Energy-efficient Algorithm Design for Wireless Sensor Networks

by

Vahid Shah-Mansouri

B.Sc., Electrical and Computer Engineering, University of Tehran, Iran, 2003
M.Sc., Electrical Engineering, Sharif University of Technology, Iran, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in
The Faculty of Graduate Studies
(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

August 2011

© Vahid Shah-Mansouri, 2011
Abstract

Wireless sensor networks (WSNs) are composed of inexpensive sensor devices called sensor nodes. Sensors have limited power supply, computational capabilities, and memory. Different types of sensors can measure either temperature, light, sound, or pressure from the environment. Because the sensors have short transmission range, the generated data are gathered via multihop transmissions at a central processor called a sink. In this thesis, we propose several power efficient algorithms for WSNs.

First, we formulate the lexicographically optimal commodity lifetime routing problem. We propose the lexicographically optimal node lifetime algorithm, which is suitable for practical implementation. Simulation results show that our proposed algorithm can increase the network lifetime compared to other schemes in the literature.

Second, we study the problem of supporting multicast traffic in WSNs with network coding. We formulate the maximum-lifetime minimum-resource coding subgraph problem to study the lifetime-resource tradeoff. Results show that the network lifetime can be substantially increased using our algorithm.

Next, we consider the problem of designing feedback mechanisms for WSNs using random linear network coding (RLNC). For an intermediate node, we determine the time at which the node can stop transmission of a particular flow. We propose novel link-by-link and end-to-end feedback mechanisms for RLNC with buffer sharing. Simulation results show that link-by-link feedback is more power-efficient compared to end-to-end feedback.

Then, we study the passive loss inference problem in WSNs using RLNC. By inspecting
the contents of packets at the sink, the sink can estimate the path loss rates from the sources and intermediate nodes. We propose a passive loss inference with RLNC algorithm. Simulation results show that our algorithm can identify the status of a higher number of links compared to a Bayesian inference algorithm.

Finally, we study the problem of cardinality estimation in radio frequency identification systems with several readers. We introduce exclusive estimators to estimate the number of tags located exclusively in the zone of a reader. Then, we propose cardinality estimation algorithms. Our simulation results show that the variance of our proposed estimation algorithms increases linearly with the number of readers while it increases exponentially for existing algorithms.
Preface

I am the first author and principal contributor of all chapters. All chapters are co-authored with Dr. Vincent W.S. Wong, who supervised the research. Chapter 2 is also co-authored with Dr. Amir-Hamed Mohsenian-Rad, who contributed in formulating the problem as a mixed integer programming problem.

The following publications describe the work completed in this thesis. In some cases, the conference papers contain materials overlapping with the journal papers.

Journal Papers Accepted/Published


Journal Paper Submitted


Conference Papers Published


Preface

# Table of Contents

1.4.2 Linearization Technique ........................................ 13
1.4.3 Regularization Technique .......................................... 14
1.5 Contributions and Results ........................................... 15
1.6 Thesis Organization .................................................. 18

2 Lexicographically Optimal Routing for WSNs with Multiple Sinks .... 20
  2.1 Lexicographically Optimal Commodity Lifetime (LOCL) Algorithm ........ 23
    2.1.1 System Model .................................................. 23
    2.1.2 Problem Formulation ......................................... 25
    2.1.3 LOCL Algorithm: First Step ................................ 26
    2.1.4 LOCL Algorithm: Subsequent Steps ......................... 31
  2.2 Lexicographically Optimal Node Lifetime (LONL) Algorithm .............. 34
    2.2.1 System Model .................................................. 35
    2.2.2 Distributed Implementation .................................. 37
  2.3 Performance Evaluation ............................................ 39
    2.3.1 Performance of LOCL Algorithm ............................. 39
    2.3.2 Performance Comparisons between LOCL, LONL, MLMS, and LMM Algorithms ............................................. 42
  2.4 Summary .......................................................... 44
  2.5 Analytical Proofs .................................................. 45
    2.5.1 Proof of Lemma \[ \] ............................................ 45
    2.5.2 Proof of Lemma \[ \] ............................................ 46
    2.5.3 Proof of Theorem \[ \] ......................................... 47

3 Lifetime-Resource Tradeoff for Multicast Traffic in WSNs ............... 50
  3.1 System Model ..................................................... 53
    3.1.1 System Constraints .......................................... 53
# Table of Contents

3.2 Problem Formulation .................................................................................. 59
  3.2.1 Maximum Lifetime Minimum Resource Problem ............................... 59
  3.2.2 Limit the Rate of Coding Operations ............................................... 70
  3.2.3 Limit the Number of Coding Devices .............................................. 71
  3.2.4 Minimize the Number of Coding Devices ........................................ 72

3.3 Performance Evaluation ............................................................................. 73
  3.3.1 Varying the Tuning Parameter $\gamma$ .............................................. 73
  3.3.2 Varying the Number of Sinks $|D^s|$ ............................................... 76
  3.3.3 Varying the Number of Sources $|S|$ .............................................. 77
  3.3.4 Comparison between MLMR and EMR Algorithms .......................... 78

3.4 Summary ................................................................................................. 80

4 Feedback Mechanism for RLNC with Buffer Sharing in Lossy WSNs .... 82
  4.1 System Model ......................................................................................... 86
  4.2 Transmission Rate Analysis .................................................................. 89
    4.2.1 When Can Node $i$ Stop Transmission of Flow $f$? ...................... 89
    4.2.2 Determine the Time $\tau^f_i$ ......................................................... 92
    4.2.3 Transmission Rate of Innovative Packets After Time $\tau^f_i$ .......... 94
  4.3 Feedback Mechanism Design ................................................................. 98
    4.3.1 Link-by-Link Feedback Mechanism ............................................ 98
    4.3.2 End-to-End Feedback Mechanism .............................................. 99
  4.4 Performance Evaluation ....................................................................... 102
  4.5 Summary ............................................................................................... 107

5 Link Loss Inference in WSNs with RLNC .................................................. 110
  5.1 System Model ....................................................................................... 113
    5.1.1 Network Structure .................................................................... 113
# Table of Contents

5.1.2 Random Linear Network Coding (RLNC) .................................... 113

5.2 Problem Formulation .......................................................... 115
  5.2.1 Path Loss Estimation .................................................... 115
  5.2.2 Identifiability Problem ................................................ 119

5.3 Performance Evaluation ...................................................... 120
  5.3.1 Number of Identifiable Links ........................................ 121
  5.3.2 Performance Comparison .............................................. 122

5.4 Summary ................................................................. 124

5.5 Analytical Proof .......................................................... 125
  5.5.1 Proof of Lemma ......................................................... 125

6 Cardinality Estimation in RFID Systems with Multiple Readers ........ 127

  6.1 System Model and Problem Statement .................................. 129
    6.1.1 Notations and Model .............................................. 129
    6.1.2 Multiple Counting Problem ..................................... 131

  6.2 Asynchronous Multiple Reader Cardinality Estimation .............. 133
    6.2.1 Asynchronous Exclusive Estimator ............................... 133
    6.2.2 A-MRCE Algorithm ............................................... 135
    6.2.3 Estimation Error and Discussion ................................ 138

  6.3 Synchronous Multiple Reader Cardinality Estimation (S-MRCE) .... 140
    6.3.1 Synchronous Exclusive Estimator ............................... 140
    6.3.2 S-MRCE Algorithm ............................................... 143

  6.4 Performance Evaluation ................................................ 147
    6.4.1 Performance Evaluation for Exclusive Estimators ............ 147
    6.4.2 Performance Evaluation for A-MRCE and S-MRCE Algorithms .. 151

  6.5 Summary ............................................................... 156
# Table of Contents

6.6 Analytical Proofs .................................................. 156
  6.6.1 Proof of Theorem 5 ........................................... 156
  6.6.2 Proof of Theorem 6 ........................................... 159

7 Conclusions and Future Work .................................. 161
  7.1 Research Contributions ........................................ 161
  7.2 Suggestions for Future Work .................................. 163

Bibliography .......................................................... 165
List of Tables

3.1 Average number of links performing network coding. . . . . . . . . . . . . . . . . . . . . .
List of Figures

2.1 A sample WSN with two sinks. .............................................. 21
2.2 LOCL algorithm for network with different number of sinks: (a) Minimum commodity lifetime; (b) Average number of iterations in LOCL algorithm. ............................................. 41
2.3 Lifetime of commodities for LOCL algorithm with different number of sources. ............................................. 41
2.4 Lifetime of commodities for LOCL algorithm when there are single and different types of sinks. ............................................. 42
2.5 Average normalized minimum lifetime in LONL algorithm versus $\gamma$. ............................................. 43
2.6 Lifetime of commodities for LOCL, LONL, LMM, and MLMS algorithms in a network with four sinks. ............................................. 45
2.7 Lifetime of nodes for LONL, LMM, and MLMS algorithms for a network with four sinks. ............................................. 46
3.1 Sample network with the source node $s$ and two sinks ($d_1$, $d_2$). ............................................. 60
3.2 Network with three sinks ($d_1$, $d_2$, $d_3$) and one source node $s$ with data generation rate of 2 units per second. (a) Information flow values, (b) coding subgraph (actual rate values), (c) coding flow values, (d) multipath packet transmission. ............................................. 62
3.3 Average network lifetime under different values of $\gamma$. ............................................. 74
3.4 Total rate of performing network coding under different values of $\gamma$. ............................................. 75
3.5 Network lifetime versus coding operation rate. ............................................. 76
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6</td>
<td>Network lifetime under different number of sinks ($\gamma = 10^{-4}$).</td>
<td>77</td>
</tr>
<tr>
<td>3.7</td>
<td>Total rate of coding operations under different number of sinks ($\gamma = 10^{-4}$).</td>
<td>78</td>
</tr>
<tr>
<td>3.8</td>
<td>Network lifetime for different number of sources ($\gamma = 10^{-4}$).</td>
<td>79</td>
</tr>
<tr>
<td>3.9</td>
<td>Total rate of coding operations under different number of sources ($\gamma = 10^{-4}$).</td>
<td>80</td>
</tr>
<tr>
<td>3.10</td>
<td>Network topology: (a) one copy, (b) three copies.</td>
<td>81</td>
</tr>
<tr>
<td>4.1</td>
<td>A network tree with two sources, one intermediate node $i$, and one sink.</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>Packets of flows 1 and 2 are generated by source $s_1$ and $s_2$, respectively.</td>
<td>90</td>
</tr>
<tr>
<td>4.2</td>
<td>Normalized average power consumption versus generation size $K$ for $\lambda = 2$.</td>
<td>105</td>
</tr>
<tr>
<td>4.3</td>
<td>Normalized average power consumption versus parameter $\lambda$.</td>
<td>104</td>
</tr>
<tr>
<td>4.4</td>
<td>The normalized decoding delay between RLNC with and without buffer sharing.</td>
<td>105</td>
</tr>
<tr>
<td>4.5</td>
<td>Normalized average power consumption versus the number of sources for $\lambda = 2$.</td>
<td>107</td>
</tr>
<tr>
<td>4.6</td>
<td>Normalized average power consumption versus the number of nodes for $\lambda = 2$.</td>
<td>108</td>
</tr>
<tr>
<td>5.1</td>
<td>A sample WSN with five sources, namely 1, 2, 3, 4, and 7.</td>
<td>114</td>
</tr>
<tr>
<td>5.2</td>
<td>Average number of links whose rates determined using PLI-RLC algorithm.</td>
<td>122</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparison of false detection for PLI-RLC algorithm and Bayesian inference algorithm.</td>
<td>123</td>
</tr>
<tr>
<td>6.1</td>
<td>An RFID system with two readers $r_1$ and $r_2$. $n_{r_1} = 6$, $n_{r_2} = 7$, $N = 11$, $f = 10$, and $p = 1$.</td>
<td>131</td>
</tr>
<tr>
<td>6.2</td>
<td>Network topologies: (a) Three readers, and (b) Ten readers.</td>
<td>146</td>
</tr>
<tr>
<td>6.3</td>
<td>Mean of estimation error for different number of interrogations $M$, (a) asynchronous exclusive estimator, and (b) synchronous exclusive estimator.</td>
<td>147</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>6.4</td>
<td>Standard deviation of estimation error for different number of interrogations $M$ when using an <em>asynchronous</em> exclusive estimator</td>
<td>148</td>
</tr>
<tr>
<td>6.5</td>
<td>Standard deviation of estimation error for different number of interrogations $M$ when using the <em>synchronous</em> exclusive estimator</td>
<td>149</td>
</tr>
<tr>
<td>6.6</td>
<td>The mean of estimation error for varying number of tags, asynchronous and synchronous exclusive estimators</td>
<td>150</td>
</tr>
<tr>
<td>6.7</td>
<td>Operational range of asynchronous and synchronous exclusive estimators for varying persistence probabilities</td>
<td>151</td>
</tr>
<tr>
<td>6.8</td>
<td>Standard deviation of error for varying persistence probabilities: (a) asynchronous exclusive estimator, (b) synchronous exclusive estimator</td>
<td>152</td>
</tr>
<tr>
<td>6.9</td>
<td>Comparison of A-MRCE and S-MRCE in terms of standard deviation of error</td>
<td>153</td>
</tr>
<tr>
<td>6.10</td>
<td>Comparing different algorithms in terms of interrogation time (number of time slots), (a) mean of estimation error, and (b) standard deviation of estimation error</td>
<td>154</td>
</tr>
<tr>
<td>6.11</td>
<td>Comparing the standard deviation of estimation error for A-MRCE, S-MRCE, EZB, and LoF algorithms</td>
<td>155</td>
</tr>
</tbody>
</table>
List of Abbreviations

A-MRCE Asynchronous Multiple Reader Cardinality Estimation
ARQ Automatic Repeat Request
EMR Evolutionary Minimum Resource
EZB Enhanced Zero Based
FDMA Frequency Division Multiple Access
FEC Forward Error Correction
LMM Lexicographical Max-Min Fair
LOCL Lexicographically Optimal Commodity Lifetime
LOCLR Lexicographically Optimal Commodity Lifetime Routing
LoF Lottery Frame
LONL Lexicographically Optimal Node Lifetime
LONLR Lexicographically Optimal Node Lifetime Routing
LONPR Lexicographically Optimal Normalized Power Consumption Routing
LP Linear Program
MAC Medium Access Control
MIP Mixed-Integer Program
ML Maximum Likelihood
MLMR Maximum Lifetime Minimum Resource
MLMR Maximum Lifetime Multiple Sinks
### List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLST</td>
<td>Maximum Lifetime Steiner Tree</td>
</tr>
<tr>
<td>NP</td>
<td>Non-deterministic Polynomial-time</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>RLNC</td>
<td>Random Linear Network Coding</td>
</tr>
<tr>
<td>S-MRCE</td>
<td>Synchronous Multiple Reader Cardinality Estimation</td>
</tr>
<tr>
<td>TDMA</td>
<td>Time Division Multiple Access</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
Acknowledgments

First and foremost, I offer my sincerest gratitude to my supervisor, Dr. Vincent Wong, for his invaluable advice, insightful guidance, and enormous patience. He has improved the quality of my research and presentation by his constructive comments from the early stages of my research until the very end. He has supported me and encouraged me throughout my PhD study. He introduced me to new ways of thinking, taught me technical writing, and provided me with excellent research topics.

I would like to express my gratitude to Dr. Victor Leung and Dr. Robert Schober for their support and help during my PhD program and Dr. Hamed Mohsenian-Rad for his constructive suggestions and guidance. I would also like to thank the members of my doctoral committee for their time and effort.

I am also thankful to my fellow labmates and friends at the Communications lab of the Department of Electrical and Computer Engineering at the University of British Columbia who inspired me along the way, especially Mr. Mohammad MohammadNia, Mr. Ehsan Bayaki, Mr. Ali Nezampour, Mr. Keivan Ronasi, and Mr. Man-Hon Cheung. I thank my friends at the University of British Columbia, Mr. Ali Mahmoudzadeh, Ms. Monir Hajiaghayi, and Ms. Maryam Beigi for their continuous support.

Lastly, and most importantly, I wish to thank my mom for her support. Without her, I could not achieve this success and finish this road. To her, I dedicate this thesis. I would also like to thank my sister Afsaneh and my brother Hamed.

This work was supported by the Natural Sciences and Engineering Research Coun-
Acknowledgments

cil (NSERC) of Canada, Bell Canada, MITACS, and the University of British Columbia Graduate Fellowship.
Dedication

With love, to my mom.
Chapter 1

Introduction

Wireless sensor networks (WSNs) and radio frequency identification (RFID) systems are two important wireless technologies that have a wide variety of applications and provide limitless future potentials. They consist of thousands of small inexpensive devices with radio frequency technology to provide wireless communication. Because the devices are inexpensive, they have several restrictions including limited power supply and memory storage. Therefore, simplicity and power efficiency are two major design goals in such systems. Moreover, the large number of wireless devices requires scalable algorithms. The conventional wireless protocols used for data transmission, collection, and relaying in ad hoc and mesh networks may not be applicable in WSNs and RFID systems. In this thesis, we consider the problem of protocol design for WSNs and RFID systems. In the next subsections, we introduce WSNs and RFID systems and describe the directions of our work. Then, we summarize the mathematical tools that we used in this thesis.

1.1 Wireless Sensor Networks (WSNs)

Recent advances in low power integrated circuits, wireless communications, and micro-electro-mechanical systems have sped up the development of various types of low cost wireless sensors, which are the building blocks of WSNs. In WSNs, each sensor node has the capability to sense the environment (e.g., temperature, pressure, light, acoustic), process the data and transmit data packets. The sensors are also capable of relaying infor-
Chapter 1. Introduction

mation of other sensors. WSNs have a wide range of applications including environment and habit monitoring, home automation, health care applications, and traffic control [2].

A WSN is a self-organizing network composed of a large number of sensor nodes deployed in a geographical area of interest. The generated information of various sensor nodes is gathered at central processors in the system called sinks. Each source node may transmit its data towards one or multiple sinks at the same time. Due to the short transmission range of sensor nodes, multihop communication is exploited instead of single hop communication used in infrastructure-based networks.

Wireless sensor nodes are low cost, battery-powered communication devices with limited power supply, memory, and computational capabilities. Because there is a large number of sensor devices in an environment, it is not practical to replace the battery when a sensor node runs out of power. Therefore, it is of interest to prolong the lifetime of the sensors by minimizing the power consumption [3, 4].

Different definitions have been proposed for the lifetime of WSNs. The most commonly used definition of lifetime is proposed in [5, 6], where lifetime is defined as the time at which the first sensor node runs out of energy. Using this definition of lifetime, the maximum lifetime routing problem is formulated as a linear programming problem [6]. However, this problem only considers the case that the information is gathered at a single sink. For the case that several sinks are responsible in data collection, one can define different lifetimes for the data flows towards the various sinks. Since intermediate nodes may relay packets of different flows, the maximum lifetime routing problem needs to be restated for WSNs with multiple sinks. In Chapter 2, we study this problem and formulate a mixed integer programming problem to find the maximum lifetime routing solution for WSNs with multiple sinks.
1.1.1 Wireless Sensor Network Terminology

In this subsection, we briefly introduce the terminology used in the next four chapters for WSNs. A WSN consists of several sensor nodes with wireless communication capability and one to several sinks. A source node is a sensor node generating information sent towards the sink(s). The information are sensed by the sensor node from the environment. The source may or may not process data before transmission. A sink in practice is a device with wireless capability and higher computational power with no energy limit compared to the sensor nodes. The sink is responsible for gathering the information from the sources. Intermediate nodes or relay nodes are the sensor nodes which relay information of other sensors towards the sink. A source node can act as an intermediate node to relay packets of other sources as well. Therefore, a source node can be an intermediate node as well. The data flow from various sources towards a sink is called a commodity. An intermediate node may relay packets of various commodities at the same time. A session is referred to the data flow from one source to one or multiple sinks. In Chapters 2 and 3, we consider multipath end-to-end communication between a source and the sink(s). This means, sources and intermediate nodes are allowed to transmit the generated and received packets towards the sink(s) via multiple concurrent paths. The portion of data transmitted on various paths can be different in this case. However, in Chapters 4 and 5, we consider single path end-to-end communication. In this case, there is only one path connecting every source node to the sink. This type of end-to-end communication results in a tree network topology.

1.2 Network Coding

Network coding refers to the notion of performing coding (e.g., binary addition, XOR) on the content of the packets at intermediate network devices. A network device performs
network coding when it combines incoming packets which have to reach different sinks and creates a coded packet which has to reach the same set of sinks. For multicast and broadcast traffic, network coding can lead to substantial improvement in network throughput of wired and wireless networks \cite{7, 8, 9, 10, 11}. It has been shown in \cite{7} that the max-flow min-cut bound on multicast capacity is achievable if nodes are allowed to perform network coding. Li et al. \cite{13} showed that linear codes are sufficient to achieve this multicast capacity. It has been shown that random codes can achieve the desired capacity and facilitate distributed implementation \cite{14, 15}. In addition to the throughput gain, network coding can reduce power consumption and congestion by reducing the total number of packets transmitted in the network for unicast and multicast traffic. Different upper bounds have been derived in \cite{16, 17, 18, 19} for the energy gain of network coding.

Consider a directed acyclic graph $G = (V, E)$ modeling the network, where $V$ and $E$ are the set of nodes and links, respectively. Assume that the links have unit capacity. Consider the problem of multicasting from one source to $N$ receivers. Assume that the min-cut capacity from the source to each receiver is $h$. The source has $h$ packets to transmit to all the receivers which are vectors over finite field $\mathbb{F}_q$. Let $P_s = (p_1^s, \ldots, p_h^s)$ denote the set of source packets. Consider link $e \in E$. Let set $\text{In}(e)$ denote the set of incoming links to the parent node of link $e$. Consider a coding vector $c(e) = (c_1(e), \ldots, c^{\text{In}(e)}(e))$ with length $|\text{In}(e)|$. Each element of the coding vector belongs to the finite field $\mathbb{F}_q$. Let $p_i$ for $i = 1, \ldots, |\text{In}(e)|$ denote the packet received from the $i$-th input link to the parent node of link $i$. Then, the following packet is transmitted on link $e$

$$p_e = \sum_{i=1}^{\text{|In}(e)|} c^i(e)p_i.$$  

This is called linear network coding. If all the network nodes perform linear network coding, receiver $j$ receives $h$ packets denoted as vector $\delta = (\delta_1, \ldots, \delta_h)$. This vector can be
represented as a linear combination of source packets \( \delta = A_j P_s \), where \( A_j \) is the transfer matrix from the source to the receiver \( j \) with size \( h \times h \). If intermediate nodes pick the random coefficient randomly from the finite field \( F_q \) and the field size is large enough, the matrix \( A_j \) is full rank and the receiver can extract the source packets successfully [20].

The problem of minimum cost multicast for both wired and wireless networks is studied in [21]. In [21], the wireless network is modeled as a hypergraph, which represents the broadcast nature of the wireless links. The performance of network coding using the hypergraph model is investigated in [22]. The problem of supporting multicast over coded packet networks can be divided into two disjoint design problems. The first problem determines the nodes in the network that perform network coding, which is also referred in the literature to as the problem of determining the coding subgraph. The second problem determines the code that should be used on the coding subgraph. Different factors may affect the problem of constructing the coding subgraph in a wireless network. The power consumption, the lifetime of the network, and the rate of performing network coding are affected by the coding subgraph. The problem of constructing a coding subgraph which minimizes power consumption is studied in [21] for wireless networks. The problem of minimizing the rate at which network coding is performed for wired networks is studied in [23]. In Chapter 3, we study the problem of constructing the coding subgraph for WSNs while we jointly maximize the network lifetime and minimize the rate of performing network coding.

For a multihop communication session with lossy links, it is shown that if intermediate nodes store the received packets and combine the buffered packets to transmit new packets, the reliability of end-to-end communication can be improved. If the coding operations are linear and the coefficients used to create coded packets are random variables, then the scheme is called random linear network coding (RLNC). It is shown that RLNC can achieve
the packet-level (i.e., max-flow min-cut) capacity of a unicast session in the presence of
delay links \cite{24, 25}. To reduce the time required for the sink to decode the packets, the
source divides the generated packets into groups called \textit{generations}. The source(s) and
intermediate nodes perform coding operations on the packets of one generation. The sink
can decode the packets of one generation as soon as it has enough packets for the decoding.
The source(s) and intermediate nodes stop transmission of packets of one generation as
soon as they receive a feedback control packet indicating that the sink can decode the
packets of that generation. The intermediate nodes can also generate the feedback for their
upstream nodes. The feedback mechanism has a direct impact on power consumption of
the intermediate nodes. Therefore, it is of interest to design feedback mechanisms that
minimize the power consumed to deliver a generation of packets to the sink. In Chapter
4, we design power efficient feedback mechanisms for WSNs using RLNC. We assume that
each intermediate node stores the packets of different flows towards the sink in a shared
buffer. We analyze and study the behavior of RLNC with buffer sharing and propose two
types of feedback mechanisms, namely link-by-link and end-to-end feedback mechanisms.

\subsection{Network Tomography}

For both wireless and wired networks, it is of interest to monitor the behavior of the
network, identify the link status, and estimate the delay and loss rate of different links.
Such a technique is called \textit{network tomography} or \textit{network monitoring} in the literature.
Network tomography for wired and wireless networks has received much attention recently
\cite{26, 27, 28, 29, 30}. Network monitoring is especially important for WSNs as the infor-
mation can allow sensor nodes to make decisions on routing and to bypass the links which are
either congested or prone to failure.

Network monitoring can be performed either in an \textit{active} or \textit{passive} manner \cite{27}. Active
network monitoring refers to the case where probe packets are transmitted by some chosen source nodes and are gathered from the sinks. Although active monitoring schemes may reveal more information about the link loss rates, they utilize additional network resources such as bandwidth and energy. On the other hand, for passive network monitoring, information from the received data packets can allow the sink to infer the link loss information. Passive monitoring schemes are more attractive for WSNs because of the limited power supply in these networks.

RLNC used for data collection in WSNs provides a tool for monitoring. The link identification problem in coded packet networks is different from routed packet networks. In a coded packet network, the loss rate of a chosen path is the maximum loss rate of all the links of that path. Therefore, it can be infeasible to infer the loss rate of the intermediate links by simply observing the path loss rate from the source node towards the sink. This behavior of the coded packet networks makes the link identification problem more challenging. However, we can benefit from the coding operations performed at the intermediate nodes and extract information about the path loss rate from the intermediate nodes to the sink. In Chapter 5, we study the problem of passive loss inference in WSNs using RLNC. By inspecting the contents of the received coded packets at the sink and adopting the subspace properties of network coding, we propose an estimation technique to estimate the path loss rates not only from the source nodes but also from several intermediate nodes to the sink.

1.3 RFID Systems

RFID systems are increasingly being deployed as automated identification systems. These systems are expected to play an important role in various applications such as warehouse and supply chain management, object tracking, and patients’ monitoring in health care.
facilities [33, 34, 35]. An RFID system consists of a set of readers and several objects. Each object is equipped with a small computer chip, called a tag. Using these inexpensive tags, every object can be uniquely identified. RFID tags can be categorized into passive and active tags. A passive tag uses backscatter modulation; its transmission power is derived from the signal of the interrogating reader [34, 36]. Passive tags can operate in different frequency bands. Low-frequency tags operate in the 124-135 kHz band and have an operating range of up to 0.5 m. Ultra high frequency tags, which operate at either 860-960 MHz or 2.45 GHz, have a range in the order of 10 m. Active tags require a power source (e.g., a battery) for data transmission and have a larger range (> 100 m).

In an RFID system with one reader and several tags, since the reader and the tags share the same wireless channel, tag-to-tag collision can occur when multiple tags transmit signals simultaneously to the same reader. This prevents the reader from recognizing any tag. Various tag anti-collision protocols are proposed in [37, 38, 39, 40, 41, 42, 43, 44]. An efficient Aloha-based tag anti-collision scheme has also been standardized recently by EPCglobal in [45]. The interrogation time is divided into time slots and called interrogation frame. The reader begins each interrogation round by informing all the tags about the frame size. Each tag then chooses a time slot at random and transmits its identifier to the reader. If the frame size is large enough, the probability of tag-to-tag collision can be reduced significantly.

Tag estimation is widely used as a preliminary phase in ALOHA-based interrogation techniques [46, 47, 48]. Readers can adjust the frame size based on the estimation of tag population. Another application of tag estimation techniques, which has recently received attention, is the anonymous tracking of objects [49, 50]. In order to preserve the privacy and anonymity of the tagged users, it may not be necessary to identify each individual user in some RFID applications. Instead, the goal is to estimate the total number of
tags (or users) in the system. This is called the *cardinality estimation* (or tag population estimation) problem in RFID systems. The potential applications include estimating the number of attendants in large exhibitions and conferences when each attendant is equipped with an RFID tag, and urban traffic monitoring at streets and intersections when cars are equipped with RFID tags. In several RFID applications (e.g., for inventory checking in a large-scale warehouse), it is necessary to deploy several readers to achieve complete interrogation coverage and also to improve read rate and correctness. Since there may be multiple tags in the overlapping coverage area of neighboring readers, the cardinality estimation techniques used for single reader systems are not accurate enough for use in multiple readers systems. In Chapter 6, we study the problem of anonymous cardinality estimation for RFID systems with multiple readers. We propose two estimation methods that are useful under different circumstances.

1.4 Mathematical Background

In this section, we briefly introduce mathematical frameworks which are used in the rest of this thesis.

1.4.1 Convex Optimization

Consider the following optimization problem

\[
\begin{align*}
\text{minimize}_{x \in \mathcal{X}} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq 0, \quad \forall \ i = 1, \ldots, N_i \\
& \quad g_i(x) = 0, \quad \forall \ i = 1, \ldots, N_e,
\end{align*}
\]

(1.2)
where $\mathbf{x}$ is the vector of optimization variables (i.e., $\mathbf{x} = (x_1, \ldots, x_N) \in \mathcal{X}$). The function $f_0 : \mathcal{R}^N \rightarrow \mathcal{R}$ is the objective function, $f_i : \mathcal{R}^N \rightarrow \mathcal{R}$ for $i = 1, \ldots, N_i$ are the inequality constraint functions, and $g_i : \mathcal{R}^N \rightarrow \mathcal{R}$ for $i = 1, \ldots, N_e$ are the equality constraint functions. The inequality and equality constraints together determine the feasible region of the optimization problem. If problem (1.2) is convex, the global optimal solution of the problem can be obtained using optimization theory. The key feature of a convex optimization problem is that local optimality implies global optimality. Problem (1.2) is convex if:

1. $\mathcal{X}$ is a convex set,
2. $f_i(\mathbf{x})$ is convex for $i = 0, \ldots, N_i$,
3. $g_i(\mathbf{x})$ is affine for $i = 1, \ldots, N_e$.

The set $\mathcal{X}$ is convex if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ and $\alpha \in [0, 1]$, $(\alpha \mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2) \in \mathcal{X}$. A function $h(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{R}$ is convex on $\mathcal{X}$ if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ and $\alpha \in [0, 1]$

$$h(\alpha \mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2) \leq \alpha h(\mathbf{x}_1) + (1 - \alpha)h(\mathbf{x}_2).$$  

(1.3)

A function $h(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{R}$ is affine if it has the form $h(\mathbf{x}) = A\mathbf{x} + b$, where $A \in \mathcal{R}^{1 \times N}$ and $b \in \mathcal{R}$. The Lagrangian function of problem (1.2) is defined by introducing Lagrange multipliers $\lambda = (\lambda_1, \ldots, \lambda_{N_i})$ and $\gamma = (\gamma_1, \ldots, \gamma_{N_e})$ as

$$L(\mathbf{x}, \lambda, \gamma) = f_0(\mathbf{x}) + \sum_{i=1}^{N_i} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{N_e} \gamma_i g_i(\mathbf{x}).$$  

(1.4)

The dual function is obtained using the Lagrangian multiplier as

$$q(\lambda, \gamma) = \inf_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}, \lambda, \gamma).$$  

(1.5)
Chapter 1. Introduction

The dual function is concave even if the primal problem (i.e., problem (1.2)) is not convex. The dual problem is defined as

\[
\text{maximize} \quad q(\lambda, \gamma) \\
\text{subject to} \quad \lambda_i \geq 0, \text{ for } i = 1, \ldots, N_i. \tag{1.6}
\]

Let \( f^* \) and \( q^* \) denote the optimal value of the primal problem (1.2) and dual problem (1.6), respectively. Then, we always have \( q^* \leq f^* \) regardless of the convexity of the primal problem. Under favorable circumstances, including convexity of the primal problem, we have \( q^* = f^* \). This is called strong duality.

