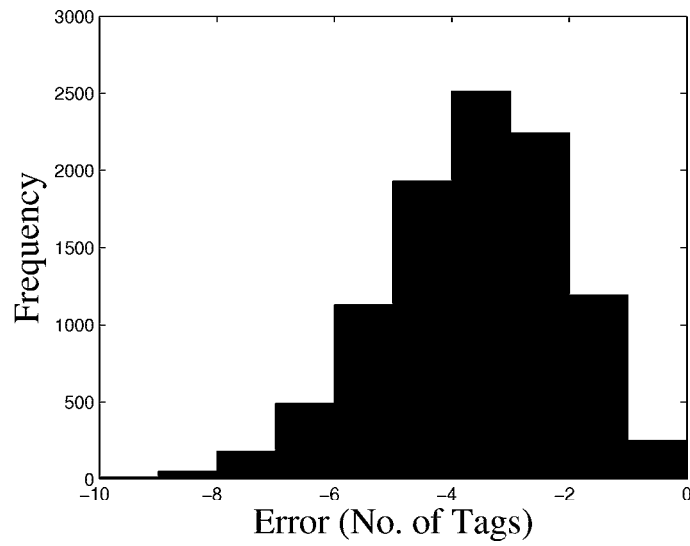
Figures 4.2(a) and 4.2(b) plot the error distribution of Vogt [69] [70]'s first function, i.e., DFSAV-I. In Figure 4.2(b), where  $n'$  matches the frame size, the mean error is lower compared to Figure 4.2(a);  $n' = 100$ . This means DFSAV-I performs better when  $n'$  is comparable to  $N$ . In both figures, the mean error is negative, indicating that the number of tags estimated is less than the actual number of tags. The variability of the error distribution reduces as the number of tags becomes comparable to the frame size. This indicates that tag estimates are becoming more stable when  $n'$  is comparable to the frame size used. Lastly, the error distribution in Figure 4.2(a) and 4.2(b) is skewed positively and negatively. Both distributions are leptokurtic.

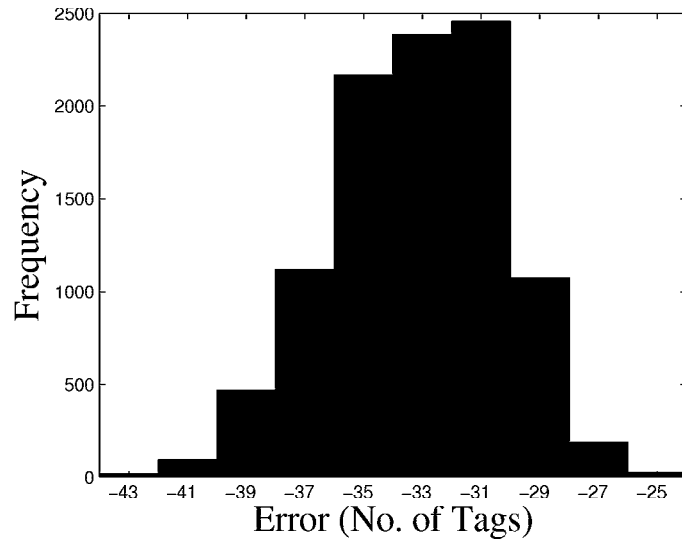
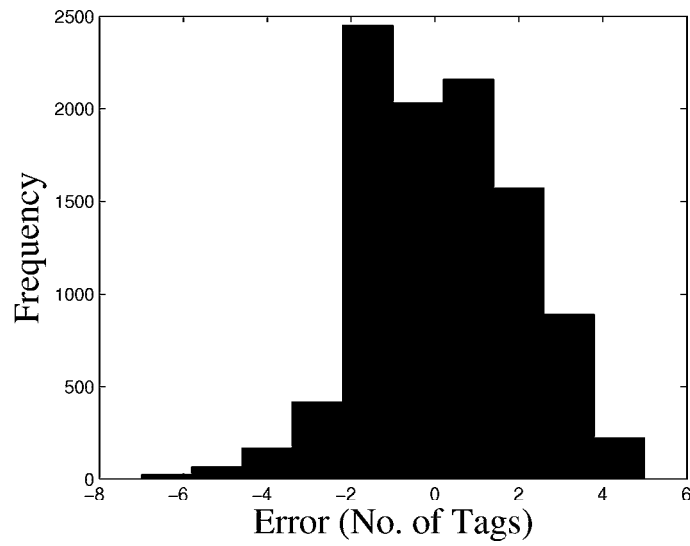
### 4.5.1.2 DFSAZ

Figures 4.3(a) and 4.3(b) plot the error distribution for Zhen et al. [65]'s functions, i.e., DFSAZ. In Figure 4.3(a), the percentage of error is  $-32.65\%$  which is very high compared to Figure 4.3(b), where the percent error is only  $0.8884\%$ . Therefore, DFSAZ is more accurate when the number of tags matches the frame size used. However, when the number of tags exceeds the frame size, the computed estimate is highly erroneous. From Figure 4.3(a), the mean error is negative, indicating that most tag estimates are less than the actual number of tags in a reader's interrogation zone. On the other hand, from Figure 4.3(b), the mean error is  $0.2853$ , indicating that most tag estimates are higher than the actual tag count. The variability of the error distribution in Figure 4.3(b) is less than Figure 4.3(a), indicating stable estimates for  $n' = 32$ . The error distribution in both figures is positively skewed and is leptokurtic.

### 4.5.1.3 DFSAC-I

Figures 4.4(a) and 4.4(b) plot the error distributions for Cha et.al [66]'s function, i.e., DFSAC-I. For both figures, the mean error percentage is approximately  $37\%$ . In other words, for DFSAC-I, the error percentage does not vary much

(a)  $N = 32, n' = 100$ (b)  $N = 32, n' = 32$ **Figure 4.2** DFSAV-I error analysis.

(a)  $N = 32, n' = 100$ (b)  $N = 32, n' = 32$ **Figure 4.3** DFSAZ error analysis.

with the number of tags, even when the number of tags is comparable to the frame size, the error estimate remains large. For both figures, the mean estimate is negative, indicating that the tag estimate is less than the actual number of tags. The variability of the error distribution however, is lower in Figure 4.4(b) compared to Figure 4.4(a), indicating that smaller frame sizes have stable error estimates. Both curves are positively skewed, where skewness for Figure 4.4(b) is higher than Figure 4.4(a). Both distributions are leptokurtic.

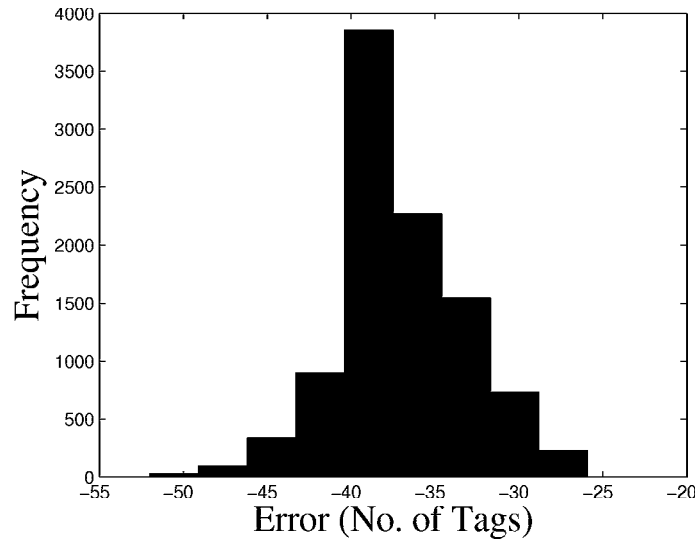
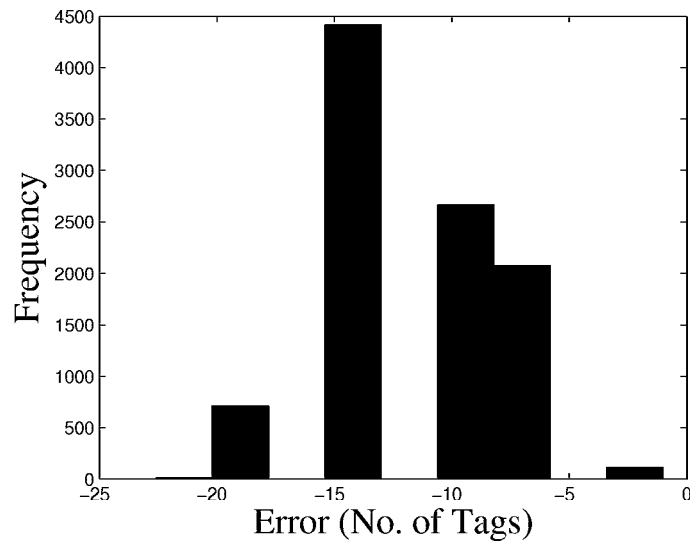
#### 4.5.1.4 DFSAC-II

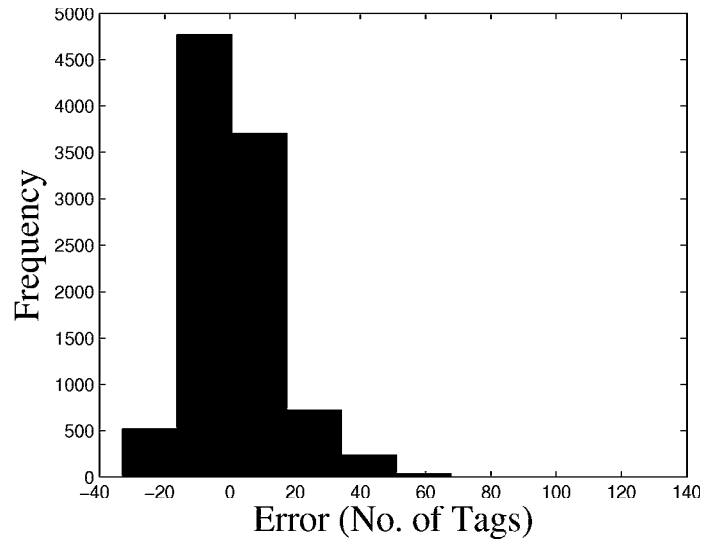
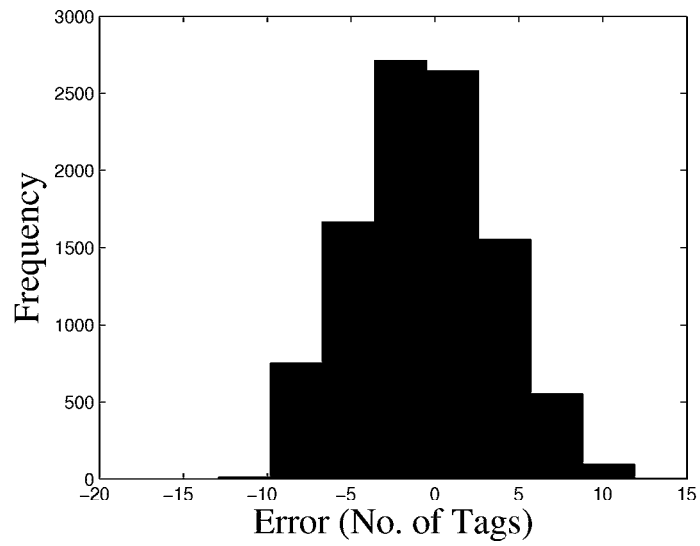
Figures 4.5(a) and 4.5(b) plot the error distribution for DFSAC-II. The error percentage for Figure 4.5(a) and 4.5(b) is 1.0827% and  $-1.4341\%$  respectively. Clearly for DFSAC-II the mean error percentage is higher when the number of tags is comparable to the frame size. However, when the number of tags is larger than the frame size, the average error percentage is very low. On the other hand, the variability in the distribution when tags are comparable to the frame size is less than the scenario where tags are significantly larger than the frame size used, meaning the estimate is stable if the  $n'$  is comparable to  $N$ . Both figures are positively skewed and are leptokurtic.

#### 4.5.1.5 DFSAV-II

Figures 4.6(a) and 4.6(b) depict the error distributions for DFSAV-II, another of Vogt [69]'s function. The mean percentage error for Figure 4.6(a) is 1.7% and for Figure 4.6(b) is  $-0.2131\%$ . This indicates that the number of estimated tags has a higher accuracy for both cases. It can be noted that when the number of tags is comparable to the frame size, the mean error is negative, indicating that the number of estimated tags is lower than the actual number of tags. However, when the number of tags is higher than the frame size, as in Figure 4.6(a), the mean error is positive, indicating that most estimates are higher than the actual tag count. The main distinction is evident from variability in the error distribution. The variance for Figure 4.6(a) is as high as 213, but for Figure 4.6(b), its only 4.8. This indicates that the error estimates for Figure 4.6(a) are very unstable. Figure 4.6(a) is positively skewed and Figure 4.6(b)



(a)  $N = 32, n' = 100$ (b)  $N = 32, n' = 32$ **Figure 4.4** DFSAC-I error analysis.

(a)  $N = 32, n' = 100$ (b)  $N = 32, n' = 32$ **Figure 4.5** DFSAC-II error analysis.

is negatively skewed. Both curves are leptokurtic.

#### 4.5.1.6 Summary

Table 4.1 gives a summary of the aforementioned statistical analysis. From Table 4.1, when  $n' = 32$ , DFSAZ has the lowest percentage of error. On the other hand, when  $n' = 100$ , DFSAC-II has the lowest percent error. DFSAV-I has the lowest variance for  $n' = 100$  as well as  $n' = 32$ . However, DFSAC-II and DFSAV-II have very high variability in their error distribution. The Kurtosis for DFSAV-II is significantly higher than other functions for  $n' = 100$ . DFSAV-I is the only entry in Table 4.1 which is negatively skewed for  $n' = 32$ . All others are positively skewed and leptokurtic.

### 4.5.2 Comparisons

In this subsection, a detailed comparison analysis of tag estimation functions based on mean error, variability, skew and Kurtosis is presented.

#### 4.5.2.1 Mean error

Figure 4.7 depicts the mean error in tag estimates. It can be seen that DFSAC-I has the highest mean error in estimates until the number of tags is less than 68. After that, DFSAV-I becomes erroneous. DFSAC-II and DFSAV-II have the lowest mean error for a wide range of tag set. DFSAC-II, however, has a lower accuracy than DFSAV-I when the number of tags is lower than 20. It is because, DFSAC-II considers collision slots only when estimating tags. When the number of tags is less than the frame size used, there are fewer collisions because there are more slots with a single reply. Thereby, the mean error for DFSAC-II is higher when the number of tags is lower than the frame size used. DFSAZ has a lower mean error than DFSAC-I and DFSAV, and similar error values to DFSAC-II and DFSAV-II for  $n' < 40$ , and becomes more erroneous when  $n' > 40$ . It can be observed that DFSAC-II outperforms DFSAV-II in accuracy when the numbers of tags starts to increase beyond the frame size. Overall, DFSAC-I, DFSAZ, DFSAV-I have better performance when the number of tags

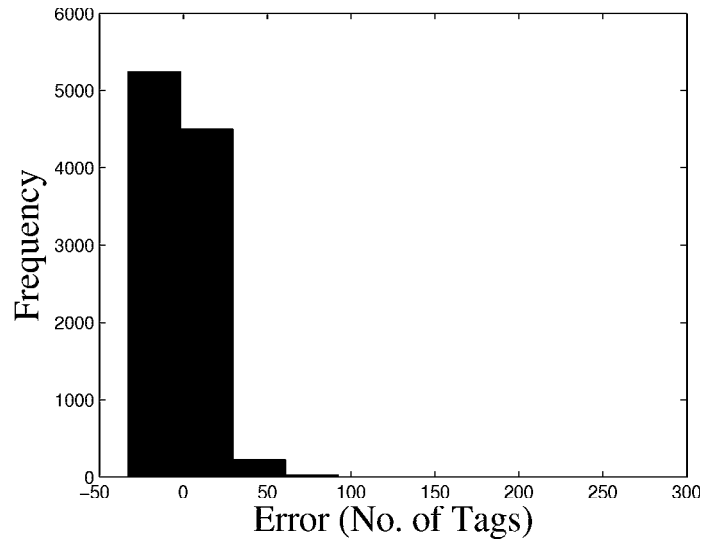
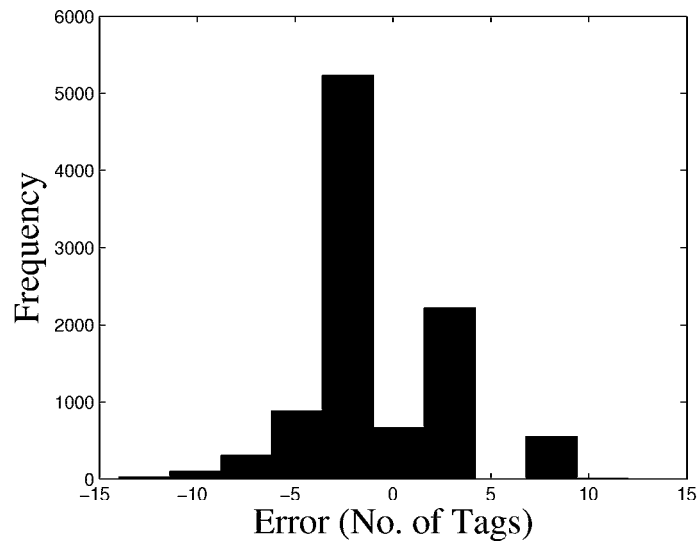
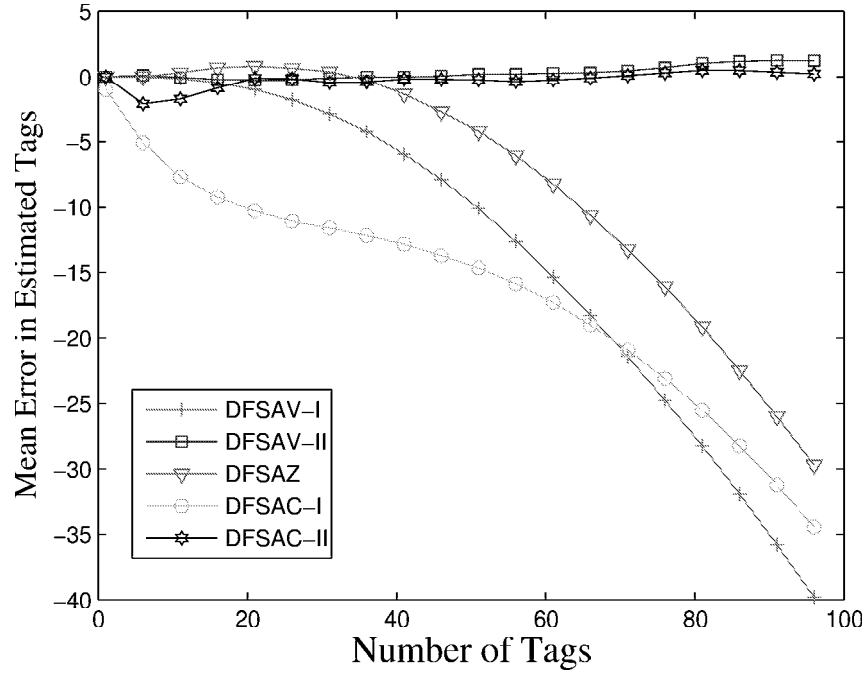
(a)  $N = 32, n' = 100$ (b)  $N = 32, n' = 32$ **Figure 4.6** DFSAV-II error analysis.

Table 4.1 Summary of statistics for error distribution plots

| Function                      | Mean Estimate | Mean error | Median | Percent error | Standard Deviation | Variance | Skew    | Kurtosis |
|-------------------------------|---------------|------------|--------|---------------|--------------------|----------|---------|----------|
| $R = 10000, n' = 100, N = 32$ |               |            |        |               |                    |          |         |          |
| DFSAV-I                       | 57.0392       | -42.9608   | -43    | -42.9608      | 2.2547             | 5.0838   | 0.0522  | 2.9431   |
| DFSAV-II                      | 101.7528      | 1.7528     | -2     | 1.7528        | 14.6118            | 213.5056 | 0.7705  | 26.0821  |
| DFSAZ                         | 67.3445       | -32.6555   | -33    | -32.6555      | 2.8684             | 8.2274   | 0.3603  | 2.9028   |
| DFSAC-I                       | 63.0545       | -36.9455   | -38    | -36.9455      | 4.0686             | 16.5534  | 0.7775  | 2.8955   |
| DFSAC-II                      | 101.0827      | 1.0827     | -4     | 1.0827        | 14.0181            | 196.5071 | 1.0396  | 8.0436   |
| $R = 10000, n' = 32, N = 32$  |               |            |        |               |                    |          |         |          |
| DFSAV-I                       | 28.8968       | -3.1032    | -3     | -9.6975       | 1.5908             | 2.5308   | -0.1946 | 3.0082   |
| DFSAV-II                      | 31.9318       | -0.0682    | 0      | -0.2131       | 2.2055             | 4.8644   | -0.0928 | 3.7691   |
| DFSAZ                         | 32.2843       | 0.2843     | 0      | 0.8884        | 1.8487             | 3.4178   | 0.4613  | 3.0047   |
| DFSAC-I                       | 20.3384       | -11.6616   | -13    | -36.4425      | 3.3170             | 11.0024  | 1.2105  | 2.8183   |
| DFSAC-II                      | 31.5411       | -0.4589    | -2     | -1.4341       | 3.6836             | 13.5687  | 1.2551  | 2.8448   |

is similar to the frame size used. DFSAC-I has the lowest accuracy compared to others.



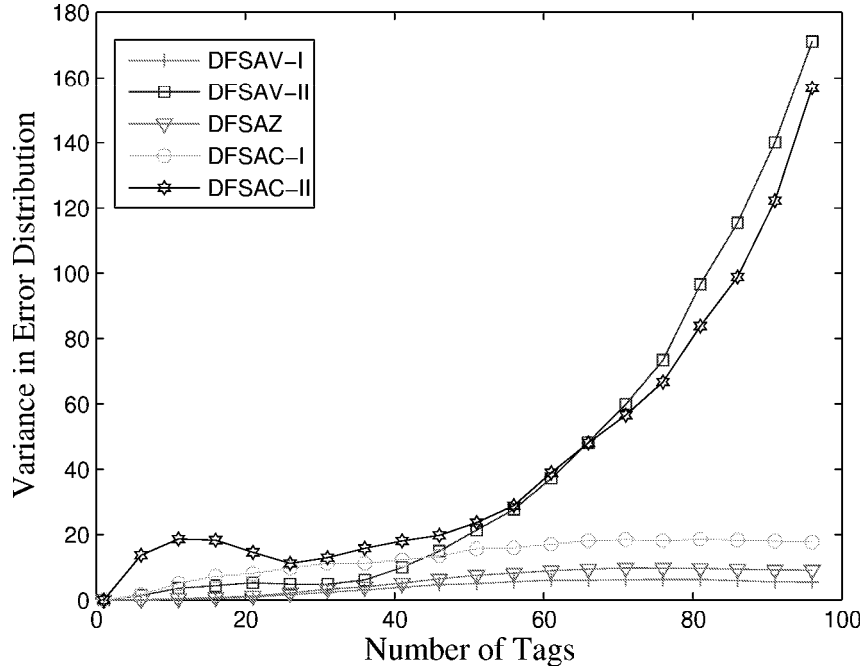
**Figure 4.7** Mean error versus number of tags.

#### 4.5.2.2 Variability

Figure 4.8 compares the variability of each function's error distribution. DFSAV-I has the most stable error estimates, followed by DFSAZ and DFSAC-I. DFSAV-II and DFSAC-II have higher variability in error distributions and therefore the error estimates are unstable. The variability of DFSAC-II is very unstable for tags ranging from 0 to 25. DFSAC-II has lower variability than DFSAV-II for  $n' > 35$ .

#### 4.5.2.3 Skew

Figure 4.9 depicts the skew observed in the error distribution of each tag estimation function. The skew of the distribution does not show a generalized pattern since it spans positive and negative values, and varies with changing tag numbers.



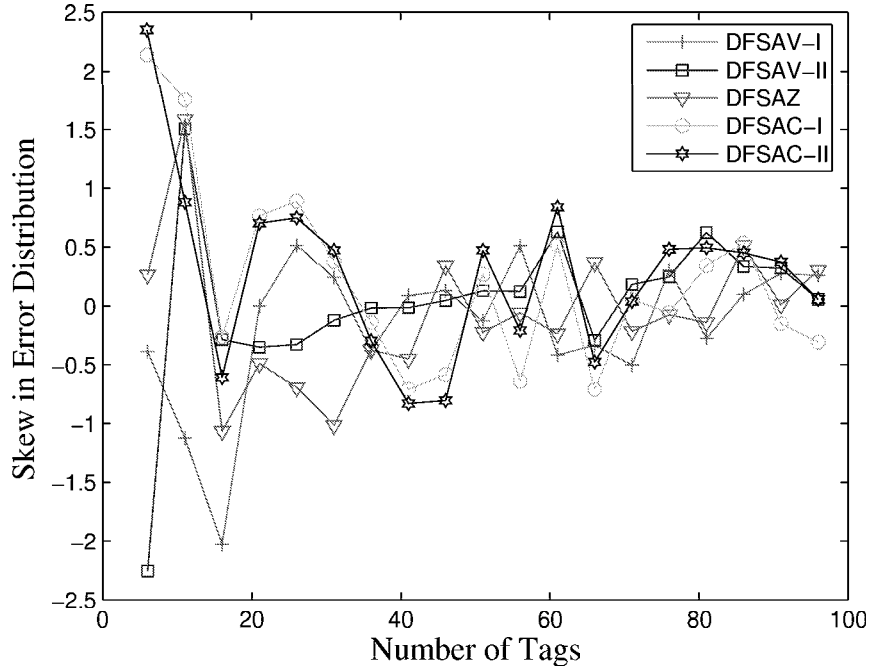
**Figure 4.8** Variability versus number of tags.

#### 4.5.2.4 Kurtosis

Figure 4.10 plots the Kurtosis for all tag estimation functions. All estimation functions are leptokurtic. The Kurtosis values for DFSAC-II vary widely compared to other estimation functions. When there are less than 10 tags, its Kurtosis is very high. This is because when the number of tags is very low; the function can estimate the tags accurately; yielding a small standard deviation value. As Kurtosis is inversely proportional to standard deviation, it reaches a very high value when the number of tags is low.

## 4.6 Conclusion

From the comparison analysis in Section 4.5.2, it can be observed that among dynamic estimation techniques, DFSAV-II has the best performance; both for low and high  $n'$  values. However, this is achieved at a higher computational



**Figure 4.9** Skew versus number of tags.

expense. DFSAC-II, although developed for muting based environments, has a higher accuracy than DFSAV-II, especially when the number of tags is higher than the frame size used. DFSAC-II is also suitable for use in non-muting tag environments, provided  $n'$  is not smaller than  $N$ , to preserve accuracy. Similarly, DFSAV-II is suitable for muting based environments, where during frame adjustments, identified tags can be omitted. DFSAC-I is the least accurate amongst all the tag estimation functions. It under estimates the tag set as it does not consider slots with single tag response. Therefore, it is unsuitable for non-muting environments.



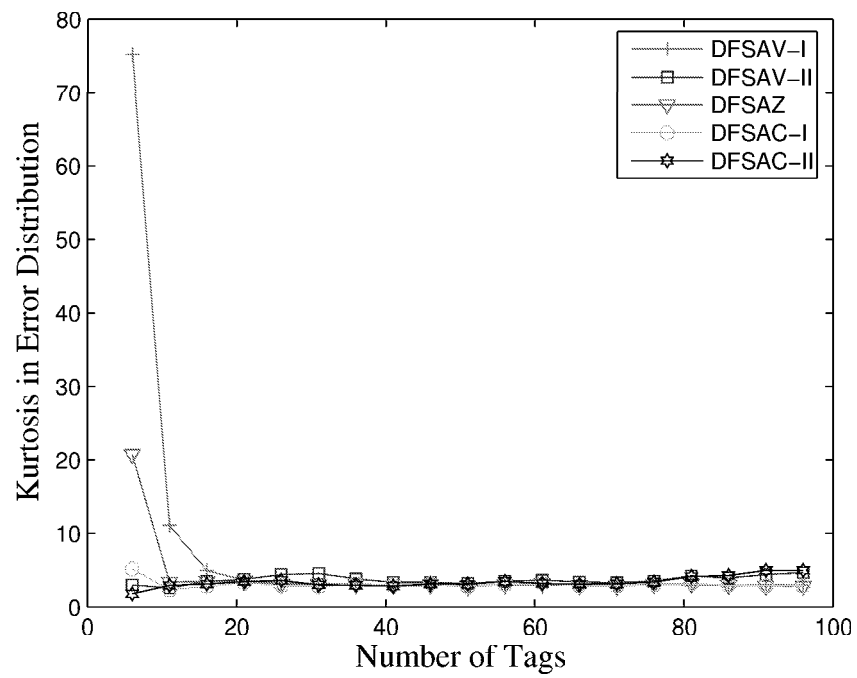


Figure 4.10 Kurtosis versus number of tags.

# Chapter 5

## Framed Slotted Aloha Based RFID Anti-Collision Protocols

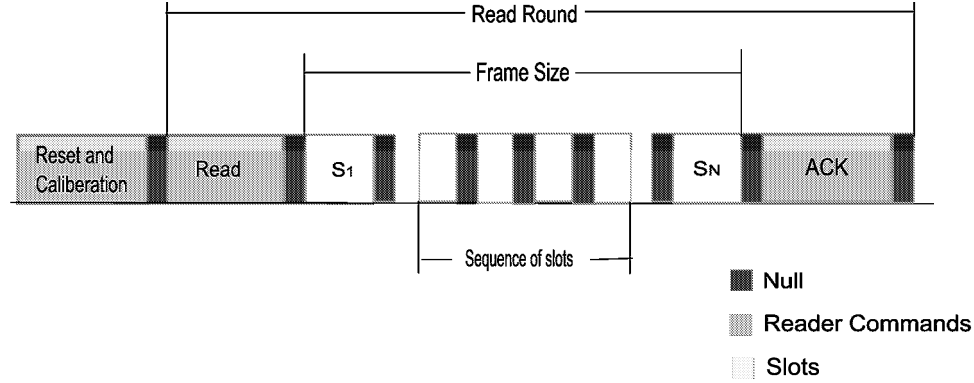
This chapter studies the energy consumption of frame slotted Aloha (FSA) based anti-collision protocols. Specifically, twelve FSA variants are investigated using a detailed qualitative and quantitative methodology to evaluate their energy efficiency with varying tag population.

The chapter is organized as follows. Section 5.1 provides a background on FSA variants. Then sections 5.2 and 5.3 outline research methodology and system model respectively. Next, results are presented in Section 5.4 followed by conclusions in Section 5.5.

### 5.1 FSA

FSA protocols group slots into a frame, where a frame's size may be fixed or variable depending on implementation [120]. A FSA with a fixed frame size is referred to as basic framed slotted Aloha (BFSA) and one that uses variable frame sizes is called dynamic framed slotted Aloha (DFSA) [12] [27]. BFSA and DFSA can be further classified according to whether they support muting and early end. Note, when the early end feature is used, a reader closes an idle or no response slot early. On the other hand, muting enables the reader to silence

tags that have been read successfully.



**Figure 5.1** Round structure for FSA variants.  $S_i$  indicates the slot number.

Figure 5.1 depicts the timing scenario for both muting and non-muting RFID systems [17] [71]. The reader-to-tag communication is controlled by commands from the reader. Initially, tags are assumed to be inactive or un-powered. Tags are activated when they receive a *reset* and *calibration* command from the reader. Once tags are activated, the reader transmits a *read* command, specifying the frame size in the current read round. Tags respond by selecting a slot randomly and transmit their ID along with a 16 bit CRC. Note, *Null* signals the completion of a command and the end of a slot, thereby serving as a synchronization mechanism that allows tags to determine slots boundary. After transmission, tags wait for an *ACK*) from the reader.

The *ACK* command is a string of bits, the length of which correspond to the frame size. Specifically, the position of each bit corresponds to a slot position, where a one indicates a successful reception and a zero indicates a failed reception or no response in the corresponding slot. In muting based systems, a positive acknowledgement mutes tags. However, in non-muting based algorithms, *ACK* is optional. For FSA systems which support early end, the reader closes an idle slot early when no responses are detected after a duration of 10 data bits [71].

### 5.1.1 Frame Adjustment

As discussed in Chapter 4, DFSA and its variants have the ability to adjust their frame size according to the number of tags in a reader's interrogation zone. The reader starts collision resolution with a predefined frame size. If a large number of responses are detected, the reader adjusts its frame size to accommodate the additional tag responses. This means the reader is required to continually adjust its frame size until it achieves an optimal frame size for a given tag population.

To achieve an optimal frame size the reader uses a tag estimation function. Based on the results in Chapter 4, the analysis to follow uses DFSAV-II for non-muting and DFSAC-II for muting based framed Aloha variants. In addition, Vogt proposed a set of frame sizes for a given tag range. According to Table 5.1, if the number of estimated tags in a reader's interrogation is in the one to nine range, the frame size should be 16 in order to achieve low reading delays.

**Table 5.1** Optimal frame sizes for a given tag range.

| <i>FrameSize(N)</i> | <i>Low(n)</i> | <i>High(n)</i> |
|---------------------|---------------|----------------|
| 1                   | —             | —              |
| 4                   | —             | —              |
| 8                   | —             | —              |
| 16                  | 1             | 9              |
| 32                  | 10            | 27             |
| 64                  | 17            | 56             |
| 128                 | 51            | 129            |
| 256                 | 112           | $\infty$       |

A limitation of all DFSA variants is that the frame size is only limited to a maximum value of 256 [70]. If the number of tags exceed this value, a reader is unable to achieve an optimal frame size. Lee et al. [77] address this issue by proposing an enhanced version of DFSA called enhanced-DFSA or EDFSA. In EDFSA, if the estimated number of tags is larger than the frame size, EDFSA divides the tags into  $M$  groups. Table 5.2 shows the value of  $M$  for a given tag range [77]. In Table 5.2,  $n$  denotes the number of tags,  $N$  is the frame size and  $M$  is the modulus operator. Lee et al. [77] also proposed frame sizes for varying tag ranges to achieve maximum system efficiency. The value of  $M$  is

one when the number of tags is lower than 355. However, when the number of tags increases, the modulo operation divides the responding tags into  $M$  groups. The reader then reads tags on a group-by-group basis. To reduce identification delay, EDFSA can be incorporated with the early end and muting features.

**Table 5.2**  $N$  and  $M$  values for EDFSA

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

From the discussion above, it is clear there are a number of FSA variants. This chapter studies all these variants, compares their energy efficiency, and determines their suitability for use in RFID-enhanced WSNs.

## 5.2 Research Methodology

In order to evaluate the energy consumption of FSA protocols, we first need to evaluate the delay incurred in different phases of the tag reading process. These phases are, i) success, ii) collision, and iii) idle listening. Note, idle listening corresponds to the scenario where the reader did not receive any responses from tags. Once we have the delay in each phase, it is used to derive the energy consumption of tag reading protocols. With the average energy consumption in hand, we can then study its effect on a sensor node's battery lifetime.

The energy consumed by a reader is determined by the duration for which it is scanning a given set of tags. If  $D$  is the total delay to read  $n$  tags then the energy consumed by a reader during scanning is,

$$E = P \times D \quad (5.1)$$

where  $E$  (Joules) is the energy consumed by a reader when scanning  $n$  tags,  $P = VI$  (Watts) is the power consumed by the reader during scanning,  $V$  (Volts) is the supply,  $I$  (Amperes) is the current consumed during scanning and  $D$  (seconds) is the scanning duration.

Equ. 5.1 can be used to calculate a sensor node's battery lifetime, which is determined by the number of tags a battery can read in its lifetime. For a given protocol, the number of tags that can be read in its lifetime is,

$$N_{given\_protocol} = n \times \frac{B}{E} \quad (5.2)$$

where  $B$  is the energy stored in a battery.

The battery energy wasted due to idle listening and collisions,  $B_{waste}$  (Joules), during tag identification is,

$$B_{waste} = B \left( 1 - \frac{n \times T}{D} \right) \quad (5.3)$$

In Equ. 5.3,  $T$  is the slot duration and it is defined as,

$$T = \frac{ID(bits)}{data\_rate(bps)} \quad (5.4)$$

where  $ID(bits)$  is a tag's identity, and  $data\_rate(bps)$  is the tags' data rate in bits per second.

The aforementioned equations, Equ. 5.1, 5.2 and 5.3, play a critical role in determining the performance of a tag reading protocol. Notice that the common parameter is  $D$  or delay. In the following sections, various methodologies to evaluate  $D$  for each FSA variant are presented. After deriving  $D$ , the following performance metrics, i) total energy consumed in the tag reading process, ii) battery lifetime, and iii) energy wastage are obtained.

### 5.2.1 BFSA

The delays incurred by BFSA variants are evaluated as follows.

### 5.2.1.1 BFSA-Non Muting

First the cycles needed to read a given set of tags with a confidence level of  $\alpha$  are evaluated. The number of read cycles is then used to determine the total delay to read a given set of tags. From the read cycles, the number of slots with idle responses and collisions are extracted, which are then used to determine the idle and collision delay respectively.

The read cycles required is evaluated as follows. With a frame size of  $N$  and the number of tags  $n$ , the probability of  $r$  tags responding in a slot in the  $i^{th}$  read round is given by [65],

$$P_r(i) = \binom{n}{r} \left(\frac{1}{N}\right)^r \left(\frac{N-1}{N}\right)^{n-r} \quad (5.5)$$

From Equ. 5.5, the probability of having an idle transmission  $p_0(i)$ , a successful transmission  $p_1(i)$ , and collisions  $p_k(i)$  in the  $i^{th}$  read round can be evaluated as,

$$p_0(i) = \left(1 - \frac{1}{N}\right)^n \quad (5.6)$$

$$p_1(i) = \frac{n}{N} \times \left(1 - \frac{1}{N}\right)^{n-1} \quad (5.7)$$

$$p_k(i) = 1 - p_0(i) - p_1(i) \quad (5.8)$$

Using Equ. 5.7 the expected number of successful transmissions in the  $i^{th}$  read round is calculated as  $Np_1(i)$  [65]. From [65], the probability of having an unread tag after  $R$  read rounds is,

$$p_{miss}(i) = \prod_{i=1}^R \left(1 - \frac{Np_1(i)}{n}\right) = 1 - \alpha \quad (5.9)$$

In Equ. 5.9,  $\alpha$  determines the confidence level of the tag reading process. Since

$p_1(i)$  is the same for all read rounds in BFSA, Equ. 5.9 becomes,

$$\left(1 - \frac{Np_1}{n}\right)^R = 1 - \alpha \quad (5.10)$$

Solving Equ. 5.10 for  $R$ , we find that the read cycles required to read a set of tags with  $\alpha$  confidence level must be at least,

$$R \geq \left\lceil \frac{\log(1 - \alpha)}{\log\left(1 - \frac{Np_1}{n}\right)} \right\rceil \quad (5.11)$$

In Equ. 5.11, to obtain an integral value and to avoid conservative delay values, the ceil function is used.

Using  $R$ , the theoretical delay in each phase of the tag reading is evaluated as follows. For a slot of duration  $T$ , the delay to read a set of tag successfully is,

$$D_{Succ\_BFSA} = NRT \quad (5.12)$$

In order to find the idle delay, the expected number of idle slots during each read cycle is determined. This can be obtained from Equ. 5.6 as  $Np_0$  for a frame size of  $N$ . Thus, the delay due to idle slots with  $\alpha$  confidence level is,

$$D_{Idle\_BFSA} = Np_0RT \quad (5.13)$$

Lastly, the delay incurred due to collisions during the reading process is given by,

$$D_{Coll\_BFSA} = NRT(1 - p_0 - p_1) \quad (5.14)$$

Note, the delays as computed by Equ. 5.12, Equ. 5.13, Equ. 5.14 assume the number of tags is known.

### 5.2.1.2 BFSA-Muting

Muting reduces the number of responses after each identification round. Hence, the number of tags in the  $(i + 1)^{th}$  read round is either equal to or fewer than



those in the  $i^{th}$  read round. The number of tags in the  $(i+1)^{th}$  round is evaluated as [65],

$$n(i+1) = n(i) - p_1(i) \times N(i) \quad (5.15)$$

In Equ. 5.15,  $p_1(i) \times N(i)$  is the number of tags identified in a read round and is denoted as  $c_1$ . Therefore,

$$n(i+1) = n(i) - c_1 \quad (5.16)$$

Based on Equ. 5.16, Algorithm 2 is used to evaluate the following metrics to read  $n$  tags: a) total delay, b) delay due to collisions, and c) delay due to idle listening.

```

BEGIN ;
Initialize unread tags = actual number of tags;
while True do
    Perform a read cycle for unread tags;
    Store the number identified tags ;
    Store the number slots filled with collisions ;
    Store the number of slots filled with idle responses ;
    Store current frame size;
    if (No Collisions) then
        Break;
    else
        Unread Tags = actual - identified tags;
    end
end
Total delay =  $T \times \sum$  stored frames;
Collision Delay =  $T \times \sum$  stored collision slots;
Idle Delay =  $T \times \sum$  stored idle slots;
END;

```

**Algorithm 2:** Pseudo-code to determine the delay in each phase of the collision resolution process for BFSA-Muting.

Algorithm 2 works as follows. A reader performs a read round and stores the frame size, number of identified tags, idle slots, and collided slots. If there are no collisions, the reader calculates the respective delays; see lines 15 to 17.