Suppose that optimization variable \( x \) has \( m \) components \( x_1, \ldots, x_m \) of dimensions \( n_1, \ldots, n_m \), respectively. Consider the following optimization problem

\[
\text{minimize} \quad \sum_{i=1}^{m} f_i(x_i) \\
\text{subject to} \quad \sum_{i=1}^{m} g_{ij}(x_i) \leq 0, \forall j = 1, \ldots, N_i \\
\sum_{i=1}^{m} h_{ij}(x_i) = 0, \forall j = 1, \ldots, N_e, \\
x_i \in X_i, \quad \forall i = 1, \ldots, m \tag{1.7}
\]

where \( f_i : X_i \to \mathbb{R}, g_{ij} : X_i \to \mathbb{R}, \) and \( h_{ij} : X_i \to \mathbb{R} \) are given functions. The dual problem of problem (1.8) can be written as

\[
\text{maximize} \quad \sum_{i=1}^{m} q_i(\lambda, \gamma) \\
\text{subject to} \quad \lambda \succeq 0, \tag{1.8}
\]

\[
\begin{array}{ll}
\max & q(\lambda, \gamma) \\
\text{s.t.} & \lambda_i \geq 0, \text{ for } i = 1, \ldots, N_i.
\end{array}
\]
where \( q_i(\lambda, \gamma) = \inf_{x_i \in X_i} \left\{ f_i(x_i) + \sum_{j=1}^{N_i} \lambda_ig_{ij}(x_i) + \sum_{e=1}^{N_e} \gamma_ih_{ij}(x_i) \right\} \). Moreover, \( \lambda \geq 0 \) means that all the elements of vector \( \lambda \) are greater than or equal to zero. The optimization involved in the calculation of the dual function has been decomposed into \( m \) simpler optimization. This separation technique is widely used to distributively solve an optimization problem in a network. If \( m \) denote the number of nodes in the network, each node \( i \) can solve a decomposed dual problem.

The subgradient method is a method to find the minimum value for a nondifferentiable convex function. It is similar to the ordinary gradient method for differentiable functions with some exceptions. In the subgradient method, the step sizes can be fixed. Unlike the ordinary gradient method, the subgradient method is not a descent method which means the function value can increase. The subgradient method is slower than Newton’s method, but it is much simpler and can be applied to a far wider variety of problems. By combining the subgradient method with dual decomposition technique, it is sometimes possible to develop a simple distributed algorithm for a problem. Suppose \( f : \mathcal{R}^n \rightarrow \mathcal{R} \) is convex. To minimize the function \( f(x) \) using the subgradient method, we can use an iterative approach where at iteration \( k + 1 \) we have,

\[
x^{k+1} = x^k - \alpha_k \pi(x^k),
\]

where \( \pi(x^k) \) is any subgradient of \( f(x) \) at \( x^k \), and \( \alpha_k \) is the step size of iteration \( k \).

The subgradient of function \( f \) at \( x \) is any vector \( \pi \) that satisfies the inequality \( f(y) \geq f(x) + \pi^T(y - x), \forall y \in \mathcal{R}^n \). A typical step size for the subgradient method is non-summable diminishing step size (i.e., \( \alpha_k \) approaches zero while \( \sum_k \alpha_k \) approaches infinity). The step size approaches zero as \( k \) grows in this case while the summation of the step sizes is unbounded. A typical example of this step size is \( \alpha_k = \beta/\sqrt{k} \), where \( \beta > 0 \) is a given constant.
In problem (1.2), if functions \( f_i(x) \), \( \forall i = 0, \ldots, N_i \) and \( g_i(x) \), \( \forall i = 1, \ldots, N_e \) are affine functions of the form

\[
\begin{align*}
    f_i(x) &= a^T x + b, \hspace{1em} \forall i = 0, \ldots, N_i, \\
    g_i(x) &= c^T x + d, \hspace{1em} \forall i = 1, \ldots, N_e,
\end{align*}
\]

(1.10)

where \( a \) and \( c \) are vectors of size \( N \), and \( b \) and \( d \) are scalar values, then the resulting problem is a linear programming problem. Linear programming problems can be solved in polynomial time, which makes them one of the most practical optimization problems. Various optimization techniques including the simplex method and the interior point method can be used to solve linear problems efficiently [51].

### 1.4.2 Linearization Technique

Consider a non-negative real variable \( x \) and a binary variable \( a \). The product of these variables can be replaced by a new real variable \( z \). The value of \( z \) corresponds to the value of \( x \) and \( a \) as follows:

\[
    z = \begin{cases} 
        0, & \text{if } a = 0, \\
        x, & \text{if } a = 1. 
    \end{cases}
\]

(1.11)

Assume that \( x_{\text{max}} \) is an upper bound for the real variable \( x \). The desired correspondence between \( x, a \), and the new variable \( z \) is obtained by requiring that [52]:

\[
\begin{align*}
    0 & \leq z \leq x, \\
    x - x_{\text{max}}(1 - a) & \leq z \leq x_{\text{max}}a.
\end{align*}
\]

(1.12)
1.4.3 Regularization Technique

Consider the following convex optimization problem with variable $x$:

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \geq 0, \quad i = 1, \ldots, m \\
& \quad x \succeq 0,
\end{align*}
\]

(1.13)

where $f_0$ and $f_i$’s can be linear or nonlinear functions. Let $p^*$ denote the optimal value (i.e., minimum value) achieved at an optimal solution $x^*$. That is, $p^* = f_0(x^*)$. Now consider the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad f_0(x) + \delta \phi(x) \\
\text{subject to} & \quad f_i(x) \geq 0, \quad i = 1, \ldots, m \\
& \quad x \succeq 0.
\end{align*}
\]

(1.14)

Mangasarian et al. \[53\] proved that if problem (1.13) is a linear programming problem, for any convex function $\phi(x)$ and for all values of $\delta$ below some positive threshold, the optimal solutions of the regularized problem (1.14) are also the optimal solution in problem (1.13).

Recently in \[54\], Friedlander et al. extended this work and proved that when the problem is nonlinear, the result still holds. They showed that this threshold is the inverse of the
Lagrange multiplier of the second inequality constraint in the following problem:

\[
\begin{align*}
\text{minimize} & \quad \phi(x) \\
\text{subject to} & \quad f_i(x) \geq 0, \quad i = 1, \ldots, m \\
& \quad f_0(x) \leq p^* \\
& \quad x \succeq 0.
\end{align*}
\] (1.15)

They proved that this problem always has a Karush-Kuhn-Tucker (KKT) point. Thus, the threshold always exists.

1.5 Contributions and Results

This thesis covers several aspects of energy-efficient protocol design for WSNs. The results are divided into four chapters for WSNs. In the last chapter, we consider the problem of cardinality estimation in RFID systems. The contributions in each chapter are as follows.

- In Chapter 2, we extend the maximum lifetime routing problem which was originally proposed for WSNs with single sink to the case that there are multiple sinks in the system. In the maximum lifetime routing problem, the goal is to find a routing solution such that the lifetime of the network is maximized. For a WSN with multiple sinks, the data flows towards different sinks are called commodities. We define the lifetime for each commodity individually and formulate the lexicographically optimal commodity lifetime routing problem. We aim to maximize the lifetime of all the commodities with emphasis on commodities with lower lifetime. A stepwise centralized algorithm, called the lexicographically optimal commodity lifetime algorithm, is proposed which can obtain the optimal routing solution and lead to lexicographical fairness among commodity lifetimes. We then show that under certain assumptions,
Chapter 1. Introduction

the lexicographical optimality among commodity lifetimes can be achieved by providing lexicographical optimality among node lifetimes. This motivates us to propose our second algorithm, called the lexicographically optimal node lifetime algorithm, which is suitable for practical implementation in contrast to the commodity lifetime solution. Simulation results show that the difference of both schemes in terms of the commodity lifetime is negligible especially when there are few sinks in the system. The work in Chapter 2 is published in [55, 56, 57, 58].

• In Chapter 3, we study the problem of supporting multicast traffic in WSNs with network coding. We formulate the maximum-lifetime minimum-resource coding subgraph problem and study the tradeoff between network lifetime and network resources used for coding operations. Our objective is to jointly maximize the network lifetime and minimize the number of packets undergoing network coding when the cost of performing coding varies. We propose the coding flow variables and the maximum lifetime minimum resource (MLMR) algorithm which enable us to determine the type of operations (e.g., forwarding, replication, coding) performed in each sensor node and also the rate at which these operations are performed. Results show that for WSNs with four sinks, the network lifetime can be increased up to 25% using our proposed MLMR algorithm compared with the classical multicast with Steiner tree. The network lifetime and the total rate of performing network coding can also be increased by 20% and 30%, respectively, when compared with another algorithm which uses network coding without taking into account the broadcast nature of wireless links. The work in Chapter 3 is published in [59, 60].

• In Chapter 4, we consider the use of random linear network coding (RLNC) for wireless sensor networks (WSNs). We study the problem of feedback mechanism design for RLNC where each intermediate node uses a shared buffers for various flows
that it relays. For an intermediate node in the network tree topology, we determine when the node can stop transmission of a particular flow. We analytically show that reducing the transmission rate at this time does not change the rate at which the node transmits the coded packets that carry new information for the downstream neighbor. We propose a novel link-by-link feedback mechanism. In this mechanism, every intermediate node generates a feedback control packets for its upstream neighboring node. We propose three different types of end-to-end feedback mechanisms for RLNC with buffer. In end-to-end feedback mechanisms, the sink is responsible for generating feedback control packets for all the nodes in the network. Simulation results show that the link-by-link feedback mechanism has better performance in terms of power consumption than the end-to-end feedback mechanism. The results also show the performance gain of using buffer sharing compared to RLNC without buffer sharing in terms of power consumption. They also show the tradeoff between the power consumption and delay when buffer sharing is employed in the network.

• In Chapter 5, we study the passive loss inference problem in WSNs while the data collection is performed using RLNC. We show that by inspecting the content of the coded packets at the sink (i.e., destination), one can estimate the path loss rates not only from the source nodes but also from various intermediate nodes to the sink. By utilizing such information at the sink, we determine the set of links whose loss rates can be identified. We propose a passive loss inference with RLNC (PLI-RLNC) algorithm to estimate the link loss rates. Results show that in coded packet WSNs, our proposed algorithm can identify the status of a higher number of links compared to a Bayesian inference algorithm. The work in Chapter 5 is published in [61].

• In Chapter 6, we consider the anonymous cardinality estimation problem in RFID systems. To achieve complete system coverage and increase the accuracy of measure-
ment, multiple readers with overlapping interrogation zones are deployed. We study the problem under two different circumstances. First, we assume that the readers cannot perform interrogations synchronously. This models the case when the readers are not equipped with accurate clocks or synchronization imposes a high overhead. Under such condition, we propose an asynchronous exclusive estimator to estimate the number of tags that are exclusively located in the zone of a selected reader. By using this estimator, we propose an asynchronous multiple-reader cardinality estimation (A-MRCE) algorithm. In the second scenario, we assume that readers can perform interrogations synchronously. We propose a synchronous exclusive estimator and a synchronous multiple-reader cardinality estimation (S-MRCE) algorithm to estimate the total number of tags. For the exclusive estimators, we show that they are asymptotically unbiased and we derive upper bounds on the variance of error. We validate our analytical model via simulations. Results show that although the A-MRCE algorithm enjoys the asynchronous operation of the readers, it performs worse than the S-MRCE algorithm in terms of estimation error. Compared to the enhanced zero-based (EZB) and lottery frame (LoF) algorithms, the variance of the estimation error for both A-MRCE and S-MRCE algorithms increases linearly with the number of readers, while it increases exponentially for EZB and LoF algorithms. The work in Chapter 6 is published in [62, 63].

1.6 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we study the maximum lifetime routing problem for WSNs with multiple sinks. We present our stepwise centralized algorithm called the lexicographically optimal commodity lifetime algorithm which obtain a routing solution leading to lexicographical fairness among commodity lifetimes. We
also propose the lexicographically optimal node lifetime algorithm, which is suitable for practical implementation in contrast to the commodity lifetime solution. In Chapter 3, we formulate the maximum-lifetime minimum-resource coding subgraph problem for multicast traffic in WSNs. We study the tradeoff between network lifetime and network resources used for coding operation. We propose the coding flow variables that enable us to determine the type of operations performed in each sensor node. Then, we present the maximum lifetime minimum resource algorithm used to jointly maximize the network lifetime and minimize the resources used for network coding. In Chapter 4, we present our feedback mechanisms for WSNs using RLNC. We model and analyze the rate at which intermediate nodes transmit coded packets which carry new information when shared buffers are used to store the packets of various flows. We propose link-by-link and end-to-end feedback mechanisms based on the proposed models. In Chapter 5, we investigate the problem of passive loss inference in WSNs while RLNC is used for data collection. We propose a passive loss inference with RLNC algorithm to estimate the link loss rates in the network. In Chapter 6, we study the anonymous cardinality estimation problem in an RFID system with several readers. We study the problem under two different circumstances. First, we assume that the readers cannot perform interrogations synchronously. We propose an asynchronous exclusive estimator and an asynchronous multiple-reader cardinality estimation algorithm. Next, we assume that readers can perform interrogations synchronously. We propose a synchronous exclusive estimator and a synchronous multiple-reader cardinality estimation algorithm to estimate the total number of tags. Finally, Chapter 7 contains discussions of main results, conclusions, and proposals of future research directions. Each of the main chapters in this thesis is self-contained and included in separate journal articles or conference papers. A review of the related work is given for each chapter accordingly. The notations are defined separately for each chapter.
Chapter 2

Lexicographically Optimal Routing for Wireless Sensor Networks with Multiple Sinks

Sensor nodes are inexpensive battery-powered devices with wireless capability. One of the design objectives for WSNs is to prolong the lifetime of the network [13]. There are various ways to define the lifetime of a WSN. It can be defined as the time at which the first node runs out of energy [14]. This time is equivalent to the time at which the first routing path is disconnected [15]. In [16], the lifetime is defined as the time at which the maximum number of times a certain data collection function can be carried out. In [17, 18], it is defined as the time at which a region within the WSN is not covered by any nodes. The maximum lifetime routing problem is formulated as a linear programming problem in [19], [20], and [21]. In [22], a distributed algorithm based on dual decomposition is proposed to solve the linear maximum lifetime problem. In [23], the maximum lifetime problem is extended by considering a variable-length TDMA (time division multiple access) frame in the MAC (medium access control) layer. In [24], the utility-lifetime tradeoff in the maximum lifetime problem is studied by considering the source rates as variables in the system. In [25], the same problem is studied by considering the scheduling constraints of the link data rates. In [26], the maximum lifetime routing problem is studied for the networks with multiple commodities. A heuristic method is proposed which can achieve a
solution close to the optimal solution. In \cite{75}, an iterative algorithm is proposed to obtain a lexicographic max-min node lifetime solution. In \cite{76}, an iterative centralized algorithm is proposed to find a Pareto-optimal routing solution for WSNs.

As the size of the network increases, it becomes inefficient (in terms of power consumption) and sometimes impossible (in terms of network capacity) to gather all information into a single sink node. To tackle this problem, one can increase the number of sinks \cite{56,77,78,79,80,81}. A sample WSN with two sinks is illustrated in Fig. 2.1, with each source node sending data to the nearest sink. WSNs with multiple sinks have recently received increasing attention. In \cite{77}, a multi-sink WSN architecture is proposed where the network is partitioned into clusters. All the sources in a cluster are assigned to send data to the sink designated to that particular cluster. A multi-drain (multi-sink) sensor network is considered in \cite{78}. Data from each source are logged in two distinct drains in order to increase the resiliency to drain failure. In \cite{57}, upper and lower bounds for the optimal solution of the multiple sink problem are obtained. It is shown that the bounds are tight for networks with a large number of nodes. In \cite{57}, we studied the problem of

![Figure 2.1: A sample WSN with two sinks.](image-url)
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

fair resource allocation among different commodities in multiple sinks WSNs.

The set of nodes sending data to a particular sink is called a commodity. The lifetimes of commodities in a network with multiple sinks are not independent. By changing the routing flows, we can increase or decrease the lifetime for each commodity. One important issue is how to obtain a routing solution that provides fairness among different commodity lifetimes. This is the main focus in this chapter. Our proposed algorithms are based on lexicographic optimization (see Section 2.1.1) to achieve fairness. The contributions of this chapter are as follows:

- We consider the commodity lifetime problem in a WSN with multiple sinks. We formulate this problem as a sequence of optimization problems.

- We propose a stepwise centralized algorithm, called the lexicographically optimal commodity lifetime (LOCL) algorithm, which lexicographically maximizes the lifetime of commodities. By using the LOCL algorithm, for a WSN with four sinks, the average normalized commodity lifetime increases by 100% and 35% compared to maximum lifetime with multiple sinks (MLMS) \([74]\) and lexicographical max-min fair (LMM) \([75]\) algorithms, respectively.

- We show that, under certain assumptions, a lexicographically optimal commodity lifetime problem can be reduced to a lexicographically optimal node lifetime problem. This problem can be solved in polynomial time. We propose the lexicographically optimal node lifetime (LONL) algorithm to solve this problem using a dual decomposition technique. The LONL algorithm is simpler for implementation and more interesting for practical deployment.

The rest of this chapter is organized as follows: In Section 2.1, we present the problem formulation and describe our proposed LOCL algorithm. In Section 2.2, we propose the
LONL algorithm and provide its distributed implementation using the dual decomposition technique. The performance comparisons between LOCL, LONL, MLMS, and LMM algorithms are presented in Section 2.3. A summary is given in Section 2.4.

2.1 Lexicographically Optimal Commodity Lifetime (LOCL) Algorithm

In this section, we first introduce the notations. The lifetime of the commodity and the lexicographically ordered commodity lifetime vectors are defined. We then present the stepwise LOCL algorithm.

2.1.1 System Model

Consider a WSN where the sensor nodes are randomly scattered over the coverage area. Each sensor node has limited energy. Nodes can cooperate to relay data towards the sinks. The commodities and the nodes can be considered as the users and resources of the system, respectively. Each sink uses the system resources to gather information within the sensing area. The source nodes in a commodity utilize the energy of the intermediate nodes in the network for a multi-hop transmission towards the sink. Since each node is battery powered, resources in the system are limited. Thus, a fair resource allocation is needed. Our objective is to obtain a routing flow that fairly shares the system resources among commodities. The lexicographic optimality is used as the notion of fairness among the lifetimes of the commodities.

Let \( \mathcal{V} \) denote the set of sensor nodes and \( \mathcal{C} \) denote the set of sinks collecting information from the network. The data rate generated by source node \( i \) to sink \( k \in \mathcal{C} \) is denoted by \( S^k_i \). The total data rate generated by source node \( i \) is denoted by \( S_i \) and is equal to
Let $\sum_{k \in C} S_i^k$. Let $\mathcal{N}_i$ be the set of neighbors of node $i \in \mathcal{V}$. Let $x_{ij}^k$ denote the data rate of commodity $k \in \mathcal{C}$ transmitted from node $i$ to node $j \in \mathcal{N}_i$. The aggregate data rate for the unidirectional logical link from node $i$ to $j \in \mathcal{N}_i$ is denoted by $x_{ij}$ and is equal to $\sum_{k \in \mathcal{C}} x_{ij}^k$. For notation simplicity, we stack up all $x_{ij}$ and denote it as vector $\mathbf{x}$.

Let $p_{ij}$ denote the power consumed in node $i \in \mathcal{V}$ for transmission of one bit of information to node $j \in \mathcal{N}_i$. The maximum data rate between nodes $i$ and $j$ is denoted by $R_{ij}$. It depends on the capacity of the link and the MAC protocol being used. Let $E_i$ represent the initial energy of node $i$. The lifetime of sensor node $i$, $T_i(\mathbf{x})$, is defined as follows:

$$T_i(\mathbf{x}) = \frac{E_i}{\sum_{j \in \mathcal{N}_i} p_{ij} \sum_{k \in \mathcal{C}} x_{ij}^k}.$$  \hspace{1cm} (2.1)

The lifetime of commodity $k$ is defined as the time at which the first path is disconnected between one of the sources and sink $k \in \mathcal{C}$. This time is equal to the time at which the first node carrying information for sink $k$ runs out of its energy. The lifetime of commodity $k \in \mathcal{C}$ under data flow vector $\mathbf{x}$ is defined as follows:

$$T^k(\mathbf{x}) = \min \left\{ T_i(\mathbf{x}) \mid i \in \mathcal{V} \text{ and } \sum_{j \in \mathcal{N}_i} x_{ij}^k > 0 \right\},$$ \hspace{1cm} (2.2)

where condition $\sum_{j \in \mathcal{N}_i} x_{ij}^k > 0$ assures that node $i$ relays packets of commodity $k$. A vector $\mathbf{T} = (T^1, T^2, \ldots, T^{|\mathcal{C}|})$ is called a lexicographically ordered commodity lifetime vector if $T^1 \leq T^2 \leq \cdots \leq T^{|\mathcal{C}|}$. That is, all the elements in the vector are sorted (or arranged) in ascending order. The commodity lifetime vector $\mathbf{T}$ is lexicographically greater than vector $\mathbf{T}'$ if and only if there exists $i$ such that $T^i > T'^i$, and for all $j < i$ we have $T^j = T'^j$. As an example, if $\mathbf{T} = (2, 2, 5, 8)$ and $\mathbf{T}' = (2, 2, 4, 9)$, then $\mathbf{T}$ is lexicographically greater than $\mathbf{T}'$. Two vectors are lexicographically equal if all of the elements of these two vectors are equal.
A vector is *lexicographically optimal* in a set if it is the lexicographically greatest vector in the set. As an example, for the set \{(1, 3, 7, 8), (2, 2, 5, 8), (2, 2, 4, 9), (2, 2, 3, 10)\}, the vector \((2, 2, 5, 8)\) is the lexicographically optimal vector. Note that for any compact set of \(\mathbb{R}^m\), there exists only one lexicographically optimal vector \([25]\).

### 2.1.2 Problem Formulation

In the LOCL algorithm, the goal is to determine the routing paths and flow rates (i.e., data flow vector \(x\)) that lead to the lexicographically optimal feasible lifetime vector. The LOCL algorithm is a stepwise algorithm. In the first step, the minimum commodity lifetime in the network is maximized. This step may have an infinite number of optimal routing solutions. In the second step, among the solutions from the first step, a solution is chosen that maximizes the second minimum commodity lifetime. The second step may also have an infinite number of optimal solutions.

In general, in step \(n\) of the LOCL algorithm, among the solutions from step \((n - 1)\), the solution is chosen that maximizes the \(n\)th minimum commodity lifetime. In other words, in step \(n\), the \(n\)th minimum commodity lifetime is maximized while all lower commodity lifetimes are being maximized. The routing solution in the last step lexicographically maximizes the lifetime of all commodities. Note that the number of steps is equal to or less than the number of commodities. The optimal routing solution in the last step is the lexicographically optimal commodity lifetime routing (LOCLR) solution:

**Definition 1.** A routing flow is LOCLR if the vector of lifetimes of commodities under this routing flow is the lexicographically greatest feasible commodity lifetime vector.

LOCLR is the solution of the LOCL algorithm or equivalently the optimal solution of the last step of this algorithm. The last step of the LOCL algorithm may have more than one (i.e., an infinite number of) optimal solutions.
Proposition 1 The lexicographically optimal commodity lifetime vector is unique but the LOCLR solution is not necessarily unique.

As we mentioned earlier, the lexicographically optimal commodity lifetime vector in a set is unique [S]. Results in [T] showed that in a lifetime maximization problem, it is possible to have an infinite number of optimal solutions. An example can be found in [U]. This example can be extended for the networks with multiple sinks or commodities.

2.1.3 LOCL Algorithm: First Step

We now present the first step of the LOCL algorithm and describe how it can be converted to a linear mixed-integer programming problem. In the first step, the minimum commodity lifetime is maximized. The problem is as follows:

\[
\text{maximize} \quad \min_{k \in C} \{ T^k(x) \} \\
\text{subject to} \quad \sum_{j \in \mathcal{N}_i} (x^k_{ij} - x^k_{ji}) = S^k_i, \quad \forall \, i \in \mathcal{V}, \quad k \in \mathcal{C} \\
\sum_{k \in \mathcal{C}} x^k_{ij} \leq R_{ij}, \quad \forall \, i \in \mathcal{V}, \quad j \in \mathcal{N}_i \\
x^k_{ij} \geq 0, \quad \forall \, i \in \mathcal{V}, \quad j \in \mathcal{N}_i, \quad k \in \mathcal{C}. \quad (2.3)
\]

The first set of constraints is the flow conservation for each commodity in all of the nodes. The second set of constraints is the data rate limits on each link. The objective function of problem (2.3) is not convex. Therefore, we perform a series of change of variables. To obtain a linear objective function, we introduce an auxiliary scalar variable \( t \), which is a lower bound for the lifetime of the minimum commodity:

\[
t \leq \min_{x, \, k \in C} \{ T^k(x) \} \quad \Rightarrow \quad t \leq T^k(x), \quad \forall \, k \in \mathcal{C}. \quad (2.4)
\]
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

The objective is to maximize \( t \). Substituting (2.2) in (2.4), and (2.4) in (2.3), the problem becomes

maximize \( t \)
subject to \( t \leq T^k \), \( \forall k \in \mathcal{C} \)
\[
T^k = \min_{i \in \mathcal{V}} \left\{ T_i \mid T_i = \frac{E_i}{\left( \sum_{j \in \mathcal{N}_i} p_{ij} \sum_{m \in \mathcal{C}} x^m_{ij} \right)} \text{ and } \sum_{j \in \mathcal{N}_i} x^k_{ij} > 0 \right\}, \quad \forall k \in \mathcal{C}
\]
\[
\sum_{j \in \mathcal{N}_i} (x^k_{ij} - x^k_{ji}) = S^k_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C}
\]
\[
\sum_{k \in \mathcal{C}} x^k_{ij} \leq R_{ij}, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{N}_i
\]
\[
x^k_{ij} \geq 0, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{N}_i, \quad k \in \mathcal{C}.
\]

The second constraint in problem (2.5) can be replaced by

\[
T^k \leq \frac{E_i}{\sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} p_{ij} x^m_{ij}} \quad \text{if} \quad \sum_{j \in \mathcal{N}_i} x^k_{ij} > 0, \quad \forall i \in \mathcal{V}.
\]

(2.6)

We replace \( T^k \) and \( t \) by their respective inverses: \( q^k = 1/T^k \) and \( q = 1/t \). The objective is changed from maximizing \( t \) to minimizing \( q \). Problem (2.5) can now be written as

minimize \( q \)
subject to \( q^k \leq q \), \( \forall k \in \mathcal{C} \)
\[
\sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} p_{ij} x^m_{ij} \leq E_i q^k, \quad \text{if} \quad \sum_{j \in \mathcal{N}_i} x^k_{ij} > 0, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C}
\]
\[
\sum_{j \in \mathcal{N}_i} (x^k_{ij} - x^k_{ji}) = S^k_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C}
\]
\[
\sum_{k \in \mathcal{C}} x^k_{ij} \leq R_{ij}, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{N}_i
\]
\[
x^k_{ij} \geq 0, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{N}_i, \quad k \in \mathcal{C}.
\]

(2.7)
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

The second constraint in problem (2.7) is conditional. To obtain a closed-form for this constraint, we multiply both sides of the inequality by \((\sum_{j \in \mathcal{N}_i} x_{ij}^k)\). The new constraint is

\[
\left(\sum_{j \in \mathcal{N}_i} x_{ij}^k\right) \left(\sum_{j \in \mathcal{N}_i, m \in \mathcal{C}} p_{ij} x_{ij}^m\right) \leq \left(\sum_{j \in \mathcal{N}_i} x_{ij}^k\right) E_i q_k, \quad \forall \, i \in \mathcal{V}, \quad k \in \mathcal{C}. \tag{2.8}
\]

In (2.8), if \(\sum_{j \in \mathcal{N}_i} x_{ij}^k = 0\), then both left hand side and right hand side become 0. If \(\sum_{j \in \mathcal{N}_i} x_{ij}^k > 0\), then constraint (2.8) is equivalent to: \(\sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} p_{ij} x_{ij}^m \leq E_i q_k\). Therefore, constraint (2.8) is equivalent to the second constraint in (2.7). The constraint in (2.8) is a nonconvex constraint. We use a series of change of variables to replace this constraint with several linear constraints. We introduce an auxiliary boolean variable

\[
b_{i}^k = \begin{cases} 
0, & \text{if } \sum_{j \in \mathcal{N}_i} x_{ij}^k = 0 \\
1, & \text{if } \sum_{j \in \mathcal{N}_i} x_{ij}^k > 0,
\end{cases} \tag{2.9}
\]

where \(b_{i}^k\) is equal to 1 when node \(i\) carries information of commodity \(k\). We can then map the new boolean variable to rate variables as follows

\[
\sum_{j \in \mathcal{N}_i} x_{ij}^k \leq b_{i}^k, \quad \forall \, i \in \mathcal{V}, \quad k \in \mathcal{C}. \tag{2.10}
\]

Constraint (2.8) can now be written as follows

\[
b_{i}^k \sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} p_{ij} x_{ij}^m \leq b_{i}^k E_i q_k, \quad \forall \, i \in \mathcal{V}, \quad k \in \mathcal{C}. \tag{2.11}
\]

This constraint is still nonlinear. We use a linearization technique and convert this constraint to a set of linear constraints. Details of the linearization technique can be found in
Section 1.4.2. We define two new variables

\[ \nu^k_i = b^k_i \sum_{j \in N_i} \sum_{m \in C} p_{ij} x^m_{ij}, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \]

\[ \gamma^k_i = b^k_i E_i q^k, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C}. \]  

(2.12)

Constraint (2.11) can be written as

\[ \nu^k_i \leq E_i \gamma^k_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C}. \]  

(2.13)

A set of constraints is added for each new variable and node \( i \) and commodity \( k \)

\[ 0 \leq \nu^k_i \leq \sum_{j \in N_i} \sum_{m \in C} p_{ij} x^m_{ij}, \]

\[ \sum_{j \in N_i} \sum_{m \in C} p_{ij} x^m_{ij} - (1 - b^k_i) P_{i,u}^{max} \leq \nu^k_i, \]

\[ \nu^k_i \leq b^k_i P_{i,u}^{max}, \]

\[ 0 \leq \gamma^k_i \leq q^k, \]

\[ q^k - q_{max}^k (1 - b^k_i) \leq \gamma^k_i \leq q_{max}^k b^k_i, \]  

(2.14)

where \( P_{i,u}^{max} = \sum_{j \in N_i} p_{ij} R_{ij} \), and \( q_{max}^k \) is a loose upper bound for \( q^k \). We re-write problem (2.7) with new variables. In this problem, the variables are \( q, q^k, x^k_{ij}, x^m_{ij}, b^k_i, \nu^k_i, \) and \( \gamma^k_i \).
while $S^k_i$, $R_{ij}$, $E_i$, $p_{ij}$, $P^\text{max}_i$, and $q_{\text{max}}^k$ are the constants:

\[
\begin{align*}
\text{minimize} & \quad q \\
\text{subject to} & \quad q^k \leq q, \quad \forall k \in \mathcal{C} \\
& \quad \sum_{j \in \mathcal{N}_i} (x^k_{ij} - x^k_{ji}) = S^k_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad \sum_{j \in \mathcal{N}_i} x^k_{ij} \leq b^k_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad \nu_i^k \leq E_i \gamma_i^k, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad 0 \leq \nu_i^k \leq \sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} p_{ij} x^m_{ij}, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad \sum_{j \in \mathcal{N}_i} x^k_{ij} \leq R_{ij}, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{N}_i \\
& \quad \nu_i^k \leq b^k_i P^\text{max}_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad 0 \leq \gamma_i^k \leq q^k, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad q^k - q_{\text{max}}^k (1 - b^k_i) \leq \gamma_i^k \leq q_{\text{max}}^k b^k_i, \quad \forall i \in \mathcal{V}, \quad k \in \mathcal{C} \\
& \quad x^k_{ij} \geq 0, \quad b^k_i \in \{0, 1\}, \quad \forall i \in \mathcal{V}, \quad j \in \mathcal{N}_i, \quad k \in \mathcal{C}.
\end{align*}
\]

Because of the linearity of the objective function, it is possible to have an infinite number of optimal lifetime solutions. To obtain a unique set of commodities with minimum lifetime in this step and the subsequent steps, one can use the regularization method. Details of the regularization method can be found in Section 1.3.3. The regularization term used in problem (2.15) is the Euclidean norm of the commodity inverse lifetime vector which is $\sum_{k \in \mathcal{C}} (q^k)^2$. Therefore, the objective function for the regularized problem is

\[
\text{minimize} \quad q + \delta \sum_{k \in \mathcal{C}} (q^k)^2,
\]

(2.16)
where $\delta$ is the regularization parameter. The work in [53] and [54] proved that when $\delta$ is less than a certain threshold, the optimal solutions of the regularized problem form a subset of the optimal solutions of the problem before regularization.

### 2.1.4 LOCL Algorithm: Subsequent Steps

The first step in the LOCL algorithm is a linear mixed-integer program (MIP). The feasible set in the second step of the LOCL algorithm is the optimal solution set of the first step. Similarly, the feasible set in the third step is the optimal solution of the second step, and so on. We call the mixed integer programming problem in step $n$ MIP-$n$. The following lemma characterizes the feasible region for the problem in step $n$:

**Lemma 1** The feasible region of problem MIP-$n$ is nonempty if there exists at least one path between each source and its associated sink.

The proof of this lemma is presented in Section 2.5.1. This lemma guarantees that the feasible region in each step, which is the optimal solution set in the previous step, is nonempty.

Assume that the minimum lifetime in the first step (problem (2.15)) is $T^1*$ and the optimal value (inverse of minimum commodity lifetime) is $q^1*$. Let $\mathcal{P}_1$ be the set of commodities that the first constraint in problem (2.15) is active for them. In the second step, the minimum lifetime among all the commodities except the members of $\mathcal{P}_1$ (i.e., $\mathcal{C}\backslash \mathcal{P}_1$) is maximized subject to the condition that the lifetime of the commodities in $\mathcal{P}_1$ is also being maximized. The problem in the second step is similar to problem (2.15) while the objective is modified to be as

$$\text{maximize } \min_{k \in \mathcal{C}\backslash \mathcal{P}_1} T^k(x).$$
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

Also, there is a constraint on the maximization of minimum lifetime. The constraint is

\[ T_l \geq T^{l_1^*}, \quad \forall l \in \mathcal{P}_1. \]

The problem in the \( n \)th step can be formulated with similar changes in the objective function and by including the additional constraints. Let \( T^{h_1^*} \) denote the maximum achievable value for the \( h \)th minimum commodity lifetime (obtained from the \( h \)th step). Let \( \mathcal{P}_h \) denote the set of commodities that their lifetimes are equal to \( T^{h_1^*} \) in the \( h \)th step. We have \( \mathcal{P}_h = \{ k \mid T^k = T^{h_1^*} \text{ and } k \in C \setminus \bigcup_{l=1}^{h-1} \mathcal{P}_l \} \). The problem in the \( n \)th step is as

\[
\text{maximize} \quad \min_{k \in C \setminus \bigcup_{h=1}^{n-1} \mathcal{P}_h} \{ T^k(x) \} \\
\text{subject to} \quad T_l \geq T^{h_1^*}, \quad \forall l \in \mathcal{P}_h, \quad h = 1, \ldots, n - 1 \\
\sum_{j \in N_i} (x^k_{ij} - x^k_{ji}) = S^k_i, \quad \forall i \in V, \quad k \in C \\
\sum_{k \in C} x^k_{ij} \leq R_{ij}, \quad \forall i \in V, \quad j \in N_i \\
x^k_{ij} \geq 0, \quad \forall i \in V, \quad j \in N_i, \quad k \in C. \tag{2.17}
\]

By letting \( q^{h_1^*} = 1/T^{h_1^*} \), the same series of changes that are applied to problem \((2.3)\) can be applied to problem \((2.17)\). The mixed integer programming problem in step \( n \) (i.e.,
MIP-\( n \) is as follows:

\[
\begin{align*}
\text{minimize} & \quad q \\
\text{subject to} & \quad q^k \leq q, \quad \forall \ k \in \mathcal{C} \setminus \bigcup_{h=1}^{n-1} \mathcal{P}_h \\
& \quad q^l \leq q^h^*, \quad \forall \ l \in \mathcal{P}_h, \ h = 1, \ldots , n - 1 \\
& \quad \sum_{j \in \mathcal{N}_i} (x^k_{ij} - x^k_{ji}) = S^k_i, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad \sum_{j \in \mathcal{N}_i} x^k_{ij} \leq b^k_i, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad \nu^k_i \leq E^k_i, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad 0 \leq \nu^k_i \leq \sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} p_{ij} x^m_{ij}, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad \sum_{j \in \mathcal{N}_i} \sum_{m \in \mathcal{C}} x^m_{ij} - (1 - b^k_i) P_i^{max} \leq \nu^k_i, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad \sum_{k \in \mathcal{C}} x^k_{ij} \leq R_{ij}, \quad \forall \ i \in \mathcal{V}, \ j \in \mathcal{N}_i \\
& \quad \nu^k_i \leq b^k_i P_i^{max}, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad 0 \leq \gamma^k_i \leq q^k, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad q^k - q_{max}\left(1 - b^k_i\right) \leq \gamma^k_i \leq q_{max} b^k_i, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad x^k_{ij} \geq 0, \ b^k_i \in \{0, 1\}, \quad \forall \ i \in \mathcal{V}, \ j \in \mathcal{N}_i, \ k \in \mathcal{C}.
\end{align*}
\]

Algorithm 2.1 shows the stepwise LOCL algorithm to determine the LOCLR solution.

**Algorithm 2.1** Stepwise LOCL Algorithm (Centralized)

1: Set \( n := 1 \)
2: While \( \bigcup_{h=1}^{n} \mathcal{P}_h \neq \mathcal{C} \)
3: Solve MIP-\( n \).
4: Set \( q^{n*} := \text{Optimal value of MIP-} n \).
5: Set \( \mathcal{P}_n := \{ k \mid q^k = q^{n*} \text{ and } k \in \mathcal{C} \setminus \bigcup_{h=1}^{n-1} \mathcal{P}_h \} \).
6: Set \( n := n + 1 \).
7: End
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

The MIP-1 is problem (2.15). The MIP-n (for $n > 1$) is problem (2.18). The number of steps in the stepwise LOCL algorithm is less than or equal to the number of commodities.

Each iteration of Algorithm 2.1 includes an NP-hard problem. There are efficient commercial software (such as CPLEX [83] and MOSEK [84]) to solve linear mixed-integer problems. Most of them use branch-and-bound algorithm [85]. The linear mixed-integer problem in the $n$th step of Algorithm 2.1 has $|C| |V|$ binary variables, $|C|(2|V| + |L|)$ real variables, $|C|(8|V| + n)$ inequality constraints, and $|C| |V|$ equality constraints, where $L$ is the set of links in the network. Notice that the computational complexity of a linear mixed-integer problem grows exponentially with the number of its integer (in our case binary) variables [86]. Thus, problem (2.18) can be solved in practice for small-scale and medium-scale WSNs. Next, we propose our second algorithm which is particularly useful for practical deployments of large scale WSNs.

2.2 Lexicographically Optimal Node Lifetime (LONL) Algorithm

In (2.2), we defined the lifetime of a commodity $k$ to be the time at which the first node carrying information in commodity $k$ runs out of energy. In a WSN with a large number of sensor nodes, if the sensor nodes and the sinks are uniformly distributed in the coverage area and all the sensor nodes have the same amount of initial energy, then the first node which runs out of its energy in commodity $k$ is most likely to be one of the neighbors of sink $k$. In the first stage, some sensor nodes that are closer to the sinks will most likely run out of energy. Since they are neighbors of the sinks, they usually carry only information belonging to their commodities. Therefore, their lifetimes are independent of each other. If the above assumptions are valid, then the lexicographically optimal commodity lifetime
problem can be reduced to a lexicographical optimal node lifetime problem. In this section, we show how to solve the latter in a distributed manner. The proposed algorithm is called the lexicographically optimal node lifetime (LONL) algorithm. Compared to the LOCL algorithm in the previous section, the LONL algorithm is simpler for implementation and more practical for deployment.

2.2.1 System Model

Given the list of nodes’ lifetimes from equation (2.1), we can determine the corresponding lexicographically ordered node lifetime vector such that \( T_1 \leq T_2 \leq \cdots \leq T_{|V|} \). The concepts of lexicographical ordering and lexicographically greater than are the same as what we defined in Section 2.1.1.

**Definition 2.** A routing is Lexicographically Optimal Node Lifetime Routing (LONLR) if the node lifetime vector under this routing flow is the lexicographically greatest among all feasible node lifetime vectors.

We call the inverse of the lifetime of each node the normalized power consumption of the node. Let \( g_i \) denote the normalized power consumption of node \( i \). The normalized power consumption of node \( i \) under data flow vector \( x = \{x_{ij}^k\} \) is

\[
g_i(x) = \sum_{j \in N_i} p_{ij} \sum_{k \in C} x_{ij}^k E_i.
\] (2.19)

For simplicity, the vector \( g = \{g_i\} \) is used to denote the normalized power consumptions of all the nodes.

**Definition 3.** A routing flow is Lexicographically Optimal Normalized Power Consumption Routing (LONPR) flow if the normalized power consumption vector under this
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

Routing flow is the *lexicographically smallest* feasible normalized power consumption vector.

Using the concept of LONPR, we can express the following lemma:

**Lemma 2** The nodes’ lifetimes vectors under LONLR flow (Definition 2) and LONPR flow (Definition 3) are lexicographically equal.

The proof of this lemma is provided in Section 2.5.2. It shows that the $i$th minimum lifetime is equal to the inverse of the $i$th maximum normalized power consumption for all values of $i = 1, \ldots, |\mathcal{V}|$.

We now formulate a convex optimization problem, which leads to the LONPR solution. Consider a convex function $f(.)$ and the following problem:

$$\begin{align*}
\text{minimize} & \quad \sum_{i \in \mathcal{V}} f(g_i) \\
\text{subject to} & \quad \sum_{j \in \mathcal{N}_i} (x_{ij}^k - x_{ji}^k) = S_i^k, \quad \forall \ i \in \mathcal{V}, \ k \in \mathcal{C} \\
& \quad \sum_{j \in \mathcal{N}_i} p_{ij} \sum_{k \in \mathcal{C}} x_{ij}^k = E_i g_i, \quad \forall \ i \in \mathcal{V} \\
& \quad \sum_{k \in \mathcal{C}} x_{ij}^k \leq R_{ij}, \quad \forall \ i \in \mathcal{V}, \ j \in \mathcal{N}_i \\
& \quad x_{ij}^k \geq 0, \quad \forall \ i \in \mathcal{V}, \ j \in \mathcal{N}_i, \ k \in \mathcal{C},
\end{align*}$$

(2.20)

Problem (2.20) has $|\mathcal{V}|(|\mathcal{C}|+1)$ (real) variables, $|\mathcal{C}|(|\mathcal{C}|+1)$ inequality constraints, and $|\mathcal{V}|(|\mathcal{C}|+1)$ equality constraints. Since all the constraints are linear and the objective function is convex, the optimization problem in (2.20) is a convex programming problem. There are several schemes that can be used to solve convex optimization problems. They include the *gradient projection methods*, *interior point method*, and *primal-dual method* [87]. Note that the convex programming algorithms have polynomial complexity. That is,
the run time is no greater than a polynomial function of the problem size $(2|V|+|L|)(|C|+1)$. The theorem below relates problem (2.20) and the LONPR solution.

**Theorem 1** If $h(x)$ is a differentiable increasing convex function, then the routing solution of problem (2.20) with $f(x) = (h(x))^{\gamma}$ approaches the LONPR solution as $\gamma \to \infty$.

The proof of this theorem is presented in Section 2.5.3. It can be shown that the LONPR flow solution is not necessarily unique. However, all of the solutions lead to a unique lexicographically ordered node lifetime vector. Based on Theorem 1, the optimal solution of problem (2.20) is the LONPR solution. Based on Lemma 1, the LONPR and LONLR flow solutions lead to the same node lifetime vector. Therefore, the solution of problem (2.20) is indeed the LONLR solution. Problem (2.20) is a tractable convex problem which can be solved in polynomial time. We now propose a distributed solution for this problem.

### 2.2.2 Distributed Implementation

In this section, we present a distributed algorithm to solve problem in (2.20). The technique that we use is *dual decomposition* [87], which has also been used in [70, 88]. Problem (2.20) is referred as the primal problem.

We first introduce Lagrange multipliers $\lambda^k_i$ for the first equality constraints in (2.20). The other constraints are local constraints in each node and do not need to be relaxed. The Lagrangian function $L(g, x, \lambda)$ is

$$L(g, x, \lambda) = \sum_{i \in V} f(g_i) + \sum_{i \in V} \sum_{k \in C} \lambda^k_i \left( \sum_{j \in N_i} (x^k_{ij} - x^k_{ji}) - S_i \right)$$

$$= \sum_{i \in V} \left( f(g_i) + \sum_{j \in N_i} \sum_{k \in C} x^k_{ij} \left( \lambda^k_i - \lambda^k_j \right) - \sum_{k \in C} \lambda^k_i S_i \right).$$
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

From the Lagrangian, the dual function and the dual problem can be defined. A subgradient algorithm [87] can be used to solve the dual problem. The subgradient algorithm is an iterative algorithm. In iteration $\tau$, given $\lambda_i^k(\tau)$, $\lambda_j^k(\tau)$ for $j \in \mathcal{N}_i$, and the local information $(p_{ij}, E_i, R_{ij})$, each node $i$ updates $x_{ij}^k(\tau)$ and $g_i^k(\tau)$ by solving the following problem:

$$\text{minimize} \quad f(g_i(\tau)) + \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{C}} x_{ij}^k(\tau) (\lambda_i^k(\tau) - \lambda_j^k(\tau)) - \sum_{k \in \mathcal{C}} \lambda_i^k(\tau) S_i$$

subject to

$$\sum_{j \in \mathcal{N}_i} p_{ij} \sum_{k \in \mathcal{C}} x_{ij}^k(\tau) = E_i g_i(\tau),$$

$$\sum_{k \in \mathcal{C}} x_{ij}^k(\tau) \leq R_{ij}, \quad \forall \ j \in \mathcal{N}_i$$

$$x_{ij}^k(\tau) \geq 0, \quad \forall \ j \in \mathcal{N}_i, \ k \in \mathcal{C}. \quad (2.21)$$

Algorithm 2.2 shows the distributed LONL algorithm performed in node $i$.

**Algorithm 2.2** LONL Algorithm in Node $i$ (Distributed)

1: Set $\tau := 1$
2: **While** not converged
3: Solve problem (2.21) to obtain $g_i(\tau)$ and $x_{ij}^k(\tau)$ for $j \in \mathcal{N}_i$, $k \in \mathcal{C}$.
4: Exchange the values of $x_{ij}^k(\tau)$ with neighboring nodes $j \in \mathcal{N}_i$.
5: Set $\lambda_i^k(\tau + 1) := \lambda_i^k(\tau) - \mu(\tau) \left( S_i - \sum_{j \in \mathcal{N}_i} x_{ij}^k(\tau) - x_{ji}^k(\tau) \right)$.
6: Exchange the values of $\lambda_i^k(\tau + 1)$ with neighboring nodes $j \in \mathcal{N}_i$.
7: $\tau := \tau + 1$.
8: **End**

In the above algorithm, $\mu(\tau)$ is a positive diminishing step size and is chosen as $\frac{1}{\tau + 1}$. The convergence of Algorithm 2.2 is guaranteed by the subgradient method [87].
2.3 Performance Evaluation

In this section, we evaluate the performance of our proposed LOCL and LONL algorithms. We assume a deterministic path loss model. The power consumed for transmission of one bit from node \( i \) to node \( j \) (i.e., \( p_{ij} \)) is \( \eta_1 + \eta_2 d_{ij}^4 \), where \( d_{ij} \) is the physical distance of nodes \( i \) and \( j \). We choose \( \eta_1 = 1 \) and \( \eta_2 = 0.1 \). Each source generates 1 kbps of information. The initial energy of each source node \( E_s \) is 3 J while the initial energy of non-source node \( i \) (i.e., \( E_i \)) is 1 J. Nodes use TDMA to access the shared communication channel. The time slots are assigned equally and statically to the different links, which have interference with each other. The maximum link rate \( R_{ij} \) is 250 kbps for all logical links.

2.3.1 Performance of LOCL Algorithm

In the first experiment, we show the performance of employing multiple sinks. There are 30 sensor nodes randomly deployed in a 50 m \( \times \) 50 m square field. The transmission range of each node is 10 m. Eight nodes are randomly chosen as the source nodes. Each source sends data to its (physically) closest sink. Fig. 2.2(a) shows the normalized minimum commodity lifetime in the network versus the number of sinks. Each sink is located in one of the four corners in the field. Results are averaged over 100 simulation runs. The values are normalized with respect to the lifetime of the network with only one sink. Simulation results predict that the lifetime increases almost linearly as the number of sinks is increased. It can be seen that the minimum commodity lifetime in the network when there are four sinks is almost 500% more compared to the network with one sink. The benefit of increasing the number of sinks is evident. Fig. 2.2(b) shows the number of steps performed in the LOCL algorithm versus the number of sinks in the network. When there are four sinks in the network, the average number of steps is about 3.5. This shows the average number of times that the MIP problem should be solved to achieve the optimal solution.
Next, we investigate the performance of the LOCL algorithm with a different number of sources. The other simulation settings are the same as in the previous experiment. Fig. 2.2 shows the average normalized commodity lifetime for the commodities under different number of sources. When the number of sources increases, the lifetime of each commodity decreases.

In this experiment, we assume that there are four different types of sources in the network, two sensors of each type. For example, there can be two temperature sensors,
two pressure sensors, two humidity meter sensors, and two airflow sensors. One sink is assigned to gather information for each data type. Sensors are then required to send data to the corresponding sink. Fig. 2.4 shows the normalized lifetime of the commodities. We compare the results when there are different types of sensors and the case that each source sends data to the nearest sink. When there are different types of sources and all sources are randomly deployed in the sensor area, there are more common nodes among commodities. The correlation between different commodities increases. Also, the data packets are traversed over longer paths. Therefore, the lifetime of commodities decreases compared to the case where each node sends data to the nearest sink.

Figure 2.3: Lifetime of commodities for LOCL algorithm with different number of sources.
2.3.2 Performance Comparisons between LOCL, LONL, MLMS, and LMM Algorithms

To perform the simulation for the LONL algorithm, we need to choose a function $h(g_i)$ and parameter $\gamma$. We set $h(g_i) = g_i$. To choose $\gamma$, we set up an experiment. The network topology is similar to the network used in the first experiment with two sinks. We increment the value of $\gamma$ starting from 1 and solve problem (2.20). Fig. 2.5 shows the average normalized minimum lifetime in the network for 100 simulation runs versus different values of $\gamma$. It can be seen that when $\gamma > 8$, the difference between consecutive lifetime values is less than 1%. We choose $\gamma = 10$ for the subsequent simulation runs. The objective function in problem (2.20) is chosen as $f(g_i) = g_i^\gamma$.

To compare our LOCL and LONL algorithms with the existing algorithms in the literature, we implemented the algorithms proposed in [74] and [75]. We solve the corre-
Chapter 2. Lexicographically Optimal Routing for WSNs with Multiple Sinks

Figure 2.5: Average normalized minimum lifetime in LONL algorithm versus $\gamma$.

Responding mixed integer programming problems by using the MOSEK [84] optimization toolbox. There are four sinks and eight sources in the network. In [72], the problem of network lifetime maximization is modeled as a concurrent multi-commodity flow optimization problem. The problem can be extended to consider the multi-sink case. We modify this algorithm and assume that each source sends data to the closest sink. Note that this makes the comparison to be fair. When there is one sink, this problem is equal to the maximum lifetime routing problem proposed in [65]. We call the modified algorithm the maximum lifetime routing for network with multiple sinks (MLMS).

A lexicographical max-min fair (LMM) algorithm is proposed in [75]. This algorithm determines a schedule for the routing flows. We extend this algorithm for WSNs with multiple sinks. We compare the results of LOCL with the first routing flow of LMM method. Notice that the LOCL algorithm provides one routing flow but not a schedule of routing flows for the network. Since both LOCL and LONL algorithms provide one routing
flow, we compare these algorithms with the first routing flow in the schedule of LMM.

Fig. 2.6 compares the lifetime of commodities between LOCL, LONL, MLMS, and LMM algorithms. Results show that for the minimum commodity lifetime, all four algorithms provide the same lifetime. For the other commodities, the LOCL algorithm provides a higher lifetime. The difference between LOCL and LONL algorithms is due to the fact that after the first node runs out of its energy, the LOCL algorithm tries to maximize the next commodity lifetime while the LONL algorithm tries to maximize the second node lifetime. The second node does not necessarily belong to the second commodity. This happens for the subsequent commodities and leads to different results for LOCL and LONL algorithms. The difference increases as the commodity number increases. The lifetimes for the commodities obtained from the MLMS algorithm are almost equal because this algorithm only maximizes the minimum lifetime in the network. For the LMM algorithm, the first routing flow from the schedule is compared.

Fig. 2.7 compares the lifetime of nodes under LONL, MLMS, and LMM algorithms. The LONL algorithm maximizes the lifetime of all the nodes in the network and consequently has better performance. The LMM algorithm has a better performance compared to the MLMS algorithm. In the MLMS algorithm, only the minimum lifetime is maximized and therefore the lifetimes of other nodes are almost equal.

2.4 Summary

Multiple sinks can be employed in a WSN to increase the lifetime of the network. In this chapter, we formulated two algorithms to fairly share the network resources among various commodities and nodes in the system using the concept of lexicographical fairness. We first proposed the centralized LOCL algorithm to obtain the exact lexicographically optimal solution. LOCL is a stepwise algorithm. In each step, a linear mixed-integer
programming problem is being solved. Our second algorithm, called the LONL algorithm, is distributed. It can obtain the optimal solution under certain assumptions, and the sub-optimal solutions in general. The LONL algorithm is easier for implementation and more practical for deployments. Simulation results showed that both LOCL and LONL algorithms have better performance compared to some existing schemes.

2.5 Analytical Proofs

2.5.1 Proof of Lemma

We use mathematical induction to prove this lemma. If there is at least one path between each source and its associated sink, the feasible set of problem MIP-1 is nonempty. Consequently, the optimal set is also nonempty. If the feasible region of problem MIP-\( n \) is
nonempty then its optimal set is nonempty. Since the feasible set of problem MIP-(n + 1) is the optimal set in problem MIP-n, therefore it is nonempty.

2.5.2 Proof of Lemma 2

We use mathematical induction to prove the following parts:

1. The inverse of the minimum lifetime is equal to the maximum normalized power consumption.

2. If the inverse of the \(i\)th minimum lifetime is equal to the \(i\)th maximum normalized power consumption, then the inverse of the \((i + 1)\)th minimum lifetime is equal to the \((i + 1)\)th maximum normalized power consumption.
Both parts can be proved by contradiction. We only prove part one; the proof for the second part can be derived in a similar manner. Let $T_1$ denote the minimum lifetime (i.e., the maximum achievable value for the minimum lifetime) and $g_1$ denote the maximum normalized power consumption (i.e., the minimum achievable value for the maximum normalized power consumption).

Now assume that $g_1 \neq \frac{1}{T_1}$. If $g_1 > \frac{1}{T_1}$, it means that it is possible to reduce maximum normalized power consumption to $\frac{1}{T_1}$ while we have assumed that $g_1$ is the minimum achievable value for the maximum normalized power consumption, which is a contradiction. If $g_1 < \frac{1}{T_1}$, it means that it is possible to reduce maximum normalized power consumption to $\frac{1}{T_1}$ while we have assumed that $g_1$ is the minimum achievable value for the maximum normalized power consumption which is a contradiction. It is also a contradiction. ■

2.5.3 Proof of Theorem

The set of constraints in problem (2.20) constructs a closed convex set. Let $g^\text{max}_i$ be a loose upper bound for the normalized power consumption of node $i \in \mathcal{V}$. Let $g$ denote the vector of normalized power consumption and $g^\gamma$ represent the optimal value of problem (2.20) when $f(x) = (h(x))^\gamma$. The closed feasible set and the loose upper bound constraint construct a compact feasible set for $g^\gamma$. Since $g^\gamma$ is a sequence in a compact set, there exists a subsequence of $\gamma$, $\{\gamma_m, m \geq 1\}$, such that $g^\gamma_m$ converges to some $g^\star$ as $m \to \infty$, where $g^\star$ is the limit point of $g^\gamma_m$. We prove that $g^\star$ is the lexicographically smallest normalized power consumption vector and is unique.

Proof by contradiction: Assume that $g^\star$ is not the lexicographically optimal solution and $g'$ is the lexicographically optimal vector. $g'$ is lexicographically less than $g^\star$. It means that, there exists $i$ such that $g^\star_j = g'_j$ for all $j < i$ and $g^\star_i > g'_i$. Let $\xi = g^\star_i - g'_i$. From the convergence of $g^\gamma_m$ to $g^\star$, there exists $m_0$ such that for all $m \geq m_0$, $g^\gamma_m$ is in the
\( g^*_n - \epsilon \leq g^{\gamma_m}_n \leq g^*_n + \epsilon. \) 

(2.22)

Consider the expression \( A_m \) defined by:

\[
A_m = \sum_{n=1}^{\lvert V \rvert} (f_m(g'_n) - f_m(g^{\gamma_m}_n)),
\]

(2.23)

where \( f_m(x) = (h(x))^\gamma_m \). From the optimality of \( g^{\gamma_m} \), we have \( A_m \geq 0 \). Since for the values of \( n < i \), the members of two vectors are equal, we have

\[
A_m = f_m(g'_i) - f_m(g^{\gamma_m}_i) + \sum_{n=i+1}^{\lvert V \rvert} (f_m(g'_n) - f_m(g^{\gamma_m}_n)).
\]

(2.24)

Since the function \( f_m(x) > 0 \), we have

\[
A_m \leq f_m(g'_i) - f_m(g^{\gamma_m}_i) + \sum_{n=i+1}^{\lvert V \rvert} f_m(g'_n)
\]

\[
= -f_m(g^{\gamma_m}_i) + \sum_{n=i}^{\lvert V \rvert} f_m(g'_n).
\]

(2.25)

The vector \( g' \) is lexicographically ordered. Thus, for all \( n > i \), \( g'_n < g'_i \), we have

\[
A_m \leq -f_m(g^{\gamma_m}_i) + M f_m(g'_i),
\]

(2.26)

where \( M = \lvert V \rvert - i + 1 \). We know that \( g'_i = g^*_i - \xi \). We also use equation (2.22) to obtain

\[
A_m \leq -f_m(g^*_i - \epsilon) + M f_m(g^*_i - \xi)
\]

\[
= -f_m(g^*_i - \epsilon) \left( 1 - M \frac{f_m(g^*_i - \xi)}{f_m(g^*_i - \epsilon)} \right).
\]

(2.27)
Since $\epsilon$ can be any number, we choose it such that $\epsilon < \xi$. The last term is equal to

$$\frac{f_m(g^*_i - \xi)}{f_m(g^*_i - \epsilon)} = \left( \frac{h(g^*_i - \xi)}{h(g^*_i - \epsilon)} \right)^{\gamma_m},$$

(2.28)

where $h(x)$ is an increasing function. When $\gamma_m \to \infty$ (i.e., $m \to \infty$), this term tends to zero. Therefore, $A_m \leq -1$, which is a contradiction.

Note that our proof is related to that of Lemma 3 in [89]. In [89], Mo et al. considered utility maximization problems and proved a similar theorem for max-min fairness; however, here we consider the normalized power consumption minimization problem and the lexicographically optimal is the notion of fairness.
Chapter 3

Lifetime-Resource Tradeoff for Multicast Traffic in Wireless Sensor Networks

The focus of this chapter is on constructing the coding subgraph for multicast traffic in WSNs. WSNs consist of battery-powered low cost wireless sensor nodes. Sensor nodes typically have limited energy supply, memory and computational capability. Sensor nodes performing network coding require additional capabilities. For example, they need to have more computational capability and memory to perform coding, especially when there is a large number of sessions in the network. Moreover, further complexity is incurred and more power is consumed when network coding is performed in the application layer. Current sensor devices either do not have these capabilities or they can be costly to implement. Hence, it is advantageous to either minimize the rate of network coding operation or limit the number of devices performing network coding in the network. The problem of choosing the minimum set of nodes to perform network coding is NP-hard in general. However, we show that the problem of minimizing the rate of network coding operations can be formulated as a linear programming problem.

As mentioned in Chapter 1, the problem of supporting multicast over coded packet networks can be divided into two disjoint design problems. The first problem determines the nodes in the network that perform network coding, which is also referred in the liter-
nature as the problem of determining the *coding subgraph*. The second problem determines the code that should be used on the coding subgraph. A linear programming formulation is used in [21] to construct the minimum cost coding subgraph. A distributed algorithm is proposed in [91] to construct the maximum utility coding subgraph. It has been shown that random codes are useful in achieving the desired capacity and facilitating distributed implementation [14, 92].

For the problem of determining the minimum resource coding subgraph, results in [93] show that for acyclic networks with two sources, the number of nodes required to perform network coding is always less than the number of sinks. However, the results cannot be extended to an arbitrary number of sources. The problem of maximizing network throughput while using the minimum set of nodes to perform network coding is inherently an NP-hard problem. In [90], a heuristic based on a genetic algorithm is used to find a suboptimal solution. In [23], the problem of minimizing network coding resources for wired networks is formulated as a linear programming problem. The objective is to minimize the rate at which packets undergo coding in the network. Although this method can provide a distributed solution, it cannot be applied to wireless networks with broadcast links.

The problem of constructing the multicast tree for routed packet networks is equivalent to the problem of finding the Steiner tree, which is NP-complete. Due to the complexity of this problem, several heuristics are proposed. A multicast incremental power algorithm is proposed in [94] to construct a multicast tree for wireless networks while exploiting the broadcast nature of the wireless links. Interestingly, the problem of constructing the coding subgraph can be formulated as a linear programming problem for coded packet networks. This problem can be solved in polynomial time and it is suitable for distributed implementation.

In this chapter, we formulate the maximum-lifetime minimum-resource (MLMR) coding
subgraph problem to support multicast traffic in WSNs with network coding. We propose an MLMR algorithm to determine the rate of performing network coding in each sensor node. The contributions of this chapter are as follows:

- We propose the coding flow variables, which are a new representation for the set of information flow and actual rate variables. The coding flow variables enable us to determine the type of operation(s) (i.e., forwarding, replication, or network coding) performed in each wireless sensor node. These variables can also determine the rate of performing network coding and replication in each sensor node.

- We formulate a linear programming optimization problem for joint maximization of the network lifetime and minimization of the rate of performing network coding. We propose the MLMR algorithm which enables us to study the tradeoff between maximizing the network lifetime and minimizing the rate of coding operations.

- We present two model extensions including the case when the objective is to minimize the number of sensors performing network coding and the case when only a fixed number of sensor nodes are capable of performing network coding.

- We investigate the lifetime-resource tradeoff in the network. Simulation results show that the network lifetime can considerably be improved when the cost of performing network coding is relatively low compared to the case that this cost is high for intermediate nodes in the network.

The rest of this chapter is organized as follows: In Section 3.1, we introduce the notations and system model. In Section 3.2, we describe our proposed coding flow variables and formulate the joint maximum lifetime minimum resource coding subgraph problem. We present our proposed MLMR algorithm. The model extensions are also described. In
Section 3.3, we evaluate the performance of our proposed MLMR algorithm and compare it with three other algorithms [23], [95], [96]. A summary is given in Section 3.4.

### 3.1 System Model

We represent the WSN as a hypergraph, which can effectively model the broadcast nature of wireless links. Let $H = (\mathcal{V}, \mathcal{A})$ denote a directed hypergraph, where $\mathcal{V}$ is the set of nodes and $\mathcal{A}$ is the set of hyperarcs. Let $\mathcal{N}_i$ denote the set of neighbors of node $i \in \mathcal{V}$. We assume that is given in the network. It can be calculated by the sink using the geographical information of the sensor node. A hyperarc is a pair $(i, \mathcal{J})$ representing a broadcast link from node $i$ to the subset of neighboring nodes $\mathcal{J} \subset \mathcal{N}_i$. Let $\mathcal{I}_i$ denote the power set (i.e., the set of all subsets of a given set) of set $\mathcal{N}_i$ except the empty set. The set of hyperarcs in the network can be represented as

$$\mathcal{A} = \{(i, \mathcal{J}) \mid i \in \mathcal{V}, \ \mathcal{J} \in \mathcal{I}_i\}.$$ 

Let $\mathcal{A}_{\text{in}}^i$ denote the set of all incoming hyperarcs of node $i \in \mathcal{V}$. These are the hyperarcs including node $i$ as the destination:

$$\mathcal{A}_{\text{in}}^i = \{(j, \mathcal{I}) \in \mathcal{A} \mid i \in \mathcal{I}\}.$$ 

The cardinality of hyperarc $(i, \mathcal{J})$ is equal to $|\mathcal{J}|$, which is the number of elements in set $\mathcal{J}$. Let $\mathcal{S}$ denote the set of source nodes in the network and $\mathcal{D}^s$ denote the set of sinks to which source $s \in \mathcal{S}$ sends data. We call source $s$ with its respective sinks as session $s$. Let $g_{i,j}^{sd}$ denote the information flow rate of session $s$ from node $i$ to node $j \in \mathcal{J}$ through hyperarc $(i, \mathcal{J})$ towards sink $d \in \mathcal{D}^s$. Let $z_{i,j}^s$ denote the actual rate at which the data of
session $s$ is sent on hyperarc $(i, \mathcal{J})$. This variable represents the data rate sent on hyperarc $(i, \mathcal{J})$ after performing coding at node $i$. The actual rate vector $\mathbf{z} = \{z_{i,\mathcal{J}}^s\}$ is called the \textit{coding subgraph} of the network.