### 5.2.1.3 BFSA-Non muting-early end

Recall that the early-end feature closes an idle slot early to reduce the total time required to read a given set of tags. However, notice that the read cycles

needed to read a set of tags remain the same. This is because read cycles are independent of slot duration, see Equ. 5.11.

Let  $t < T$  be the duration after which a reader closes a slot if no responses are detected. Let's say there are  $N_{Idle\_early}$  no response slots, meaning tags will not transmit for a time period of  $(T - t)N_{Idle\_early}$ . Therefore, the average delay to read a tag in BFSA with the early end feature is calculated as,

$$D_{Success\_early} = D_{Succ\_BFSA} - (T - t)N_{Idle\_early} \quad (5.17)$$

The expected number of idle transmissions in a frame of size  $N$  in a single read round is  $Np_0$ . Thus, for  $R$  read rounds, the number of idle slot is  $N_{Idle\_early} = NRp_0$ . Inserting  $D_{Succ\_BFSA}$  and  $N_{Idle\_early}$  into Equ. 5.17, we get,

$$D_{Success\_early} = NR(T - (T - t)p_0) \quad (5.18)$$

The delay due to idle transmissions is,

$$D_{Idle\_early} = tNRp_0 \quad (5.19)$$

Note, collision delay remains unchanged in BFSA-non muting-early-end since the probability of collision is independent of slot duration. Moreover, the delay due to muting and early-end can be evaluated using Algo. 2 and using Equ. 5.17, 5.18 and 5.19.

## 5.2.2 DFSA

This section presents the methodology used to evaluate the delay incurred by DFSA-muting and DFSA-non-muting. Once the total identification delay for both protocols is computed, the methodology presented in Section 5.2.1.3 is used to obtain the delay incurred by DFSA-muting and DFSA-non-muting with the early-end feature.

### 5.2.2.1 DFSA-Non Muting

In order to evaluate the delays to read a set of tags in DFSA-Non Muting, we need to determine the i) total delay incurred to estimate a set of tags, denoted

as estimation delay, and ii) the delay incurred in reading the estimated tags with  $\alpha$  confidence level, denoted as reading delay. Summing these two delays therefore give us the delay to read a set of tags with  $\alpha$  confidence level.

```

BEGIN;
while True do
    Perform a read round;
    Estimate tag numbers;
    if (current > last estimate) then
        Adapt and store frame size;
        Store the number collided slots;
        Store the number of idle slots;
    else
        Break;
    end
end
Store tag estimate;
Store frame size;
/* Estimation delay calculations */;
Total Delay =  $T \times \sum$  stored frames;
Collision Delay =  $T \times \sum$  stored collision slots;
Idle Delay =  $T \times \sum$  stored idle slots;
END;
```

**Algorithm 3:** Pseudo-code to determine the estimation delay in each phase of the collision resolution process for DFSA-Non Muting.

Algorithm 3 [70] is used to evaluate the estimation delay incurred by DFSA-Non-Muting. In the algorithm, we first estimate the number of tags in a reader's interrogation zone before evaluating the delay due to the tag estimation function. The algorithm estimates the number of tags in each read round and compare the current tag estimate to that of the previous round. If the estimate is higher, the loop exits and the algorithm stores the current tag estimate and frame size. Lastly, the estimation delays are calculated according to lines 16-18.

Once we have the estimation delay, we need to determine the reading delay, which is calculated from the number of read cycles required to read an estimated number of tags, see Equ. 5.11. These two values are then fed into Equ. 5.12, Equ. 5.13 and Equ. 5.14 to obtain the reading delay. Finally, as mentioned, the total delay incurred by DFSA-Non Muting is obtained by adding the estimation and reading delay of each phase.

### 5.2.2.2 DFSA-Muting

To evaluate the reading delay of DFSA-muting, lines 9 to 13 of Algorithm 2 are replaced by the lines shown in Algorithm 4. The algorithm works as follows.

The reader performs a read cycle and stores the values as in Algorithm 2. If there is a collision, the reader estimates the number of tags and adapts its frame size accordingly. In addition, it determines the remaining number of unread tags in its interrogation zone. If there are no collisions, the loop exits and delays are calculated as per lines 16-18 of Algorithm 2.

```

if (No Collisions) then
  Break ;
else
  Estimate tags;
  Adapt frame size;
  Unread Tags = actual - identified tags;
end

```

**Algorithm 4:** Computing identification delays for DFSA-Muting

### 5.2.3 EDFSA

Lastly, this section presents the methodology used to evaluate the delay for EDFSA; with or without muting. Note, EDFSA-non-muting with early-end and EDFSA-muting with early-end are omitted from discussions since they follow the early-end methodology of BFSA.

#### 5.2.3.1 EDFSA-Non Muting

In EDFSA, the grouping of tags and frame adjustments are based on Table 5.2. The delay in EDFSA-non muting encompasses both estimation and reading delays, similar to DFSA-non muting. Table 5.2 can be implemented as a look up table from which the value of  $M$  is obtained along with estimation delay. For EDFSA, the delay in each phase will therefore be the summation of the estimation delay, and  $M$  times the reading delay of each group, where the reading delay of each group is evaluated similarly to DFSA-non-muting.

#### 5.2.3.2 EDFSA-Muting

EDFSA-muting can be evaluated by inserting Table 5.2 in Algorithm 4 as a look-up table. As long as the value of  $M = 1$  in Table 5.2, the delay evaluation for EDFSA resembles DFSA, hence we can apply Algorithm 4. However, when the number of tags increases, tags will be partitioned into  $M > 1$  groups. Once

Algorithm 4 finishes, the first group of tags will have been read completely, thus there will be  $M - 1$  groups remaining. Note, when  $M > 1$ , the frame size is fixed to 256 according to Table 5.2, which equates to BFSA-Muting with a frame size of 256. Therefore, the delay evaluation for the remaining  $M - 1$  groups can be based on Algorithm 2. Finally, adding the delays for each group yields the total delay incurred by EDFSA.

### 5.3 System Model

The system consists of an RFID reader and  $n$  tags in its interrogation zone. The reader model is based on the design features of SkyeTek's M1-Mini RFID reader developed to mate directly with the MICA2DOT sensor mote [48]. The reader operates from a Lithium rechargeable battery ( $B$ ) which has 0.48 Kilo-joules of energy. The tag to reader data rate is 26 kbps (ISO 15693). The power consumed during scanning ( $\psi$ ) is 180 milli-watts.

For non-muting environments, an RFID reader is assumed to transmit energy until tags are read with 99% confidence level. For muting based environments, the reader is assumed to transmit until it receives no collisions in a particular read cycle. It is assumed that tags are synchronized upon receiving a new reader command. The performance degradation due to a reader's orientation is assumed to be absent from the system. The reader detects collisions when the CRC check fails and transmits an *ACK* only when an ID is received correctly. The delay due to *null* commands are assumed to be negligible. The frame size is  $N$  and the slot duration is  $T$ .

Tags are assumed to be passive, hence have no power source, and they are used in read-only mode. Further, tags are static and can be read regardless of their orientation. Finally, tags have an ID that is 112 bits in size, which includes 16-bits of CRC.

For muting based protocols, *ACK* is used to mute tag responses. If the early-end feature is supported, the reader sends a *close slot* command after waiting for a

duration of  $t$ , which is assumed to be  $\frac{T}{10}$  in our analysis. Further, the analysis assumes a noise free channel and considers packet losses are due to collisions only.

## 5.4 Results

The algorithms developed in Section 5.2 are now used to evaluate the energy consumption of FSA variants, and study their battery lifetime and wastage.

Two cases are considered, i) low tag densities ( $n < 100$ ), and ii) high tag densities ( $n \geq 100$ ). Note, BFSA-Non-muting and BFSA-non-muting with early-end are evaluated using the theoretical formulations in sections 5.2.1.1 and 5.2.1.3. On the other hand, DFSA-non muting, EDFSA-non muting and their early end counterparts require both the use of simulations to obtain delays incurred by the tag estimation function and theoretical formulations to evaluate reading delays; see Section 5.2.2.1 and Section 5.2.3.1. Other than that, the rest of the FSA variants rely on simulations where 1000 read rounds are performed on a given tag set. The mean delay value is then computed and used to determine the energy consumption in different phases of the reading process. The delay variance (in seconds) is also recorded. The initial frame size for DFSA variants is set to 16, and BFSA variants have a fixed frame size of 32.

### 5.4.1 Low tag densities ( $n < 100$ )

First, the energy consumption when the number of tags in a reader's interrogation is less than 100 is evaluated.

#### 5.4.1.1 Total energy consumed to read $n$ tags

Figure 5.2 depicts the energy consumed by each FSA variant to read  $n$  tags. The figure comprises of non-muting and muting based FSA variants. For non-muting based variants, DFSA-non-muting with early-end has the lowest energy consumption for most tag ranges. This is because of their ability to adjust their frame sizes in accordance with tag population.

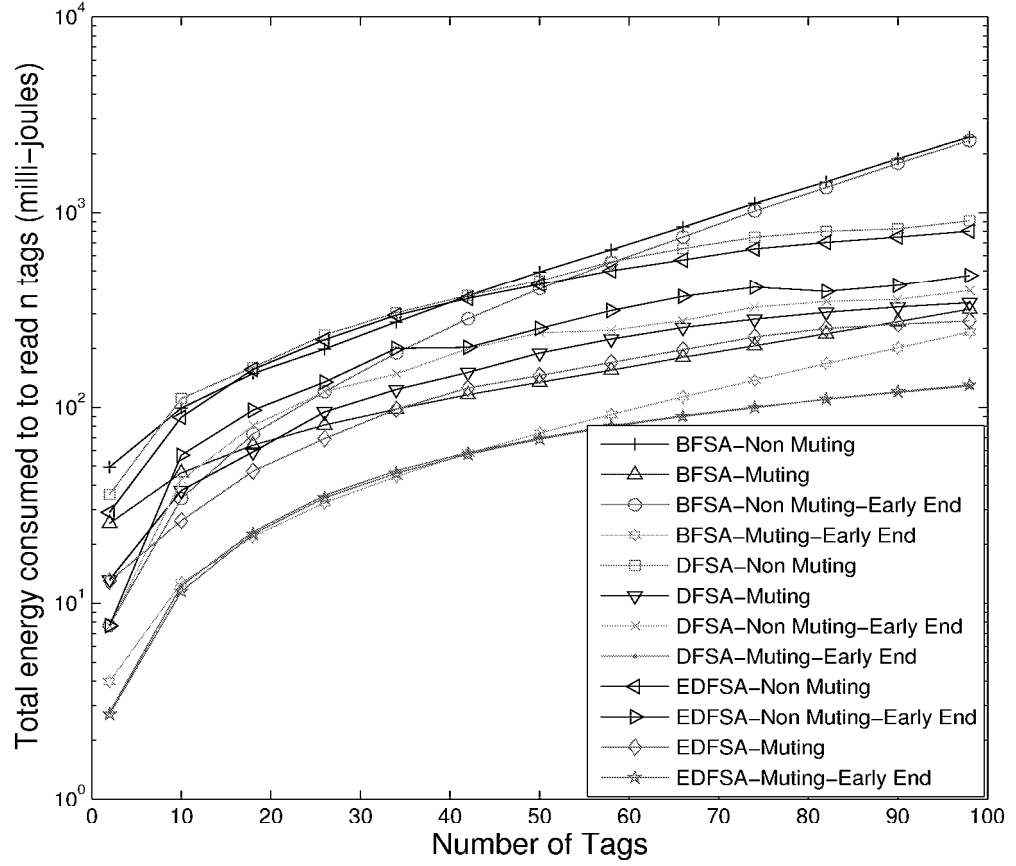
For DFSA variants employing muting, DFSA-muting has the highest energy consumption whereas EDFSA-muting with early-end has the lowest energy consumption for most tag ranges. The discrepancy in energy consumption between these two variants is due to the different methodologies used for frame adjustments. EDFSA frame sizes are smaller than those of DFSA for the same tag population. Hence, EDFSA-muting with early-end has a lower energy consumption due to its unique frame adjustment algorithm.

EDFSA-muting with early-end is found to have a delay variance in the range of 0 to  $0.9 \times 10^{-3}$  seconds, which is the lowest among all variants compared. The maximum variance is observed for DFSA non-muting; 0 to 0.38 seconds. This indicates that the energy consumption distribution of EDFSA-muting with early-end is very stable whereas DFSA Non-muting is unstable. This is because of the variability in tag estimates and the different frame adjustment methodologies used by EDFSA and DFSA variants.

Overall, FSA variants with muting have the lowest energy consumption compared to those without muting. Moreover, these variants can further reduce their energy consumption using early-end.

#### 5.4.1.2 Total energy consumed in idle listening to read $n$ tags

Figure 5.3 depicts the energy consumption incurred by each FSA variant in idle listening. DFSA non-muting consumes the most energy in idle listening. On the other hand, BFSA-muting early-end has the lowest energy wastage in idle listening for most tag ranges. Among the FSA variants, lower energy consumption is observed for those based on BFSA compared to DFSA due to it using a fixed frame. As the number of tags increases, using a fixed frame means slots is likely to be filled with a tag response, thereby reducing idle listening. In DFSA's case, the frame size varies with tag population, and is increased if large number of responses is observed, which may result in higher idle listening delay if the frame size used is non-optimal. Lastly, EDFSA muting with early-end has the least variability in energy consumption.

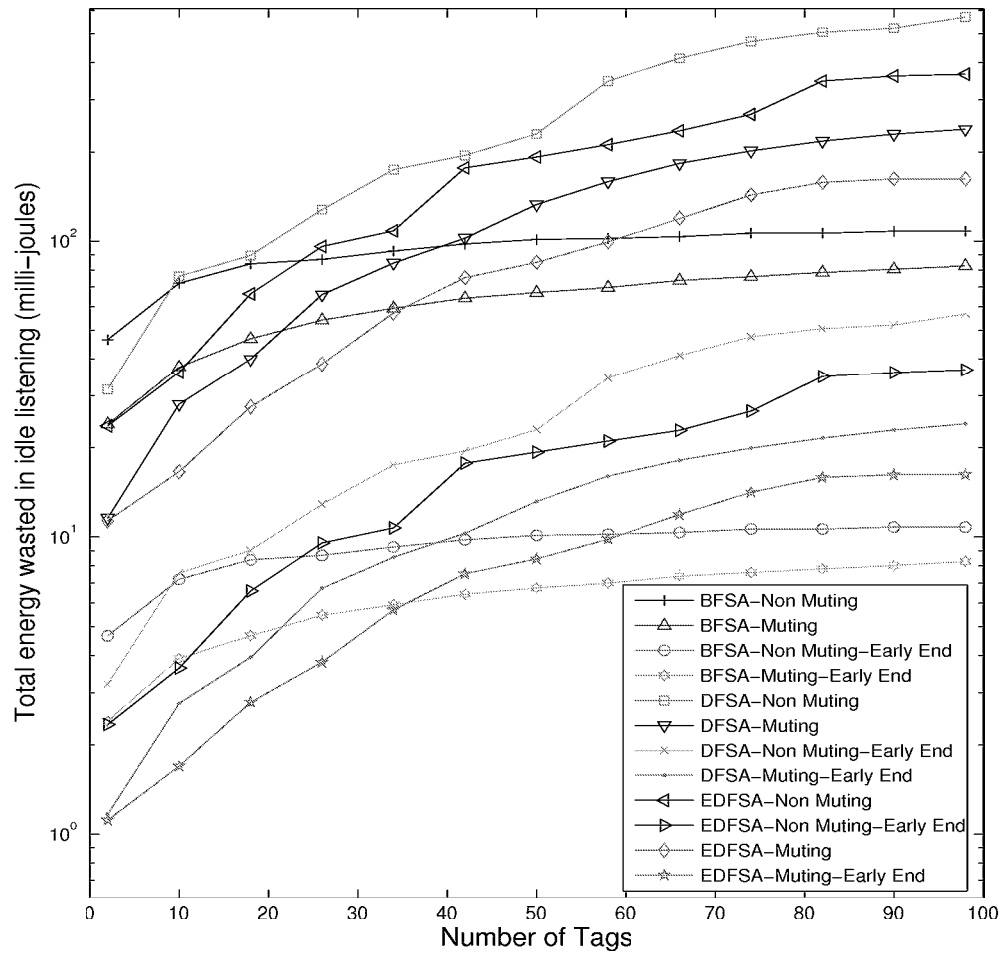


**Figure 5.2** Total energy consumed versus number of tags for FSA variants in low tag density environments.

#### 5.4.1.3 Total energy wasted in collisions to read $n$ tags

Figure 5.4 depicts the total energy wasted due to collisions when reading  $n$  tags. Firstly, it can be observed that with increasing number of tags, the energy wasted due to collisions increases for each FSA variant. The lowest energy consumption is observed for BFSA muting, and BFSA-muting with early-end when  $n < 34$ . This is because their frame size is fixed to 32 and is comparable to the number of tags for  $n < 34$ . However, for DFSA-muting and DFSA-muting with early-end, the initial frame size is 16, which results in a large number of collisions and frame adjustments when the number of tags exceeds the frame size used to read them. FSA variants which do not support muting incur significant



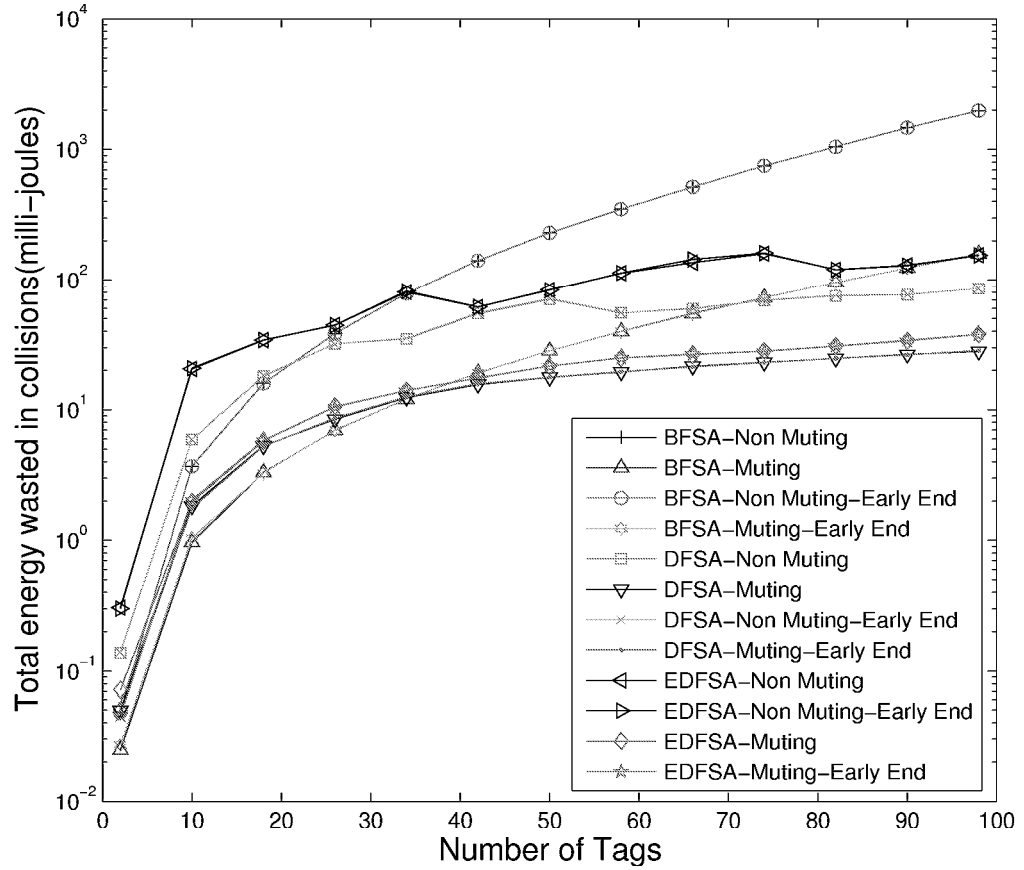


**Figure 5.3** Total energy wasted in idle listening versus number of tags for FSA variants in low tag density environments.

energy wastage due to collisions. In addition, it can be observed that early-end has no effect on the energy consumed resulting from collisions.

#### 5.4.1.4 Battery lifetime

Figure 5.5 plots the battery lifetime for each FSA variant. DFSA-muting-early-end can read the highest number of tags in its lifetime, followed by BFS-muting-early-end. DFSA-non-muting has the lowest battery lifetime when there are less than 42 tags. Otherwise, BFS non-muting has the lowest battery lifetime.