When node $i$ performs network coding on packets of session $s$, it combines several incoming packets of this session destined for different sinks and transmits one coded packet. Such a coded packet should reach all the sinks that the original packets had been destined for. This happens by means of replication on the route of the coded packet to the sinks of session $s$. Node $i$ sends a packet on hyperarc $(i, \mathcal{J})$ with cardinality greater than one if the packet needs to reach different sinks through different neighbors in set $\mathcal{J}$. In case that the coded packet is sent on a hyperarc and this coded packet traverses different routes to the sinks, replication is not performed. The following lemma shows that it is not necessary to consider all the hyperarcs in the network graph to obtain the coding subgraph.

\textbf{Lemma 3} \textit{It is not necessary to consider hyperarcs with cardinality greater than $|\mathcal{D}^s|$ (i.e., the number of sinks of session $s$) to obtain the coding subgraph of session $s$.}

\textbf{Proof} For session $s \in \mathcal{S}$, let $b_{i,\mathcal{J}}^{sD}$ denote a packet sent on the hyperarc $(i, \mathcal{J})$ destined for the sinks in $D \subset \mathcal{D}^s$. For node $i$ and outgoing hyperarc $\mathcal{J} \in \mathcal{I}_i$, the coded packet $b_{i,\mathcal{J}}^{sD}$ is created by combining several packets as follows:

$$
  b_{i,\mathcal{J}}^{sD} = \sum_{j \in \mathcal{J}} \omega_{ij} b_{ij}^{sD_j},
$$

(3.1)

where $b_{ij}^{sD_j}$ is a packet of session $s$ destined for the set of destinations in $D_j \subset \mathcal{D}^s$ and it is sent from node $i$ to $j$ if hyperarc $(i, \mathcal{J})$ is not used. If $|D_j| > 1$, then the packet $b_{ij}^{sD_j}$ is a coded packet. $\omega_{ij}$ is the coding coefficient (chosen from a Galois field) that node $i$ used for packet $b_{ij}^{sD_j}$. We notice that set $D$ is the union of sets $D_j$ (i.e., $D = \bigcup_{j \in \mathcal{J}} D_j$). Since the packets destined for the same destinations cannot be combined, the sets $D_j$
should be disjoint for different \( j \in J \). The number of such packets is \(|J|\) while the number of destinations in session \( s \) is \(|D^s|\). Hence, the maximum number of packets which are combined can be at most \(|D^s|\) (i.e., \(|J| \leq |D^s|\)). Therefore, node \( i \) does not send any packet on hyperarc \((i, J)\) if \(|J| > |D^s|\) (i.e., \(z_{iJ}^s = 0\)). Consequently, it is not necessary to consider these hyperarcs to obtain the coding subgraph of session \( s \).

Based on Lemma \( \text{Lemma 3} \) and the definition of \( \mathcal{A}, \mathcal{A}_i^{\text{in}}, \) and \( \mathcal{I}_i \), for session \( s \), we define the following sets:

\[
\mathcal{A}^s = \{(i, J) \mid (i, J) \in \mathcal{A}, \ |J| \leq |D^s|\},
\]

(3.2)

\[
\mathcal{A}_i^{\text{in}, s} = \{(j, I) \mid (j, I) \in \mathcal{A}_i^{\text{in}}, \ |I| \leq |D^s|\}, \ i \in V
\]

(3.3)

\[
\mathcal{I}_i^s = \{J \in \mathcal{I}_i \mid |J| \leq |D^s|\}, \ i \in V.
\]

(3.4)

In WSNs, each node has a large number of neighboring nodes while the number of sinks gathering information in each session is limited. Therefore, the number of variables in the system can be considerably reduced by removing the hyperarcs with cardinality greater than \(|D^s|\).

### 3.1.1 System Constraints

There are several constraints in our problem formulation. Let \( r_s \) denote the data generation rate by source node \( s \in S \). The flow conservation constraint in node \( i \in V \) for data sent from source \( s \) towards sink \( d \in D^s \) is

\[
\sum_{J \in \mathcal{I}_i^s} \sum_{j \in J} g^{sd}_{iJ} - \sum_{(j,I) \in \mathcal{A}_i^{\text{in},s}} g^{sd}_{jI} = \beta_i,
\]

(3.5)
Chapter 3. Lifetime-Resource Tradeoff for Multicast Traffic in WSNs

where

$$\beta_i = \begin{cases} 
    r_s, & \text{if } i = s, \\
    -r_s, & \text{if } i = d, \\
    0, & \text{otherwise.}
\end{cases}$$

This constraint guarantees that the outgoing information flow rate from node $i$ is equal to the input rate plus the rate of information generated by node $i$ for session $s$.

By the flow-sharing property of network coding and the broadcast nature of a wireless channel, the actual flow rate on each hyperarc needs only be the maximum of the individual sinks flows. The actual rate constraint for session $s$ on hyperarc $(i, J)$ can be expressed as

$$z_{i, J}^s \geq \sum_{j \in J} g_{i, J, j}^{s, d}, \quad \forall d \in D^s. \ (3.6)$$

We extend the protocol interference model in multi-hop wireless networks to hypergraph. Let $\delta(i, j)$ and $\kappa(i, j)$ denote the communication range and interference range between nodes $i$ and $j$, respectively. Let $\delta(i, J) = \max_{j \in J} \delta(i, j)$ and $\kappa(i, J) = \max_{j \in J} \kappa(i, j)$ denote the communication range and interference range between node $i$ and the farthest node in set $J$, respectively. Two hyperarcs $(i_1, J_1)$ and $(i_2, J_2)$ interfere with each other (i.e., cannot be active simultaneously) if any of the following conditions are satisfied: (a) $i_1 = i_2$, (b) there exists $j \in J_1$ such that $\delta(i_2, j) \leq \kappa(i_2, J_2)$, (c) there exists $j \in J_2$ such that $\delta(i_1, j) \leq \kappa(i_1, J_1)$.

The concept of contention graph and maximal clique can also be extended to a hypergraph. In a contention hypergraph, each node corresponds to a hyperarc. There is an edge between two hyperarcs if they cannot be active simultaneously. The maximal cliques in a hypergraph can be determined accordingly. Let $C$ denote the ordered set of all maximal cliques, with $C_k$ representing the $k^{th}$ element in set $C$. The link scheduling
Chapter 3. Lifetime-Resource Tradeoff for Multicast Traffic in WSNs

Constraints are

\[
\sum_{\{(i,J), s | (i,J) \in C_k, s \in S, \text{ and } (i,J) \in \tilde{A}^s\}} z_{i,J}^s/c_{i,J} \leq \lambda_k, \quad k = 1, \ldots, |C| \quad (3.7)
\]

and

\[
\sum_{k=1}^{|C|} \lambda_k \leq 1, \quad (3.8)
\]

where \( \lambda_k \) represents the fraction of time that hyperarcs in set \( C_k \) can be active, \( c_{i,J} = \min_{j \in J} c_{ij} \) is the capacity of hyperarc \((i,J)\), and \( c_{ij} \) is the capacity of the link between node \( i \) and \( j \).

The network lifetime is defined as the time when the first node runs out of its energy \([6, 70, 98]\). Whenever a sensor node is no longer operational, the coding subgraph is updated based on the new network topology and the remaining energy of the active sensor nodes. Nodes employ power control to save energy. The transmission power of a sender is adjusted based on the distance of the two nodes. Let \( E_i \) denote the initial energy of node \( i \) and \( p_{i,J} \) represent the power consumed to transmit one bit of data on hyperarc \((i,J)\). This is equal to the power consumed to transmit data to the farthest node in set \( J \).

Thus, \( p_{i,J} = \max_{j \in J} p_{ij} \), where \( p_{ij} \) is the power consumed to transmit one bit of data from node \( i \) to \( j \). For node \( j \) to receive the bit successfully, it requires that the received power \( p_r = p_t/d(i,j)^a > \eta_2 \), where \( d(i,j) \) is the distance between nodes \( i \) and \( j \), \( a \) is the path loss exponent with \( 2 \leq a \leq 4 \), and \( \eta_2 \) is a threshold value. The total transmit power is given by \([1, 70, 93, 98]\): \( p_{ij} = \eta_1 + \eta_2 d(i,j)^a \), where \( \eta_1 \) is the power consumed in the transmitter circuitry. Given the maximum transmission power \( p_{\text{max}} \), the corresponding communication range \( \delta(i,j) \) can be determined.
Given $E_i$ and $p_{i,J}$, the lifetime of node $i$ under actual rate vector $z = \{z_{i,J}^s\}$ is

$$
\Gamma_i(z) = \frac{E_i}{\sum_{s \in S} \sum_{J \in I^s_i} p_{i,J} z_{i,J}^s}, \quad \forall \ i \in \mathcal{V}.
$$

(3.9)

Let $q$ denote an upper bound on the inverse of the lifetime of all the nodes (i.e., $1/\Gamma_i(z) \leq q$, for all $i \in \mathcal{V}$). Minimizing the variable $q$ is equivalent to maximizing the minimum node’s lifetime in the network (i.e., $\max \ \min_{i \in \mathcal{V}} \Gamma_i(z) = \min q$). Using equation (3.9), the network lifetime constraint can be expressed as

$$
\sum_{s \in S} \sum_{J \in I^s_i} p_{i,J} z_{i,J}^s \leq E_i q, \quad \forall \ i \in \mathcal{V}.
$$

(3.10)

Considering the constraints introduced on the information flow rate and the actual rate variables, the coding subgraph of the network is characterized in [7, Theorem 2], which we reproduce here, slightly adapted:

**Theorem 2** The actual rate vector $z = \{z_{i,J}^s\}$ is part of the set constructed by the constraints (3.5)–(3.8) and (3.10) if and only if there exists a network code for each session $s$ that sets up a multicast connection at a rate arbitrarily close to $r_s$.

Since designing codes for optimal inter-session coding for practical wireless networks is an active and open research area, we restrict network coding operation within sessions (i.e., intra-session coding) in this chapter. The only relation between various sessions is imposed by the link scheduling constraint in (3.7)–(3.8). In practice, random linear network coding can achieve the rate of multicast connection $r_s$ while it can be performed distributively in the network. Nodes on the coding subgraph which are responsible for network coding select the coding coefficients randomly from a Galois field $GF(2^u)$ and combine the incoming packets with the corresponding rate. Sinks can decode the received packets with probability close to one for large values of $u$. [99]
3.2 Problem Formulation

Our goal is to formulate the joint problem of maximizing the network lifetime and minimizing the rate of performing coding operations. On one hand, higher rates of coding operations require more resources and capabilities in the system. On the other hand, a larger number of packets undergoing network coding reduce the total number of packet transmissions in the network. This decreases the average power consumption. Consequently, network lifetime can be increased when the rate of coding operations increases.

To formulate this problem, it is necessary to determine the rate of coding operations at each network device. The information flow and actual rate variables alone are not sufficient to determine the corresponding rates on the incoming and outgoing hyperarcs of each node.

3.2.1 Maximum Lifetime Minimum Resource Problem

In this section, we introduce a new set of variables which allows us to determine the type of the operations and the rate at which these operations are performed by each node. Let $\mathcal{P}^s$ denote the power set of $\mathcal{D}^s$ except the empty set. For simplicity, we label the sinks of session $s$ by positive integers. For example, if session $s$ has three sinks, then $\mathcal{D}^s = \{1, 2, 3\}$ and $\mathcal{P}^s = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$. For hyperarc $(i, \mathcal{J})$, we fix the ordering of the nodes in $\mathcal{J}$. Let $j_m$ denote the $m^{th}$ member of $\mathcal{J}$.

We now introduce the coding flow variables $n^s_{i, \mathcal{J}}$. We show that using this variable, one can obtain the rate of performing coding at each sensor node. For session $s$, hyperarc $(i, \mathcal{J})$, and $P_1, \ldots, P_{|\mathcal{J}|} \in \mathcal{P}^s$, let $n^s_{i, \mathcal{J}}(P_1, \ldots, P_{|\mathcal{J}|})$ represent the rate of transmission of packets which are forwarded towards sinks in set $P_m$ by neighbor $j_m$ for $m = 1, \ldots, |\mathcal{J}|$. These packets have to reach the sinks in set $\bigcup_{m=1}^{|\mathcal{J}|} P_m$ via different neighbors of node $i$. The size of the coding variable for each hyperarc depends on the cardinality of the hyperarc. For example, for hyperarc $(i, \mathcal{J})$ with cardinality two, the coding flow variable is $n^s_{i, \mathcal{J}}(P_1, P_2)$,
where $P_1$ and $P_2 \in \mathcal{P}^s$.

For illustration purpose, consider Fig. 3.1. Nodes $h_1$ and $h_2$ has two outgoing links while node $h_3$ has one outgoing hyperarc. Node $s$ is the source and nodes $d_1$ and $d_2$ are the sinks. Assume that node $h_3$ sends one packet per second towards $d_1$ and one packet per second towards $d_2$. Node $h_3$ can combine these packets and send one packet over hyperarc $(h_3, \mathcal{J})$ where $\mathcal{J} = \{d_1, d_2\}$. Then, we have $n_{i,\mathcal{J}}^s(P_1, P_2) = 1$, where $P_1 = \{d_1\}$ and $P_2 = \{d_2\}$.

A coded packet is routed towards a sink only from one path. It implies that for coding flow variable $n_{i,\mathcal{J}}^s(P_1, \ldots, P_{|\mathcal{J}|})$, the sets $P_1, \ldots, P_{|\mathcal{J}|}$ are disjoint (i.e., $P_a \cap P_b = \emptyset$ for $a, b = 1, \ldots, |\mathcal{J}|, a \neq b$). Let $P_{i,\mathcal{J}}^s$ denote the set of all disjoint subsets of $\mathcal{P}^s$ with $|\mathcal{J}|$ members. Let $P_{i,\mathcal{J}}^s(d, m)$ denote the set of members of $P_{i,\mathcal{J}}^s$ while $d \in P_m$. These sets can be defined as follows:

$$P_{i,\mathcal{J}}^s = \{(P_1, \ldots, P_{|\mathcal{J}|}) \mid P_1, \ldots, P_{|\mathcal{J}|} \in \mathcal{P}^s \text{ and } P_a \cap P_b = \emptyset \text{ for } a, b = 1, \ldots, |\mathcal{J}|, a \neq b\},$$
\[ P_{i,J}^s(d,m) = \{(P_1, \ldots, P_{|J|}) \in P_{i,J}^s \mid d \in P_m\}, \quad m = 1, \ldots, |J|. \quad (3.11) \]

For example, consider hyperarc \((i, J)\) with cardinality two (i.e., \(J = \{j_1, j_2\}\)) and session \(s\) with two sinks, namely sink 1 and 2. Then, we have \(P_{i,J}^s = \{\{(1), \{2\}\}, \{(2), \{1\}\}\}\), \(P_{i,J}^s(1, 1) = P_{i,J}^s(2, 2) = \{\{(1), \{2\}\}\}\), and \(P_{i,J}^s(1, 2) = P_{i,J}^s(2, 1) = \{\{(2), \{1\}\}\}\).

We now show how to relate the information flow variables \(g_{s_{i,J}}^{sd}\) to the coding flow variables \(n_{s_{i,J}}^i\). For simplicity, we assume that the size of each packet is \(M\) bits. The information flow variable \(g_{s_{i,J}}^{sd}\) is equal to the packet size (i.e., \(M\)) times the total rate at which packets which have to reach sink \(d\) of session \(s\) through neighbor \(j_m\) (i.e., \(\forall n_{s_{i,J}}^i(P_1, \ldots, P_{|J|})\) while \(d \in P_m\)). The relation between coding flow and information flow variables can be expressed as

\[ g_{s_{i,J}}^{sd} = M \sum_{(P_1, \ldots, P_{|J|}) \in P_{i,J}^s(d,m)} n_{s_{i,J}}^i(P_1, \ldots, P_{|J|}). \quad (3.12) \]

For illustration purpose, consider the following example. Assume that there are three sinks in session \(s\), namely 1, 2, 3. For hyperarc \((i, J)\) with cardinality two (i.e., \(J = \{j_1, j_2\}\)), equation (3.12) can be written for sink 1 and neighbors \(j_1\) and \(j_2\) as follows:

\[ \frac{g_{s_{i,J}}^{1}}{M} = \sum_{(P_1, P_2) \in P_{i,J}^s(1,1)} n_{s_{i,J}}^i(P_1, P_2) \quad (3.13) \]

\[ = \sum_{\{P_1, P_2 \in P^s \mid P_1 \cap P_2 = \emptyset, 1 \in P_1\}} n_{s_{i,J}}^i(P_1, P_2) \]

\[ = n_{s_{i,J}}^i(\{1\}, \{2\}) + n_{s_{i,J}}^i(\{1\}, \{3\}) + n_{s_{i,J}}^i(\{1\}, \{2, 3\}) + n_{s_{i,J}}^i(\{1, 2\}, \{3\}) \]

\[ + n_{s_{i,J}}^i(\{1, 3\}, \{2\}), \]

61
The actual rate of hyperarc \((i, J)\) is the summation of the rate of packet transmissions towards all the sinks (i.e., all the coding flow variables of that hyperarc multiplied by the packet size) as:

\[
z^s_{i,J} = M \sum_{(P_1, \ldots, P_{|J|}) \in P^s_{i,J}} n^s_{i,J}(P_1, \ldots, P_{|J|}).
\]  

Consider the following example. Assume that there are three sinks in session \(s\), namely 1, 2, 3. Consider hyperarc \((i, J)\) with cardinality two (i.e., \(J = \{j_1, j_2\}\)). The actual rate on this hyperarc is the summation of all coding flow variables multiplied by the packet size as follows:

\[
z^s_{i,J} = M \left( n^s_{i,J}(\{1\}, \{2, 3\}) + n^s_{i,J}(\{2\}, \{1, 3\}) + n^s_{i,J}(\{3\}, \{1, 2\}) + n^s_{i,J}(\{1, 2\}, \{3\}) 
+ n^s_{i,J}(\{1, 3\}, \{2\}) + n^s_{i,J}(\{2, 3\}, \{1\}) \right).
\]

Fig. 3.2 shows a sample network with three sinks and one source node. The source is generating information with a rate of 2 units per second and the data is sent towards all three sinks. The packet size is assumed to be one. The information flow, actual rate, and coding flow values of all hyperarcs plus the paths that packets travel are shown on each hyperarc.

Equations (3.12) and (3.15) imply that the information flow and actual rate variables...
Figure 3.2: Network with three sinks ($d_1$, $d_2$, $d_3$) and one source node $s$ with data generation rate of 2 units per second. (a) Information flow values, (b) coding subgraph (actual rate values), (c) coding flow values, (d) multipath packet transmission.

are dependent variables while the coding flow variables are independent variables. We keep these dependent variables since they make the presentation of the formulation clear and easier to understand.

The coding flow variables provide information on the packet transmission rates for different disjoint sets of sinks on each hyperarc. We can determine the type of operation performed in node $i$ by inspecting variables $n^s_{i,j}$. Let $\phi^s_i(R)$ denote the total rate at which node $i$ sends coded packets which have to reach sinks in set $R \in P^s$ on the outgoing
Chapter 3. Lifetime-Resource Tradeoff for Multicast Traffic in WSNs

Let $\phi_s^i(R)$ denote the rate at which these packets are sent on hyperarc $(i, \mathcal{J})$. Variable $\phi_s^i(R)$ is the summation of variables $\phi_s^i(J)$ over all outgoing hyperarcs of node $i$ (i.e., $(i, \mathcal{J}), \mathcal{J} \in \tilde{I}_s^i$). The coding flow variable $n_s^i(P_1, \ldots, P_{|\mathcal{J}|})$ is defined as the rate at which packets which have to reach sinks in set $\bigcup_m P_m$ are sent over hyperarc $(i, \mathcal{J})$. Therefore, the rate at which node $i$ sends coded packets which have to reach sinks in set $R \in \mathcal{P}^s$ over hyperarc $(i, \mathcal{J})$ is equal to the summation of coding flow variables $n_s^i(P_1, \ldots, P_{|\mathcal{J}|})$ such that $\bigcup_m P_m = R$. This can be expressed as

$$
\phi_s^i(R) = \sum_{\mathcal{J} \in \tilde{I}_s^i} \phi_s^i(J) = \sum_{\mathcal{J} \in \tilde{I}_s^i} \sum_{\{P_1, \ldots, P_{|\mathcal{J}|}\} \in \mathcal{P}_s^i, \bigcup_m P_m = R} n_s^i(P_1, \ldots, P_{|\mathcal{J}|}).
$$

(3.16)

Let $\psi_s^i(R)$ denote the rate at which node $i$ receives coded packets which have to reach the sinks in set $R \in \mathcal{P}^s$. This is equal to the summation of rate of those packets over all incoming hyperarcs of node $i$ (i.e., $(j, I) \in \tilde{A}_s^{in,i}$). For hyperarc $(j, I)$, which is an incoming hyperarc of node $i$, the coding variable $n_s^j(P_1, \ldots, P_{|I|})$ contains the information of sinks in set $R$ for node $i$ if there exists $v \in \{1, \ldots, |I|\}$ such that $i = I_v$ (i.e., $i$ is the $v$th member of set $I$) and $P_v = R$. Therefore, we have

$$
\psi_s^i(R) = \sum_{(j, I) \in \tilde{A}_s^{in,i}} \sum_{\{P_1, \ldots, P_{|I|}\} \in \mathcal{P}_s^j, v \in \{1, \ldots, |I|\}, P_v = R, i = I_v} n_s^j(P_1, \ldots, P_{|I|}).
$$

(3.17)

Variables $\phi_s^i(R)$ and $\psi_s^i(R)$ represent the rate at which node $i$ sends and receives packets destined for the sinks in set $R$. The value of $\phi_s^i(R)$ is less than $\psi_s^i(R)$ if node $i$ combines received packets destined for sinks in set $R$ and creates a coded packet. On the other hand, the value of $\phi_s^i(R)$ is greater than $\psi_s^i(R)$ if node $i$ replicates a coded packet towards the sinks in set $R$. From these variables, we can determine the rate at which node $i$ performs coding or replication on the packets destined for different sets of sinks. Let $\mathcal{T}^s$ denote the set of all collections of two or more disjoint subsets of $\mathcal{P}^s$. Any replication or
coding operation performed at each node corresponds to one of the members of $\mathcal{T}^s$. Node $i$ performs replication with respect to set $T \in \mathcal{T}^s$ if it transmits $|T|$ copies of a coded packet to $|T|$ different neighbors. Each copy is transmitted towards the sinks in a member of $T$. Node $i$ performs coding with respect to set $T$ if it combines $|T|$ incoming packets which have to reach sinks in members of $T$ and sends one coded packet. For example, for a network with three sinks, we have $\mathcal{T}^s = \{\{1\},\{2\}, \{1,2\}\}$. If node $i$ receives three packets which have to reach sinks 1, 2, and 3 and creates one coded packet which have to reach the sinks in set $\{1,2,3\}$, then it performs coding with respect to set $\{\{1\},\{2\},\{3\}\}$. As another example, assume that node $i$ receives one packet which has to reach the sinks in set $\{1,2,3\}$ and replicates two packets. One of the packets is sent towards sinks $\{1,2\}$ and the other one is sent towards sink 3. In this case, node $i$ performs replication with respect to set $\{\{1\},\{2\},\{3\}\}$.

Let $\mu^s_i(T)$ and $\nu^s_i(T)$ denote the rate of performing replication and network coding with respect to set $T \in \mathcal{T}^s$, respectively. The final step in determining the rate of performing network coding is to relate variables $\mu^s_i(T)$ and $\nu^s_i(T)$ to variables $\phi^s_i(R)$ and $\psi^s_i(R)$. In general, for node $i \in \mathcal{V}$, session $s \in \mathcal{S}$, and set $R \in \mathcal{P}^s$, we have

$$
\phi^s_i(R) - \psi^s_i(R) = \sum_{\{T \in \mathcal{T}^s \mid R \in T\}} (\mu^s_i(T) - \nu^s_i(T)) - \sum_{\{T \in \mathcal{T}^s \mid \cup_{t \in T} t = R\}} (\mu^s_i(T) - \nu^s_i(T)).
$$

(3.18)

Equation (3.18) which quantifies the amount of coding and replication operations is a flow conservation constraint. In (3.18), the output rate minus the input rate is equal to the rate at which packets are generated for packets which have similar destinations. In (3.18), the left hand side is the difference between the output and input rate for the packets which have to reach the sinks in set $R$. On the right hand side, the term $\sum_{\{T \in \mathcal{T}^s \mid R \in T\}} \mu^s_i(T)$ denotes the rate of replication operations creating outgoing packets that have to reach the sinks in
set \( R \), the term \( \sum_{\{T \in \mathcal{T}^s \mid R \in T\}} \nu_i^s(T) \) denotes the rate of coding operations performed on incoming packets that have to reach the sinks in set \( R \), the term \( \sum_{\{T \in \mathcal{T}^s \mid \cup_{i \in T^t = R}\}} \mu_i^s(T) \) denotes the rate of replication operations performed on incoming packets that have to reach the sinks in set \( R \), and the last term \( \sum_{\{T \in \mathcal{T}^s \mid \cup_{i \in T^t = R}\}} \nu_i^t(T) \) denotes the rate of coding operations creating outgoing packets that have to reach the sinks in set \( R \).

As an example, consider the network in Fig. 3.2 (d). The source transmits two packets \( b_1 \) and \( b_2 \) towards the sinks. Node 2 performs routing, nodes 6 and 7 broadcast received packets using the corresponding outgoing hyperarc, nodes 1 and 3 replicate packets, and nodes 4 and 5 perform network coding. Node 1 receives \( b_1 \) destined for sinks \( d_1 \) and \( d_2 \). It replicates one copy of the packet to sink \( d_1 \) and one to node 4. The second packet should reach sink \( d_2 \). For this node, we have \( \mu_1(\{1\}, \{2\}) = 1 \). Similarly, we have \( \mu_3(\{2\}, \{3\}) = 1 \) for node 3. Node 4 receives packet \( b_1 \) destined for \( d_2 \) and packet \( b_2 \) destined for \( d_1 \). It combines these packets using coding coefficients 1 and 2 for \( b_1 \) and \( b_2 \), respectively. A coded packet is transmitted to node 6. Therefore, we have \( \nu_4(\{1\}, \{2\}) = 1 \). Similarly, node 5 uses coding coefficients 2 and 1 to combine \( b_1 \) and \( b_2 \), respectively. We have \( \nu_7(\{2\}, \{3\}) = 1 \) for node 3.

Note that in (3.18), the outgoing and incoming rates of packet transmission and reception (i.e., \( \psi_i^s(R) \) and \( \phi_i^s(R) \), respectively) are given. These rates are obtained using the coding flow variables in (3.10) and (3.17). Moreover, equations (3.12) and (3.15) relate the coding flow variables to the actual rate and information flow rate. Therefore, if the actual rate and flow information rate satisfy the constraints in (3.5) – (3.8) and (3.10), then the rates of coding and replication obtained from (3.18) will also satisfy those constraints. From Theorem 2, it is guaranteed that there exists a network code achieving these rates. Linear random network coding can be used to achieve the rates obtained from (3.18).

As an example, we show how linear random network coding can be used in the network
while nodes perform both coding and routing. Consider the network in Fig. 3.1. Assume that the source transmits two packets per second towards sinks $d_1$ and $d_2$. One of them is sent via node $h_1$ and the other is sent via node $h_2$. Nodes $h_1$ and $h_2$ replicate those packets to node $h_3$ and also to sinks $d_1$ and $d_2$, respectively. Every second, node $h_3$ receives two different packets destined for sinks $d_1$ and $d_2$. Node $h_3$ can either use network coding and transmit one coded packet on a hyperarc including sinks $d_1$ and $d_2$, or it can perform routing and transmit them separately. Assume that as a solution of problem (3.20), the rate of performing network coding at node $h_3$ is $1/2$ per second. In this case, node $h_3$ transmits a coded packet every two seconds while transmitting two separate packets at other times. The coded packet is constructed by randomly selecting two coefficients and combining two packets using the coefficients. In this case, node $h_3$ performs both network coding and routing with rate $1/2$.

Equation (3.18) shows a linear relation between variables $\phi_s^i$ and $\psi_s^i$ and also $\mu_s^i$ and $\nu_s^i$. This equation is internal to each node. Both the input and output coding rates and the rate of coding and replications are variables of the joint problem. Therefore, given $\phi_s^i$ and $\psi_s^i$, the values of $\mu_s^i$ and $\nu_s^i$ cannot be uniquely determined using this set of linear equations. We select the objective function of the joint problem such that we can find the optimal rate of coding operations among feasible solutions. The total rate of performing network coding is

$$\sum_{s \in S} \sum_{i \in V \setminus \{s\}} \sum_{T \in T^s} \nu_s^i(T).$$

Note that source $s$ does not perform coding on the packets that it generates. In the maximum lifetime minimum resource coding subgraph problem, the objective is to

$$\text{minimize } q + \gamma \sum_{s \in S} \sum_{i \in V \setminus \{s\}} \sum_{T \in T^s} \nu_s^i(T),$$

(3.19)
where $\gamma$ is the cost of performing network coding. The unit of $\gamma$ depends on the unit of $\beta_i$ in (3.13). The value of the tuning parameter $\gamma$ determines the effect of each part in the joint objective function. The feasible set of the joint coding subgraph problem is constructed by equations (3.5), (3.7), (3.8), (3.10), (3.12), and (3.15)–(3.18). Problem (3.20) is the maximum lifetime minimum resource coding subgraph problem. This is a linear programming problem. It can be solved by various techniques such as the simplex
method.

\[
\begin{align*}
\text{minimize} & \quad q + \gamma \sum_{s \in S} \sum_{i \in \mathcal{V}(s)} \sum_{T \in T^s} \nu_i^s(T) \\
\text{subject to} & \quad \sum_{J \in \mathcal{T}_i} \sum_{j \in J} g_{i,j}^{sd} - \sum_{(j,l) \in \tilde{A}_{i,m,s}} g_{j,l}^{sd} = \beta_i, \quad \forall i \in \mathcal{V}, \ d \in \mathcal{D}^s, \ s \in S \\
& \quad \sum_{s \in S} \sum_{J \in \mathcal{T}_i} p_{i,J} z_{i,J}^s \leq E_i q, \quad \forall i \in \mathcal{V} \\
& \quad g_{i,J|m}^{sd} = \sum_{(P_1,\ldots,P_{|J|}) \in P_{i,J}^{s}(d,m)} n_{i,J}^s(P_1,\ldots,P_{|J|}), \quad \forall (i,J) \in \tilde{A}^s, m = 1,\ldots,|\mathcal{J}|, d \in \mathcal{D}^s, s \in S \\
& \quad z_{i,J}^s = M \sum_{(P_1,\ldots,P_{|J|}) \in P_{i,J}^s} n_{i,J}^s(P_1,\ldots,P_{|J|}), \quad \forall (i,J) \in \tilde{A}^s, s \in S \quad (3.20) \\
& \quad \phi_i^s(R) = \sum_{J \in \mathcal{T}_i} \sum_{\{(P_1,\ldots,P_{|J|}) \in P_{i,J}^s \mid \cup_{m} P_m = R\}} n_{i,J}^s(P_1,\ldots,P_{|J|}), \quad \forall i \in \mathcal{V}, \ s \in \mathcal{S}, \ R \in \mathcal{P}^s \\
& \quad \psi_i^s(R) = \sum_{(j,l) \in \tilde{A}_{i,m,s}} \sum_{\{(P_1,\ldots,P_{|J|}) \in P_{j,l}^s \mid \cup_{i=1}^{m} P_i = R, \ i = I_i\}} n_{j,l}^s(P_1,\ldots,P_{|J|}), \quad \forall i \in \mathcal{V}, \ s \in \mathcal{S}, \ R \in \mathcal{P}^s \\
& \quad \phi_i^s(R) = \psi_i^s(R) + \sum_{\{T \in T^s \mid R \in T\}} \left(\mu_i^s(T) - \nu_i^s(T)\right) - \sum_{\{T \in T^s \mid \cup_{i \in T} i = R\}} \left(\mu_i^s(T) - \nu_i^s(T)\right), \forall R \in \mathcal{P}^s, i \in \mathcal{V}, s \in \mathcal{S} \\
& \quad n_{i,J}^s(P_1,\ldots,P_{|J|}^s) \geq 0, \ \mu_i^s(T) \geq 0, \ \nu_i^s(T) \geq 0, \ \lambda_k \geq 0, \ q \geq 0, \\
& \quad \forall (i,J) \in \tilde{A}^s, (P_1,\ldots,P_{|J|}^s) \in \mathcal{P}^s, T \in \mathcal{T}^s, s \in \mathcal{S}, k = 1,\ldots,|\mathcal{C}|.
\end{align*}
\]

From Theorem 3.2, it follows that the solution of problem (3.20) is the maximum lifetime minimum resource coding subgraph for an asymptotically achievable rate for multicast sessions. Although all the nodes are capable of performing network coding, only some of them perform network coding in the optimal coding subgraph.
In problem (3.20), only variables \( n_{ JS}^s \), \( q \), \( \mu_i^s \), \( \nu_i^s \), and \( \lambda_k \) are independent variables. Variables \( g_{d_i J}^{sd} \), \( z_{i J}^s \), \( \phi_i^s \), and \( \psi_i^s \) are auxiliary variables used to simplify the notation and formulation. Also, the second, third, fourth, and fifth constraints can be removed if the variables \( g_{i J}^{sd} \), \( z_{i J}^s \), \( \phi_i^s \), and \( \psi_i^s \) are replaced by variable \( n_{ i J}^s \), accordingly. Thus, the number of constraints and variables of the problem can further be reduced. The number of constraints in problem (3.20) and the number of independent variables \( n_{ i J}^s \), \( \mu_i^s \), \( \nu_i^s \) increase linearly with the number of sessions \(|S|\) and the number of nodes \(|V|\), and they increase exponentially with the number of the sinks \(|D^s|\) of each session \( s \in S \). Thus, our formulation is suitable for networks with a large number of sessions while each session has a small number of sinks. Algorithm 3.1 below is the MLMR algorithm for determining the coding subgraph in the network.

**Algorithm 3.1 - MLMR Algorithm**

1. Collect topology information (including \( V \) and \( A \)), set of sources \( S \), and set of sinks \( D^s \).
2. Determine the ordered set of all maximal cliques \( C \) in the hypergraph.
3. Set variable \( \gamma \) equal to the cost of performing network coding.
4. Solve problem (3.20) to obtain \( n_{ i J}^s \), \( \mu_i^s \), \( \nu_i^s \), \( \lambda_k \), and \( q \).
5. Forward the values of \( n_{i J}^s \) to each sensor node \( i \in V \).

We now describe how to extend the formulation proposed earlier to model some problems.

**3.2.2 Limit the Rate of Coding Operations**

Because of limited computational capabilities and memory, the rate at which intermediate nodes can perform network coding is limited in practice. For an intermediate node \( i \), this depends on the resources available at the node and it is different for various nodes. The total rate at which intermediate node \( i \) performs network coding is \( \sum_{s \in S} \sum_{T \in T^s} \nu_i^s(T) \). The maximum lifetime coding subgraph problem with limited coding rates can be formulated by
setting $\gamma = 0$ in the objective function of problem (3.20) and by including the following constraint to this problem:

$$
\sum_{s \in S} \sum_{T \in T^s} \nu^s_i(T) \leq \eta_i, \quad \forall i \in V,
$$

where $\eta_i$ is the maximum rate at which node $i$ can perform coding operations. We notice that constraint (3.21) is a linear constraint. Therefore, the maximum lifetime coding subgraph problem with a limited coding rates is a linear programming problem which can be efficiently solved by using various polynomial time algorithms.

### 3.2.3 Limit the Number of Coding Devices

Sometimes only a fixed number of sensor devices in the WSN are capable of performing network coding. In this case, there are two types of sensor devices. The first set of sensors called the **coding devices** are capable of performing network coding, routing, and replication. The second set of sensors simply perform routing or replication. The coding devices can be randomly deployed in the network. It is also possible to deploy coding devices in specific locations. The maximum lifetime coding subgraph problem for network with a fixed number of coding devices can be formulated with some modifications on problem (3.20). The objective of this new problem is to maximize the network lifetime, which is equivalent to minimizing the variable $q$. The **maximum lifetime coding subgraph problem with limited number of coding devices** can be formulated by setting $\gamma = 0$ in the objective function in problem (3.20) and by including the following constraint in problem (3.20):

$$
\nu^s_i(T) = 0, \quad \forall i \in V \setminus V_c, \ s \in S, \ T \in T^s,
$$

where $V_c \subset V$ denotes the set of coding devices. The constraint in (3.22) guarantees
that only coding devices perform network coding. The maximum lifetime coding subgraph
problem with a limited number of coding devices is also a linear programming problem. It
can also be efficiently solved by using various polynomial time algorithms.

3.2.4 Minimize the Number of Coding Devices

Consider the case where the number of coding devices is not fixed. However, the coding
devices are more expensive compared to typical wireless sensor nodes. The objective is to
jointly maximize network lifetime and minimize the number of coding devices. In other
words, this formulation answers the following question: What is the minimum set of sensor
devices that should perform network coding in order to achieve a certain level of the network
lifetime? The difference between this problem and the problem presented in the previous
subsection is that the number of coding devices is not fixed.

The problem of finding the coding subgraph while the objective is to jointly maximize
the lifetime and minimize the number of coding devices can be formulated with modification
in problem (3.20). Let \( \alpha_i \) be an indicator variable which denotes whether node \( i \) performs
network coding or not. If \( \alpha_i = 1 \), then node \( i \) performs network coding. This boolean
variable can be mapped to \( \nu^s_i \) for node \( i \) as follows:

\[
\alpha_i = \begin{cases} 
1, & \text{if } \sum_{s \in S} \sum_{T \in T_s} \nu^s_i(T) > 0, \\
0, & \text{otherwise.}
\end{cases}
\]

This problem can be formulated by (a) replacing the objective function in problem
(3.20) by minimizing \( q + \zeta \sum_{i \in V} \alpha_i \), where \( \zeta \) is a design factor, and (b) including the
following constraints in problem (3.20):

\[
\alpha_i \geq \frac{\sum_{s \in S} \sum_{T \in T_s} \nu^s_i(T)}{H_{\max}}, \quad \alpha_i \in \{0, 1\}, \quad (3.23)
\]
where $H_{\text{max}}$ is an upper bound for the rate of performing network coding at all nodes. It is a constant and is used to guarantee that $\left( \sum_{s \in S} \sum_{T \in T^s} \nu_i^s(T) / H_{\text{max}} \right)$ is always less than 1. The formulated problem is a mixed integer linear programming problem with binary variables $\alpha_i$. It can be solved by using branch-and-bound method \cite{51}.

### 3.3 Performance Evaluation

In this section, we evaluate the performance of our proposed MLMR algorithm and compare it with the minimal resource algorithm \cite{23}, the maximum lifetime Steiner tree (MLST) algorithm \cite{95}, and evolutionary minimum resource (EMR) algorithm \cite{96}. For the MLST algorithm, a mixed integer linear programming model is used to construct the maximum lifetime multicast Steiner tree. We use the MOSEK \cite{84} optimization toolbox to solve the problem. In the MATLAB simulations, we consider a square coverage area with each side equal to 70 m. There are 40 sensor nodes randomly deployed in the coverage area. One of them is the source node. There are four sinks located at the corners of the coverage area. The random topology has been used widely in WSNs in the literature \cite{6, 70, 98}. Each source generates 10 kbps of information (i.e., $r_s = 10$ kbps, $\forall s \in S$). The size of each packet is 256 bits. The maximum transmission range is 10 m. The path loss exponent $a$ is set to 4 \cite{6}. We choose $\eta_1 = 100$ pJ/sec/m and $\eta_2 = 1$ nJ/sec \cite{70}. The initial energy of each sensor node is 1 kJ. Each point is averaged over 100 simulation runs.

#### 3.3.1 Varying the Tuning Parameter $\gamma$

In the first experiment, we determine the maximum network lifetime and the total rate of performing coding by varying the tuning parameter $\gamma$. There are four sinks deployed at the corners of the coverage area. One sensor node is randomly selected to operate as the source transmitting data towards all the sinks. A large value of $\gamma$ (i.e., $\gamma \gg 1$) implies
Figure 3.3: Average network lifetime under different values of $\gamma$.

that network lifetime is more important than the rate of performing network coding. On the other hand, a small value of $\gamma$ (i.e., $\gamma << 1$) decreases the effect of network lifetime in finding the optimal coding subgraph.

Figs. 3.3 and 3.4 show the network lifetime and the total rate of performing network coding under different values of $\gamma$, respectively. When the value of $\gamma$ is small, the joint maximum lifetime minimum resource coding subgraph problem is reduced to the maximum lifetime coding subgraph problem. In this case, both the network lifetime and also the rate of performing coding achieve their maximum values. For small values of $\gamma$, the MLMR algorithm outperforms the minimal resource algorithm and the MLST algorithm by 20% and 25%, respectively. The total rate of performing network coding is 30% more in our proposed MLMR algorithm when compared with the minimal resource algorithm. When the value of $\gamma$ increases, the rate of coding operation is decreased and eventually becomes zero when $\gamma$ is greater than 10. In this case, the MLST algorithm performs better than the minimal resource algorithm since MLST makes use of the broadcast nature of the wireless
Figure 3.4: Total rate of performing network coding under different values of $\gamma$.

links while the minimal resource algorithm does not. The MLMR algorithm performs better than MLST since it uses multipath routing as well as the broadcast nature of the wireless links. Multipath routing is known to outperform single path routing in terms of energy consumption and network throughput.

Fig. 3.5 shows the lifetime of the network versus the network coding rates. Different points of this figure relate to different values of parameter $\gamma$ and are derived from Figs. 3.3 and 3.4. When moving along this curve from left to right, the value of $\gamma$ decreases from $10^{-4}$ to $10^{-3}$. A further decrease of the value of $\gamma$ can neither improve the network lifetime nor increase the rate of coding operations. This figure shows that higher rates of coding operations in the network can lead to higher network lifetime. It is because network coding can reduce the total number of packets transmitted in the network and consequently can reduce the power consumption.
3.3.2 Varying the Number of Sinks $|D^c|$ 

In the second experiment, we compare the performance of our MLMR algorithm with the minimal resource and MLST algorithms in the presence of different number of sinks. One sensor node is randomly selected as the source node. The number of sinks increases in each step starting from one to four. In each step, the source node transmits data towards all the available sinks. Figs. 3.6 and 3.7 show the network lifetime and the rate of performing network coding under different number of sinks, respectively. When the network only has one sink, the problem is reduced to the unicast problem. Thus, our proposed MLMR algorithm and the minimal resource algorithm provide similar performance. These algorithms perform better than the MLST algorithm because they use multipath routing rather than single path routing as in the MLST algorithm. The MLST problem in the unicast case can be solved in polynomial time. In a unicast problem, the rate of performing network coding is zero. On the other hand, when the number of sinks increases, the amount of data flow in the network increases. The lifetime of the network decreases and the rate
Chapter 3. Lifetime-Resource Tradeoff for Multicast Traffic in WSNs

Figure 3.6: Network lifetime under different number of sinks ($\gamma = 10^{-4}$).

of performing coding operation increases accordingly.

3.3.3 Varying the Number of Sources $|S|$  

The next experiment evaluates the performance of different algorithms when the number of sources varies in the network. There are four sinks located at the four corners of the coverage area. The number of sources is increased in each step of the simulation from 1 to 5. Each source node sends data towards the two sinks which are closer to itself. Figs. 3.8 and 3.9 show the network lifetime and the rate of performing network coding under different number of sources, respectively. When the number of sources increases, there are more packet transmissions and it causes a higher energy depletion in each sensor node. Thus, the network lifetime decreases. As the number of sources increases in the network, the rate of performing network coding is increased as well. The percentage of difference between our proposed MLMR algorithm and the MLST algorithm increases from 25% for a network with one source to 35% for a network with five source nodes. Compared with the minimal
resource algorithm, our proposed MLMR algorithm has a higher network lifetime and a higher network coding rate.

### 3.3.4 Comparison between MLMR and EMR Algorithms

In the last experiment, we compare the MLMR algorithm with the evolutionary minimum resource (EMR) algorithm proposed in [96]. We consider the network constructed by cascading multiple copies of the topology as shown in Fig. 3.10 (a). The network constructed by cascading three copies is shown in Fig. 3.10 (b). We consider the cascaded networks with 1, 3, 7, and 15 copies. The number of sinks in these networks are 2, 4, 8, and 16, respectively. The source node is placed at the root of each topology. Table 3.1 compares the number of links performing network coding between the MLMR and EMR algorithms for different copies of topology shown in Fig. 3.10 (a). Since the multicast rates are achievable without network coding, the minimum number of links performing network coding is zero. Our proposed MLMT algorithm achieves the optimal solution for values of $\gamma$ greater...
Figure 3.8: Network lifetime for different number of sources ($\gamma = 10^{-4}$).

Table 3.1: Average number of links performing network coding.

<table>
<thead>
<tr>
<th>Number of copies</th>
<th>1</th>
<th>3</th>
<th>7</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMR algorithm</td>
<td>0.23</td>
<td>0.65</td>
<td>2.15</td>
<td>5.35</td>
</tr>
<tr>
<td>MLMR algorithm with $\gamma = 10^{-2}$</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>MLMR algorithm with $\gamma = 10^{-1}$</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>MLMR algorithm with $\gamma = 1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MLMR algorithm with $\gamma = 10$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

When $\gamma$ is less than $10^{-1}$, the EMR algorithm performs better than the MLMR algorithm especially for larger networks. It is because our proposed MLMR algorithm maximizes the network lifetime as the main objective and minimizes the network resources as the secondary objective when $\gamma$ is small.
3.4 Summary

In this chapter, we studied the tradeoff between network lifetime and system resources utilized to perform network coding in energy-constrained WSNs. We first introduced the constraints of the system. We then proposed the coding flow variables which enable us to determine the type of operation performed in each sensor node. The rate of performing network coding and replication in each node can be determined by using our proposed variables. We formulated the joint problem of constructing maximum-lifetime minimum-resource coding subgraph as a linear programming problem. The complexity of our algorithm increases linearly with the number of nodes and number of sessions, and it increases exponentially with the number of sinks. Hence, the algorithm is suitable for networks with a large number of sessions while each session has a small number of sinks. We described two directions for the model extension of this problem. Simulation results showed that our proposed MLMR algorithm has a better performance when compared with the minimal resource algorithm [23], the MLST algorithm [95], and the EMR algorithm [96].

Figure 3.9: Total rate of coding operations under different number of sources ($\gamma = 10^{-4}$).
Figure 3.10: Network topology: (a) one copy, (b) three copies.
Chapter 4

Feedback Mechanism for Random Linear Network Coding with Buffer Sharing in Lossy Wireless Sensor Networks

The wireless links are prone to failure due to fading and interference in WSNs. Therefore, improving the reliability can help to achieve power gain (i.e., lower power consumption). It is shown that random linear network coding (RLNC) can improve the reliability of end-to-end communication in the presence of lossy links [24, 25]. For a multihop communication session between a source and the sink, if the intermediate nodes use RLNC to store the received packets and transmit coded packets by linearly combining the buffered packets, the packet-level (i.e., max flow min-cut) capacity of a unicast session can be achieved [24, 25]. To reduce the time for the sink to decode the packets, the source divides the generated packets into groups called generations. The source and intermediate nodes perform coding operations on the packets of one generation. The sink can decode the packets of one generation as soon as it has enough packets for the decoding.

The performance of RLNC is comparable to link-by-link automatic repeat request (ARQ) in terms of power consumption. However, it is shown in [10] that RLNC can
achieve substantially higher power gain compared to end-to-end ARQ. The decoding delay of RLNC in multicast networks is investigated and compared to conventional scheduling techniques in [101]. The delay analysis of RLNC for unicast session on a line topology is presented in [102]. A comprehensive study of RLNC for time division duplexing with unicast session on a line network is performed in [103, 104, 105]. The source transmits packets consecutively and then waits for the sink to announce the number of additional packets it requires for decoding. In order to minimize the power consumption, there is an optimal number of packets to send before waiting for the feedback from the sink [103]. Energy efficient techniques based on opportunistic variants of RLNC are proposed in [106, 107]. RLNC can also be used in other areas such as network tomography [61], peer-to-peer content distribution [108], and code updates [109].

The use of feedback in routing packet networks and networks using RLNC is different. In routing packet networks with lossy links, feedback-based techniques such as ARQ are used to trigger the retransmission of lost packets. With delay and error free link-by-link feedback, the end-to-end packet transmission with ARQ can achieve the max-flow min-cut capacity [104]. However, perfect feedback may not be possible in real situations. On the other hand, the feedback packets in RLNC are used to acknowledge the receipts of enough coded packets for decoding. Similar to ARQ, both link-by-link and end-to-end feedback mechanisms can be used in RLNC. When link-by-link feedback is used, each intermediate node sends feedback to its upstream neighboring node when it has sufficient packets with independent coding coefficients. With end-to-end feedback, the sink informs the source and all the intermediate nodes to stop transmission of a generation by sending feedback packets to them. RLNC can achieve the max-flow min-cut capacity when the code size and the generation size are sufficiently large [24, 25].

In RLNC, when intermediate nodes keep different buffers for various flows (i.e., RLNC
without buffer sharing), each transmitted packet contains information of only one flow. For the case of link-by-link feedback, an intermediate node \(i\) will receive a feedback control packet for a particular flow when its immediate downstream node has enough coded packets to decode the packets of one generation of that flow. For the case of end-to-end feedback, an intermediate node \(i\) will receive a feedback control packet from the sink for a particular flow when the sink has enough coded packets for decoding. The work in \[110\] presented different ways that feedback can help for network coding, including the benefit of feedback for quality of service, resource management, and improving the reliability. A feedback-based online network coding approach is presented in \[111\], where the destinations send individual feedback for each received packet. It has a lower decoding delay compared to the generation-based coding approaches.

The idea of using shared buffer for different flows passing an intermediate node has been employed in several network coding based techniques. COPE \[112\] presents a practical way to implement linear network coding in wireless networks while packets of various flows share the buffer. It increases the throughput of the network by taking advantage of network coding and opportunistic listening. The performance of COPE is improved by jointly considering network coding and medium access control (MAC). COPE has also been extended in different directions \[113, 114, 115\]. In SenseCode \[116\], the intermediate nodes combine the overheard packets from different neighbors with its generated packets. It increases the amount of redundant information in the network, which can improve the reliability.

We consider a WSN with a single sink and several sources, while data are gathered through a reverse multicast tree rooted at the sink. The feedback mechanism design for RLNC with buffer sharing is different from RLNC without buffer sharing. We notice that the feedback control packet is generated in order to inform a node to stop packet
transmission for a particular flow. In RLNC without buffer sharing, an intermediate node can stop transmission of a flow \( f \) after successfully transmitting \( K_f \) packets, where \( K_f \) is the generation size of flow \( f \). On the other hand, for RLNC with buffer sharing, the transmitted packets contain information of all the flows passing the intermediate node. Therefore, it is not possible to distinguish to which flow a transmitted packet belongs. Consequently, it is a challenge to determine when a node can stop transmission of a particular flow.

In this chapter, we study the problem of feedback mechanism design for RLNC with buffer sharing. Although the problem of feedback mechanism design for RLNC without buffer sharing is studied in [25], [103], and [110], to the best of our knowledge, there is no feedback mechanism in the literature for RLNC with shared buffers. The contributions of this chapter are as follows:

- For an intermediate node \( i \) in a reverse multicast tree topology, we determine when the node can stop transmission of a particular flow \( f \). We analytically show that reducing the transmission rate at this time does not change the rate at which node \( i \) transmits the coded packets which carry new information for its downstream neighbor.

- We propose a novel link-by-link feedback mechanism. In this mechanism, every intermediate node generates feedback control packet for its upstream neighboring node.

- We propose three different types of end-to-end feedback mechanisms for RLNC with buffer. In end-to-end feedback mechanisms, the sink is responsible for generating feedback control packet for all the nodes in the network.

- Simulation results show that the link-by-link feedback mechanism has better performance in terms of the power consumption than the end-to-end feedback mechanism. The results also show the performance gain of using buffer sharing compared to RLNC.
without buffer sharing in terms of power consumption. They also show the tradeoff between the power consumption and delay when using buffer sharing is employed.

The model and mechanisms we present in this chapter are unique when compared to the prior work done in the area of feedback mechanism design. In terms of buffering, the work in [24, 25] require an individual buffer for each flow at the intermediate nodes. Our proposed feedback mechanisms use buffer sharing. In terms of overhead, COPE [112] requires an acknowledgement packet be transmitted for each received or overheard packet. In contrast, our proposed mechanisms require a downstream node (or sink) to transmit a feedback packet when it has received a generation of packets. For the work in Sensecode [116], all the flows stop transmission at the same time while in our proposed approaches, various flows stop transmissions based on the order the packets of one generation are successfully transmitted.

The rest of this chapter is organized as follows: The system model along with a description of RLNC with buffer sharing is summarized in Section 4.1. In Section 4.2, we model and analyze the rate at which the intermediate nodes transmit and receive the packets with new information. In Section 4.3, we propose the link-by-link and end-to-end feedback mechanisms. Performance evaluation and comparison are presented in Section 4.4. A summary is given in Section 4.5.

4.1 System Model

We consider a WSN with a set of sources and one sink. Without loss of generality, we assume that the sources do not relay information of other flows. Let $\mathcal{V}$ denote the set of sensor nodes. Data from different sources are gathered at the sink via a reverse multicast tree. Therefore, there is a one-to-one mapping between each sensor node and its outgoing link. We use the same set $\mathcal{V}$ to denote the set of links. Thus, link $i$ is the output link
Chapter 4. Feedback Mechanism for RLNC with Buffer Sharing in Lossy WSNs

of node $i$. We use $\alpha_i$ to denote the packet success probability of the output link of node $i$. Let $\mathcal{F}$ denote the set of all the flows in the network. A flow is the data flow from a source to the sink. The set of nodes relaying packets of flow $f \in \mathcal{F}$ is denoted by $\mathcal{P}_f$. The set of different flows passing through an intermediate node $i \in \mathcal{V}$ is denoted by $\mathcal{F}_i$. Each intermediate node keeps one buffer for all the flows.

Each source divides the generated packets into groups called generations. The downstream intermediate nodes buffer the received packets of one generation from the upstream flows. The sources and the intermediate nodes create coded packets by combining the buffered packets using random coefficients taken from a Galois field with size $q$ (i.e., $\text{GF}(2^q)$). They append the coding coefficients in the packet header. The vector of coding coefficients is called the encoding vector.

The source and intermediate nodes transmit coded packets until they receive a feedback control packet from either the sink or a downstream node. The rate at which packets of flow $f \in \mathcal{F}$ are being transmitted is $r_f$. Since various sources may generate packets at different rates, we set the generation size of different flows such that all the flows have the same generation size to transmission rate ratio. This ratio is denoted by $\Delta$. Let $K_f$ denote the generation size of flow $f$. We have $K_f/r_f = \Delta$ for all flow $f \in \mathcal{F}$. This condition ensures that if the links are lossless, then the sink can decode the packets $\Delta$ seconds after receiving the first packet. In RLNC with buffer sharing, all the sources start transmission of a generation at the same time. The difference between the time that the sources start transmission of the packets and the time that the sink can decode the packets is called the decoding delay.

Let $\Lambda_i(t)$ denote a matrix composed of the encoding vectors of the packets in the buffer of node $i$ at time $t$. Each column in matrix $\Lambda_i(t)$ contains the encoding vector of a packet stored in the buffer of node $i$. Each row in matrix $\Lambda_i(t)$ corresponds to the coefficients of a
packet from a generation of a particular flow. The rank of this matrix, denoted as $|\Lambda_i(t)|$, represents the number of packets with independent encoding vectors in the buffer of node $i$.

We define $|\Lambda_i(t)|_F$ as the rank of matrix $\Lambda_i(t)$ with respect to the set of flows in $F \subseteq \mathcal{F}_i$. It is computed by replacing the rows of matrix $\Lambda_i(t)$ which do not belong to the flows in $F$ by zero vectors and then computing the rank of the resulting matrix. A packet $W$ transmitted by node $i$ at time $t$ is an innovative packet for downstream node $j$ if it increases the rank of matrix $\Lambda_j(t)$ by one upon reception. For large Galois field size, $|\Lambda_i(t)| > |\Lambda_j(t)|_F$ would result that $W$ is an innovative packet with probability close to one [25].

The flux of innovative packets for flow $f$ can be modeled as a queueing system [20, Chapter 4]. Each intermediate node is modeled as a server while innovative packets are customers traveling from the sources to the sink via intermediate nodes. Note that this queue is different from the actual buffer of the node which stores all the received packets. To distinguish this queue from the buffer of the node, it is called the virtual queue of the node.

For an intermediate node $i$, the number of packets in the virtual queue of node $i$ is increased by one when it receives an innovative packet from one of its upstream nodes. The virtual queue of intermediate node $i$ may receive packets from multiple flows. The input rate to the virtual queue of node $i$ is the aggregate rate at which node $i$ receives innovative packets from its upstream neighboring nodes. The number of packets in the virtual queue of node $i$ is decreased by one when node $i$ successfully transmits an innovative packet. Therefore, the service rate of the virtual queue is the rate at which packets are successfully transmitted. We call the product of the packet success probability of the output link of node $i$ and the transmission rate of node $i$ as the effective transmission rate of node $i$. This is also the service rate of the virtual queue of node $i$. The rate at which the innovative
packets are transmitted from node $i$ depends on the effective transmission rate of the node and the input rate of the innovative packets. The number of innovative packets in the buffer of node $j$ (i.e., the downstream node of node $i$) received from node $i$ at time $t$ is $|\Lambda_j(t)|_{\mathcal{F}_i}$.

### 4.2 Transmission Rate Analysis

The goal of the feedback mechanism is to generate feedback control packets which acknowledge the receipt of a generation of source packets. An intermediate node $i$ reduces its transmission rate by $r_f$ upon receiving the feedback for flow $f$. Therefore, to design the feedback mechanism, we need to determine when an intermediate node can stop transmitting the packets of one generation for flow $f$. In this section, we determine the time that an intermediate node $i$ can stop transmission of a particular flow $f$ by analyzing the rate at which it transmits innovative packets.

#### 4.2.1 When Can Node $i$ Stop Transmission of Flow $f$?

For RLNC without buffer sharing, the intermediate node $i$ stores the packets of different flows in various buffers. It can stop transmitting the packets of flow $f$ as soon as it successfully transmits the $K_f$-th innovative packet. After this time, the rate at which node $i$ transmits the innovative packets is zero. Therefore, node $i$ can reduce its transmission rate by $r_f$.

On the other hand, for the case that the intermediate node $i$ stores the packets of various flows in a *shared buffer*, it starts transmitting the packets from the shared buffer at the aggregate rate of $\sum_{f \in \mathcal{F}_i} r_f$. The rate at which node $i$ transmits the innovative packets depends on the rate at which it receives the innovative packets and its effective transmission rate. Node $i$ can stop transmission of flow $f$ (i.e., reduce its transmission rate
Figure 4.1: A network tree with two sources, one intermediate node $i$, and one sink. Packets of flows 1 and 2 are generated by source $s_1$ and $s_2$, respectively.

by $r_f$) if the rate at which innovative packets are transmitted does not alter after reducing the rate. Every transmitted packet contains information of all the flows in $F_i$. Thus, one cannot distinguish to which flow a transmitted packet belongs to. Consequently, it is not feasible to determine the time that node $i$ can stop transmission of flow $f$ by counting the number of transmitted innovative packets of a particular flow.

As an example, consider Fig. 4.1 with two flows (from sources 1 and 2 to the sink $d$) and an intermediate node $i$. The transmission rates of flows 1 and 2 are $r_1 = r_2 = 1$ packet per second. The packet success probability of the links in the network are as follows: $\alpha_1 = 0.75$, $\alpha_2 = 0.9$, and $\alpha_i = 1$. Node $i$ receives innovative packets of flows 1 and 2 at the average rate of 0.75 and 0.9 packets per second at the beginning, respectively. Therefore, the input rate of innovative packets to the virtual queue of node $i$ is 1.65 packets per second. The transmission rate of node $i$ is 2 packets per second, which is equal to the effective transmission rate of the node. If the generation size of flow 1 is $K_1$, node $i$ receives the last innovative packet of flow 1 on average after $\tau_1 = K_1/0.9$ seconds. After this time, the input rate of the innovative packets to the virtual queue of node $i$ drops from 1.65 to 0.75.
As soon as node $i$ can manage to transmit the packets of flow 1 after time $\tau_1$, it can reduce its transmission rate from two packets per second to one packet per second. Since the rate at which this node receives the innovative packets is 0.75 packets per second, reducing the transmission rate from 2 to 1 does not change the rate at which this node transmits the innovative packets.

Now, we determine when each node on path set $\mathcal{P}^f$ can stop transmission of flow $f$. We start from the source of flow $f$ and move along path $\mathcal{P}^f$ to the sink. Among flows in $\mathcal{F}$, we only study flow $f$ in this section. For the sake of simplicity, we denote the set of nodes in the path set $\mathcal{P}^f$ by $\{1, 2, \ldots, |\mathcal{P}^f|\}$, where node 1 is the source of flow $f$ and node $|\mathcal{P}^f|$ is the sink. Since the source of flow $f$ does not relay the packets of other flows, it can stop transmission after successfully transmitting $K_f$ packets with independent encoding vectors.

Let $\tau_1^f$ denote the time that the source of flow $f$ successfully transmits the $K_f$-th innovative packet. At this time, the virtual queue of the source is empty. Since after time $\tau_1^f$, node 2 does not receive any innovative packet from the source of flow $f$, it can stop transmission of the flow as soon as it can transmit all the packets in its virtual queue after time $\tau_1^f$. Note that the packets transmitted by node 2 contain information of all the packets in its buffer. Therefore, we cannot distinguish to which flow a transmitted packet belongs if node 2 relays packets of more than one flow. The first time that the virtual queue of node 2 is empty after $\tau_1^f$ is the first time that we can ensure all the information of flow $f$ are transmitted. Let $\tau_2^f$ denote the first time after $\tau_1^f$ at which the virtual queue of node 2 is empty. For an intermediate node $i$, we use $\tau_i^f$ to denote the first time after $\tau_{i-1}^f$ that the virtual queue of node $i$ is empty. We define this time in a recursive manner as

$$
\tau_i^f = \min \left\{ t \mid t > \tau_{i-1}^f, |\Lambda_i(t)| = |\Lambda_{i+1}(t)|_{\mathcal{F}_i} \right\}, \quad i = 1, \ldots, |\mathcal{P}^f| - 1
$$

(4.1)
where $\tau_0^f = 0$. First, we show how this time can be calculated in practice. Then, we show that reducing the rate by $r_f$ at time $\tau_i^f$ does not change the rate at which node $i$ transmits the innovative packets, which means node $i$ can reduce its transmission rate at $\tau_i^f$.

### 4.2.2 Determine the Time $\tau_i^f$

Let $g_i^f(t)$ denote the number of packets with *exclusive* information of flow $f$ in the buffer of node $i$ at time $t$. This is the number of innovative packets in the buffer of node $i$ minus the number of innovative packets of all the flows except flow $f$ at time $t$. $g_i^f(t)$ is increased by one at time $t$ if node $i$ receives an innovative packet which is not innovative for flows in $\mathcal{F}_i \backslash \{f\}$.

For better understanding of the concept, consider the following example. Assume that node $i$ receives packets of three flows, namely, $f_1$, $f_2$, and $f_3$. Assume that the generation size is one (i.e., $K_1 = K_2 = K_3 = 1$) and node $i$ currently has two packets with encoding vectors $(2, 4, 3)$ and $(2, 3, 1)$. Both of these packets are innovative. They are also innovative packets for flows $f_1$ and $f_2$. Therefore, the number of packets with exclusive information of flow $f_3$ is zero. At time $t$, node $i$ receives a packet with encoding vector $(1, 4, 1)$. Since this packet is innovative, the total number of innovative packets in the buffer of node $i$ would be 3. Moreover, this packet is not innovative for flows $f_1$ and $f_2$. Therefore, $g_i^3(t) = 1$. We notice that the newly received packet is not innovative for flow $f_3$ while it has exclusive information of flow $f_3$.