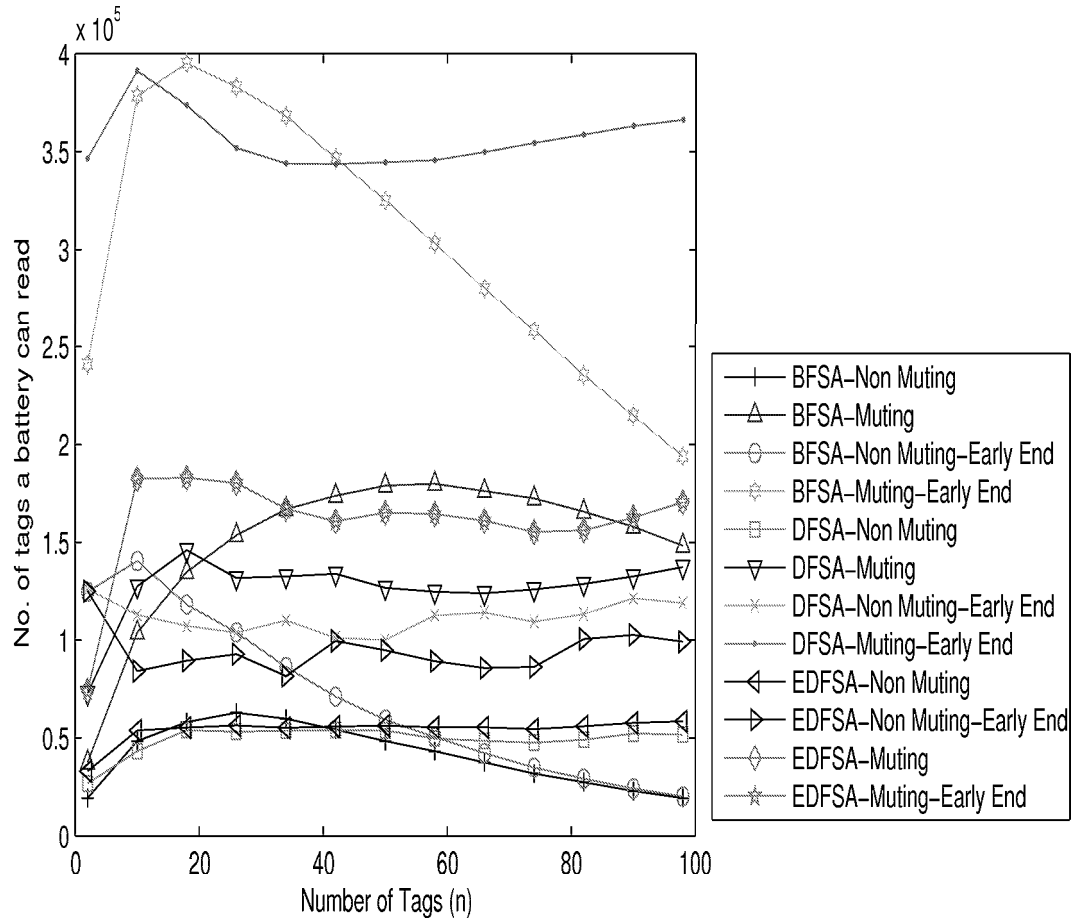
**Figure 5.4** Total energy wasted in collisions versus number of tags for FSA variants in low tag density environments.

#### 5.4.1.5 Battery wastage

Figure 5.6 plots the battery wastage versus the number of tags for each FSA variant. EDFSA-muting-early-end and DFSA-muting-early-end has a lower battery wastage as compared to all FSA variants except when  $n$  is between 17 and 42, where DFSA-non-muting-early-end has the lowest battery wastage.

#### 5.4.1.6 Summary

DFSA-Non muting has the highest energy consumption than all FSA variants when  $n < 42$ . This is because it incurs both reading and estimation delays.

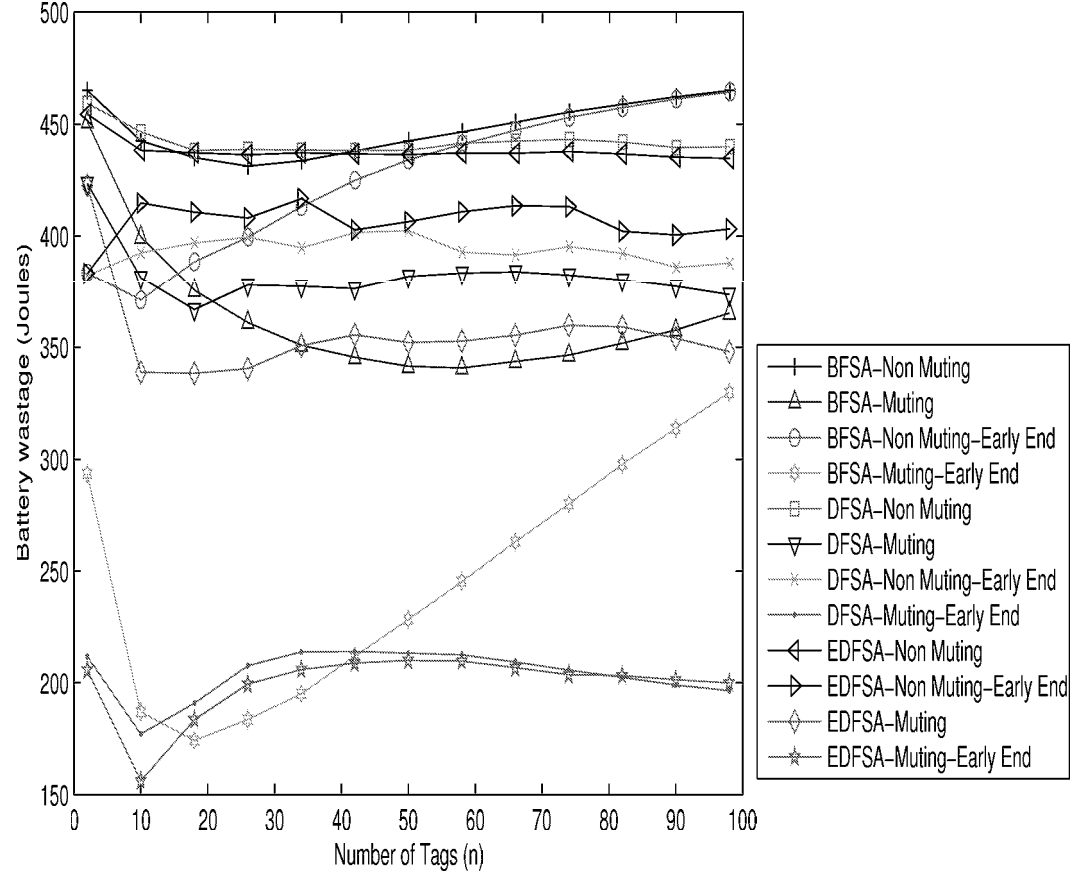


**Figure 5.5** Battery lifetime.

The energy consumption of BFSA-non muting is at its highest after  $n > 42$  because of its fixed frame size. Thus, DFSA-Non muting has a lower energy consumption than BFSA for high tag numbers. For low tag densities and non muting environments, DFSA-Non muting with early-end has the lowest energy consumption followed by EDFSA-non muting with early-end. DFSA-muting with early-end protocol has the lowest energy consumption among all FSA variants.

#### 5.4.2 High tag densities ( $n > 100$ )

This section evaluates the energy consumption when the number of tags in a reader's interrogation is more than 100. BFSA variants are omitted from our



**Figure 5.6** Battery energy wasted to read  $n$  tags.

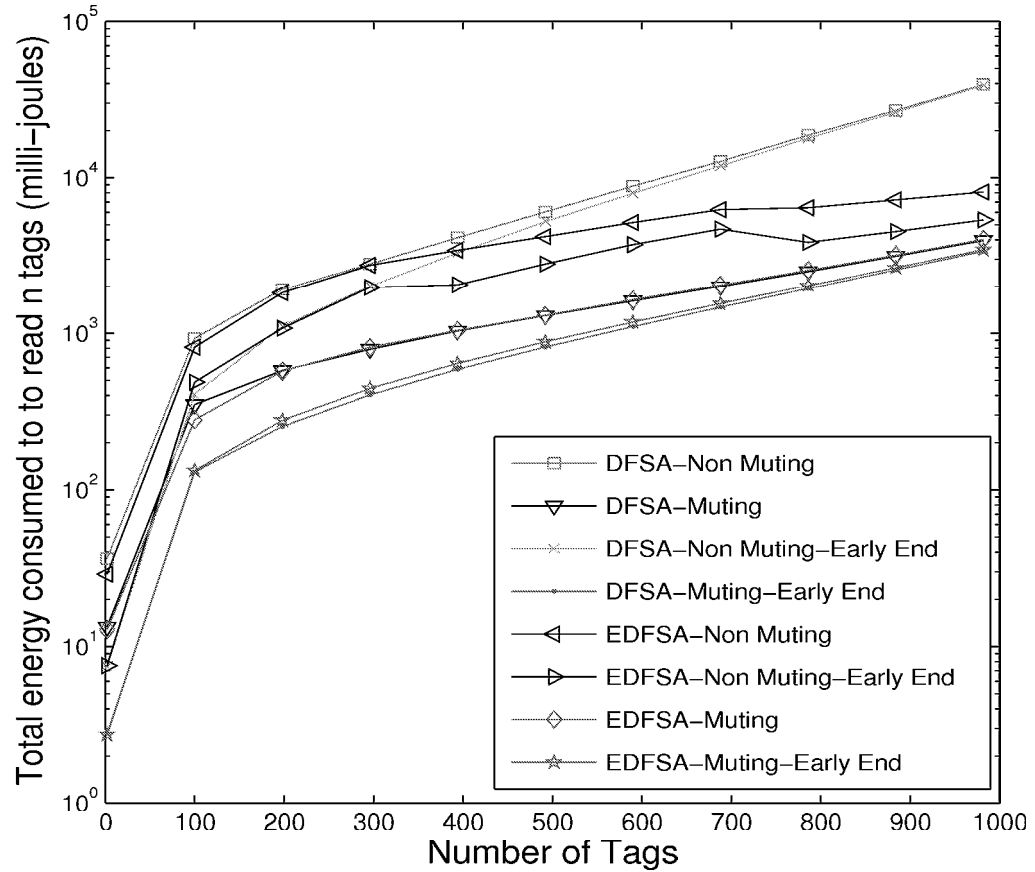
plots since they experience an exponential rise in delay, hence are unsuitable for use in high tag densities scenarios.

#### 5.4.2.1 Total energy consumed to read $n$ tags

Figure 5.7 plots the energy consumption of DFSA variants. DFSA-non muting has the highest energy consumption. On the other hand, EDFSA-non-muting with early-end has the lowest energy consumption. In muting based DFSA variants, EDFSA-muting has the highest energy consumption. On the other hand, DFSA-muting-early end has the lowest energy consumption.

Both DFSA-muting early-end and EDFSA-muting with early-end have a low

delay variance. On the other hand, DFSA-non muting and DFSA non-muting have the highest delay variance with values as high as 1500 seconds when  $n$  is close to 1000 tags.

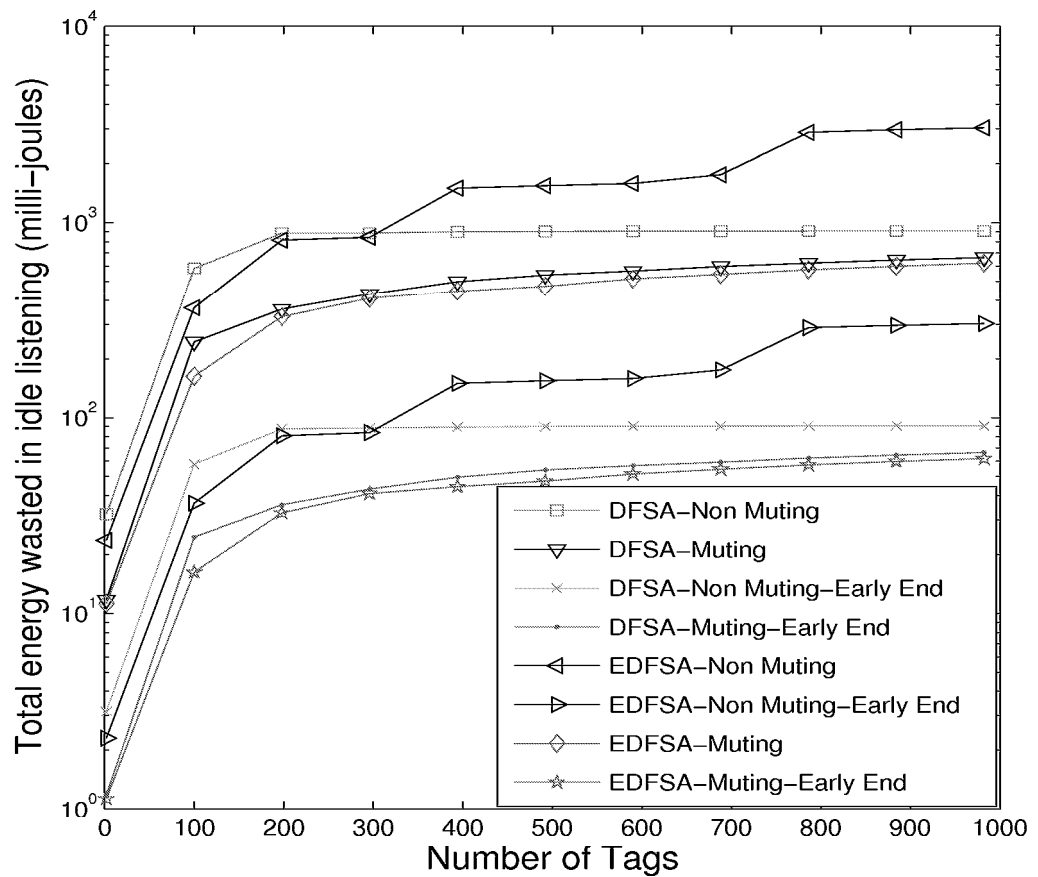


**Figure 5.7** Total energy consumed versus number of tags for DFSA variants in high tag density environments.

#### 5.4.2.2 Total energy consumed in idle listening to read $n$ tags

Figure 5.8 plots the energy wastage of DFSA variants due to idle listening. EDFSA-non-muting wastes a large amount of energy due idle listening compared to other FSA variants when  $n > 300$ . For  $n < 300$ , DFSA-non-muting has the highest idle listening delay. For non-muting variants, EDFSA-non muting with early-end experiences minimal idle listening delay. EDFSA-muting and EDFSA-muting with early-end have a lower energy wastage due to idle listening

compared to DFSA-muting and DFSA-muting with early-end because they rely on different frame adjustment techniques discussed in Section 5.1.1. The frame sizes proposed for DFSA are larger than those proposed for EDFSA, thereby causing higher energy wastage due to idle listening. Overall, EDFSA-muting with early-end has the lowest energy wastage due to idle listening, and lowest energy consumption variability.

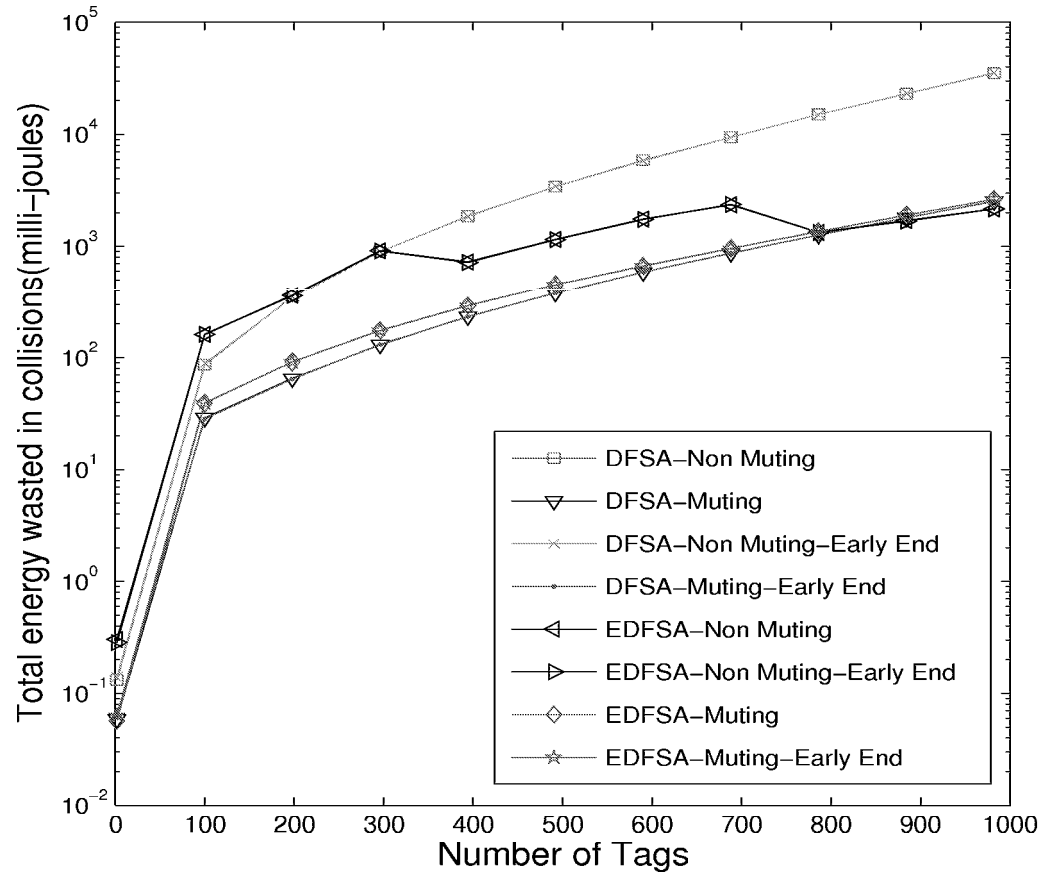


**Figure 5.8** Total energy wasted in idle listening versus number of tags for DFSA variants in high tag density environments.

#### 5.4.2.3 Total energy wasted in collisions to read $n$ tags

Figure 5.9 plots the energy wastage of DFSA variants due to collisions. DFSA-muting with and without early-end has the lowest energy wastage due to collisions for most tag ranges, and also have the lowest delay variance, maximum

being 0.33 seconds when  $n = 982$ .



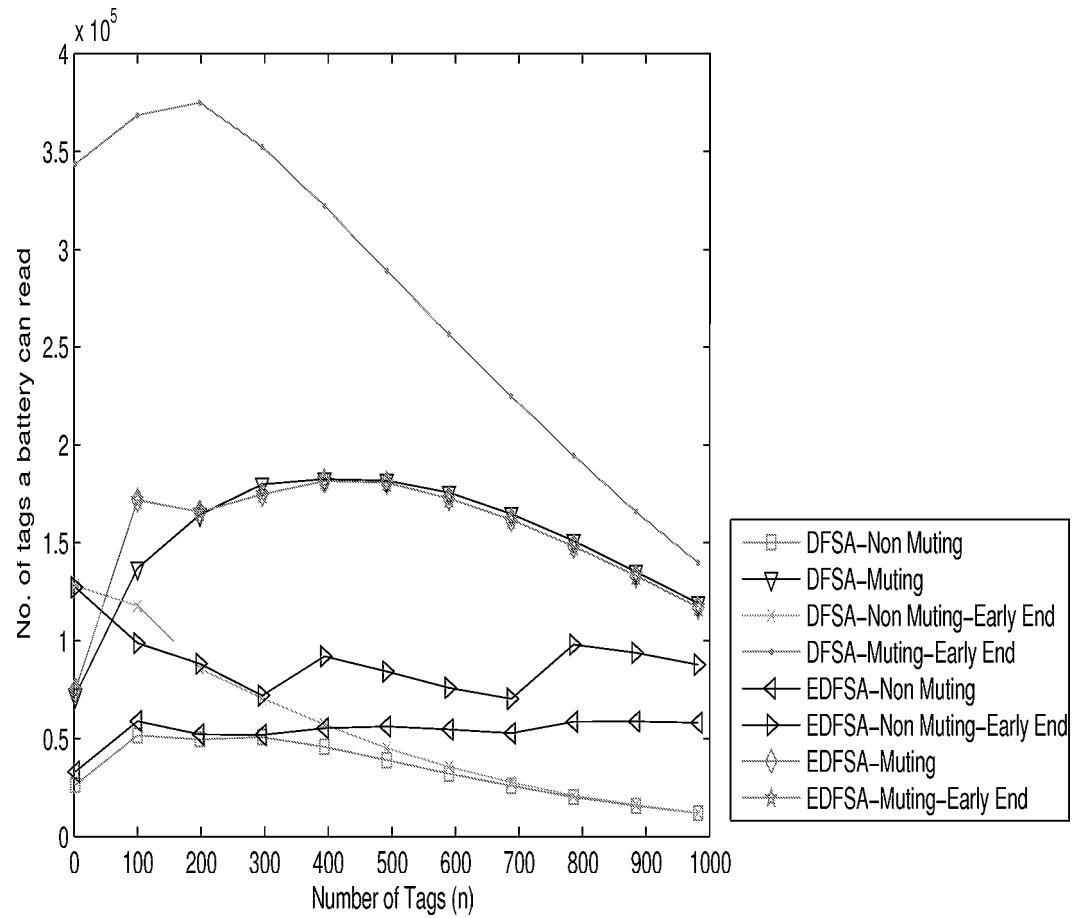
**Figure 5.9** Total energy wasted in collisions versus number of tags for DFSA variants in low tag density environments.