From the definition of $g_i^f(t)$, the value of $g_i^f(t)$ at time $t$ can be calculated as

$$g_i^f(t) = |\Lambda_i(t)| - |\Lambda_i(t)\backslash\{f\}|,$$

(4.2)

where $|\Lambda_i(t)\backslash\{f\}\}$ is the number of innovative packets of all the flows except flow $f$ at time $t$. This can be calculated by setting the coding coefficients corresponding to the flow $f$ for
the packets in $\Lambda_i(t)$ to zero and then calculating the rank of this resulting matrix. The following lemma shows how the function $g_f^i(t)$ can be used to determine time $\tau_f^i$.

**Lemma 4** The time $\tau_f^i$ is the first time at which $g_{i+1}^f(t)$ is equal to $K_f$

$$\tau_f^i = \min \left\{ t \mid g_{i+1}^f(t) = K_f \right\}, \ i = 1, \ldots, |\mathcal{P}| - 1, \ f \in \mathcal{F}_i \quad (4.3)$$

**Proof** The function $g_f^i(t)$ is an increasing function since it is not possible that node $i$ receives a packet which is innovative for all the flows except $f$ while this packet is not innovative for node $i$. Nevertheless, the number of packets with exclusive information of flow $f$ is less than or equal to the number of innovative packets of flow $f$. Therefore, function $g_f^i(t)$ is upper bounded by $K_f$. Now, we proof the lemma by mathematical induction. At time $\tau_1^f$, node 2, which is the immediate downstream node of node 1 (i.e., the source node of flow $f$), has received the $K_f$-th innovative packet from the source. Since every packet in the buffer of node 2 contains either information of $f$ or other flows, we have

$$|\Lambda_2(\tau_f^1)| = |\Lambda_2(\tau_f^1)\backslash \{f\}| + |\Lambda_2(\tau_f^1)|_{\{f\}}.$$  

Therefore, we have $g_2^f(\tau_f^1) = K_f$. This is the basis of our induction. Now as the hypothesis of the induction, assume that $g_{i+1}^f(\tau_i^f)$ is equal to $K_f$ while $g_{i+1}^f(t) < K_f$ for $t < \tau_i^f$. We need to prove that $g_{i+2}^f(\tau_{i+1}^f) = K_f$, while $g_{i+2}^f(t) < K_f$ for $t < \tau_{i+1}^f$. Since $\tau_{i+1}^f > \tau_i^f$ and $g_f^i(t)$ is an increasing function and is upper bounded by $K_f$, we have $g_{i+1}^f(\tau_{i+1}^f) = K_f$. Since the virtual queue of node $i+1$ is empty at time $\tau_{i+1}^f$, node $i+2$ has the same information that node $i+1$ has about flows in set $\mathcal{F}_{i+1}$, which implies $g_{i+2}^f(\tau_{i+1}^f)$ is equal to $K_f$. To complete the proof, we need to show that $g_{i+2}^f(t) < K_f$ for $t < \tau_{i+1}^f$. Since $g_{i+2}^f(t) < g_{i+1}^f(t)$ and $g_{i+1}^f(t) < K_f$ for $t < \tau_i^f$, therefore, $g_{i+2}^f(t) < K_f$ for $t < \tau_i^f$. Now, assume that there exists a time $\zeta$ such that $\tau_i^f < \zeta < \tau_{i+1}^f$ and $g_{i+2}^f(\zeta) = K_f$. Thus, at time $\zeta$, node $i+2$ has received a packet with exclusive information of flow $f$. It means there is no packet in the virtual queue of node $i+1$ with information of other flows except $f$. However, this is the last packet sent by node
\[ i + 1 \text{ with exclusive information of flow } f. \text{ Therefore, after time } \zeta, \text{ the virtual queue of node } i + 1 \text{ is empty while this contradicts the definition of } \tau_{i+1}^f. \text{ Therefore, } g_{i+2}(t) < K_f \text{ for } t < \tau_{i+1}^f. \]

### 4.2.3 Transmission Rate of Innovative Packets After Time \( \tau_i^f \)

In this section, we show that reducing the transmission rate after time \( \tau_i^f \) does not change the rate at which node \( i \) transmits innovative packets. Our model analyzes the system on a long term operation period and the transmission rates are expected values. Let \( z_i(F) \) denote the rate at which node \( i \) transmits the innovative packets of flows in set \( F \). We focus on node \( i \) and one of the flows \( f \in \mathcal{F}_i \). We assume that flow \( f \) has the lowest value of \( \tau_i^f \) among all the flows in \( \mathcal{F}_i \) (i.e., \( \tau_i^f = \min_{s \in \mathcal{F}_i} \tau_s^i \)). For the sake of simplicity, we assume that \( \tau_i^f \) is not equal to \( \tau_s^i \) for \( s \in \mathcal{F}_i \setminus \{f\} \) in the rest of the chapter. The following theorem shows that node \( i \) can reduce its transmission rate at time \( \tau_i^f \) without changing the rate at which innovative packets are transmitted.

**Theorem 3** The value of \( z_i(\mathcal{F}_i \setminus \{f\}) \) does not change if node \( i \) reduces its transmission rate from \( \sum_{s \in \mathcal{F}_i} r_i \) to \( \sum_{s \in \mathcal{F}_i \setminus \{f\}} r_i \) (i.e., reduces by \( r_f \)), where \( \tau_i^f \) is equal to \( \min_{s \in \mathcal{F}_i} \tau_s^i \).

**Proof** Since \( \tau_i^f = \min_{s \in \mathcal{F}_i} \tau_s^i \), \( z_i(\mathcal{F}_i \setminus \{f\}) \) is constant in time period \([0, \tau_i^f]\) as long as node \( i \) receives innovative packets from flows in \( \mathcal{F}_i \setminus \{f\} \). At time \( \tau_i^f \), the virtual queue of node \( i \) is empty. Hence, all the received innovative packets up to time \( \tau_i^f \) have been transmitted by node \( i \). The number of innovative packets transmitted by node \( i \) up to time time \( \tau_i^f \) is \( \left| \Lambda_i(\tau_i^f) \right| \), which is equal to \( g_i^f(\tau_i^f) \). Based on Lemma 4, we have \( g_i^f(\tau_i^f) = K_f \). Since the transmission rate of the innovative packets for all the flows except flow \( f \) is \( z_i(\mathcal{F}_i \setminus \{f\}) \), the average number of innovative packets transmitted by node \( i \) up to time \( \tau_i^f \) is \( z_i(\mathcal{F}_i \setminus \{f\}) \tau_i^f + K_f \). The average number of successful transmissions of node \( i \) up to time \( \tau_i^f \) is \( \tau_i^f \alpha_i \sum_{s \in \mathcal{F}_i} r_s \). The average number of innovative packets that node \( i \)
has transmitted is less than or equal to the number of successful transmissions of node $i$. Therefore, we have

$$z_i(\mathcal{F}_i \setminus \{f\}) \tau_i^f + K_f \leq \alpha_i \sum_{s \in \mathcal{F}_i} r_s,$$  \hspace{1cm} (4.4)

which can be re-arranged as

$$z_i(\mathcal{F}_i \setminus \{f\}) \leq \alpha_i \sum_{s \in \mathcal{F}_i} r_s - \frac{K_f}{\tau_i^f}. \hspace{1cm} (4.5)$$

If $K_f/\tau_i^f > \alpha_i r_f$, then we obtain $z_i(\mathcal{F}_i \setminus \{f\}) < \alpha_i \sum_{s \in \mathcal{F}_i \setminus \{f\}} r_s$ from (4.4). Now, we consider the case that $K_f/\tau_i^f \leq \alpha_i r_f$, which is equal to $\Delta/\tau_i^f \leq \alpha_i$. Since the flows in set $\mathcal{F}_i \setminus \{f\}$ are still active, the total number of innovative packets transmitted by node $i$ from these flows up to time $\tau_i^f$ is strictly less than the total number of packets for one generation of these flows (i.e., $z_i(\mathcal{F}_i \setminus \{f\}) \tau_i^f < \sum_{s \in \mathcal{F}_i \setminus \{f\}} K_s$). Therefore, we have

$$z_i(\mathcal{F}_i \setminus \{f\}) < \frac{\sum_{s \in \mathcal{F}_i \setminus \{f\}} K_s}{\tau_i^f} \sum_{s \in \mathcal{F}_i \setminus \{f\}} r_s \leq \alpha_i \sum_{s \in \mathcal{F}_i \setminus \{f\}} r_s. \hspace{1cm} (4.6)$$

This means the rate at which node $i$ transmits innovative packets is less than its effective transmission rate after reducing the rate by $r_f$. Hence, the rate at which this node transmits the innovative packets does not alter by reducing the rate.

The following lemma shows that as soon as node $i$ reduces its transmission rate once, the queuing delay of its virtual queue is upper bounded.

**Lemma 5** For time $t$ greater than $\tau_i^f$, where $\tau_i^f = \min_{s \in \mathcal{F}_i} \tau_i^s$, the number of packets in the virtual queue of node $i$ is upper bounded.

**Proof** The virtual queue of node $i$ is empty at time $\tau_i^f$. Based on Theorem 3, at time $t = \tau_i^f$, the input rate of the virtual queue is less than the service rate of the queue which
Chapter 4. Feedback Mechanism for RLNC with Buffer Sharing in Lossy WSNs

is $\alpha_i \sum_{s \in F_i \setminus \{f\}} r_s$. Since the input rate of the virtual queue is not increased over time, the input rate is always less than $\sum_{s \in F_i \setminus \{f\}} r_s$ for $t > \tau_i^f$. We notice that, the transmission rate of node $i$ is not reduced such that it falls below the input rate of the virtual queue after time $\tau_i^f$. Therefore, since $J_i(\tau_i^f) = 0$ while the input rate of the queue is always less than the service rate during the operation of the queue, the number of packets in the virtual queue of node $i$ is bounded and the probability that the virtual queue is empty is non-zero for $t > \tau_i^f$.

Another interpretation of Lemma 5 is that the rate at which node $i$ transmits innovative packets is always less than the rate at which this node receives the innovative packets after time $\tau_i^f$. Let $\varphi(t) \subseteq F_i$ denote the set of flows that node $i$ receives innovative packets from at time $t$. The following lemma characterizes the rate at which node $i$ transmits innovative packets of flows $\varphi(t)$.

**Lemma 6** The rate at which node $i$ transmits the innovative packets of flows $\varphi(t)$ from time 0 to $t$ is the weighted sum of the rates of all these flows. That is,

$$z_i(\varphi(t)) = \sum_{s \in \varphi(t)} r_s \beta_i^s(\varphi(t)),$$

(4.7)

where $\beta_i^s(\varphi(t))$ is a non-negative coefficient for node $i$ and flow $s$ corresponding to the set of flows $\varphi(t)$.

**Proof** We use mathematical induction to prove this lemma. The rate at which the source of flow $f$ (i.e., node $s_f$) transmits the innovative packets is $z_{s_f}(\{f\}) = \alpha_{s_f} r_{s_f}$. This is the basis of induction. Consider intermediate node $i$ where $N_i$ denotes the set of upstream nodes of node $i$. As inductive step, assume that upstream node $j \in N_i$ transmits packets of flows $\varphi(t) \cap F_j$ with rate $z_j(\varphi(t) \cap F_j) = \sum_{s \in \varphi(t) \cap F_j} r_s \beta_j^s(\varphi(t) \cap F_j)$. Then, the rate at which node $i$ receives innovative packets of flows $\varphi(t)$ is the sum of the rate at which nodes
in set $N_i$ transmit innovative packets. The rate at which node $i$ transmits the innovative packets is the minimum of the rate at which it receives innovative packets and its effective transmission rate. That is,

$$z_i(\varphi(t)) = \min \left\{ \sum_{j \in N_j} z_j(\varphi(t) \cap F_j), \alpha_i \sum_{s \in \varphi(t)} r_s \right\}$$

$$= \min \left\{ \sum_{j \in N_j} \sum_{s \in \varphi(t) \cap F_j} r_s \beta_s^i(\varphi(t) \cap F_j), \alpha_i \sum_{s \in \varphi(t)} r_s \right\} \tag{4.8}$$

Both arguments of the minimum function on the right hand side of equation (4.8) are linear combination of $r_s$ for all $s \in \varphi(t)$. Therefore, $z_i(\varphi(t))$ is a linear combination of $r_s$ for all $s \in \varphi(t)$ which can be denoted by $z_i(\varphi(t)) = \sum_{s \in \varphi(t)} r_s \beta_s^i(\varphi(t))$ where $\beta_s^i(\varphi(t))$ is a constant coefficient.

Using Lemma 6, we are able to generalize Theorem 3 as

**Theorem 4** The value of $z_i(\varphi(t))$ does not change if node $i$ reduces its transmission rate by $r_f$ at time $\tau_i^f$.

**Proof** If all the flows being relayed by node $i$ are active at time $\tau_i^f$, based on Theorem 3, node $i$ can reduce its transmission rate by $r_f$. Now, assume that flow $f$ is not the first flow whose rate is reduced by node $i$. Since node $i$ has reduced its transmission rate before time $\tau_i^f$ once, the rate at which it receives the innovative packets before time $\tau_i^f$ is less than its effective transmission rate. According to Lemma 3, the rate at which this node receives the packets of flows in set $\varphi(t)$ is $z_i(\varphi(t))$. The average rate at which it received the packets of flow $f$ before $\tau_i^f$ is $K_f/\tau_i^f$. Therefore, we have

$$z_i(\varphi(t) \setminus \{f\}) + \frac{K_f}{\tau_i^f} \leq \alpha_i \sum_{s \in \varphi(t)} r_s. \tag{4.9}$$
Equation (4.9) is similar to (4.5). We can follow the rest of proof of Theorem 3 after equation (4.5) and show that \( z_i(\varphi(t) \setminus \{f\}) < \alpha_i \sum_{s \in \varphi(t) \setminus \{f\}} r_s \). This shows that reducing the transmission rate of node \( i \) by \( r_f \) does not change the rate at which this node transmits the innovative packets.

Theorem 3 proves that \( \tau_{f_i}^f \) is the time that node \( i \) can reduce the transmission rate by \( r_f \).

### 4.3 Feedback Mechanism Design

The goal of the feedback mechanism is to generate feedback control packet for an intermediate node and a particular flow such that the node can stop transmission of that flow. As shown in Theorem 3, for any flow \( f \in F_i \), node \( i \) can reduce its transmission rate by \( r_f \) at time \( \tau_{f_i}^f \). However, as Lemma 3 shows, node \( i \) cannot determine this time by itself. The feedback control packet generated by downstream nodes of node \( i \) is used to notify node \( i \) to reduce its transmission rate at certain times. We propose two feedback mechanisms, namely link-by-link and end-to-end feedback, which are useful under different circumstances.

#### 4.3.1 Link-by-Link Feedback Mechanism

In link-by-link feedback mechanism, each intermediate node is responsible to generate feedback control packet for its upstream neighboring nodes. It is applicable if the intermediate nodes are capable of calculating the matrix ranks. In this mechanism, each intermediate node checks the stored packets and informs the upstream neighboring nodes to reduce the rate for a particular flow when it exclusively has full rank information of the flow. Based on Theorem 3, node \( i \) can stop transmission of flow \( f \) at time \( \tau_{i}^f \) where \( g_j \left( \tau_{i}^f \right) = K_f \).
Algorithm 4.1 Link-by-Link Feedback Mechanism at Node $i$.

1: input: $\mathcal{F}_i, K_f, \forall f \in \mathcal{F}_i$
2: for each generation
3: $\mathcal{A} := \emptyset$
4: while $\mathcal{A} \neq \mathcal{F}_i$ and a new packet is arrived at time $t$
5: for flow $f \in \mathcal{F}_i \setminus \mathcal{A}$
6: if $|\Lambda_i(t)| - |\Lambda_i(t)\setminus \{f\}|$ is equal to $K_f$
7: Send feedback control packet for flow $f$ to the upstream node that relays packets of flow $f$
8: $\mathcal{A} := \mathcal{A} \cup \{f\}$
9: end if
10: end for
11: end while
12: end for

and $j$ is the downstream neighboring node of node $i$. Therefore, node $j$ can generate the feedback control packet of flow $f$ for node $i$ as soon as $g_j(\tau_i^f) = K_f$. Algorithm 4.1 shows the link-by-link feedback mechanism at the intermediate node $i$.

Upon receiving a packet from an upstream node, node $i$ checks whether it has enough exclusive packets for any of the flow (Step 4). If there is any flow with enough exclusive packets (Step 6), node $i$ will send the feedback control packet to the appropriate upstream neighboring node (Step 7). The algorithm continues until node $i$ sends feedback control packet for all the flows in $\mathcal{F}_i$.

### 4.3.2 End-to-End Feedback Mechanism

As mentioned earlier, the link-by-link feedback mechanism is beneficial if the intermediate sensor nodes are capable of calculating the rank of the matrix. However, sensor nodes are inexpensive devices with low computational capabilities. Sensor nodes may not able to calculate the matrix ranks online, especially when there are several flows. In this case, the sink, which has higher computational power, can generate the feedback control packet for the sources and all the intermediate nodes.
When the sink is responsible to generate the feedback, there are three different feedback mechanisms:

**Type 1.** The sink transmits feedback control packets to all the nodes as soon as it can decode the packets of one generation of all the flows. In this case, the sink needs to monitor the total number of received innovative packets. Once the number of innovative packets is equal to \( \sum_{s \in \mathcal{F}} K_s \), the sink sends feedback control packets to all the nodes to stop transmission. Although this has a low computational load since the sink does not need to monitor various flows individually, this wastes power at the intermediate nodes via unnecessary transmissions. We call this mechanism the *end-to-end feedback mechanism after decoding*.

**Type 2.** The sink generates a feedback control packet for flow \( f \) at time \( \tau^f_d \). This feedback packet acknowledges all the nodes on the path set \( \mathcal{P}^f \) relaying packets of flow \( f \). We call it *per flow end-to-end feedback mechanism*. The following proposition shows that the sink \( d \) can generate a feedback packet to acknowledge flow \( f \) at time \( \tau^f_d \). We notice that \( g^f_d(t) = |\Lambda_d(t)| - |\Lambda_d(t)\setminus\{f\}|. \)

**Proposition 2** If \( g^f_d(t) \) is equal to \( K_f \) at time \( t \), then all the nodes relaying packets of flow \( f \) can reduce their transmission rates by \( r_f \).

**Proof** Based on Lemma 2, the immediate upstream node of the sink \( d \), which relays packets of flow \( f \), can stop transmission of flow \( f \) as soon as \( g^f_d(t) \) is equal to \( K_f \). Since \( \tau^f_{i+1} \) is strictly less than \( \tau^f_i \) for \( i = 1, \ldots, |\mathcal{P}^f| - 1 \), all the nodes in path set \( \mathcal{P}^f \) can stop their transmissions at this time as well.

**Type 3.** The sink sends individual feedback packet for any intermediate node \( i \in \mathcal{V} \) and flow \( f \in \mathcal{F}_i \). This type of feedback is unique to the network which uses shared buffer. If the intermediate nodes keep separate buffers for various flows, it is not possible to create
feedback per intermediate node. An intermediate node with multiple incoming links can receive individual feedback from the sink. We call this type of the feedback per flow per node end-to-end feedback mechanism. Proposition 4 provides a sufficient condition for the generation of feedback for node \(i\) and flow \(f\). The following lemma, however, provides a necessary condition for the feedback generation.

**Lemma 7** The sink \(d\) should send feedback control packet of flow \(f\) to intermediate node \(i \in V\) at time \(t\) if

\[
|\Lambda_d(t)|_{\mathcal{F}_i} - |\Lambda_d(t)|_{\mathcal{F}_i \setminus \{f\}} = K_f. \tag{4.10}
\]

**Proof** In the buffer of node downstream node \(i + 1\) of node \(i\), there are two independent sets of packets. The first set is the packets which have information of flows in \(\mathcal{F}_i\). The second set of packets include those with information of the flows in \(\mathcal{F}_{i+1} \setminus \mathcal{F}_i\). The packets of flows in \(\mathcal{F}_i\) at the sink are linear combination of the packets in the buffer of node \(i + 1\). Assume that there are \(n\) packets in the buffer of the sink which contain information of flows \(\mathcal{F}_i\).

We have \(|\Lambda_d(t)|_{\mathcal{F}_i} = \min\{n, |\Lambda_{i+1}(t)|_{\mathcal{F}_i}\}\) and \(|\Lambda_d(t)|_{\mathcal{F}_i \setminus \{f\}} = \min\{n, |\Lambda_{i+1}(t)|_{\mathcal{F}_i \setminus \{f\}}\}\). This shows that,

\[
|\Lambda_d(t)|_{\mathcal{F}_i} - |\Lambda_d(t)|_{\mathcal{F}_i \setminus \{f\}} = \begin{cases} 
0, & \text{if } n \leq |\Lambda_{i+1}(t)|_{\mathcal{F}_i \setminus \{f\}}, \\
n - |\Lambda_{i+1}(t)|_{\mathcal{F}_i \setminus \{f\}}, & \text{if } |\Lambda_{i+1}(t)|_{\mathcal{F}_i \setminus \{f\}} < n < |\Lambda_{i+1}(t)|_{\mathcal{F}_i}, \\
|\Lambda_{i+1}(t)|_{\mathcal{F}_i} - |\Lambda_{i+1}(t)|_{\mathcal{F}_i \setminus \{f\}}, & \text{if } n \geq |\Lambda_{i+1}(t)|_{\mathcal{F}_i}. 
\end{cases} \tag{4.11}
\]

This equation shows that \((|\Lambda_d(t)|_{\mathcal{F}_i} - |\Lambda_d(t)|_{\mathcal{F}_i \setminus \{f\}}) \leq (|\Lambda_{i+1}(t)|_{\mathcal{F}_i} - |\Lambda_{i+1}(t)|_{\mathcal{F}_i \setminus \{f\}}) = g_f^i(t)\). To guarantee that \(g_f^i(t) = K_f\), it is necessary that \(|\Lambda_d(t)|_{\mathcal{F}_i} - |\Lambda_d(t)|_{\mathcal{F}_i \setminus \{f\}} = K_f\).

Algorithm 4.2 shows type 3 end-to-end feedback mechanism performed at the sink \(d\). Upon receiving a packet from an upstream node (Step 4), for any unacknowledged flow at
Algorithm 4.2 Type 3 End-to-End Feedback Mechanism at Sink d.

1: input: $\mathcal{F}_i, \forall i \in \mathcal{V}, K_f, \forall f \in \mathcal{F}$
2: for each generation
3: $\mathcal{A}_i := \emptyset, \forall i \in \mathcal{V}$
4: while a new packet is arrived at time $t$
5: for $i \in \mathcal{V}$
6: for $f \in \mathcal{F}_i \setminus \mathcal{A}_i$
7: if $|\Lambda_d(t)|_{\mathcal{F}_i} - |\Lambda_d(t)|_{\mathcal{F}_i \setminus \{f\}}$ is equal to $K_f$
8: Send feedback control packet for the flow $f$ to node $i$
9: $\mathcal{A}_i := \mathcal{A}_i \cup \{f\}$
10: end if
11: end for
12: end for
13: end while
14: end for

node $i$ (i.e., set $\mathcal{A}_i$), the sink checks whether it has enough exclusive packets for that flow (Step 7). If there exists a flow $f \in \mathcal{F}_i$ with enough exclusive packets, the sink will send the feedback packet to the appropriate node (Step 8) and the node will reduce its transmission rate by $r_f$. Then, the sink adds flow $f$ to the set $\mathcal{A}_i$ (Step 9).

4.4 Performance Evaluation

In this section, we present the performance analysis and comparison for the proposed feedback mechanisms as well as the power efficiency of RLNC with buffer sharing. We construct the reverse multicast tree from the sources to the sink which is the root of the tree. For the packet success probability, we follow the model used in [117]. For each outputs link of node $i$, the packet success probability $\alpha_i$ is randomly chosen with probability $\lambda \alpha_i^{(\lambda-1)}$, where $\lambda$ is a tunable design parameter. The mean of the packet success probability is $\lambda/ (\lambda + 1)$. The higher the value of $\lambda$, the lower rate the packet loss in the network. The results are averaged over 1000 simulation runs. The power consumption of node $i \in \mathcal{V}$ is the number of packets transmitted by node $i$ times the power consumed to transmit one
packet. To normalize the power consumption, we divide it by the number of flows, the generation size, and the power consumed to transmit one packet.

We compare RLNC with buffer sharing and RLNC without buffer sharing for both link-by-link and end-to-end feedback mechanisms. The comparison is carried out in terms of the power consumption and the decoding delay. First, we consider a topology with 100 intermediate sensor nodes and 30 sources randomly deployed in a 100 m × 100 m square area. The sink is placed at the upper-right corner. The generation size of different flows are equal. We set \( \lambda \) equals to 2. Fig. 4.2 compares the normalized power consumption of various feedback mechanisms for the generation size equal to 10, 50, 100, 150, and 200. The normalized power consumption decreases by increasing the number of packets in each generation. We notice that the results are normalized by the generation size. For values greater than 100, the improvement in the power consumption achieved by increasing the generation size is negligible. Therefore, for the rest of simulations, we set \( K_f = 100 \) for
all the flows. For $K_f = 100$, by using buffer sharing in RLNC, the power consumption can be reduced by 15% and 22% for link-by-link and end-to-end feedback mechanisms, respectively.

Figs. 4.3 (a) and (b) compare the normalized power consumption of RLNC with/without buffer sharing using link-by-link and end-to-end feedback mechanisms. We vary the design parameter $\lambda$ and measure the power consumption. Results shows that the link-by-link feedback mechanism performs better when compared to the end-to-end feedback mechanism.
Figure 4.4: The normalized decoding delay between RLNC with and without buffer sharing.

for RLNC with buffer sharing and RLNC without buffer sharing. For all the mechanisms, the power consumption decreases as $\lambda$ increases since a higher packet success probability will lead to a lower number of packet losses in the network. For large values of $\lambda$ (i.e., $\lambda \geq 7$), since the loss rate of the links approach zero, the benefit of using network coding is negligible. For RLNC with end-to-end feedback mechanism, we observe that the performance of RLNC with buffer sharing and end-to-end feedback mechanism is close to that of RLNC without buffer sharing and link-by-link feedback mechanism. This means, even if the intermediate sensor nodes are not capable of generating feedback, end-to-end feedback mechanism can be used with buffer sharing while the performance is close to RLNC without buffer sharing and link-by-link feedback mechanism.

Next, we present the decoding delay performance. The decoding delay is the difference between the first time that the sink can decode all the packets of a generation and the time that the sources start transmission of that generation. The decoding delay is equal to

$$\min \left\{ t \mid g_f^d(t) = K_f, \forall f \in F \right\}.$$  

We divide the total delay by the generation size in order to obtain the normalized decoding delay. In RLNC without buffer sharing, the sink decodes packets from different flows independently. Therefore, the decoding delay is different for
various flows. For RLNC without buffer sharing, we consider the average delay of various flows in the network. Fig. 4.4 compares the normalized decoding delay for RLNC with and without buffer sharing. Note that the type of feedback has no effect on the decoding delay since the rate at which the sink receives the innovative packets is independent of the feedback type. For $\lambda = 1$, the normalized decoding delay of RLNC with buffer sharing is 30% higher than RLNC without buffer sharing. As $\lambda$ increases, the decoding delay of both techniques approaches since the number of lost packet diminishes and the effect of using network coding disappears. As Figs. 4.2 and 4.4 show, the power gain achieved by buffer sharing is at the cost of a higher decoding delay.

Next, we vary the number of sources from 10 to 30 with steps of 5. The number of intermediate nodes is 100 and the field size is 100 m $\times$ 100 m. The parameter $\lambda$ is set to 2. Figs. 4.5 (a) and (b) show the power consumption of RLNC with and without buffer sharing, respectively. These results are normalized to the number of sources. For RLNC with buffer sharing, increasing the number of sources decreases the normalized power consumption for all both end-to-end and link-by-link feedback mechanisms. It is because the average number of flows passing each intermediate node increases. It provides more opportunities to combine the packets of different flows and save more power at the intermediate nodes. For RLNC without buffer sharing, increasing the number of sources has a superposition effect and does not change the normalized power consumption.

Finally, we vary the number of nodes from 60 to 140 with steps of 20 in the network while we keep the number of sources to be equal to 30. The field size is increased from $\sqrt{60}$ m $\times$ $\sqrt{60}$ m to $\sqrt{140}$ m $\times$ $\sqrt{140}$ m. The parameter $\lambda$ is set to 2. Figs. 4.6 (a) and (b) compare different versions of the end-to-end and link-by-link feedback mechanisms. For both RLNC with and without buffer sharing, the power consumption is increased by increasing the number of sources because the number of hops from the sources to the sink
Chapter 4. Feedback Mechanism for RLNC with Buffer Sharing in Lossy WSNs

Figure 4.5: Normalized average power consumption versus the number of sources for $\lambda = 2$.

is increased and more transmissions are required to deliver one packet to the sink.

4.5 Summary

In this chapter, we studied the use of RLNC with buffer sharing in wireless sensor networks. In RLNC with buffer sharing, intermediate sensor nodes which relay information of multiple sources maintain one buffer for flows towards the sink. Since the transmitted packets convey
information of multiple flows, it is not possible to distinguish which flow a packet belongs to after transmission. Through analytical modeling, we showed when an intermediate node can stop transmission of the packets for a particular flow without changing the average decoding delay of the sink. Then, we designed two feedback mechanisms, namely link-by-link and end-to-end, for RLNC with buffer sharing. In link-by-link feedback mechanism, an intermediate node transmits feedback control packet to its upstream neighboring node and acknowledges each flow individually. The upstream node reduces its transmission
rate when it receives a feedback packet. In end-to-end feedback mechanism, the sink is responsible of transmitting the feedback. We investigated the power efficiency of both feedback mechanisms and compared the case with and without buffer sharing, separately. Simulation results showed that for both feedback mechanisms, RLNC with buffer sharing has a better power efficiency compared to RLNC without buffer sharing. The cost of achieving this gain is a longer delay in decoding the packets.
Chapter 5

Link Loss Inference in Wireless Sensor Networks with Random Linear Network Coding

Due to the stochastic nature of the channels, communication links between sensors are also subject to fading and interference [1, 3]. Although techniques such as ARQ and forward error correction (FEC) can improve the reliability of packet transmission between the source nodes and the sink, energy and bandwidth constraints in WSNs impose barriers to deploying those techniques. Recent studies have shown that network coding can not only increase the network throughput in wired and wireless networks [7, 8, 11], but can also improve the robustness in systems with erasure channels [20, 25, 118]. Network coding outperforms ARQ and FEC techniques in terms of a lower delay and a higher network throughput [20]. Network coding also outperforms erasure codes since intermediate sensor nodes do not need to decode packets. It reduces the complexity of operation at intermediate sensor nodes and reduces the end-to-end delay.

For both wireless and wired networks, it is of interest to monitor the behavior of the network, identify the link status, estimate the delay and loss rate of different links. Such a technique is called network tomography or network monitoring in the literature. Network tomography for wired and wireless networks has received much attention recently [20, 27, 28, 41, 60]. Network monitoring is especially important for WSNs as the information can
allow sensor nodes to make decisions on routing and to bypass the links which are either congested or prone to failure.

As mentioned in Chapter 1, network monitoring can be performed either in an active or passive manner [27]. Passive monitoring schemes are more attractive for WSNs because of the limited power supply and bandwidth constraints in these networks [31, 32]. The problem of supporting passive network monitoring by data aggregation techniques is studied in [117, 119]. Identification of poorly performing links using end-to-end observations and maximum likelihood estimate in a WSN is studied in [120]. However, none of the above works use network coding for network monitoring. In [121], Gui et al. used a linear algebraic approach and proposed an active monitoring scheme for packet loss inference on mesh topologies. Active network monitoring using network coding is studied in [11]. The identifiability challenge in active monitoring of coded packet networks is studied in [122]. Results in [123] show that for active monitoring techniques using network coding, appropriate selection of the number and location of sources and destinations have a significant impact on the accuracy of estimation. Network coding has also been used for topology inference. In [124], network coding and active network monitoring are used to study the topology inference problem. The passive monitoring of coded packet networks employing RLNC is studied in [125].