#### 5.4.2.4 Battery lifetime

Figure 5.10 plots the battery lifetime of FSA variants. DFSA-muting-early-end has the highest lifetime. DFSA-non-muting, on the other hand, has the lowest battery lifetime.

#### 5.4.2.5 Battery wastage

Figure 5.11 plots the battery wastage versus the number of tags for FSA variants. Battery wastage is the highest for DFSA-non-muting and lowest for



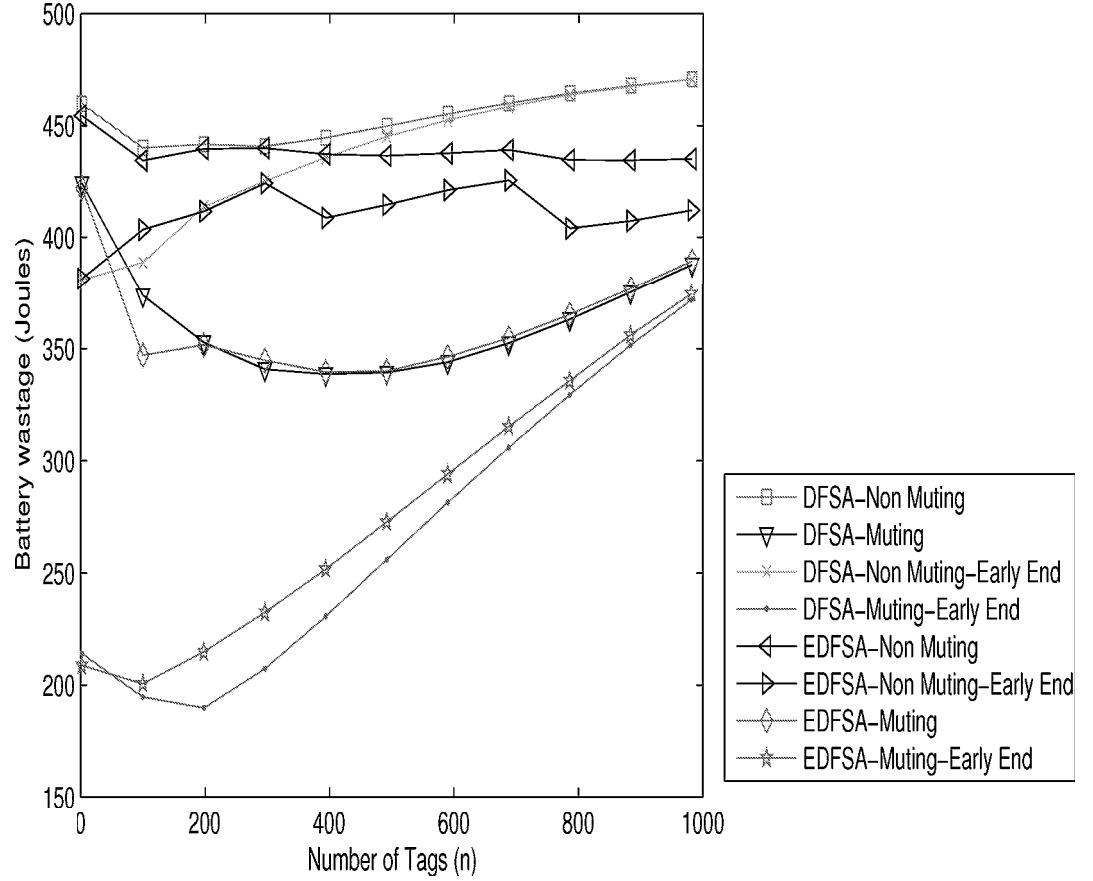
**Figure 5.10** Battery lifetime.

DFSA-muting-early-end.

#### 5.4.2.6 Summary

From our analysis, DFSA-non-muting consumes the most energy and DFSA-muting with early-end has the lowest energy consumption. On the other hand, for non-muting DFSA variants, EDFSA non-muting with early-end has the lowest energy consumption in high tag density scenarios.





**Figure 5.11** Battery energy wasted to read  $n$  tags.

## 5.5 Conclusion

The main findings of this chapter are that for low tag densities and in non muting environments, DFSA-non-muting with early-end has the lowest energy consumption, whereas EDFSA-non-muting with early-end performs well when tag density is high. In muting based systems, DFSA-muting with early-end has the lowest energy consumption for both low and high tag density environments. Amongst the twelve FSA variants, DFSA with muting and early-end is best suited for energy efficient identification. However, all the protocols are required to re-read tags if used for monitoring, thereby making them unsuitable for use in RFID-enhanced WSNs.

Based on our observations herein, a new protocol called ResMon is presented in Chapter 6 that uses frame adjustment properties of DFSA in conjunction with muting and early end features. In addition, ResMon uses small reservation slots that precedes tag identification. Moreover, ResMon uses a monitor frame that is specially designed to reduce energy consumption during monitoring.

# Chapter 6

## ResMon

A key observation in Chapter 3 and 5 is that existing anti-collision protocols consume a significant amount of energy during identification. Critically, they cannot be used to monitor tags in an energy efficient manner. Thus, making them unsuitable for use in energy constrained RFID-enhanced nodes. In this respect, no work has specifically addressed energy efficient monitoring in RFID-enhanced WSNs.

This chapter proposes ResMon, an energy efficient dynamic framed slotted Aloha (DFSA) protocol called ResMon that is suitable for RFID-enhanced WSNs. ResMon uses three frames to identify and monitor tags. Moreover, it uses Cha et al. [66]’s DFSAC-II estimation function, which has been specifically developed for muting based RFID systems and has been shown in Chapter 4 to yield accurate estimates. In addition, ResMon uses the optimal frame size proposed by Vogt [69] for a given tag population.

The closest work to ResMon is [68], where the authors use reservation slots to resolve contention. Their algorithm, called detection and jump (DJ), involves the use of a detection frame, where tags contend with each other in 16-bit slots to transmit a randomly generated bit string. The main limitation of their protocol is that it needs to re-identify all tags whenever there is a change in tag numbers, which leads to inefficient use of energy.

This chapter is organized as follows. Section 6.1 describes ResMon, including the frames that it uses and how they are used to read and monitor tags. After that, the research methodology and simulation results used to verify ResMon are presented in Section 6.2 and 6.3 respectively. Lastly, Section 6.4 concludes the chapter.

## 6.1 ResMon

ResMon has three frames: reservation ( $R_{Frame}$ ), body ( $B_{Frame}$ ), and monitor ( $M_{Frame}$ ). Firstly each frame is defined before providing an example that illustrates how they are used to read and monitor tags.

### 6.1.1 Frames

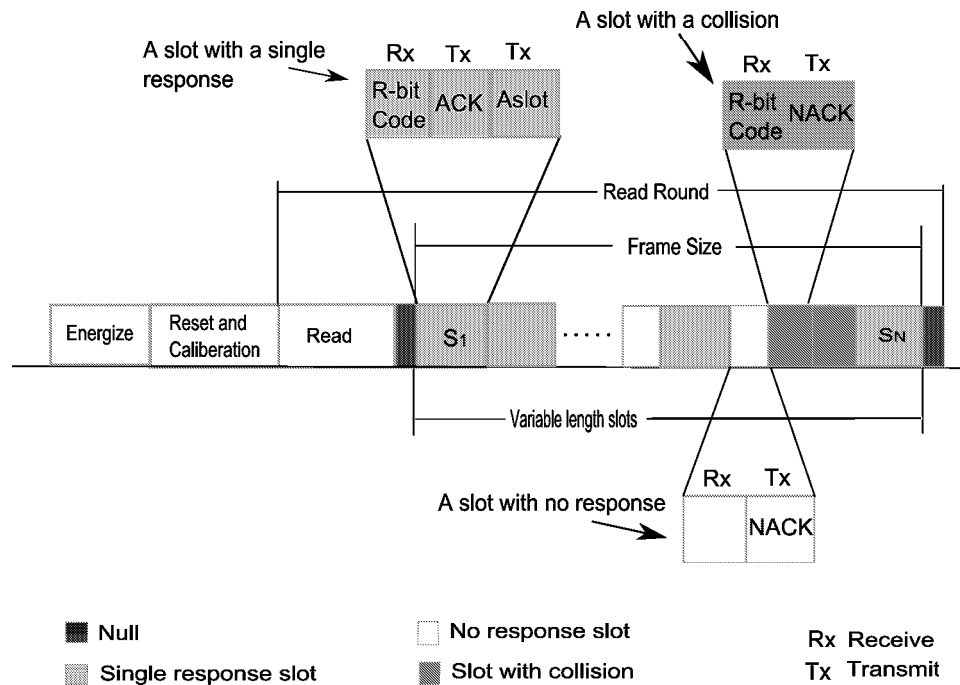
#### 6.1.1.1 Reservation

$R_{Frame}$  is used to allocate each tag a unique slot in the forthcoming  $B_{Frame}$ , where tags transmit their ID to the reader. Figure 6.1 depicts the  $R_{Frame}$  structure. After energizing tags, the reader transmits a *reset and calibration* command. Following that, a *read* command is transmitted, which specifies the number of reservation slots  $N_i$  to follow.  $N_i$  is restricted to values that are powers of two, and a maximum value of 256 [66]. The reader then transmits a *Null* command, which signals the start of reservation slots. If a single response is received in a slot, the reader transmits an acknowledgement (*ACK*); otherwise a negative ACK (*NACK*) is transmitted. The start and end of a slot is controlled by *ACK/NACK*. This allows the reader to vary a slot's duration depending on whether a slot is collision free. Specifically, if a slot is collision free, then the slot includes the transmission of an  $A_{Slot}$  command followed by an *ACK* command. Otherwise, the reader ends a slot with a *NACK* command after detecting a collision or an idle slot.

Each tag selects a slot,  $S_i \in \{1, N_i\}$ , randomly, and transmits a randomly generated bit string that is  $R$  bits in length, where  $length(ID) > length(R)$ . If the reader successfully receives  $R$ , the reader allocates a  $B_{Frame}$  slot to the

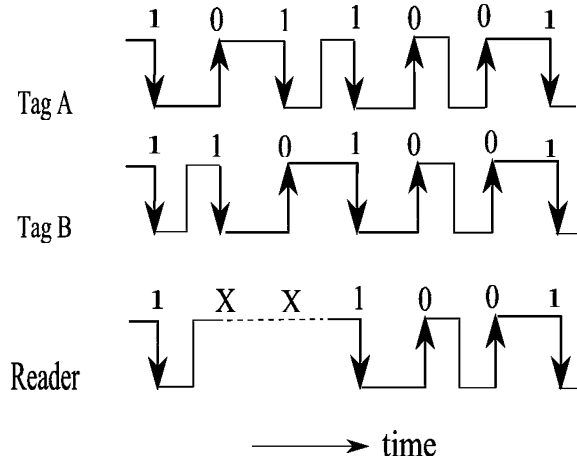
responding tag via an  $A_{Slot}$  command. A tag then stores the allocated slot in its unique frame slot counter (UFSC) before entering mute state.

The  $A_{Slot}$  command is 9 bits in length, meaning the  $B_{frame}$  can only accommodate 512 tags. For high tag density environments, the length of  $A_{Slot}$  command should be adjusted accordingly. Finally, after transmitting an  $A_{Slot}$  command, the reader increments its body frame slot counter (BFSC); a counter that tracks the last allocated slot in the upcoming  $B_{Frame}$ .



**Figure 6.1** Reservation frame structure.

Tags' transmissions are Manchester encoded to facilitate collision detection. In Manchester encoding, each bit is transmitted in a fixed duration called a bit period. Each encoded bit contains a transition at the midpoint of the bit period. The direction of a transition determines whether the received bit is a 0 or 1. A '0' is expressed by a low to high transition, and a '1' by high to low transition. Figure 6.2 shows a Manchester coded bit string transmitted by tags A and B simultaneously. Also shown is the resulting signal from these two transmissions. As can be seen, bits 2 and 3 cause no change in the received signal level, meaning a collision has occurred.



**Figure 6.2** Collision detection using Manchester encoding.

The reader relies on a tag estimation function to adjust the  $R_{frame}$  size used in every round. Recall that ResMon uses muting, hence an estimation function is required that take muting into account. As seen in Chapter 5, Cha et al. [66]’s Cha-I function is specifically devised for muting based DFSA protocols and is used in resMon. Once a tag estimate is obtained, the reader updates its frame size according to Table 6.1 [69].

**Table 6.1** Optimal frame sizes for a given tag range.

| $FrameSize(N)$ | Tags range (n) |
|----------------|----------------|
| 16             | 1 – 9          |
| 32             | 10 – 27        |
| 64             | 17 – 56        |
| 128            | 51 – 129       |
| 256            | 112 – $\infty$ |

### 6.1.1.2 Body

A  $B_{Frame}$  is transmitted after the  $R_{Frame}$  when  $BFSC > 1$ , and no collisions have been detected in the  $R_{Frame}$ , thereby indicating that all tags have been allocated a  $B_{Frame}$  slot and are muted.

Figure 6.3 depicts the structure of a  $B_{Frame}$ . The reader transmits an *unmute* command to activate all tags. After that, the reader transmits a synch pulse to synchronize tag responses. Following that, the reader transmits a body

frame ( $BF$ ) command to inform tags to respond with their ID. The reader then transmits a slot offset ( $SO$ ) command, which is used to skip tags identified in previous  $B_{Frame}$  rounds.

$SO$  is initially set to zero. On reception of a  $SO$  command, tags with  $UFSC \leq SO$  are muted, which corresponds to those tags identified in the last  $B_{Frame}$ . On the other hand, tags with  $UFSC > SO$  remain active, and respond according to Algorithm 5. Once  $B_{Frame}$  ends, the reader sets  $SO = BFSC - 1$ .

```

if ( $SO=0$ ) then
  | Tag transmits if the current reader slot is equal to tag's UFSC ;
else
  |  $UFSC_{tmp} = UFSC$ ;
  | Tag transmits if the current reader slot is equal to ( $UFSC_{tmp} - SO$ ) ;
end

```

**Algorithm 5:** Pseudo code to avoid re-identification of tags.

The size of  $B_{Frame}$  equals  $BFSC - 1$  when  $SO = 0$ , and  $BFSC - 1 - SO$  for  $SO > 0$ . The number of slots are limited to 512, which is twice the maximum frame size available in  $R_{Frame}$ . However, in practice, a higher number of slots can be used if the reader has sufficient resources to manage more than 512 tags.

The reader transmits an  $ACK$  when an ID is received successfully and a  $NACK$  when it experiences collisions. Collisions occur when two or more tags transmit the same random bit string  $R$  simultaneously in a  $R_{Frame}$ , thereby causing the reader to assign them the same UFSC. Therefore, the reader must update their UFSC to resolve future collisions. To do this, each tag maintains a next frame slot counter (NFSC) that operates as follows.

On reception of a  $BF$  command, each tag initializes its NFSC to UFSC. Each tag then transmits in the  $UFSC^{th}$  slot. For every slot with a single tag response, the reader transmits an  $ACK$ . On reception of an  $ACK$ , a tag first sets its UFSC to NFSC before going to sleep. On the other hand, for every slot with a collision, the reader transmits a  $NACK$ , which causes each unread tag to decrement its NFSC by one, and collided tags to set their UFSC and NFSC to zero.

Table 6.2 illustrates how slots with collisions are removed from a  $B_{Frame}$ . In round 1, tags C and D's transmission resulted in a collision, causing the reader to

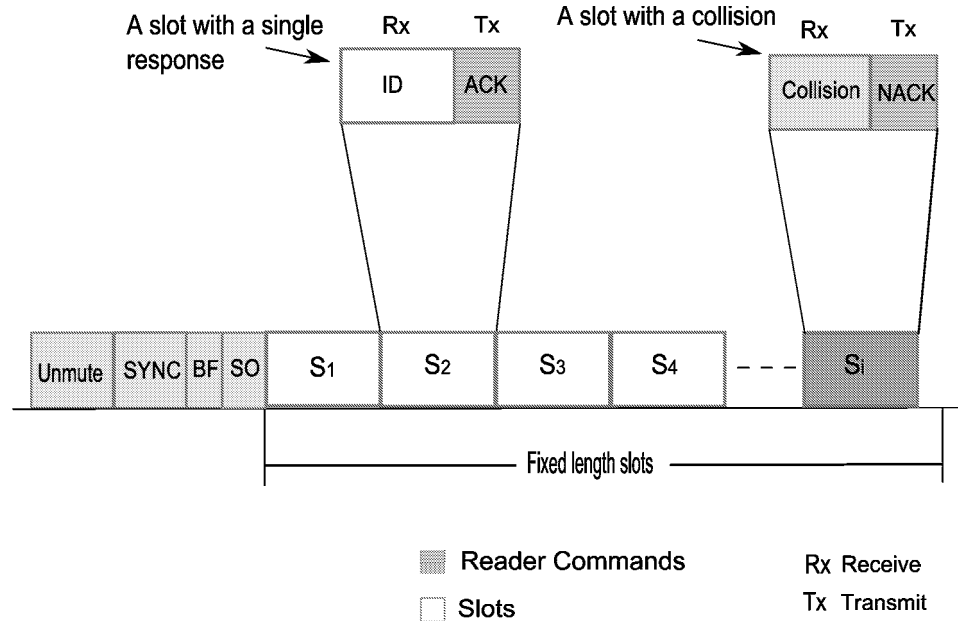
transmit a *NACK*. On reception of a *NACK*, the tags operate as per Algorithm 6. That is, tags C and D set their UFSC and NFSC to zero, whereas tag E decrements its NFSC by 1. Setting C and D's UFSC to zero effectively barred them from participating in future  $B_{Frame}$  and  $M_{Frame}$  because slots always begin from one. Hence, tags C and D will have to contend again in the next  $R_{Frame}$  to gain a unique slot in order to transmit in the following  $B_{Frame}$ .

```

if  $UFSC == Current_{ReaderSlot} \& NACK$  then
  |  $NFSC = 0$ ;
  |  $UFSC = 0$ ;
else if  $UFSC == Current_{ReaderSlot} \& ACK$  then
  |  $UFSC = NFSC$ ;
else if  $((UFSC \neq Current_{ReaderSlot}) \& (NACK) \& (NotMuted))$  then
  |  $NFSC = NFSC - 1$ ;
else
  | No Change;
end

```

**Algorithm 6:** Pseudo code used by a tag to remove collided slots in a  $B_{Frame}$ .



**Figure 6.3** Body frame structure.  $S_i = BFSC - 1$  when  $SO = 0$ .  $S_i = BFSC - 1 - SO$  when  $SO > 0$ .



Table 6.2 Removal of collided slots from  $B_{Frame}$ .

|                           | Tags      | Response slot (UFSC) | Reader's slot | Reader's response | NFSC before ACK/NACK | Update NFSC using Algorithm 6 | Update UFSC using Algorithm 6 |
|---------------------------|-----------|----------------------|---------------|-------------------|----------------------|-------------------------------|-------------------------------|
| <b>Body Frame Round 1</b> | Tag A     | 1                    | 1             | ACK               | 1                    | 1 (Unchanged)*                | 1                             |
|                           | Tag B     | 2                    | 2             | ACK               | 2                    | 2 (Unchanged)*                | 2                             |
|                           | Tag C & D | 3                    | 3             | NACK              | 3                    | 0 (Changed)(**)(***)          | 0                             |
|                           | Tag E     | 4                    | 4             | ACK               | 4                    | 3 (Changed)                   | 3                             |

\* Tag A or B updates its UFSC to NFSC because its UFSC equals the current reader slot, and it received an ACK (See Algorithm 6).  
 \*\* Tag C and D set their UFSC and NFSC to zero because their UFSC matches the current reader slot, and they received a NACK (See Algorithm 6).  
 \*\*\* Tag E decrements its NFSC (4-1=3) because it is not muted, its UFSC is not equal to the current reader's slot, and it received a NACK (See Algorithm 6).

### 6.1.1.3 Monitor

The reader uses  $M_{Frame}$  to monitor tags. Note that  $M_{Frame}$  is collision free, given that every tag has a unique transmission slot.

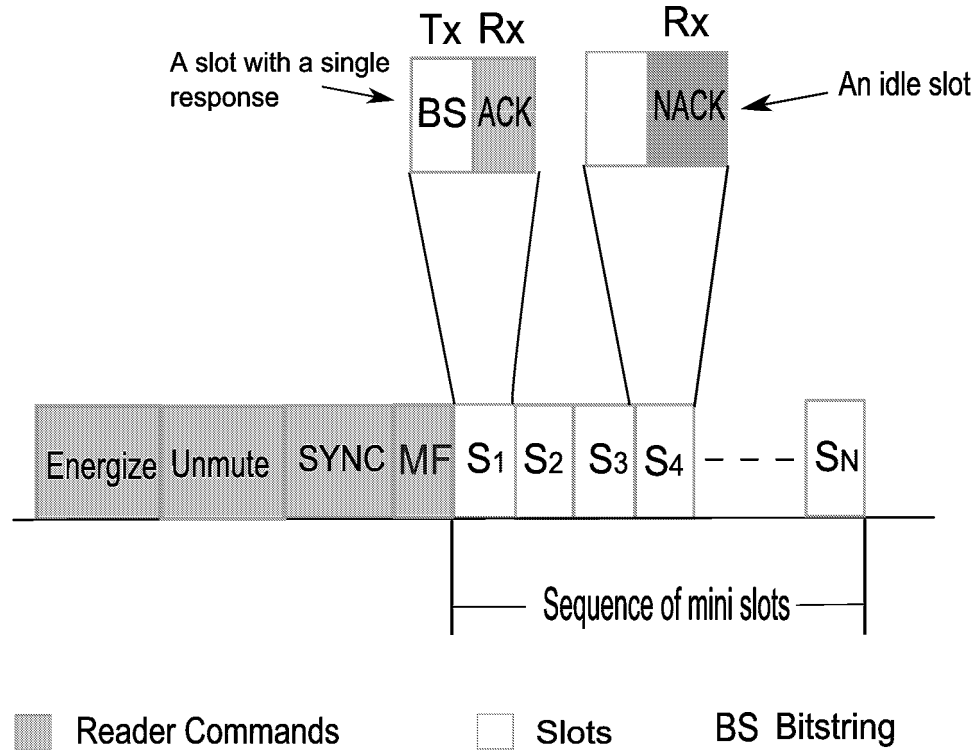
Figure 6.4 depicts the  $M_{Frame}$  structure. The reader first energizes tags before transmitting a *Sync* pulse to synchronize tag responses. After that, the reader sends a monitor frame ( $MF$ ) command to mark the start of the monitoring phase, with the number of slots set to  $BFSC - 1$ .

Each tag responds with the same predefined 4-bit Manchester coded bit string. If the bit string is received successfully, the reader transmits an *ACK*. In addition to acknowledging the tag, the *ACK* also has the effect of muting the tag. If a tag has departed from the reader's interrogation zone, the reader will receive no response, and hence experiences an idle slot. In this case, the reader transmits a *NACK*. Both *ACK* and *NACK* assist the reader in removing idle slots, as elaborated in Section 6.1.4.

## 6.1.2 ResMon Operation

This section shows the use of the aforementioned frames are used to identify tags. Let's say there are four tags A, B, C, and D. During initialization, the reader initializes BFSC to one, and tags set their UFSC to zero. The reader begins by transmitting a  $R_{Frame}$ . Assume that the reader identifies these tags in the following order: A, B, C, and D. Hence, tag A will be assigned a UFSC value of one via the  $A_{Slot}$  command, tag B will have a value of two, and so forth. After each tag is identified, the reader increases its BFSC by one. Thus, at the end of the  $R_{Frame}$ , BFSC will have a value of five.

The reader then starts a  $B_{Frame}$ , given that tags were detected in  $R_{Frame}$  with a *SO* value of zero. The number of slots in the  $B_{Frame}$  is  $BFSC - 1$ , which equals four. Tags then respond according to Algorithm 1, and upon conclusion, the reader sets the value of *SO* to  $BFSC - 1$ . If there are collisions, tags use Algorithm 6 to update their UFSC. The reader then keeps track of these collided



**Figure 6.4** Monitor frame structure.

slots and subtracts the number of slots with collision from the BFSC at the end of the frame.

Lastly, the reader transmits  $M_{Frame}$  with  $BFSC - 1$  slots. As specified in Section 6.1.1.3, all tags respond with the same 4-bit bit string in their allocated slots. Upon receiving a reply, the reader notes that a tag is still present. Otherwise, if an idle slot is encountered, the ResMon initiates the idle slot removal process outlined in Section 6.1.4.

### 6.1.3 New Tags

Assume there are four new tags in the reader's interrogation zone: D, E, F, and G. After some time, the reader transmits a  $R_{Frame}$  and the slot allocation begins from the current BFSC value, i.e., 5. Let's assume the tags are identified in the following sequence: D, E, F, and G. Thus, the UFSC of these tags is 5, 6, 7, and 8 respectively, and the reader would have a BFSC value of 9.

To read these new tags, the reader transmits a  $B_{Frame}$ . Recall that tags A, B, C, and D have already been identified. Hence, we will need to omit them. To do this, the reader transmits a  $SO$  command with a value of 4. Then, based on Algorithm 5, tags D, E, F, and G, transmit in the  $(UFSC_{tmp} - SO)^{th}$  slot, i.e., slots 1, 2, 3, and 4 respectively. Note, the UFSC value for tags D, E, F, and G remains unchanged. Hence, at the end of this process, tags A to G have a UFSC value of 1 to 9 respectively.

#### 6.1.4 Departing Tags

Tags may leave a reader's interrogation zone, thereby causing idle slots in  $M_{Frame}$ , and unnecessarily prolonging the monitoring process. To remove these idle slots, a tag uses its next frame slot counter (NFSC).

On reception of the  $MF$  command, each tag initializes its NFSC to UFSC. Each tag then transmits in the  $UFSC^{th}$  slot. For every idle slot, as indicated by a  $NACK$ , each unread tag decrements its NFSC by one. On the other hand, if a single response is received, the reader transmits an  $ACK$ . Those tags with UFSC that matches the current reader slot and received an  $ACK$  set their UFSC to NFSC before going into mute state.

Table 6.3 illustrates how idle slots are removed. In monitoring round 1, no tag has departed. Thus, the NFSC and UFSC value of tags remains unchanged. In monitoring round 2, Tag- C departs. This causes slot-3 to be idle, thereby, causing the reader to transmit a  $NACK$ . Since tags A and B have been muted, the  $NACK$  does not affect their UFSC. Tag-D, however, will decrement its UFSC by one. As a result, in monitoring round 3, tag D responds in slot 3 instead of slot 4.

The reader uses an idle slot counter (IDSC) to record the number of idle slots appearing in its  $M_{Frame}$ . After removing all idle slots, the number of slots in a  $M_{Frame}$  is computed as  $BFSC = BFSC_{Current} - 1 - IDSC$ , where  $BFSC_{Current}$  is the value of BFSC at the beginning of a  $M_{Frame}$ .

Table 6.3 Departing tags example.

|  | Tags  | Response slot<br>UFSC | NFSC before<br>ACK/NACK | Reader's Resposne | NFSC after<br>ACK/NACK | Assign NFSC to UFSC<br>after NACK/ACK |
|--|-------|-----------------------|-------------------------|-------------------|------------------------|---------------------------------------|
| Monitoring Round 1                     | Tag A | 1                     | 1                       | ACK               | 1 (Unchanged)          | 1 (Unchanged)                         |
|  | Tag B | 2                     | 2                       | ACK               | 2 (Unchanged)          | 2 (Unchanged)                         |
|  | Tag C | 3                     | 3                       | ACK               | 3 (Unchanged)          | 3 (Unchanged)                         |
|  | Tag D | 4                     | 4                       | ACK               | 4 (Unchanged)          | 4 (Unchanged)                         |
| Monitoring Round 2 (Tag C departs)     | Tag A | 1                     | 1                       | ACK               | 1 (Unchanged)          | 1 (Unchanged)                         |
|  | Tag B | 2                     | 2                       | ACK               | 2 (Unchanged)          | 2 (Unchanged)                         |
|  | -     | -                     | -                       | NACK              | -                      | -                                     |
|  | Tag D | 4                     | 4                       | ACK               | 3 (Changed)            | 3 (Changed)                           |
| Monitoring Round 3 (Idle slot removed) | Tag A | 1                     | 1                       | ACK               | 1 (Unchanged)          | 1 (Unchanged)                         |
|  | Tag B | 2                     | 2                       | ACK               | 2 (Unchanged)          | 2 (Unchanged)                         |
|  | Tag D | 3                     | 3                       | ACK               | 3 (Unchanged)          | 3 (Unchanged)                         |

## 6.2 Simulation Methodology

To study ResMon, a simulator is written using Matlab 7.0.4. The system consists of an RFID-enhanced sensor node with  $n$  tags in its interrogation zone. It is assumed that each node is equipped with a SkyeTek M1-Mini RFID reader [48]. The node operates from a Lithium rechargeable battery (B) that has 480 joules of energy. The tag to reader data rate is 26 kbps, as per ISO 15693 [48]. The power consumed by a RFID reader for scanning and sleeping is 180 milli-watts and 150 micro-watts respectively.

In simulation, propagation and processing delay are omitted. Further, the simulation considers a noise free channel, i.e., packet losses are due to collisions only. Finally, tag ID is 96 bits in size.

Tags are assumed to be passive, have no power source, static, and are used in read-only mode. Further, it is assumed that tags' antenna is never at 90 degrees with respect to the reader. Otherwise, tags become unreadable, and hence they do not contribute to the offered load. In other words, a reader is unaware of tags that are displaced by 90 degrees since they are not energized to participate in any communications [106] [27].

The ResMon uses eleven different commands, see Table 6.4. The duration of the *Reset and Energize* command is hardware dependent. In simulation, it is set to 1 milliseconds according to [71]. Table 6.4 also defines the duration of slots that appear in  $R_{frame}$ ,  $B_{frame}$  and  $M_{frame}$ .

The average delay experienced by the reader during the identification and monitoring phase is recorded. The identification phase comprises of a reservation followed by a body frame, whereas the monitoring phase only involves periodic transmissions of the monitor frame.

In the identification phase, the 1) average delay incurred to read a given number of tags, 2) the average delay due to collisions, and 3) the average delay due to idle slots are computed. In the monitoring phase, the time it takes to monitor a

**Table 6.4** Simulation parameters.

| Parameter          | Length (Bits) | Duration(millisecons)<br>(Length/26kbps) |
|--------------------|---------------|--|
| Reset and Energize | -             | 1.00                                     |
| Sync               | 9             | 0.35                                     |
| Read               | 13            | 0.50                                     |
| BF                 | 4             | 0.15                                     |
| MF                 | 4             | 0.15                                     |
| $A_{slot}$         | 9             | 0.35                                     |
| ACK                | 6             | 0.23                                     |
| NACK               | 6             | 0.23                                     |
| Null               | 3             | 0.12                                     |
| Unmute             | 6             | 0.23                                     |
| SO                 | 9             | 0.35                                     |
| Reservation slot   | 10            | 0.38                                     |
| Body frame slot    | 96            | 3.69                                     |
| Monitor frame slot | 4             | 0.15                                     |

given tag set when a fixed number of tags depart from the reader's interrogation zone is computed.

After computing delays, the energy consumed can be computed by multiplying the delay with the power consumed during scanning. The total energy consumed by the reader to identify and monitor a given number of tags is then obtained by adding the energy consumption in the identification and monitoring phases along with the energy consumed due to sleeping.

To analyze the overall protocol, the frequency in which the reservation and monitor frame are transmitted is varied. Three different settings for both frames are tested, see Section 6.3.3 for details.

In the simulation studies, ResMon is compared with three framed slotted Aloha (FSA) protocols, namely Basic FSA (BFSA) [65], Dynamic FSA (DFSA) [66] [118], and Enhanced DFSA (EDFSA) [77]. Moreover, for all protocols, following two features are considered:

1. Muting, where a tag is muted after it is identified.
2. Muting and early end, where a protocol combines both muting and early end; the later feature allows a reader to close an idle slot early

## 6.3 Results

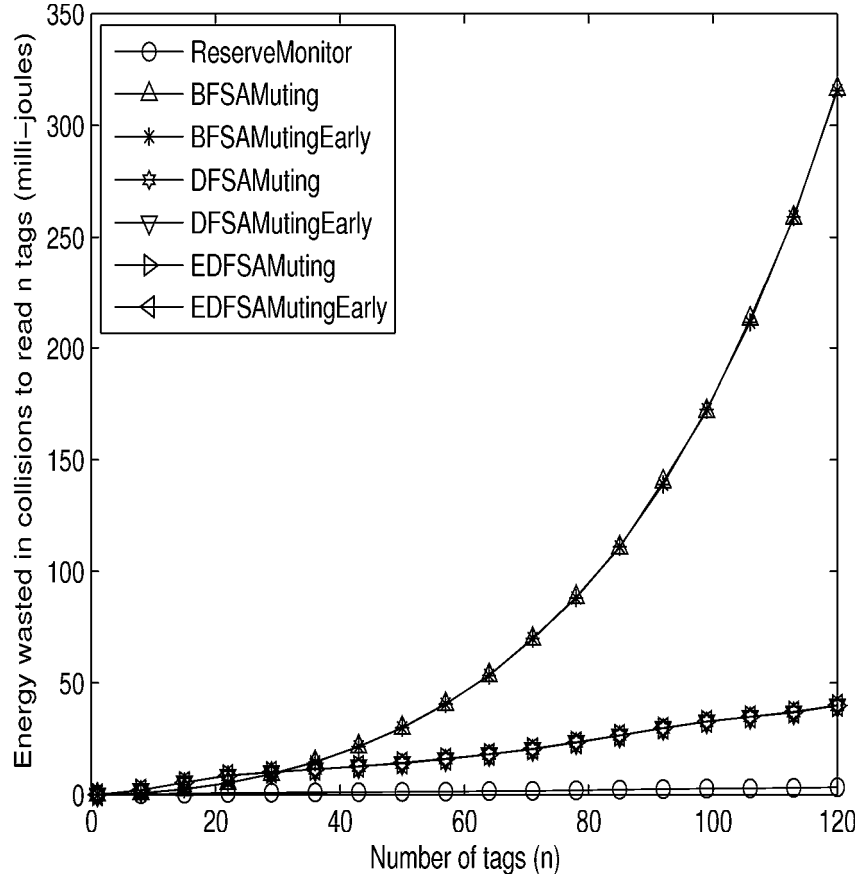
In this section, a comparison of the energy consumption of ResMon with six FSA variants using the simulation settings presented in Section 6.2. Firstly, results concerning the identification phase is presented, followed by the energy consumption incurred in the monitoring phase when a fixed number of tags depart from the reader's interrogation zone. Lastly, results when the number of arriving and departing tags is varied are presented.

### 6.3.1 Identification Phase

Figure 6.5 presents the energy wasted from collisions. ResMon, labeled as 'ReserveMonitor', consumes the lowest energy compared to FSA variants. The key reason for this is the use of 0.38 ms or 10-bit reservation slots, compared to FSA variants that have a slot of duration 4.3 ms. This means, every collision in FSA variants is approximately 10 times longer, thereby consumes ten times more energy. As the number of tags increases, the energy consumption of BFSMA variants increases exponentially. This is because they use a fixed frame size, which has a high collision probability when the number of tags exceeds the frame size. Lastly, DFSA and EDFSA have lower energy wastage from collisions compared to BFSMA variants because they use varying frame sizes to reduce the probability of collisions. Nevertheless, they consume more energy than ReserveMonitor, due to their longer slot duration.

Figure 6.6 presents the energy consumed by idle slots. Here, BFSAMutingEarly wastes the least amount of energy. This is because of two reasons. Firstly, the early-end feature results in a significant reduction in energy wastage associated with idle listening. Secondly, because of the use of a fixed frame, the probability of having an idle slot reduces as the number of tags increases. ReserveMonitor has a slightly higher energy consumption than BFSAMonitorEarly because it is difficult to set a frame size that reduces both collision and idle slots simultaneously. Lastly, DFSAMuting and EDFSAMuting have the highest energy wastage due to idle listening because each idle slot is 4.3 ms in length.



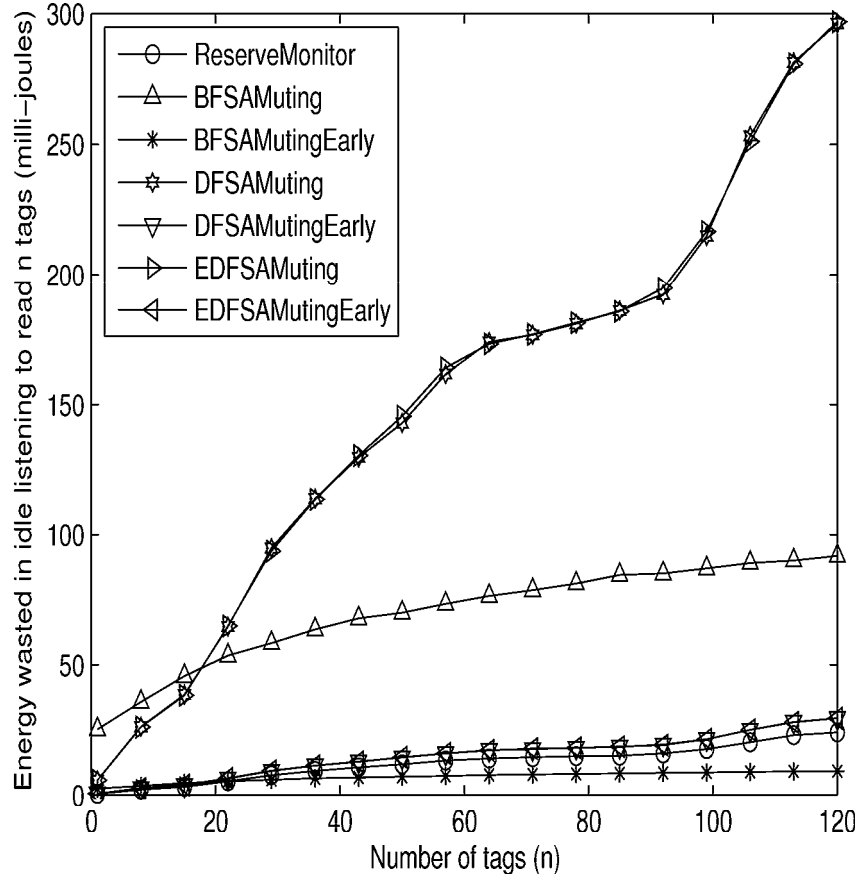


**Figure 6.5** Energy wasted in collisions during the identification phase.

Figure 6.7 shows the energy consumption to read a given number of tags successfully. ReserveMonitor consumes the lowest energy compared to FSA variants. This is because of the use of small reservation slots, which reduces the energy wastage due to idle listening as well as collisions.

### 6.3.2 Monitoring Phase

Figure 6.8 compares the energy consumption when FSA variants are used for monitoring. The energy consumed by ReserveMonitor is significantly less than FSA protocols. This is because each identified tag has a monitor frame with a unique 1.5 ms response slot, where they transmit a predefined 4-bit bit-string. On the other hand, FSA variants have to re-identify all tags to determine

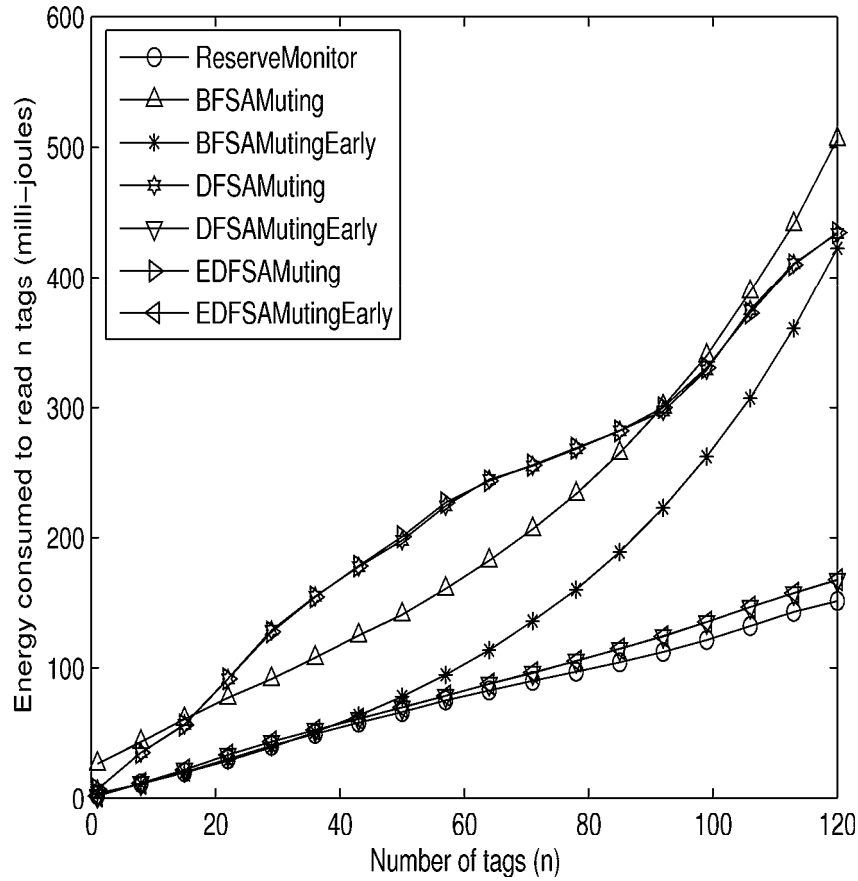


**Figure 6.6** Energy consumed to read  $n$  tags during the identification phase.

whether certain tags have been removed from the reader's interrogation zone. The energy consumption of each protocol reduces as tags are removed after every monitor round because there are less contention, hence tags can be identified quicker.