In this chapter, we focus on designing algorithms to infer the loss rate of different links in WSNs by passively monitoring the traffic at the sink. All sensor nodes (including source nodes and intermediate nodes) are capable of performing RLNC. The link identification problem in coded packet networks is different from the traditional routed packet networks. In a coded packet network, the loss rate of a chosen path is the maximum loss rate of all the links of that path [21]. Therefore, it can be infeasible to infer the loss rate of the intermediate links by simply observing the path loss rate from the source node towards the
sink. This behavior of the coded packet networks makes the link identification problem more challenging. However, we can benefit from the coding operations performed at the intermediate nodes and extract information about the path loss rate from the intermediate nodes to the sink. The contributions of this chapter are as follows:

- By inspecting the contents of the received coded packets at the sink and adopting the *subspace* properties of network coding, we propose an estimation technique to estimate the path loss rates not only from the source nodes but also from several intermediate nodes to the sink.

- We characterize the conditions under which the link loss rate of a particular link can be estimated.

- We propose a passive loss inference with random linear network coding (PLI-RLC) algorithm using the proposed estimation technique to determine the loss rate of intermediate links in the network. Results show that our proposed algorithm has a lower percentage of false detection compared to a Bayesian inference algorithm [126].

The work in this chapter differs from the existing related work in the literature in several aspects. The problem of passive inference of the possible locations of link failures for multicast traffic is studied in [127]. Our work differs from [127] in that (a) we consider unicast traffic, and (b) sensor nodes are able to buffer data packets and perform coding in order to improve the reliability of the system. In [126], the problem of passive loss inference in WSNs with network coding is studied. Our work differs from [126] in that we study the identifiability problem and compute the loss rate of links precisely instead of merely categorizing the links as either good or bad. To the best of our knowledge, this work is the first attempt to use the subspace properties in network coding for the link loss inference problem in WSNs.
The rest of this chapter is organized as follows: In Section 5.1, the system model is introduced. In Section 5.2, we propose estimators to determine the path loss rate and an PLI-RLC algorithm to infer the link loss rate. In Section 5.3, we evaluate the performance of our proposed algorithm. A summary is given in Section 5.4.

5.1 System Model

5.1.1 Network Structure

Consider a WSN. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of sensor nodes and $\mathcal{L} = \{1, 2, \ldots, L\}$ denote the set of directed wireless links. Let $\mathcal{S}$ denote the set of source nodes where $\mathcal{S} \subset \mathcal{N}$. There is one sink node in the network. The data is gathered at the sink over a directed tree rooted at the sink. The leaves of the tree are source nodes. We notice that the intermediate nodes can be sources as well.

With a tree structure, there is a unique path from every source node to the sink. We call the packets transmitted from source $s$ to the sink as the flow of source $s$. Let $\mathcal{P}$ denote the set of paths in the network. Let $P_s \in \mathcal{P}$ denote the path from source node $s$ to the sink and $\mathcal{L}_s$ denote the set of links on path $P_s$. For any intermediate node $i \in \mathcal{N}$ in the tree, the upstream flows are flows of those sources in $\mathcal{S}$ using node $i$ as a relay. Fig. 5.1 shows a sample WSN with nodes 1, 2, 3, 4, and 7 as the source nodes.

5.1.2 Random Linear Network Coding (RLNC)

Coded packet networks can provide reliable data gathering for WSNs. Each source has a stream of data packets to transmit towards the sink. Each packet is a sequence of symbols over a Galois field $GF(2^q)$. For practical purposes, the source nodes group the packets into generations $[118]$. Each source transmits one generation at a time. When the sink
can decode the packets of one generation, the source moves to the next generation. Let $K$ denote the size of each generation. Every source transmits packets periodically. Without loss of generality, sources transmit with a rate of one packet per second.

Each sensor node can perform network coding by combining several incoming packets to create a coded packet. We assume the use of RLNC \footnote{128}. Each coded packet transmitted by node $i \in \mathcal{N}$ is a weighted linear combination of the packets in the buffer of node $i$ using a set of random weights taken from $GF(2^q)$ called the \textit{coding coefficients}. Nodes append the coding coefficients in the header of the coded packet to enable decoding at the sink. The vector of coefficients is called the \textit{coding vector}. Each sensor node has several buffers to store the packets received from different upstream flows. Upon receiving a packet from flow $s$, the node checks whether the packet is linearly independent of the other packets currently stored in the buffer of flow $s$. An incoming packet is placed in the buffer if it is linearly independent of the currently buffered packets. When a node transmits a packet of flow $s$, it linearly combines the stored packets in the buffer of flow $s$ using the random coding coefficients.

We slightly modify the randomized linear network coding for the monitoring purposes. In this modified version, an intermediate node with more than one incoming link randomly selects two flows from two different incoming links and combines the buffers of those flows.
These flows do not have common information about one source since they are from different incoming links. We notice that the intermediate node still relays two separate flows. In this case, some of the transmitted packets have information of more than one source. Since we assume that the maximum loss rate is 0.5 in the network, the input rate for innovative packets of both flows which are combined is at least 0.5. Therefore, for each of the transmitted flows, the intermediate node always has innovative packets to transmit and acts as a source. We show that merging the buffers can help to derive information of the path from the intermediate node to the sink. The sink receives packets from $|S|$ flows and stores the packets of every flow $s$ at buffer $B_s$. Since some of the intermediate nodes have combined different flows, buffer of flow $s$ may contain information of other sources as well. To decode the packets, the sink uses the buffers of all flows. The sink can decode the data if it has $K|S|$ linearly independent packets in different buffers. We notice that the sink needs to keep different buffers for monitoring purposes but not for decoding purposes.

### 5.2 Problem Formulation

The objective is to study the link identification and loss inference problem for WSNs using the information obtained at the sink. We first need to estimate the end-to-end path loss rates from the sources to the sink. Then the loss rate of links can be inferred using the information of path loss rates. In the next two subsections, we consider the path loss estimation and link identifiability problems.

#### 5.2.1 Path Loss Estimation

In this subsection, the objective is to accurately estimate the loss rate of the paths in the WSNs by passively observing the data traffic received at the sink. This is based on tracking the propagation of packets known as innovative packets from different source nodes. Such
packets are innovative as they carry new, yet unknown, information generated by the source nodes. Let $\Pi_s(t)$ represent the subspace spanned by those coefficients of the packets in $B_s$ corresponding to the source $s$ at time $t$. For example, if the first $K$ coefficients of each coding vector belong to the source $s \in S$, then $\Pi_s(t)$ is the subspace spanned by the first $K$ coefficients of the packets in $B_s$ at time $t$.

The time at which the sink expects to receive the first packet from a source depends on the distance (i.e., number of hops) between the source node and the sink. Let $t^1_s$ denote the earliest time at which the sink can receive the first packet of source $s$. Let $\bar{t}_s$ denote the time at which the subspace $\Pi_s$ becomes full rank. The maximum rank of subspace $\Pi_s$ is equal to the number of packets generated by source $s$ which is $K$. That is $\dim(\Pi(\bar{t}_s)) = K$.

The rate at which buffer $B_s$ receives innovative packets from source $s$ is the minimum rate at which the nodes on the path $P_s$ receives packets \(^{20}\). This is equal to the minimum rate at which packets are transmitted on the links in $L_s$ multiplied by the corresponding link success rates. Since the rate at which intermediate nodes relay the packets of flow $s$ is one, the rate at which buffer $B_s$ receives innovative packets of source $s$ is the minimum of the link success rate for the links in $L_s$. This is also considered as the the path loss rate of path $P_s$ which can be written as\(^3\):

$$1 - \alpha_s = \min_{l \in L_s} \{1 - e_l\}, \quad \forall \ s \in S$$

(5.1)

where $\alpha_s$ is the loss rate of the path $P_s$ and $e_l$ denotes the link loss rate of link $l$. We assume the link loss rate is less than 0.5, since a link with loss rate greater than 0.5 should not be used in practice. At every second, buffer $B_s$ receives an innovative packet of source $s$ with probability $\alpha_s$. Therefore, the dimension of subspace $\Pi(\bar{t}_s)$ has a negative binomial

\(^1\)In the case of routing without network coding, we have $(1 - \alpha_s) = \prod_{l \in L_s} (1 - e_l)$.
distribution and it can be used to estimate the path success rate as:

\[
\tilde{\alpha}_s(t) = 1 - \frac{\dim(\Pi_s(t))}{t - t^1_s + 1}, \quad \forall s \in \mathcal{S}.
\] (5.2)

where \(\tilde{\alpha}_s(t)\) is the estimation of \(\alpha_s\) at time \(t\). We notice that for values of \(t > \bar{t}_s\), since the dimension of the subspace is not incremented, the estimator uses the last time slot which contains useful information for estimation which is \(\bar{t}_s\). Therefore, we have \(\tilde{\alpha}_s(t) = \tilde{\alpha}_s(\bar{t}_s)\) for \(t > \bar{t}_s\). The estimation error in \(\tilde{\alpha}(t)\) asymptotically follows a normal distribution with zero mean and variance \(\sigma^2_s(t)\) as [123]:

\[
\sigma^2_s(t) \approx \alpha_s(1 - \alpha_s) \frac{t - t^1_s + 1}{t - t^1_s + 1}, \quad \forall t \in [t^1_s, \bar{t}_s], \ s \in \mathcal{S}.
\] (5.3)

We notice that \(t\) in fact cannot go to infinity. The maximum value of \(t\) which can be used to estimate \(\alpha_s\) is \(\bar{t}_s\). At time \(\bar{t}_s\), the network experiences the minimum achievable variance of estimation error. The minimum value for \(\bar{t}_s\) which can be achieved when there is no loss is equal to the size of the generation \(K\), which is a design parameter. Thus, higher values of \(K\) can increase the accuracy of the estimation. However, it can impose additional delay to the decoding. Also, the intermediate nodes need additional memory capacity and the sink needs more computational capability for the decoding process. Hence, there is an inherent tradeoff between the accuracy and the complexity.

Given the path loss rates observed at the sink, it is challenging to infer the link loss rates as the number of variables (i.e., link loss rates) is more than the number of equations (i.e., path loss rates). From (5.1), the path loss rate \(\alpha_s\) of a path \(P_s\) is an upper bound on the link loss rate \(e_l\) for all link \(l \in \mathcal{L}_s\).

We now show that more information can be obtained for the link loss rates in the network by taking into account the properties of the coded packet networks. We call an
intermediate node with more than one incoming link as a virtual source. We note that an intermediate node can be a source node as well. In this case, we do not consider it as a virtual source since this assignment cannot provide more information. Let $\mathcal{V}$ denote the set of virtual sources in the network. For virtual source $v$, let $P_v$ denote the path from the virtual source to the sink and $\mathcal{L}_v$ denote the set of links constructing path $P_v$. From the RLNC, every virtual source $v \in \mathcal{V}$ combines the packets of two upstream flows. Therefore, among $|\mathcal{S}|$ flows received at the sink, there exists at least one flow which carries information of more than one upstream flows of the virtual source $v$. Let $\mathcal{B}_v$ denote the buffer for that flow at the sink and let $\mathcal{S}_v$ denote the set of flows whose buffers have been combined by the node $v$. We notice that this flow may contain information of other sources as well. For a virtual source $v \in \mathcal{V}$, let $\Pi_v(t)$ denote the subspace spanned by those coefficients of the packets in $\mathcal{B}_v$ corresponding to the sources in $\mathcal{S}_v$ at time $t$. Let $t_{v}^1$ denote the earliest time at which the sink can receive the first packet in $\mathcal{B}_v$. Let $\bar{t}_v$ denote the time at which the subspace $\Pi_v$ becomes full rank.

The rate at which the node $v$ receives the innovative packets of the sources in $\mathcal{S}_v$ is greater than or equal to one since the packets received are from two different flows each with rate at least equal to 0.5 (this is from the assumption that we made on the minimum loss rate of the links). Therefore, the packets transmitted from node $v$ are always innovative packets with respect to the generated packets of sources in $\mathcal{S}_v$. Therefore, the node $v$ can be considered as a source generating packets which contribute to the dimension of $\Pi_v$. Thus, the behavior of the actual and virtual sources are similar if we monitor the dimension of $\Pi_v$ instead of $\Pi_a$. We can use the following estimator to estimate the path loss rate for a path from virtual source $v$ to the sink

$$\tilde{\alpha}_v(t) = 1 - \frac{\dim(\Pi_v(t))}{t - t_{v}^1 + 1} \quad \forall \; v \in \mathcal{V},$$

(5.4)
where $\tilde{\alpha}_v(t)$ is the estimate of $\alpha_v$ at time $t$. We notice that similar to $\alpha_s(t)$, for values of $t > \bar{t}_s$, we have $\alpha_v(t) = \alpha_v(\bar{t}_v)$. Similar to the approximation proposed for the variance of the error obtained for $\tilde{\alpha}_s(t)$, the variance of error in the estimation of $\alpha_v$ can be obtained as follows:

$$
\sigma^2_v(t) \approx \frac{\alpha_v(1 - \alpha_v)}{t - t_s^1 + 1}, \quad \forall \ t \in [t_s^1, \bar{t}_v], \ v \in V.
$$

(5.5)

5.2.2 Identifiability Problem

The main objective of this chapter is infer the loss rate of the network links. However, it is not always possible to infer loss rates for all the links using end-to-end measurements. Given the topology of the network, we can classify the links in three sets: links which are not identifiable, links which are possibly identifiable, and links which are identifiable. A link $l$ is not identifiable if for any assignment for the loss rates of the links in the network, we are not able to infer the link loss rate of link $l$ using end-to-end measurement. The possible identification is a new notion of identifiability introduced in this chapter. A link is possibly identifiable if the identifiability of the link depends on the loss rate of other links. That is, the link loss rate can be inferred for some assignments, but not for other assignments. For an identifiable link, the loss rate can always be inferred independent of the loss rate of other links.

For each directed link $l \in \mathcal{L}$, let head($l$) and tail($l$) denote the nodes attached to the head and tail of link $l$, respectively. That is, head($l$) $\in \mathcal{N}$ denotes the transmitter node of link $l$ and tail($l$) $\in \mathcal{N}$ denotes the receiver node of link $l$. The following lemma helps to distinguish between these three groups of links.

**Lemma 8** For the directed link $l \in \mathcal{L}$:

(a) It is identifiable if head($l$) $\in \mathcal{S} \cup \mathcal{V}$ and tail($l$) is the sink.

(b) It is possibly identifiable if head($l$) and tail($l$) $\in \mathcal{S} \cup \mathcal{V}$.
(c) Otherwise, it not identifiable.

The proof of Lemma 8 is given in Section 5.5.1. Using the estimators presented in Section 5.2.1 and Lemma 8, we propose the passive loss inference with random linear network coding (PLI-RLC). Algorithm 5.1 shows the PLI-RLC algorithm. The algorithm monitors the incoming packets to the sink and estimates the loss rate of the links. The algorithm is composed of two parts. In the first part (lines 3–18), the algorithm calculates the dimension of all subspaces at every time slot according to the buffered packets and the packets which are received in that time slot. Then, it estimates the packet loss rate based on the dimension of the subspaces. We notice that the PLI-RLC algorithm is a passive monitoring algorithm and time $t$ is the system time instead of the algorithm time. If at a time instance $t$, the variance of estimation error of all the path loss rates obtained using equation (5.3) is less than a certain threshold $E_{th}$, then the algorithm stops monitoring the received packets. In the second part of the algorithm (lines 19–23), the algorithm determines the loss rates.

5.3 Performance Evaluation

In this section, we evaluate the performance of the proposed estimators and our proposed PLI-RLC algorithm. Moreover, we compare our proposed PLI-RLC algorithm with the Bayesian inference algorithm [126]. We developed a discrete-event packet level simulator for the WSNs using MATLAB. For RLNC, the size of the generation $K$ is set to 100. The Galois field size $q$ is equal to 8. We set the threshold value $E_{th}$ to be equal to $5 \times 10^{-4}$. 
Algorithm 5.1 - Passive Loss Inference with Random Linear Network Coding (PLI-RLC)

Algorithm

1: input: network topology, sets $S$ and $V$; output: link loss rates
2: for every generation
3: While at time slot $t$
4: for each source node $s \in S$
5: if $t \geq t^1_s$,
6: Calculate $\dim(\Pi_s(t))$.
7: Calculate $\tilde{\alpha}_s(t)$ using (5.2).
8: Calculate $\sigma^2_s(t)$ using (5.3) and $\alpha_s(t) \approx \tilde{\alpha}_s(t)$.
9: end for
10: for each virtual source $v \in V$
11: if $t \geq t^1_v$,
12: Calculate $\dim(\Pi_v(t))$.
13: calculate $\tilde{\alpha}_v(t)$ using (5.4).
14: Calculate $\sigma^2_v(t)$ using (5.5) and $\alpha_v(t) \approx \tilde{\alpha}_v(t)$.
15: end for
16: if $(t \geq \max_{i \in S \cup V} t^1_i)$ and $(\sigma^2_i(t) < \epsilon_{th}, \forall i \in S \cup V)$
17: Break while.
18: end while
19: for each link $l \in L$,
20: if $(\text{head}(l), \text{tail}(l) \in S \cup V)$
21: if $\tilde{\alpha}_{\text{head}(l)} > \tilde{\alpha}_{\text{tail}(l)}$,
22: Set $e_l := \tilde{\alpha}_{\text{head}(l)}$
23: end for
24: end for

5.3.1 Number of Identifiable Links

We investigate the performance of PLI-RLC algorithm on inferring the link loss rates. We consider the network in Fig. 5.1. For the link loss rate model, we follow the model proposed in [117]. For each link $l \in L$, the probability distribution function that the loss rate of the link becomes $e_l$ is

$$f(e_l) = \lambda(1 - e_l)^{(\lambda - 1)}, \quad e_l \in (0, 1],$$

(5.6)

where $\lambda$ is a tunable parameter. The mean of the link loss error rate is $1/(\lambda + 1)$. The link loss rate on each link is randomly chosen from the distribution in (5.6). If the chosen link loss rate is greater than 0.5, we set it to 0.5. We vary the value of $\lambda$ from 5 to 10. The
Figure 5.2: Average number of links whose rates determined using PLI-RLC algorithm.

PLI-RLC algorithm is run 100 times for each value of $\lambda$. Fig. 5.2 shows the average number of links whose loss rates can be inferred using our proposed PLI-RLC algorithm. Higher values of $\lambda$ decreases the average of link loss rate in the network. When $\lambda$ is increased from 5 to 10, the average number of link with determined loss rate is slightly decreased from 4.12 to 4.04. We notice that there are 2 identifiable links and 5 possibly identifiable links in this network. That is, from 5 possibly identifiable links, on average the link loss rate of 2 can be determined by PLI-RLC algorithm.

5.3.2 Performance Comparison

In this section, we compare our proposed PLI-RLC algorithm with the Bayesian inference algorithm proposed in [126]. In [126], the posterior probability of the link loss rate is derived given the observed path loss rates from sources to the sink. Then, a link is categorized as a good link if with probability greater than 0.5 the link loss rate is greater than a threshold $\gamma$. To compare our scheme with the one proposed in [126], we classify each link to be either a good or a bad link using the observations obtained from the PLI-RLC algorithm.
For link $l \in \mathcal{L}$ whose loss rate is determined by the algorithm, we classify the link as a good link if the loss rate of the link is higher than $\gamma$. For a link $l \in \mathcal{L}$ whose link loss rate cannot be determined using PLI-RLC algorithm, we derive the posterior probability of loss rate $e_l$ using the observed path loss rates. Let $f(e_l \mid \{\alpha_i, \forall i \in \mathcal{S} \cup \mathcal{V}\})$ denote the probability distribution of link loss rate $e_l$ for link $l$ whose rate cannot be determined using the PLI-RLC algorithm. Assuming no a priori information on $e_l$, the posterior probability distribution function of $e_l$ can be written as

$$f(e_l \mid \{\alpha_i, \forall i \in \mathcal{S} \cup \mathcal{V}\}) = \begin{cases} \frac{1}{\min_{\{i \in \mathcal{S} \cup \mathcal{V} \mid l \in \mathcal{L}_i\}} \{\alpha_i\}}, & \text{if } e_l \leq \min_{\{i \in \mathcal{S} \cup \mathcal{V} \mid l \in \mathcal{L}_i\}} \{\alpha_i\}, \\ 0, & \text{otherwise.} \end{cases}$$

Link $l$ is classified as a bad link if:

$$\int_0^\gamma f(e_l \mid \{\alpha_i, \forall i \in \mathcal{S} \cup \mathcal{V}\}) \, de_l \geq 1/2. \quad (5.7)$$
We use the PLI-RLC algorithm along with equation (5.7) to classify all the links as either good or bad links. We compare the performance of LPI-RLC and Bayesian inference algorithms using the ratio of false detection. A detection is false if we declare a good link as a bad link, or vice versa. We consider a WSN with 50 sensor nodes randomly deployed in a 100 m × 100 m field. 25 sensor nodes are randomly selected as the sources. Each source node selects the path with minimum power consumption among all the paths it has towards the sink. The link loss rates are selected randomly based on the distribution function in equation (5.6). The value of \( \lambda \) is varied from 5 to 10 and we measure the fraction of links whose status are detected incorrectly. The threshold value \( \gamma \) is set to 0.7. The results are averaged over 100 simulation runs. We notice that the number of links which participate in data communication in each simulation run depends on the topology of the network and the routing tree. Fig. 5.3 shows the false detection ratio when \( \lambda \) is varied from 5 to 10.

Fig. 5.3 compares LPI-RLC and Bayesian inference algorithms for varying \( \lambda \). Higher values of \( \lambda \) reduce the number of bad links in the network. As Fig. 5.3 shows, the false detection ratio is decreased when \( \lambda \) is increased. When \( \lambda \) is equal to 5, 32% of the links are bad links and 68% of them are good links. In this case, the percentage of the false detection in our algorithm is 14 percent less that of Bayesian inference algorithm. When \( \lambda \) is equal to 10, 11% of the links are bad links and 89% of them are good links and the percentage of the false detection in our algorithm is 11 percent better than that of the Bayesian inference algorithm.

5.4 Summary

In this chapter, we studied the loss inference problem in WSNs while nodes perform randomized linear network coding. By using the subspace property of network coding, we
proposed estimators to determine the path loss rate between a sink and a source. We showed that the intermediate nodes with multiple incoming links can be considered as virtual sources and we are able to determine the path loss rates from the virtual sources to the sink as well. We considered the assumption that there is no link with loss rate higher than 0.5 in the network in deriving the estimators. We characterized the conditions for a link to be either identifiable, possibly identifiable, or not identifiable. We proposed a passive loss inference with random linear network coding (PLI-RLC) algorithm to determine the link loss rates. Simulation results showed that our proposed algorithm is accurate. It also has a lower percentage of false detection compared to the Bayesian inference algorithm.

5.5 Analytical Proof

5.5.1 Proof of Lemma

Consider a link \( l \in \mathcal{L} \) with two actual/virtual sources at the head and tail. Let \( \alpha_{\text{head}(l)} \) and \( \alpha_{\text{tail}(l)} \) denote the path loss rate from the nodes located at the head and tail of \( l \) to the sink, respectively. The relationship between the path and link loss rates can be expressed as

\[
\alpha_{\text{head}(l)} = \max\{e_l, \alpha_{\text{tail}(l)}\},
\]

where \( \alpha_{\text{head}(l)} \) and \( \alpha_{\text{tail}(l)} \) are observed values and \( e_l \) is an unknown variable. The loss rate of link \( l \) can be determined as \( \alpha_{\text{head}(l)} \) if \( \alpha_{\text{head}(l)} > \alpha_{\text{tail}(l)} \). However, if \( \alpha_{\text{head}(l)} \leq \alpha_{\text{tail}(l)} \), then the loss rate can have any value less than or equal to \( \alpha_{\text{head}(l)} \). It means that for link \( l \), we have

\[
e_l \begin{cases} 
= \alpha_{\text{head}(l)} & \text{if } \alpha_{\text{head}(l)} > \alpha_{\text{tail}(l)}, \\
\leq \alpha_{\text{head}(l)} & \text{if } \alpha_{\text{head}(l)} \leq \alpha_{\text{tail}(l)}. 
\end{cases}
\]

Therefore, such a link is possibly identifiable. A link \( l \) is identifiable if for any assignment
of link loss rates, we have $\alpha_{\text{head}(l)} > \alpha_{\text{tail}(l)}$. This only happens when $\alpha_{\text{tail}(l)} = 0$ which means tail($l$) is the sink. Now, consider link $l$ while either head($l$) or tail($l$) is not actual/virtual source. Consider the case that tail($l$) is not an actual/virtual source. tail($l$) has one incoming link which is $l$. Let $l'$ denote the outgoing link of node tail($l$). Any path including $l$ should contain $l'$ as well. Therefore, from the point of view of sink, $l$ and $l'$ are not distinguishable. Therefore, link $l$ is not identifiable for any pattern of link loss rates. The case that head($l$) is not an actual/virtual source is similar. \hfill \blacksquare
Chapter 6

Cardinality Estimation in RFID Systems with Multiple Readers

Tag estimation is widely used as a preliminary phase in ALOHA-based interrogation techniques [46, 47, 48]. Readers can adjust the frame size based on the estimation of tag population. Another application of tag estimation techniques, which has recently received attention, is the anonymous tracking of objects [49, 50]. In order to preserve the privacy and anonymity of the tag users, it may not be necessary to identify each individual user in some RFID applications. Instead, the goal is to estimate the total number of tags (or users) in the system. This is called the cardinality estimation (or tag population estimation) problem in RFID systems.

In [49], Kodialam et al. proposed the zero-based and collision-based tag estimation techniques using a framed-slotted ALOHA model with a single reader. In [50], they extended their work by introducing the enhanced-zero based (EZB) estimator, which is an asymptotically unbiased estimator. Using this technique, the mean and variance of the estimation error approach zero when the estimation process is repeated multiple times. Although the EZB algorithm can also be used for RFID systems with multiple readers, the variance of estimation error increases exponentially with the number of readers. In [47], Qian et al. proposed the lottery frame (LoF) scheme, which is a replicate-insensitive estimation protocol. LoF applies the hash functions with geometric distribution to tag IDs to select the time slots for transmission.
For large scale RFID systems, it is necessary to deploy multiple readers with overlapped interrogation zones to fully cover the area and achieve a high accuracy in the estimation. Consequently, a tag can be within the interrogation zone of several readers simultaneously. For tracking applications, which require privacy and anonymity of the users, each tag only transmits a portion of its ID to the reader when it is being queried. Thus, readers cannot identify uniquely the individual tags. Thus, those tags which are within the range of multiple readers may be counted multiple times. We call this problem the *multiple counting problem*. This motivates us to propose estimation algorithms which are capable of estimating the number of tags in such systems. In this chapter, we study the problem in two different conditions. First, we assume that readers cannot perform interrogation synchronously. We develop an asynchronous multiple-reader cardinality estimation (A-MRCE) algorithm under this condition. Then, we study the problem where the readers can be synchronized for interrogations, and we develop a synchronous multiple-reader cardinality estimation (S-MRCE) algorithm. The contributions of this chapter are as follows:

- We propose two maximum likelihood (ML) estimators, namely asynchronous and synchronous exclusive estimators to estimate the number of tags, which are exclusively within the interrogation zone of a reader.

- We show that the error for these estimators is asymptotically normal and the estimators are asymptotically unbiased. We derive upper bounds on the variance of the estimation error. The accuracy of these bounds is validated via simulations.

- We develop two estimation algorithms namely, A-MRCE and S-MRCE algorithms using asynchronous and synchronous exclusive estimators, respectively.

- We validate the analytical models, investigate the performance of our proposed es-
timators, and compare our proposed A-MRCE and S-MRCE algorithms with the EZB [53] and LoF [17] algorithms. Although all these algorithms are asymptotically unbiased, the variance of the estimation error for A-MRCE and S-MRCE algorithms increases linearly with the number of readers, while it increases exponentially for EZB and LoF algorithms.

To the best of our knowledge, there is no prior work specifically considering the problem of tag population estimation for RFID systems with multiple readers. Although EZB and LoF algorithms can be used in RFID systems with multiple readers, since they are not designed for such systems, they can have poor performance under some scenarios as shown in Section 6.4. On the contrary, both A-MRCE and S-MRCE algorithms can be used in large scale RFID systems. Since S-MRCE algorithm needs synchronous operation of readers, this algorithm is suitable for systems where readers can operate synchronously.

The rest of this chapter is organized as follows: The system model is presented in Section 6.1. In Section 6.2, we first propose an asynchronous exclusive estimator. Then, we propose an A-MRCE algorithm to estimate the total number of tags in the RFID system. In Section 6.3, we propose a synchronous exclusive estimator and an S-MRCE algorithm. Performance evaluation and comparison are presented in Section 6.4. A summary is given in Section 6.5.

6.1 System Model and Problem Statement

6.1.1 Notations and Model

Consider an RFID system with multiple readers. Readers use framed-slotted ALOHA protocol for interrogations. This is the model proposed in EPCglobal Gen2 standard [16]. We consider a fixed frame size for an interrogation process. Each reader broadcasts a query
message, which includes information such as the frame size $f$, persistence probability $p$, and a random seed $q$ at the beginning of the interrogation process. Each tag decides whether or not to transmit in the current frame based on the persistence probability $p$. If a tag decides to transmit, it selects a time slot based on a uniform distribution related to its ID, the persistence probability $p$, and the random seed $q$. Note that given the specific values of the frame size $f$, the random seed $q$, and persistence probability $p$, the tag selects exactly the same slot in a frame of size $f$, regardless of how many times it has received the query message. To perform different interrogations, readers can alter the seed $q$. The interrogation results are independent whenever a different seed $q$ is being used. Multiple interrogations are used to improve the accuracy of the estimation process. To preserve the anonymity and privacy of the users, each tag only transmits part of its ID to the reader in the reply message. Therefore, the reader cannot individually identify the tags. Throughout this chapter, we assume that the interrogation time is fast enough such that the tags do not leave or enter the interrogation zone of a reader. According to EPCglobal Gen2 standard [45], the average duration of a time slot is less than 1 ms which is much smaller than the average time that the tags move around the system.

We now introduce some of the notations. Let $N$ and $\mathcal{R}$ denote the number of tags and the set of readers in the system, respectively. Let $\mathcal{T}_r$ denote the set of tags within the zone of reader $r \in \mathcal{R}$. Let $n_r$ denote the number of tags in the interrogation zone of reader $r \in \mathcal{R}$ (i.e., $n_r = |\mathcal{T}_r|$). We use $n_W$ to denote the number of tags within the range of a set of readers $\mathcal{W}$ (i.e., $n_W = |\cap_{w \in \mathcal{W}} \mathcal{T}_w|$). Two readers are called neighboring readers if there is a tag which is in the interrogation range of both of the readers. Let $\mathcal{H}_r$ denote the set of other readers which are neighbors of reader $r$. For reader $r$, let $\mathbf{v}_r = (v^1_r, \ldots, v^f_r)$ denote the vector created after performing an interrogation process, where $v^l_r$ (with $l = 1, \ldots, f$) indicates whether the $l^{th}$ time slot is empty (i.e., $v^l_r = 0$) or has at least one transmission (i.e.,
Chapter 6. Cardinality Estimation in RFID Systems with Multiple Readers

Figure 6.1: An RFID system with two readers $r_1$ and $r_2$. $n_{r_1} = 6$, $n_{r_2} = 7$, $N = 11$, $f = 10$, and $p = 1$. $v^l_r = 1$). The number of elements of vector $v_r$ is equal to the frame size $f$. We call vector $v_r$ as the *interrogation vector* of reader $r$. Assume reader $r$ performs $M$ interrogations using $M$ different seed values. We use vectors $v^1_r, \ldots, v^M_r$ to denote these interrogation vectors. Fig. 6.1 shows a two-reader RFID system with overlapped interrogation zones and the interrogation vectors.