### 6.3.3 Realistic Scenarios

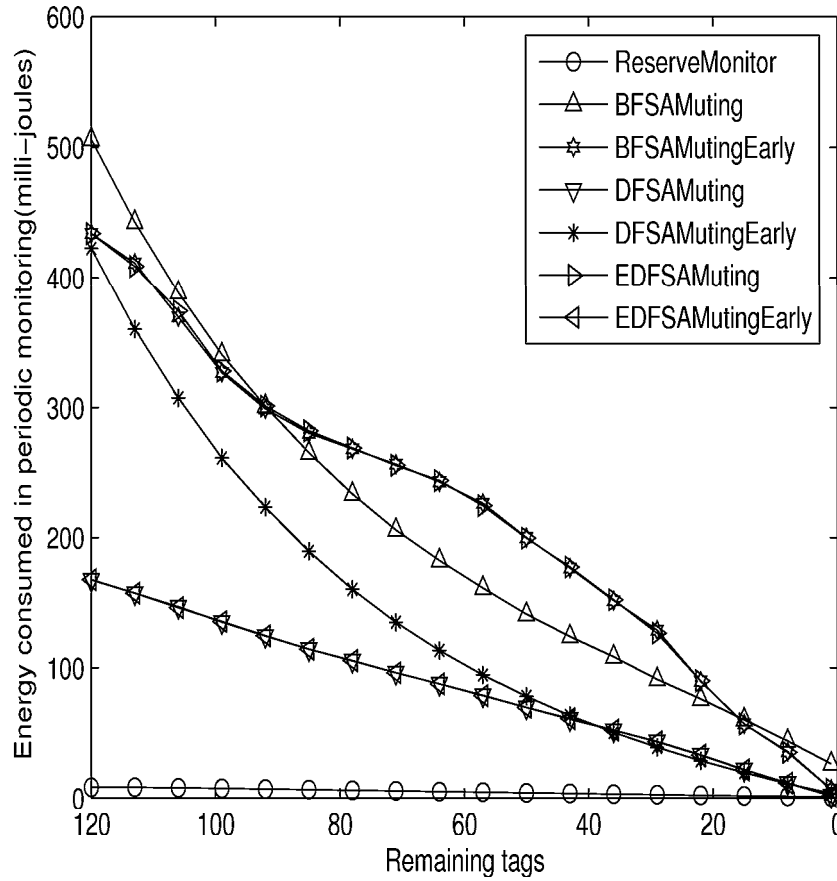
In this section, the energy consumption and battery lifetime of the reader when it is used to monitor a given region where tags arrive and depart randomly is computed. Three experiments are performed where the frequency of identification and monitoring rounds is varied. In the first scenario, an identification round and a monitor round is performed daily, with half a day sleep time be-



**Figure 6.7** Energy consumed to read  $n$  tags during the identification phase.

tween them. In the second scenario, both identification and monitoring rounds are transmitted hourly with a half an hour gap between them. Lastly, both frames are transmitted half hourly with a 15 minutes gap. Initially, there are 10 tags in the reader's interrogation zone. A random number of tags depart from the reader's interrogation zone before the arrival of the monitor frame. Once a monitor frame finishes, more random number of tags depart from the reader's interrogation zone. The simulation ends when the reader finishes its battery.

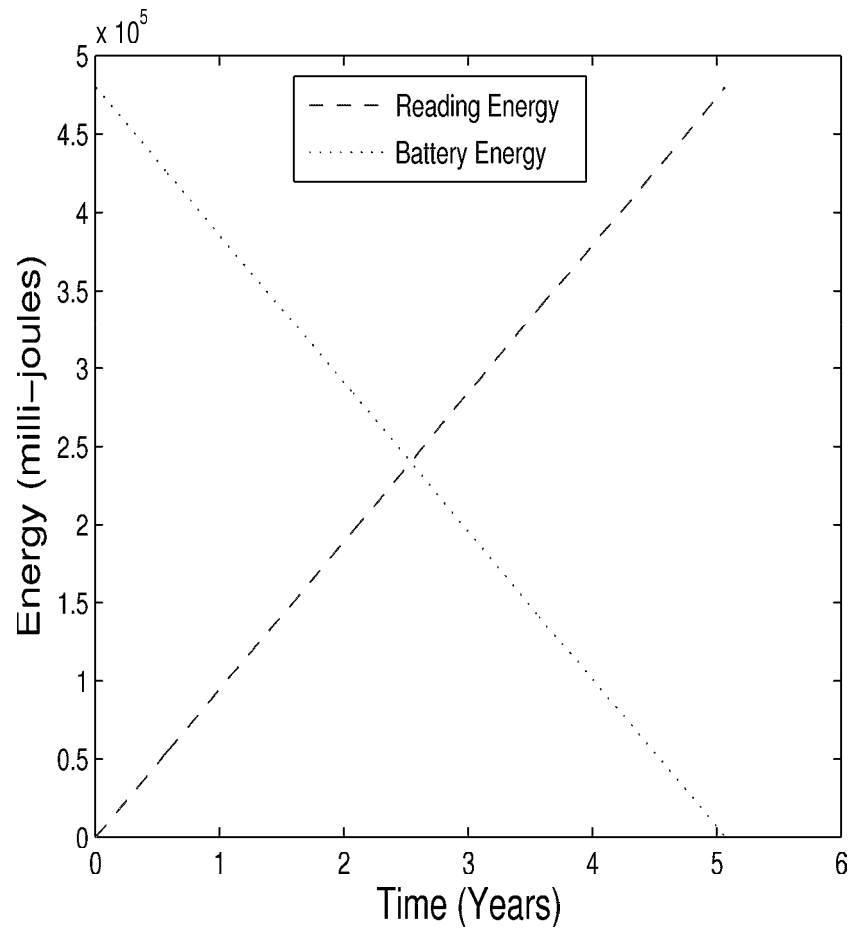
From Figure 6.9, if the reservation frame and monitor frame are transmitted daily, the reader has a lifetime of 5.1 years. On the other hand, if the reservation



**Figure 6.8** Energy consumed while monitoring a given tag set.

frame and monitor frame are transmitted with an hourly sleep period, then the reader finishes its battery in 8.4 months, as shown in Figure 6.10. Lastly, as shown in Figure 6.11, if the respective frames are transmitted every half hour, the reader is only able to operate for 138 days or 4.6 months. Table 6.5 summarizes these results; on average, for each scenario, the number of tags is approximately 200.

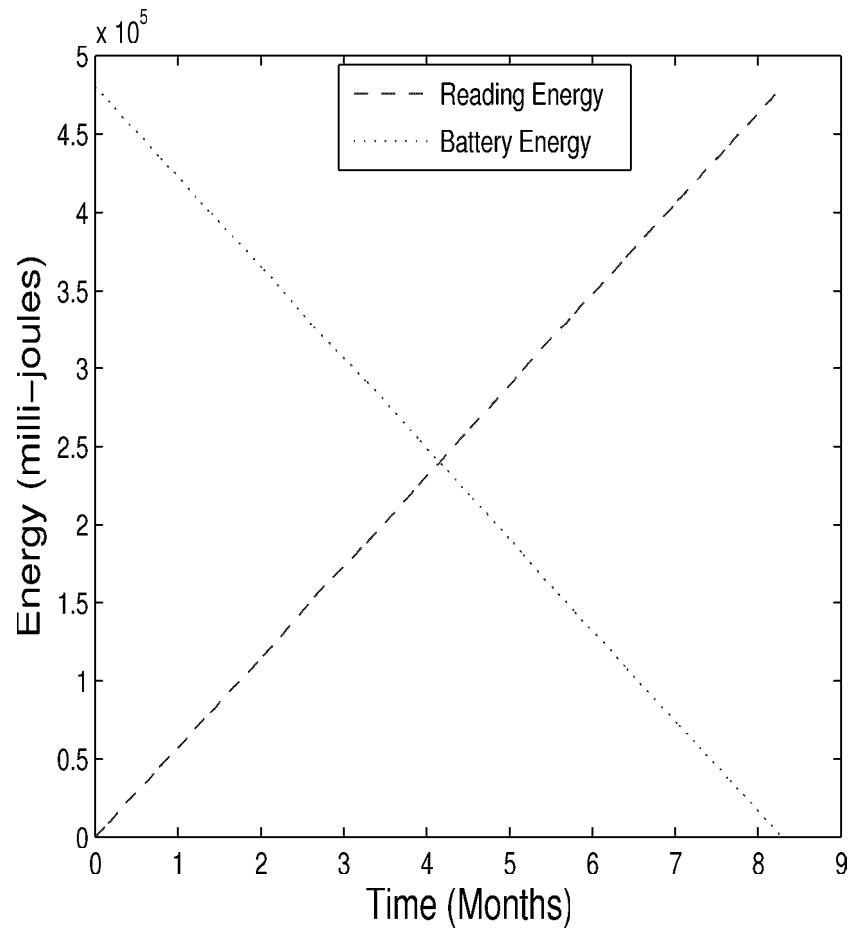
From above results, it is clear that the frequency of reservation and monitor frames significantly affect the reader's energy consumption. Hence, applications need to ensure the correct frequency is used to track tagged items; i.e., one that balances energy usage and application requirements.



**Figure 6.9** Frequency of identification and monitoring round = One Day.

## 6.4 Conclusion

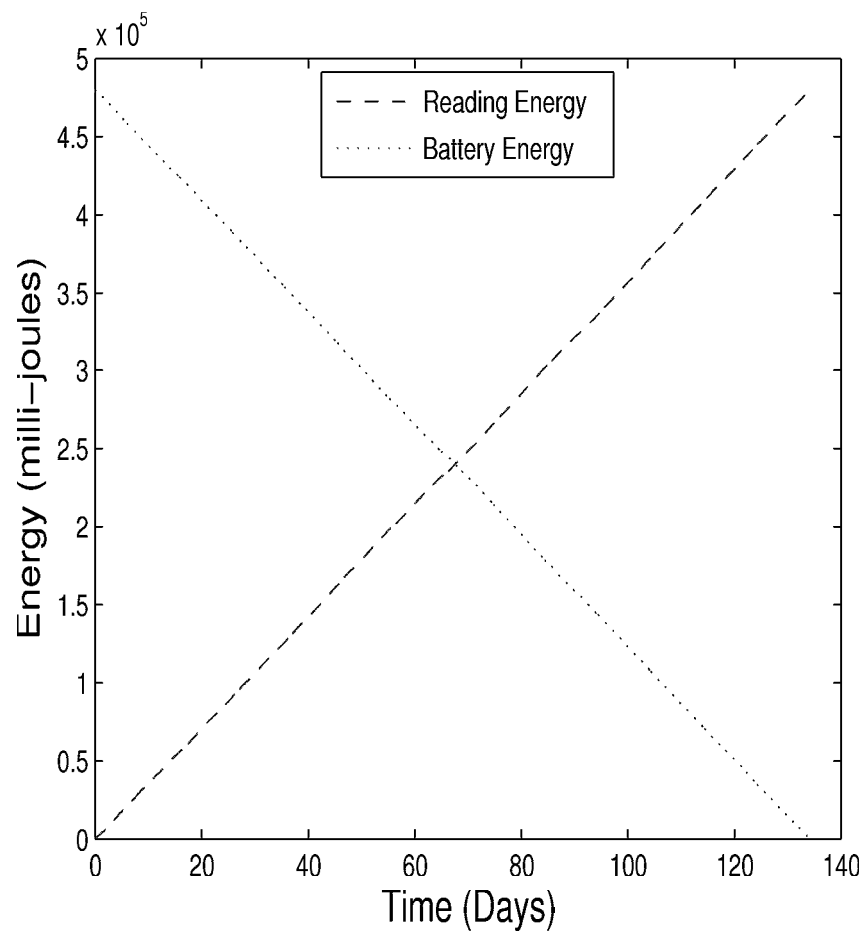
ResMon is a DFSA based protocol that uses a reservation frame for slot allocation, a body frame for tag identification, and a monitor frame to monitor RFID tags. Simulation results show that the protocol is significantly more energy efficient than existing FSA protocols, primarily because it uses small reservation and monitor slots, and it removes idle slots quickly. Moreover, it resolves collisions promptly. Thereby, making it suitable for use in RFID-enhanced WSNs.



**Figure 6.10** Frequency of identification and monitoring round = One Hour.

**Table 6.5** Summary of results involving realistic scenarios.

| Figure | Reservation Frame Frequency | Monitor Frame Frequency | Mean number of tags | Battery Lifetime |
|--------|-----------------------------|-------------------------|---------------------|------------------|
| 6.9    | 1 Day                       | 1 Day                   | 199.35              | 5.1 Years        |
| 6.10   | 1 Hour                      | 1 Hour                  | 202.75              | 8.4 Months       |
| 6.11   | 30 Minutes                  | 30 Minutes              | 201.68              | 138 Days         |



**Figure 6.11** Frequency of identification and monitoring round = Half Hour.

## Conclusions

This thesis has conducted a comprehensive study on the energy consumption of RFID anti-collision protocols. Such a study is needed to determine whether existing anti-collision protocols are suitable for next generation WSNs that have RFID tag reading capabilities. As shown in Chapter 1, reading a single tag with a 96 bit ID using Skyetek's RFID readers consume 162 to 1188 micro-joules of energy. On the other hand, a sensor node consumes 67.5 micro joules and 130 micro-joules when receiving and transmitting 96-bit of data respectively. Clearly, the energy consumed by an RFID reader is higher than a sensor node transmitting 96 bits of data. Moreover, in practice, a reader has multiple tags in its interrogation zone, which may respond simultaneously to a reader's read requests, resulting in collisions and wastage of energy. Therefore, anti-collision protocols are critical to the operation of RFID readers. In particular, they govern a reader's energy expenditure. To this end, this thesis contributes to the understanding of anti-collision protocols' energy expenditure in the following ways.

Chapter 3 evaluates the energy consumption of twelve pure and slotted Aloha variants. Analytical results show that Pure Aloha, when used in combination with fast mode and muting, consumes the lowest energy among these twelve variants. Specifically, fast mode and muting minimizes collisions significantly and therefore the protocols using both features have the least energy wastage;



fast mode has a bigger impact on energy usage because it reduces the vulnerability period significantly.

Chapter 4 studies tag estimation functions. In particular, it quantifies and compares the accuracy of five tag estimation functions using descriptive statistics. From Monte-Carlo simulations, an estimation function proposed by Vogt [69], which is based on Chebychev's inequality, achieves the best accuracy for a wide range of tag population. On the other hand, a function proposed by Cha et al. [66] for muting based environments is more accurate when the number of tags increases beyond the current frame size. These results are critical to the operation of framed slotted Aloha based tag reading protocols because they have a significant impact on the frame size used to read tags. A sub-optimal frame size results in more collisions or idle slots, both of which have a significant impact on energy expenditure.

Chapter 5 presents a qualitative and quantitative energy consumption analysis of twelve framed Aloha variants. A distinguishing feature of these variants is the use of a tag estimation function. Extensive simulation studies show that DFSA with muting and early end to have the lowest energy consumption compared to other framed Aloha variants.

Overall, collision is the key cause of energy wastage in Aloha variants. Idle listening also wastes significant energy when the number of tags is low. More importantly, none of the Aloha variants promises energy efficient monitoring of RFID tags. This is because monitoring can only be achieved by re-reading all tags, hence are subjected to collisions and idle listening, both of which waste energy. Moreover, the problem becomes critical when tag population changes frequently.

In light of the aforementioned results, there is a clear need for anti-collision protocols that have high identification rates as well as energy efficient monitoring capabilities. In this respect, this thesis is the first to propose such a protocol: called ResMon. The results in Chapter 6 show that ResMon achieves 11% and 90% improvement in terms of energy saved when compared to state of the art

framed Aloha protocols. On the negative side, ResMon has a higher complexity than existing Aloha variants. In addition, tags are required to have two separate random number generators for slot selection and pseudo ID generation respectively.

A key future research direction with respect to ResMon is to extend its capabilities to multi-hop settings. For example, ResMon can be integrated with a self organizing protocol to create a sensor network that allows collaborative tag reading. Moreover, such networks can be used to mitigate the tag orientation problem. The observation here is that given the number of deployed sensor nodes, it is likely that one of them will be better oriented to read tags that otherwise would be unreadable if there is only one reader.

Lastly, there is a need to investigate hybrid protocols. In other words, those that combine Aloha and tree protocols, for example [101] [102] [104]. Most hybrid protocols combine the query tree protocol with an Aloha variant. This is because query tree helps a reader separates tags into smaller groups, thereby lowering contention. Each group can then be read using a tree or an Aloha variant. The results in [104] show that hybrid protocols consume lower energy than the QT protocol. Moreover, in [101], the authors show that TSA achieves a higher system efficiency as compared to DFSA, EDFSA, QT, and QT with an aggressive enhancement when there are more than 60 tags. These results validate the use of hybrid protocols in RFID-enhanced WSNs. However, further research is required before they can be used to achieve energy efficient identification and monitoring.

# Bibliography

- [1] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu, “Wireless Sensor Networks: a Survey on the State of the Art and the 802.15.4 and ZigBee Standards,” *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.
- [2] Sentilla, “Tmote Sky.” Datasheet. <http://moteiv.com/products/docs/tmote-sky-datasheet.pdf>.
- [3] Crossbow, “UCB Mica2 Mote Power Benchmark Summary Numbers.” Webpage. <http://www.eecs.harvard.edu/shnayder/ptossim/mica2bench/summary.html>.
- [4] Crossbow, “MICA2/DOT Professional Kit (MOTE-KIT 5x4x).” Webpage. <http://www.xbow.com/Products/productsdetails.aspx?sid=69>.
- [5] Crossbow, “Micaz.” Webpage. [http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/MICAz\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf).
- [6] E. Zurich, “BTnodes.” Webpage. <http://www.btnode.ethz.ch/>.
- [7] E. Zurich, “Distributed Systems Group.” Webpage. <http://www.vs.inf.ethz.ch/>.
- [8] Crossbow, “IRIS Datasheet.” Webpage. [http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/IRIS\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf).

- 
- [9] Crossbow, "Imote Datasheet." Webpage.  
[http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/Imote2\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf).
- [10] Crossbow, "TELOSB Datasheet." Webpage.  
[http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/TelosB\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf).
- [11] Jae-Hyun Kim, "Multi-reader/multi-tag anti-collision."  
[www.krnet.or.kr/board/include/download.asp?no=3&db=program&fileno=2](http://www.krnet.or.kr/board/include/download.asp?no=3&db=program&fileno=2).
- [12] E. Zurich and L. A. Burdet, "RFID Multiple Access Methods." Technical Report. [http://www.vs.inf.ethz.ch/edu/SS2004/DS/reports/06\\_rfid-mac\\_report.pdf](http://www.vs.inf.ethz.ch/edu/SS2004/DS/reports/06_rfid-mac_report.pdf).
- [13] D. Baddeley, "ISO/IEC FCD 14443-3."  
<http://www.waazaa.org/download/fcd-14443-3.pdf>.
- [14] ISO, "RFID Standards." [http://www.iso.org/iso/search.htm?qt=RFID&published=on&active\\_tab=standards](http://www.iso.org/iso/search.htm?qt=RFID&published=on&active_tab=standards).
- [15] OTI, "ISO 14443." <http://www.otiglobal.com/objects/ISO%2014443%20WP%204.11.pdf>.
- [16] A. Technology, "EPCglobal class 1 gen 2 RFID specifications." Whitepaper. [http://www.alientechnology.com/docs/AT\\_wp\\_EPCGlobal\\_WEB.pdf](http://www.alientechnology.com/docs/AT_wp_EPCGlobal_WEB.pdf).
- [17] Auto-ID Center, "13.56 MHz ISM band class 1 radio frequency identification tag interface specification, version 1.0." HF RFID standard.
- [18] EPCglobal, "Electronic product code."  
<http://www.epcglobalinc.org/home/>.
- [19] NXP Semiconductors, "I Code." [http://www.nxp.com/acrobat\\_download/other/identification/sl040616.pdf](http://www.nxp.com/acrobat_download/other/identification/sl040616.pdf).
- [20] NXP Semiconductors, "Ucode." Datasheet.  
[http://www.nxp.com/acrobat\\_download/literature/9397/75014922.pdf](http://www.nxp.com/acrobat_download/literature/9397/75014922.pdf).

- 
- [21] NXP, “MIFARE.” [http://en-origin.nxp.com/#/homepage/cb=\[t=p,p=/53420/53422\]—pp=\[t=pf,i=53422\]](http://en-origin.nxp.com/#/homepage/cb=[t=p,p=/53420/53422]—pp=[t=pf,i=53422]).
- [22] RFID Journal, “DOD Releases Final RFID Policy.” <http://www.rfidjournal.com/article/articleview/1080/1/1>.
- [23] RFID Journal, “Wal-Mart Begins RFID Process Changes.” <http://www.rfidjournal.com/article/articleview/1385>.
- [24] R. Want, “An Introduction to RFID Technology,” *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.
- [25] Raghu Das and Peter Harrop, “Complete RFID Analysis and Forecasts 2008-2018.” <http://www.idtechex.com/>.
- [26] In-Stat, “Explosive Growth Projected in Next Five Years for RFID Tags.” <http://www.instat.com>.
- [27] K. Finkenzeller, *RFID Handbook , Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley and Sons Ltd, 2003.
- [28] S. Lahiri, *RFID Sourcebook*. USA: IBM Press, 2005.
- [29] R. Weinstein, “RFID: A Technical Overview and its Application to the Enterprise,” *IT Professional*, vol. 7, no. 3, pp. 27–33, 2005.
- [30] L. Ho, M. Moh, Z. Walker, T. Hamada, and C.-F. Su, “A Prototype on RFID and Sensor Networks for Elder Healthcare: Progress Report,” in *Proceeding of the 2005 ACM SIGCOMMWorkshop on Experimental Approaches to Wireless Network Design and Analysis*, (Philadelphia, Pennsylvania, USA), pp. 70–75, ACM Press, 2005.
- [31] S. Chen and V. Thomas, “Optimization of Inductive RFID Technology,” in *Proceedings of the IEEE International Symposium on Electronics and the Environment*, (Denver, CO, USA), pp. 82–87, 2001.
- [32] IDTechEX, “Chipless RFID Tags.” <http://www.idtechex.com/products/en/articles/00000435.asp>.

- 
- [33] I.F.Akyildiz, Y. S. W.Su, and E.Cayirci, "Wireless Sensor Networks: A Survey," *The International Journal of Computer and Telecommunications Networking*, vol. 38, no. 4, pp. 393–422, 2002.
- [34] M. Ilyas and I. Mahgoub, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. Taylor and Francis Group, 1970.
- [35] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 2–16, 2003.
- [36] Berkeley, "Chipcon CC2420 Datasheet." Webpage. <http://inst.eecs.berkeley.edu/cs150/Documents/CC2420.pdf>.
- [37] Texas Instruments, "Chipcon CC1000 datasheet." Webpage. <http://focus.ti.com/lit/ds/symlink/cc1000.pdf>.
- [38] SkyeTek, "SkyeTek RFID Reader." Webpage. [www.skyetek.com](http://www.skyetek.com).
- [39] S. K. Das and D. J. Cook, "Designing and Modeling Smart Environments (Invited Paper)," in *Proceedings of the International Symposium on on World of Wireless, Mobile and Multimedia Networks*, (Niagara-Falls, Buffalo-NY), pp. 490–494, 2006.
- [40] B. Lee and H. Kim, *A Design of Context aware Smart Home Safety Management using by Networked RFID and Sensor*. USA: Springer, 2008.
- [41] S.-H. Baeg, J.-H. Park, J. Koh, K.-W. Park, and M.-H. Baeg, "Building a Smart Home Environment for Service Robots Based on RFID and Sensor Networks," in *International Conference on Control, Automation and Systems*, (Niagara-Falls, Buffalo-NY), pp. 1078–1082, 2007.
- [42] M. RFID, "New RFID Sensor Network to Monitor Bushfire & Wildfire Ignitions." Webpage. [http://www.morerfid.com/details.php?subdetail=Report&action=details&report\\_id=2916&display=RFID](http://www.morerfid.com/details.php?subdetail=Report&action=details&report_id=2916&display=RFID).

- 
- [43] Telepathx, “Smart Grid Intelligent Insulators Current Leakage Sensors.” Webpage. <http://www.telepathx.com/solutions.htm>.
- [44] S. Saha and M. Matsumoto, “A Framework for Disaster Management System and WSN Protocol for Rescue Operation,” in *TENCON IEEE Region 10 Conference*, (Taipei, Taiwan), pp. 1–4, 2007.
- [45] Intel, “Intel Sensor Nets/RFID Research Areas.” Webpage. [http://www.intel.com/research/exploratory/wireless\\_sensors.htm](http://www.intel.com/research/exploratory/wireless_sensors.htm).
- [46] Jonathan Collins, “NASA Creates Thinking RF Sensors.” [http://www.sensorwaresystems.com/historical/press/RFIDJrnl\\_NASA\\_Oct04.pdf](http://www.sensorwaresystems.com/historical/press/RFIDJrnl_NASA_Oct04.pdf).
- [47] M. Roberti, “BP Leads the Way on Sensors.” Webpage. <http://www.rfidjournal.com/article/articleview/1216/1/2/>.
- [48] Skyetek, “Skyemodule M1-Mini.” Datasheet. [http://www.skyetek.com/Portals/0/SkyeModule\\_M1Mini\\_060426.pdf](http://www.skyetek.com/Portals/0/SkyeModule_M1Mini_060426.pdf).
- [49] Skyetek, “Skyemodule M1.” Datasheet. [http://www.skyetek.com/Portals/0/SkyeModule\\_M1\\_060426.pdf](http://www.skyetek.com/Portals/0/SkyeModule_M1_060426.pdf).
- [50] PDC, “DR1000 dual RFID/bar code reader.” Datasheet. [http://www.pdcorp.com/healthcare/rfid\\_products.html](http://www.pdcorp.com/healthcare/rfid_products.html).
- [51] Socket, “Socket RFID reader card.” Datasheet. [http://www.communica.se/socket/rfid\\_datasheet.pdf](http://www.communica.se/socket/rfid_datasheet.pdf).
- [52] T. Sense, “Micro-1356 multi-protocol reader.” Datasheet. <http://www.tagsense.com/ingles/products/products/Micro-1356-v2.PDF>.
- [53] Panasonic, “ZU-1870 series and ZU-9A series.” Webpage. <http://www.panasonic.com/industrial/other/>.
- [54] RF-ID.com, “Cf reader.” Webpage. <http://www.rf-id.com/2rfid13.56/cfreader.htm>.

- 
- [55] Innovision-group, “Low-cost 13.56MHz (RFID) Reader from IO.” Datasheet. [http://www.innovision-group.com/images/io\\_Technical\\_Datasheet\\_1v1.pdf](http://www.innovision-group.com/images/io_Technical_Datasheet_1v1.pdf).
- [56] Brooks-Automation, “CFCard transponder reader.” Webpage. [http://www.ready-for-rfid.com/rfid\\_index.php?menu=lf40semi&spr=e#29](http://www.ready-for-rfid.com/rfid_index.php?menu=lf40semi&spr=e#29).
- [57] SkyeTek, “Skyemodule M8.” Datasheet. [http://www.skyetek.com/Portals/0/SkyeModule\\_M8\\_060524.pdf](http://www.skyetek.com/Portals/0/SkyeModule_M8_060524.pdf).
- [58] Tag Sense, “Nano-UHF RFID reader.” Datasheet. <http://www.tagsense.com/ingles/products/products/Nano-UHF.pdf>.
- [59] D. H. Shih, P. L. Sun, D. C. Yen, and S. M. Huang, “Taxonomy and Survey of RFID Anti-collision Protocols: Short Survey,” *Computer Communications*, vol. 29, no. 11, pp. 2150–2166, 2006.
- [60] A. Rohatgi and G. D. Durgin, “RFID Anti-Collision System Using the Spread Spectrum Technique,” in *Technical Report*, pp. 1–19, 2005. [http://www.propagation.gatech.edu/Archive/PG\\_TR\\_050425\\_AR/PG\\_TR\\_050425\\_AR.pdf](http://www.propagation.gatech.edu/Archive/PG_TR_050425_AR/PG_TR_050425_AR.pdf).
- [61] Z. Tang and Y. He, “Research of multi-access and anti-collision protocols in RFID systems,” in *IEEE International Workshop on Anti-counterfeiting, Security, Identification*, (Xiamen, China), pp. 377–380, 2007.
- [62] EM Microelectronics, “Supertag category protocols.” Datasheet. <http://www.gaw.ru/doc/EM-Marin/P4022.PDF>.
- [63] D. K. Klair, K.-W. Chin, and R. Raad, “An Investigation into the Energy Efficiency of Pure and Slotted Aloha Based RFID Anti-Collision Protocols,” in *In IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM’07)*, (Helsinki, Finland), IEEE, 2007.



- 
- [64] M. Schwartz, *Telecommunication Networks Protocols, Modeling and Analysis*. USA: Addison-Wesley, 1988.
  - [65] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for Multiple RFID Objects Identification," *IEICE-Transactions on Communications*, vol. E88-B, pp. 991–999, 2005.
  - [66] J.-R. Cha and J.-H. Kim, "Novel Anti-collision Algorithms for Fast Object Identification in RFID System," in *The 11th Intl. Conf. on Parallel and Distributed Systems*, (Fukuoka, Japan), pp. 63–67, July 2005.
  - [67] T.-W. Hwang, B.-G. Lee, Y. S. Kim, D. Y. Suh, and J. S. Kim, "Improved Anti-collision Scheme for High Speed Identification in RFID System," in *Proceedings of the First International Conference on Innovative Computing, Information and Control*, (China), pp. 449–452, 2006.
  - [68] J. Wang, Y. Zhao, and D. Wang, "A novel fast anti-collision algorithm for RFID systems," in *International Conference on in Wireless Communications, Networking and Mobile Computing (WiCom 2007)*., (Shanghai, China), pp. 2044–2047, Sept 2007.
  - [69] H. Vogt, "Multiple Object Identification with Passive RFID Tags," in *IEEE Intl. Conf. on Man and Cybernetics*, (Tunisia), pp. 6–13, Oct 2002.
  - [70] H. Vogt, "Efficient Object Identification with Passive RFID Tags," in *IEEE PerCom*, (TX, USA), 2002.
  - [71] G. Khandelwal, A. Yener, K. Lee, and S. Serbetli, "ASAP: a MAC protocol for dense and time constrained RFID systems," in *IEEE International Conference on Communications (ICC'06)*, (Istanbul, Turkey), 2006.
  - [72] C. Floerkemeier and M. Wille, "Comparison of Transmission Schemes for Framed Aloha based RFID Protocols," in *Proceedings of the International Symposium on Applications on Internet Workshops*, (Phoenix, AZ, USA), 2006.

- 
- [73] C. Floerkemeier, "Transmission control scheme for fast RFID object identification," in *The 4th Annual Intl. Conference on Pervasive Computing and Communications Workshops*, (Pisa, Italy), 2006.
  - [74] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," in *SIGMOBILE: ACM Special Interest Group on Mobility of Systems, Users, Data and Computing*, pp. 322–333, 2006.
  - [75] W.-T. Chen and G.-H. Lin, "An Efficient Anti-Collision Method for RFID System," *IEICE Trans. Commun.*, vol. E89, no. B, pp. 3386–3392, 2006.
  - [76] W. Feller, *An Introduction to Probability Theory and its Applications*. New York: Addison-Wesley, 1970.
  - [77] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An Enhanced Dynamic Framed Slotted Aloha Algorithm for RFID Tag Identification," in *The 2nd Intl. Annual Conference on Mobile and Ubiquitous Systems: Networking and Services*, (San Diego, USA), pp. 166–172, 2005.
  - [78] D. K. Klair, K.-W. Chin, and R. Raad, "On the Suitability of Framed Aloha Based RFID Anti-Collision Protocols for RFID-Enhanced WSNs," in *The 16th IEEE International Conference on Computer Communications and Networks (IEEE ICCCN'2007)*, (Honolulu, Hawaii, USA), 2007.
  - [79] F. C. Schoute, "Dynamic Frame Length Aloha," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 565–568, 1983.
  - [80] C. J. "Tree Algorithms for Packet Broadcast Channels," *Information Theory, IEEE Transactions on*, vol. 25, no. 5, pp. 505–515, 1979.
  - [81] D. R. Hush and C. Wood, "Analysis of Tree Algorithms for RFID Arbitration," in *The IEEE Intl. Symposium on Information Theory*, (Mexico City, USA), pp. 107–107, 1998.
  - [82] J. Myung and W. Lee, "Adaptive Binary Splitting: A RFID Tag Collision Arbitration Protocol for Tag Identification," in *IEEE BROADNETs*, (Boston, MA, USA), pp. 347–355, Oct. 2005.

- 
- [83] J. Myung, W. Lee, and J. Srivastava, "Adaptive Binary Splitting for Efficient RFID Tag Anti-collision," *IEEE Personal Communications Magazine*, vol. 10, no. 3, pp. 144–146, 2006.
  - [84] W.-C. Chen, S.-J. Horng, and P. Fan, "An enhanced anti-collision algorithm in rfid based on counter and stack," in *Second International Conference on in Systems and Networks Communications, (ICSNC 2007).*, pp. 21–24, 2007.
  - [85] C. Law, K. Lee, and K.-Y. Siu, "Efficient memoryless protocol for tag identification (extended abstract)," in *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, (Toronto, CA), pp. 75–84, Aug. 2000.
  - [86] H.-S. Choi, J.-R. Cha, and J.-H. Kim, "Fast wireless anti-collision algorithm in ubiquitous ID system," in *The 60th IEEE Vehicular Technology Conference*, pp. 4589–4592, 2004.
  - [87] J. Myung, W. Lee, and T. Shih, "An Adaptive Memoryless Protocol for RFID Tag Collision Arbitration," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1096–1101, 2006.
  - [88] J. Myung and W. Lee, "An Adaptive Memoryless Tag Anti-collision Protocol for RFID Networks," in *the 24th IEEE Annual Conference on Computer Communications (INFOCOM 2005)*, (Miami, USA), Mar. 2005.
  - [89] 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 2004 international symposium on Low power electronics and design*, pp. 357–362, 2004.
  - [90] J. H. Choi, D. Lee, and H. Lee, "Query tree-based reservation for efficient RFID tag anti-collision," *IEEE Communications Letters*, vol. 11, no. 1, pp. 85–87, 2007.

- 
- [91] M. A. Bonuccelli, F. Lonetti, and F. Martelli, "Randomized Hashing for Tag Identification in RFID Networks," in *Technical Report: Computer Communications and Networks*, 2005.
  - [92] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent Query Tree (IQT) Protocol to Improve RFID Tag Read Efficiency," in *9th International Conference on Information Technology (ICIT'06)*, pp. 46–51, 2006.
  - [93] S. Yu, Y. Zhan, Z. Wang, and Z. Tang, "Anti-collision algorithm based on jumping and dynamic searching and its analysis," *Computer Engineering*, vol. 31, pp. 19–20, 2005.
  - [94] L. Liu, Z. Xie, J. Xi, and S. Lai, "An Improved Anti-collision Algorithm in RFID Aystem," in *2nd International Conference on Mobile Technology, Applications, and Systems*, pp. 1–5, 2005.
  - [95] F. Bo, L. Jin-Tao, G. Jun-Bo, and D. Zhen-Hua, "ID-Binary Tree Stack Anticollision Algorithm for RFID," in *Proceedings of the 11th IEEE Symposium on Computers and Communications*, (Sardinia, Italy), pp. 207–212, Apr. 2006.
  - [96] M. Jacomet, A. Ehram, and U. Gehrig, "Contact-less Identification Device with Anti-Collision Algorithm," in *Conference on Circuits Systems, Computers and Communications*, (Athens, Greece), July 1999.
  - [97] S. H. Kim, M. K. Shin, and P. Park, "A new tree-based tag anti-collision protocol for RFID systems," in *Communications in Computing*, pp. 83–86, 2006.
  - [98] S. H. Kim and P. Park, "An Efficient Tree-Based Tag Anti-Collision Protocol for RFID Systems," *IEEE Communications Letters*, vol. 11, no. 5, pp. 449–451, 2007.
  - [99] Chris Turner, "Deterministic or Probabilistic Which is Best?." Weblink. [//www.rfipsolutions.com/downloads/Deterministic\\_or\\_Probabilistic.pdf](http://www.rfipsolutions.com/downloads/Deterministic_or_Probabilistic.pdf).

- 
- [100] Chris Turner, "Evaluating the Merits of Various RFID Tagging Protocols with respect to the Protocol Characteristics." Technical Report. <http://rfip.eu/paper.php?pid=tech:9>.
- [101] M. A. Bonuccelli, F. Lonetti, and F. Martelli, "Tree Slotted Aloha: a New Protocol for Tag Identification in RFID Networks," in *International Symposium on on World of Wireless, Mobile and Multimedia Networks*, (Niagara-Falls, USA), pp. 603–608, 2006.
- [102] J. Ryu, H. Lee, Y. Seok, T. Kwon, and Y. Choi, "A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID systems," in *IEEE ICC*, (Scotland), pp. 5981–5986, June 2007.
- [103] J.-D. Shin, S.-S. Yeo, T.-H. Kim, and S. K. Kim, *Hybrid Tag Anti-collision Algorithms in RFID Systems*. UK: Springer Berlin / Heidelberg, 2007.
- [104] V. Namboodiri and L. Gao, "Energy-Aware Tag Anti-Collision Protocols for RFID Systems," in *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, (NY, USA), pp. 23–46, 2007.
- [105] H. Zhang, L. Han, and Y.-L. Li, "Design of Hash-Tree Anti-collision Algorithm," in *Third International Conference on Natural Computation ICNC 2007*, pp. 176–179, 2007.
- [106] Steve Lewis, "Think RFID Technology." <http://hosteddocs.ittoolbox.com/laran032604.pdf>, 2004.
- [107] L. Kleinrock and S. Lam, "Packet switching in multiaccess broadcast channel: Performance evaluation," *IEEE Transactions on Communications*, vol. 23, pp. 410–423, 1975.
- [108] L. Kleinrock and S. lam, "Packet-switching in a slotted satellite channel," in *AFIPS Conference Proceedings*, pp. 703–710, 1973.
- [109] V. Namboodiri and L. Gao, "Energy-aware tag anti-collision protocols for RFID systems," in *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (IEEE PerCom 2007)*, 2007.

- 
- [110] R. J. Fernandes, “Energy analysis of RFID arbitration protocols,” 2005. <http://rci.rutgers.edu/~rohanf/Documents/Project-553.ps>.
  - [111] 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 2004 international symposium on Low power electronics and design*, pp. 357–362, 2004.
  - [112] N. Abramson, “The throughput of packet broadcasting channels,” *IEEE Transactions on Communications*, vol. 24, no. 1, pp. 117–128, 1977.
  - [113] N. Abramson, “Developement of the alohanet,” *IEEE Transaction on Information Theory*, vol. 31, pp. 119–123, 1977.
  - [114] C. Namislo, “Analysis of mobile radio slotted aloha networks,” *IEEE Trans. on Vehicular Technology*, vol. 33, pp. 199–204, 1984.
  - [115] R. Binder, N. Abramson, F. Kuo, A. Okinaka, and D. Wax, “Aloha packet broadcasting-a retrospect.,” in *AFIPS Conf Proc*, pp. 203–216, 1975.
  - [116] R. Rom and M. Sidi, *Multiple Access Protocols Performance and Analysis*. Springer-Verlag, 1989.
  - [117] A. Ruiz, D. K. Klair, and K.-W. Chin, “A simulation study of pure and slotted aloha based RFID tag reading protocols,” in *The 6th IEEE Consumer Communications and Networking Conference (IEEE CCNC’09)*, Las Vegas, CA, USA, Jan. 2009.
  - [118] J.-R. Cha and J.-H. Kim, “Dynamic Framed Slotted Aloha Algorithms using Fast Tag Estimation Method For RFID System,” in *The 3rd IEEE Consumer Communications and Networking Conference*, (Las Vegas, Nevada, USA), pp. 768–772, Jan 2006.
  - [119] R. E. Walpole, R. H. Myers, and S. L. Myers, *Probability and Statistics for Engineers and Scientists*. Prentice Hall, 1997.

- [120] J. Wieselthier, A. Ephremides, and L. Michaels, “An exact analysis and performance evaluation of framed aloha with capture,” *IEEE Transactions on Communications*, vol. 37, no. 2, pp. 125–137, 1989.