### 6.1.2 Multiple Counting Problem

For an RFID system with multiple readers, by adding up the number of tags within the zone of all readers, one can obtain an estimator for the number of tags. Since some tags may appear in the zone of several readers, they are counted multiple times by different readers. Therefore, the estimation may not be accurate especially for dense RFID systems. To obtain an accurate estimate for the number of tags, the number of tags in the overlapped areas of neighboring readers is required in addition to the number of tags within the zone of each reader. In the estimation process, once a tag is counted by a reader, other readers should exclude that tag from their estimations. The total number of tags is the summation of the number of tags estimated by all the readers, while each reader excludes the tags that have already been counted by other readers. In general, for $|\mathcal{R}|$ readers $r_1, \ldots, r_{|\mathcal{R}|} \in \mathcal{R}$
with overlapping interrogation zones, the total number of tags \( N \) in the system is

\[
N = |\mathcal{T}_{r_1}| + |\mathcal{T}_{r_2} \setminus \mathcal{T}_{r_1}| + \cdots + |\mathcal{T}_{r_{|\mathcal{R}|}} \setminus \{\mathcal{T}_{r_1} \cup \cdots \cup \mathcal{T}_{r_{|\mathcal{R}|-1}}\}|. \tag{6.1}
\]

Note that the order chosen to calculate \( N \) has no effect on the final result. Moreover, if reader \( r_j \in \mathcal{R} \) shares no tags with readers \( r_1, \ldots, r_{j-1} \), then \( \mathcal{T}_{r_j} \setminus \{\mathcal{T}_{r_1} \cup \cdots \cup \mathcal{T}_{r_{j-1}}\} = \mathcal{T}_{r_j} \).

As (6.1) suggests, in order to estimate \( N \), the number of tags within the zone of a reader needs to be estimated. We call such an estimator \textit{single reader estimator}. Moreover, we need to estimate the number of tags which are only within the zone of a reader but are not in the zone of some other readers (e.g., \( |\mathcal{T}_{r_2} \setminus \mathcal{T}_{r_1}| \)). To estimate the number of tags which are exclusively located within the zone of a reader, we propose two estimators in the next sections, namely asynchronous exclusive and synchronous exclusive estimators. For the single reader estimator, we use the estimator proposed in [50]. Assume that reader \( r \) has performed \( M \) interrogations using \( M \) different seed values, the we have \( \mathbf{v}_r^1, \ldots, \mathbf{v}_r^M \). Let \( t^m \) denote the number of empty slots in \( \mathbf{v}_r^m \), \( m = 1, \ldots, M \). Using the EZB algorithm, the number of tags within the zone of reader \( r \) can be estimated as [51]:

\[
\tilde{n}_r = -f/p \ln \left( \sum_{m=1}^{M} t^m / M f \right). \tag{6.2}
\]

It is shown in [51] and [130] that the estimation error is asymptotically normal and unbiased. The variance of the estimation error is \( \sigma_n^2 = f (\exp(pn_r/f) - (1 + p^2n_r/f)) / (M p^2) \). We will use this estimator in both A-MRCE and S-MRCE algorithms.
6.2 Asynchronous Multiple Reader Cardinality Estimation

In some practical cases, it may not be possible for the readers to perform interrogations in a synchronous manner. This models the case when the readers are not equipped with accurate clocks or synchronization imposes a high overhead. Under such condition, we develop an asynchronous multiple reader cardinality estimation (A-MRCE) algorithm. The A-MRCE algorithm is used to estimate the total number of tags while readers perform interrogations independently and forward the information to a central controller. The A-MRCE algorithm implements (6.1) using asynchronous exclusive estimator described in the following subsection.

6.2.1 Asynchronous Exclusive Estimator

The asynchronous exclusive estimator is used to estimate the number of tags within the zone of a particular reader excluding the tags shared with some other readers, while readers perform interrogations independently. It facilitates implementation of equation (6.1).

Consider reader \( r \) and a set of readers \( \mathcal{W} \subseteq \mathcal{H}_r \). Let \( n_{r \setminus \mathcal{W}} \) denote the number of tags within the zone of reader \( r \) that do not belong to any of the readers in set \( \mathcal{W} \) (i.e., \( |T_r \cup \bigcup_{w \in \mathcal{W}} T_w| \)). Given the number of tags within the zone of readers in \( \mathcal{W}, n_{\mathcal{W}} \), we propose an asynchronous exclusive estimator to estimate \( n_{r \setminus \mathcal{W}} \).

Consider a time slot which is non-empty in vector \( v_r \). This indicates that one or more tags within the range of reader \( r \) has chosen that time slot. If this time slot is empty in \( v_w \) for any \( w \in \mathcal{W} \), it ensures that none of the tags within the range of readers in \( \mathcal{W} \) has chosen that slot. Let variable \( Z \) denote the number of time slots which are nonempty in \( v_r \) and empty in \( v_w, \forall w \in \mathcal{W} \). That is, given reader \( r \in \mathcal{R} \) and set \( \mathcal{W} \), we have
Chapter 6. Cardinality Estimation in RFID Systems with Multiple Readers

\[ Z = \sum_{l=1}^{f} v_r^l \prod_{w \in W}(1 - v_w^l). \]

The time slots, which are non-empty in \( v_r \) and empty in \( v_w \), are chosen by a subset of tags in the set \( \{ \mathcal{T}_r \setminus \bigcup_{w \in W} \mathcal{T}_w \} \). This suggests that the variable \( Z \) can be used to estimate \( n_{r \setminus W} \). The following theorem characterizes the distribution of variable \( Z \).

**Theorem 5** The random variable \( Z \) has a normal distribution with mean \( \mu_z \) and variance \( \sigma_z^2 \) for large values of \( f, n_{r \setminus W} \) and \( n_W \):

\[
\mu_z = f \left( 1 - \exp \left( -\frac{pn_{r \setminus W}}{f} \right) \right) \exp \left( -\frac{pn_W}{f} \right). \tag{6.3}
\]

\[
\sigma_z^2 = \mu_z \left( 1 - \mu_z/f \left( 1 + \frac{p^2 n_W}{f} \right) \right) - p^2 n_{r \setminus W} \exp \left( -\frac{2p}{f} (n_{r \setminus W} + n_W) \right). \tag{6.4}
\]

The proof of Theorem 5 is given in Section 6.6.1. By using the interrogation vectors \( v_r^1, \ldots, v_r^M, \) and \( v_w^1, \ldots, v_w^M, \forall w \in W \), one can obtain \( M \) samples of the random variable \( Z \), namely \( z^1, \ldots, z^M \). Let \( \mathbf{z} \) denote a vector composed of all the samples (i.e., \( \mathbf{z} = (z^1, \ldots, z^M) \)). From Theorem 5, the likelihood function of \( \mathbf{z} \) given \( n_{r \setminus W} \) and \( n_W \) is

\[
L(\mathbf{z}; n_{r \setminus W}, n_W) = \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp \left( -\frac{(z^m - \mu_z)^2}{2\sigma_z^2} \right). \tag{6.5}
\]

The maximum likelihood (ML) estimate of \( n_{r \setminus W} \) is the value that maximizes the log-likelihood function as follows:

\[
\tilde{n}_{r \setminus W}^M = \arg \max_{n_{r \setminus W}} \ln L(\mathbf{z}; n_{r \setminus W}, n_W) = \arg \max_{n_{r \setminus W}} \left\{ -\frac{M}{2} \ln \left( \sigma_z^2 \right) - \sum_{m=1}^{M} \frac{(z^m - \mu_z)^2}{2\sigma_z^2} \right\}. \tag{6.6}
\]

In practice, \( n_W \) is not given and an estimation should be used instead. Let \( \tilde{n}_W^M \) denote the estimation of \( n_W \) using \( M \) interrogation vectors. Since \( \ln L(\mathbf{z}; n_{r \setminus W}, n_W) \) is convex, by taking its derivative and equating to zero, we can find the closed form expression for the
ML estimator. For large values of $M$ (e.g., $M > 10$) and $f$ (e.g., $f > 100$), the exclusive estimator in (6.6) can be approximated as

$$\tilde{n}_W \approx \left\{ n_W \left| \mu_z - \frac{1}{M} \sum_{m=1}^{M} z^m = 0 \right. \right\} = -\frac{f}{p} \ln \left( 1 - \frac{\sum_{m=1}^{M} z^m}{M f} \exp \left( \frac{p \tilde{n}_W}{f} \right) \right).$$

(6.7)

Eq. (6.7) is the asynchronous exclusive estimator. In the next subsection, we explain how $n_W$ can be estimated. The inaccuracy in estimation of $n_{r\setminus W}$ comes from the random nature of samples taken from the system and also the error in estimation of $n_W$. The following theorem characterizes the error of the estimator in (6.7).

**Theorem 6** Given the error in estimation of $n_W$ follows a normal distribution, for large values of $M$ and $f$, the error in estimation of $n_{r\setminus W}$ using (6.7) has a normal distribution. Moreover, this estimator is asymptotically unbiased if the mean of error in estimation of $n_W$ approaches zero for large values of $M$. The variance of error is bounded by

$$\sigma^2_{n_{r\setminus W}}(M) \leq \frac{1}{p^2} \exp \left( \frac{2p (n_{r\setminus W} + n_W)}{f} \right) \frac{\sigma^2_z}{M} \left( \exp \left( \frac{pm_{r\setminus W}}{f} \right) - 1 \right)^2 \sigma^2_{n_W}(M),$$

(6.8)

where $\sigma^2_{n_{r\setminus W}}(M)$ and $\sigma^2_{n_W}(M)$ are the variance of errors in estimation of $n_{r\setminus W}$ and $n_W$ using $M$ independent set of interrogation vectors, respectively.

The proof of Theorem 6 is given in Section 6.6.2. We will compare $\sigma^2_{n_{r\setminus W}}(M)$ and its upper bound in Section 6.4.

### 6.2.2 A-MRCE Algorithm

We now present the A-MRCE algorithm to estimate the total number of tags in an RFID system with multiple readers. The A-MRCE algorithm is a centralized algorithm. The controller is a centralized unit which is responsible to estimate the total number...
Chapter 6. Cardinality Estimation in RFID Systems with Multiple Readers

Algorithm 6.1 A-MRCE Algorithm executed at reader $r \in \mathcal{R}$.

1: Upon receiving parameters $p$, $q$, and $f$ from the controller, reader $r$ performs $M$ interrogation processes using seed values $q$.
2: Reader $r$ sends its location info and the interrogation vectors $v^1_r, \ldots, v^M_r$ to the controller.

Algorithm 6.2 A-MRCE Algorithm executed at the controller.

1: Input: Set of readers $\mathcal{R}$, neighboring set $\mathcal{H}_r$ of reader $r$, and the interrogation vectors of reader $r$, $\forall r \in \mathcal{R}$.
2: Initialization: Set $\tilde{N} := 0$, and $\Gamma := \{}$
3: while $\Gamma \neq \mathcal{R}$
4: Select a reader $r$ randomly from the set $\mathcal{R}\setminus\Gamma$.
5: Set $W := \Gamma \cap \mathcal{H}_r$.
6: if $W = \{}$
7: Calculate $\bar{n}_r$ using (6.2).
8: Set $\tilde{N} := \tilde{N} + \bar{n}_r$.
9: else
10: Calculate $\bar{n}_W$ using (6.2) if $|W| = 1$ or using A-MRCE algorithm if $|W| > 1$.
11: Calculate $\bar{n}_{r\setminus W}$ using asynchronous exclusive estimator (6.7).
12: Set $\tilde{N} := \tilde{N} + \bar{n}_r\setminus W$.
13: end if
14: Set $\Gamma := \Gamma \cup \{r\}$.
15: end while

of tags in the system. Each reader performs interrogations individually and transmits its interrogation vectors to the controller. The controller uses these vectors to estimate the number of tags. The A-MRCE algorithm, which is shown in Algorithms 6.1 and 6.2, implements equation (6.1) using the single reader and the asynchronous exclusive estimators. Algorithm 6.1 shows part of the A-MRCE algorithm performed by a reader $r \in \mathcal{R}$. Algorithm 6.2 shows part of the A-MRCE algorithm performed by the controller. When the algorithm is invoked, the controller informs the persistence probability $p$, frame size $f$, and a set of $M$ random seeds $q$ (i.e., $|q| = M$) to all the readers. Each reader $r \in \mathcal{R}$ then performs $M$ interrogation processes and sends the interrogation vectors $(v^1_r, \ldots, v^M_r)$ to the controller. The readers also inform the controller about their location by sending the location information to the controller. $\mathcal{H}_r$ is the set of readers whose interrogation
vectors overlap with the interrogation zone of reader $r$. Based on the interrogation range of each reader and its location, the set $\mathcal{H}_r$ of neighboring readers of reader $r$ can also be determined. The readers may encounter reader-to-tag and reader-to-reader collision during their interrogations. To avoid the collision, several techniques have been proposed in the literature [131, 132, 133, 134, 135] for RFID systems with multiple readers. However, it is not the focus of this chapter and we assume that the readers employ one of these techniques to perform interrogations.

After receiving the required information (interrogation vectors and location information) from all the readers, the controller estimates the total number of tags using asynchronous exclusive estimator by invoking Algorithm 6.2. Algorithm 6.2 is equivalent to applying equation (6.1) iteratively to estimate the total number of tags. Steps 4 – 14 denote one iteration of the algorithm. At each iteration, the algorithm selects a reader $r$ randomly from the readers which have not been selected yet (i.e., $\mathcal{R}\setminus\mathcal{I}$). Then, the controller calculates the number of tags within the range of the selected reader which have not already been counted and adds it to $\tilde{N}$. The set $\mathcal{W}$ denotes the set of neighbors of reader $r$ which have been selected by the algorithm in the previous iterations (i.e., the tags within the range of the readers in the set $\mathcal{W}$ have been counted till now). If this set is empty, the algorithm uses the single reader estimator in (6.2) to calculate $\tilde{n}_r$ and adds this number to the current estimation of the total number of tags $\tilde{N}$ (Steps 6 – 9). If the set $\mathcal{W}$ is non-empty, then the algorithm uses the asynchronous exclusive estimator to calculate $\tilde{n}_{r\setminus\mathcal{W}}$. To do so, the asynchronous exclusive estimator requires the value of $\tilde{n}_\mathcal{W}$. If the set $\mathcal{W}$ has only one member, then $\tilde{n}_\mathcal{W}$ can be calculated using single reader estimator in (6.2). Otherwise, the controller invokes the A-MRCE algorithm (Algorithm 6.2) recursively to estimate the number of tags within the range of readers in the set $\mathcal{W}$. The recursion ends at maximum after $|\mathcal{W}|$ iteration. This shows a recursive operation of
the A-MRCE algorithm. We notice that since the controller already has the interrogation vectors of all the readers including those in set \( W \), it does not need to ask the readers to perform interrogations again.

### 6.2.3 Estimation Error and Discussion

To estimate \( \tilde{n}_{r\setminus W} \), the asynchronous exclusive estimator uses the estimation of \( n_W \). Therefore, the A-MRCE algorithm needs to calculate \( \tilde{n}_W \). If the set \( W \) has one element, then the single reader estimator can be used and the estimation error in estimating \( n_W \) has a normal distribution with zero mean. If the set \( W \) has more than one reader, the algorithm is invoked again to estimate \( n_W \). This gives a recursive operation of the A-MRCE algorithm. In the recursive procedure, the first level of the estimation is obtained by using the asynchronous exclusive estimator which has an error with normal distribution and zero mean. Based on Theorem 6, the next levels have also an estimation error with normal distribution and zero mean. Consequently, the estimation of \( n_W \) has a normal distribution with zero mean.

We now describe how to determine the estimation error for the A-MRCE algorithm. The value of \( \tilde{N} \) obtained by the A-MRCE algorithm is composed of several estimations from the single reader and the asynchronous exclusive estimators. All the readers contribute to the estimation of \( \tilde{N} \). Since different terms for \( \tilde{N} \) are from either single reader or asynchronous exclusive estimators, they have asymptotically normal distributions with zero mean. Therefore, \( \tilde{N} \) has asymptotically normal distribution with zero mean. In general, different terms of \( \tilde{N} \) are not independent. However, the summation of the variance of error of these terms can give an upper bound for the variance of the error for A-MRCE algorithm. The summation of the variances depends on the order that readers are selected in the algorithm (Step 4 of Algorithm 6.2).
For pure random selection of the readers (Step 4 of Algorithm 6.2), we calculate the expected value of the summation of variances over different runs of the algorithm. For reader \( r \in \mathcal{R} \) with \( |\mathcal{H}_r| \) neighbors, let \( \mathcal{Q}_r \) denote the power set of \( \mathcal{H}_r \). The power set of a set is the set of all subsets of that set. In different runs of the A-MRCE algorithm, the number of tags within the zone of reader \( r \) may appear in various forms in \( \tilde{N} \). For example, it can be either \( \tilde{N} := \tilde{N} + \tilde{n}_r \) or \( \tilde{N} := \tilde{N} + \tilde{n}_{r \setminus W} \) for any \( W \in \mathcal{Q}_r \). For set \( W \in \mathcal{Q}_r \), the probability that \( \tilde{N} \) contains \( \tilde{n}_{r \setminus W} \) is equal to the probability that only readers in set \( W \) from the neighbors of \( r \) are selected before reader \( r \) within the algorithm run. Assuming that the reader selection is purely random, this probability is equal to the probability that among reader \( r \) and its neighbors (i.e., \( \mathcal{H}_r \)), readers in set \( W \) are selected before reader \( r \). The probability that any subset of neighbors of \( r \) with cardinality \( |W| \) is selected by the algorithm before reader \( r \) is \( 1/(|\mathcal{H}_r| + 1) \) and the number of such subsets is \( \binom{|\mathcal{H}_r|}{|W|} \). Let \( P_{n_{r \setminus W}} \) denote the probability that \( n_{r \setminus W} \) appears in \( \tilde{N} \). This probability can be written as

\[
P_{n_{r \setminus W}} = \frac{1}{(|\mathcal{H}_r| + 1) \binom{|\mathcal{H}_r|}{|W|}}.
\]

(6.9)

Therefore, when the reader \( r \) is selected within the algorithm run, with probability \( P_{n_{r \setminus W}} \) we have \( W = \Gamma \cap \mathcal{H}_r, \forall W \in \mathcal{Q}_r \). The expected value for the summation of variances of errors in \( \tilde{N} \) gives an upper bound on the variance of estimation error \( \sigma_{\text{asyn}}^2 \):

\[
\sigma_{\text{asyn}}^2 \leq \sum_{r \in \mathcal{R}} \sum_{W \in \mathcal{Q}_r} P_{n_{r \setminus W}} \sigma_{n_{r \setminus W}}^2.
\]

(6.10)
6.3 Synchronous Multiple Reader Cardinality Estimation (S-MRCE)

We proposed the A-MRCE algorithm in the previous section based on the assumption that readers cannot be synchronized for interrogations. In this section, we consider the case that readers have the ability to be synchronized for interrogations (e.g., they are equipped with accurate clocks). We develop a synchronous multiple reader cardinality estimation (S-MRCE) algorithm, which is suitable for RFID systems with multiple synchronized readers. The readers operate synchronously in a sense that they start interrogation at certain times and perform interrogations periodically one after another. We first propose a synchronous exclusive estimator, which is a building block of the S-MRCE algorithm.

6.3.1 Synchronous Exclusive Estimator

The synchronous exclusive estimator is developed to estimate $n_{r\setminus \mathcal{W}}$ for $\mathcal{W} \subseteq \mathcal{H}_r$ using interrogation vectors obtained from synchronous operation of the readers. The main idea behind the operation of the synchronous exclusive estimator is the use of different seed values for different tags in one interrogation. In fact, the reader performs interrogation while the shared tags (i.e., the tags in $\mathcal{T}_r \cap \{ \bigcup_{w \in \mathcal{W}} \mathcal{T}_w \}$) and the tags which are exclusively located within the range of the reader have different seed values. Consider reader $r$, the set of readers $\mathcal{W} \subseteq \mathcal{H}_r$, and the vector $\mathbf{v}_r$ obtained from the interrogation of tags within the range of $r$ using seed value $q$. Also, consider vector $\mathbf{u}_r$ obtained from the interrogation of tags in $\mathcal{T}_r$ while the tags in $\mathcal{T}_r \setminus \{ \bigcup_{w \in \mathcal{W}} \mathcal{T}_w \}$ use seed value $q$ and the tags in $\mathcal{T}_r \cap \{ \bigcup_{w \in \mathcal{W}} \mathcal{T}_w \}$ use seed value $q'$. Since the tags select the same slot whenever they are interrogated with the same seed value, vectors $\mathbf{v}_r$ and $\mathbf{u}_r$ have common information about $n_{r\setminus \mathcal{W}}$. The difference between these two vectors comes from the shared tags, which have been interrogated with
different seed values. The number of slots with at least one transmission in either \( v_r \) or \( u_r \) represents the existence of a tag in set \( T_r \) while the tags in \( T_r \cap \{ \cup_{w \in W} T_w \} \) are counted twice. Let \( Y \) denote the number of time slots which are nonempty either in \( v_r \) or \( u_r \). That is, given reader \( r \in R \) and set \( W \), we have \( Y = \sum_{l=1}^{f} (v^l_r + u^l_r - v^l_r \cdot u^l_r) \). The samples of random variable \( Y \) can be used to estimate the value of \( n_{r \setminus W} \). The following theorem characterizes the distribution of variable \( Y \).

**Theorem 7** The random variable \( Y \) has a normal distribution with mean \( \mu_y \) and variance \( \sigma_y^2 \) for large values of \( f, n_r, \) and \( n_{r \setminus W} \) as

\[
\mu_y = f \left( 1 - \exp\left(-p\nu/f\right) \right), \quad (6.11)
\]

\[
\sigma_y^2 = f \exp\left(-p\nu/f\right) \left( 1 - (1 + p^2 \nu) \exp\left(-p\nu/f\right) \right), \quad (6.12)
\]

where \( \nu = (2n_r - n_{r \setminus W}) \).

The proof is similar to the proof given in [136] for the occupancy problem. Using \( M \) different pairs of seed values for the tags in \( T_r \setminus \{ \cup_{w \in W} T_w \} \) and \( T_r \cap \{ \cup_{w \in W} T_w \} \), \( M \) different samples of random variable \( Y \) can be obtained, namely \( y_1, \ldots, y_M \). Let \( y \) denote the vector of all samples (i.e., \( y = (y_1, \ldots, y_M) \)). The likelihood function of \( n_{r \setminus W} \) using \( y \) is similar to that obtained in \((6.5)\) if the mean and variance of variable \( Z \) is replaced by variable \( Y \). The ML estimate of \( n_{r \setminus W} \) with \( M \) different samples can be obtained by taking derivative from the likelihood function and equating it to zero similar to the approach used for \((6.6)\). For large values of \( M \) (e.g., \( M > 10 \)) and \( f \) (e.g., \( f > 100 \)), the synchronous exclusive
estimator can be approximated as

\[
\hat{n}^{M}_{r \setminus W} \approx \left\{ n_{r \setminus W} \mid \mu_y - \frac{1}{M} \sum_{m=1}^{M} y^m = 0 \right\} = 2n_r + \frac{f}{p} \ln \left( 1 - \frac{\sum_{m=1}^{M} y^m}{Mf} \right),
\]

(6.13)

where \(\hat{n}^{M}_{r \setminus W}\) denotes the estimation of \(n_{r \setminus W}\) using the synchronous exclusive estimator with \(M\) samples. When using the synchronous exclusive estimator, we use an estimation of \(n_r\). This estimation is obtained using the single reader estimator which is asymptotically unbiased. For large values of \(M\), \(\sum_{m} y^m / M\) converges to \(\mu_y\) and \(\ln (1 - \sum_{m} y^m / (Mf))\) converges to \(-p/f \nu\). Since the error in the estimation of \(n_r\) approaches zero, the synchronous exclusive estimator is asymptotically unbiased. Since it is an ML estimator, the error distribution is asymptotically normal \([137]\). The error in the estimation of \(n_r\) is also asymptotically normal and is added to the error generated from the random samples taken from the system. Assume that there is no error in estimation of \(n_r\). In this case, the error in \(\hat{n}^{M}_{r \setminus W}\) originates from the second term in (6.13). Since the synchronous exclusive estimator is an ML estimator, the variance of error can be approximated using the Cramer-Rao bound \([137]\). The Fisher information for this exclusive estimator can be written as

\[
\mathcal{I}_{\text{syn}}(n_{r \setminus W}) = E_{y|n_{r \setminus W}} \left\{ \left( \frac{\partial}{\partial n_{r \setminus W}} \ln L(y; n_{r \setminus W}) \right)^2 \right\}
= E_{y|n_{r \setminus W}} \left\{ \left( \frac{\partial}{\partial n_{r \setminus W}} \ln \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi} \sigma_y} \exp \left( -\frac{(y^m - \mu_y)^2}{2\sigma_y^2} \right) \right)^2 \right\}.
\]

(6.14)

Equation (6.14) can be approximated as

\[
\mathcal{I}_{\text{syn}}(n_{r \setminus W}) \approx \frac{(\mu'_y)^2}{\sigma_y^2} = \frac{p^2}{f \left( 1 - (1 + p^2 \nu) \exp \left( -p \nu / f \right) \right)},
\]

(6.15)
where $\mu'_y$ denotes the derivative of $\mu_y$ with respect to $n_{r\setminus W}$. Now, we calculate the variance of estimation error. The variance of estimation error for $n_{r\setminus W}$ is $E[(\tilde{n}_{r\setminus W} - n_{r\setminus W})^2]$. Replacing $\tilde{n}_{r\setminus W}$ from (6.13), we have

$$\sigma^2_{n_{r\setminus W}}(M) = E \left[ (\tilde{n}_{r\setminus W} - n_{r\setminus W})^2 \right]$$

$$= E \left[ \left( 2(\tilde{n}_r - n_r) + \left( \frac{f}{p} \ln \left( 1 - \frac{\sum_{m=1}^{M} y^m}{M f} \right) - \nu \right) \right)^2 \right].$$  \hspace{1cm} (6.16)

If we replace $\tilde{n}_r$ from (6.12), we obtain

$$\sigma^2_{n_{r\setminus W}}(M) = E \left[ \left( 2 \left( -\frac{f}{p} \ln \left( \sum_{m=1}^{M} t^m/Mf \right) - n_r \right) + \left( \frac{f}{p} \ln \left( 1 - \frac{\sum_{m=1}^{M} y^m}{M f} \right) + \nu \right) \right)^2 \right].$$

Let $\gamma_1 = \left( -\frac{f}{p} \ln \left( \sum_{m=1}^{M} t^m/Mf \right) - n_r \right)$ and $\gamma_2 = \left( \frac{f}{p} \ln \left( 1 - \frac{\sum_{m=1}^{M} y^m}{M f} \right) + \nu \right)$. Then, we have $\sigma^2_{n_{r\setminus W}}(M) = 4E[\gamma_1^2] + E[\gamma_2^2] + 2E[\gamma_1 \gamma_2]$. $t^m$ denotes the number of empty slots in the interrogation vector using $m$th seed value while $y^m$ denotes the number of non-empty slots. If the average of $t^m$ for various seed values is less than $\mu_{n_r}$, then the average of $y^m$ is above $\mu_{n_r}$. Therefore, $E[\gamma_1 \gamma_2]$ is always non-positive and we have $\sigma^2_{n_{r\setminus W}}(M) \leq 4E[\gamma_1^2] + E[\gamma_2^2]$. Replacing $E[\gamma_1^2]$ by $\sigma^2_{n_r}$ and $E[\gamma_2^2]$ by $(MI_{\text{syn}}(n_{r\setminus W}))^{-1}$, we obtain

$$\sigma^2_{n_{r\setminus W}}(M) \leq (MI_{\text{syn}}(n_{r\setminus W}))^{-1} + 4\sigma^2_{n_r}. \hspace{1cm} (6.17)$$

### 6.3.2 S-MRCE Algorithm

We now present the S-MRCE algorithm to estimate the total number of tags in a system covered by multiple readers. On contrary to the A-MRCE algorithm, the S-MRCE algorithm is a distributed approach where readers perform interrogation and estimate the number of tags individually. Then, they transmit their estimation to the controller. The
controller just sums up those numbers to obtain the estimation of the total number of tags. The computation load of estimation is at the reader’s side. The S-MRCE algorithm implements (6.1) using the synchronous exclusive estimator. Algorithm 6.3 shows the S-MRCE algorithm. In this algorithm, all the readers are assumed to be synchronized with a central clock.

We divide the system time into several time periods and each reader is assigned a period number. Each reader performs an interrogation within the assigned period. To prevent collision between neighboring readers, we use a scheduling algorithm based on graph coloring [138]. We assign different colors to various readers such that two neighboring readers are not assigned the same color. The chromatic number of the system is the minimum number of colors needed to color all the readers. Let $C$ denote the chromatic number of our system. The colors are interpreted as different time periods of the system. This provides a time division multiple access method for interrogation of all the readers. There are $C$ interrogation periods assigned to the readers. Let $s_r$ denote the period number which is assigned to the reader $r$. Each period is long enough for an interrogation process. At the beginning, the controller informs the persistence probability $p$, frame size $f$, the

---

**Algorithm 6.3** S-MRCE Algorithm to calculate the total number of tags in the system.

1: The controller sends the parameters $p$, $f$, vector $\mathbf{q}$, and random slot numbers $\{s_1, \ldots, s_{|\mathcal{R}|}\}$ to all readers.
2: \textbf{for} $k = 0, \ldots, |\mathcal{R}| - 1$
3: \hspace{1em} \textbf{if} $s_r = k$,
4: \hspace{2em} Reader $r$ starts periodic interrogation process using seed vector $\mathbf{q}$.
5: \hspace{1em} \textbf{end if}
6: \textbf{end for}
7: \textbf{for} $r \in \mathcal{R}$
8: \hspace{1em} Reader $r$ calculates $\tilde{n}_r$ using (6.2) and calculates $\tilde{n}_{r\setminus W}$ using synchronous exclusive estimator (6.13).
9: \hspace{1em} Reader $r$ sends the estimated value $\tilde{n}_{r\setminus W}$ to the controller.
10: \textbf{end for}
11: The controller sums up the received value from all readers to find the estimation of $N$. 
random seed vector \( q = (q_1, \ldots, q_M) \), where \( q_i \neq q_j, \forall \ i, j \), and the slot number to all the readers (Step 1). We call the \( C \) slots together an interrogation round. Within an interrogation round, all the readers perform the interrogation once. To estimate the number of tags, all the readers perform interrogation twice in two consecutive rounds. In the first round, all the readers announce their seed value and then perform interrogation. In the second round, however, the readers just perform the interrogation and do not announce the seed value. For the reader \( r \), the first interrogation results in measuring interrogation vector \( v_r \) while all the tags within range of reader \( r \) are interrogated with the same seed value. In the second interrogation, the reader \( r \) measures vector \( u_r \) while the tags in the range of reader \( r \) which are in the range of other readers have different seed value compared to those exclusively in the range of reader \( r \). To make sure that neighboring readers are not using the same seed value, reader with slot number \( s_r \) used the \( s_r \)-th seed value. We assume that the chromatic number \( C \) is less that the number of samples \( M \). To obtain \( M \) samples, the interrogation rounds are repeated \( M \) times with different seed values. We call this synchronous process performed by every reader the *periodic interrogation process* (Step 4).

After performing the interrogations by all the readers, each reader \( r \) uses the vectors obtained by announcing the seed value to estimate \( n_r \). Reader \( r \) calculates the estimated value of \( n_{r \setminus W} \) using the estimation of \( n_r \) and also two sets of interrogation vectors \( v_r \) and \( u_r \) (Step 8). Reader \( r \) sends the estimation of \( n_{r \setminus W} \) to the controller (Step 9). The controller can calculate the estimation of the total number of tags by adding up all values received from the readers (Step 11).

Since the error of synchronous exclusive estimator has normal distribution with zero mean, the estimation of \( N \) obtained by using the S-MRCE algorithm has normal error distribution and is asymptotically unbiased. The upper bound for the variance of estimation
error for the S-MRCE algorithm can be calculated using a similar approach employed to calculate the variance of the A-MRCE algorithm. The upper bound can be written as

$$\sigma_{\text{syn}}^2 \leq \sum_{r \in \mathcal{R}} \sum_{W \in \mathcal{Q}_r} P_{n_r \setminus W} \sigma_{n_r \setminus W}^2,$$

(6.18)

where $\sigma_{n_r \setminus W}^2$ is derived in (6.17). Compared to the A-MRCE algorithm, the S-MRCE algorithm provides better performance in terms of the estimation of error. The probability of error of asynchronous exclusive estimator increases exponentially with $n_W$. When the number of neighboring readers increases, the probability of error for asynchronous exclusive estimator increases rapidly. However, the error of synchronous exclusive estimator depends on the number of tags within the zone of the reader, but not the neighboring readers. The cost which is paid to achieve this better performance is the need of the synchronous operation of readers.

The estimation error of A-MRCE and S-MRCE algorithms depends on the choice of design parameters $f$ and $p$. To choose these parameters appropriately, the controller requires to know the number of tags in the system. In case that the controller does not have a priori information about the number of tags in the system, the controller can choose
predetermined values for \( f \) and \( p \) and perform a round of interrogations. Then, based on the estimation of the number of tags, the controller chooses \( f \) and \( p \) for the next round of interrogation.

6.4 Performance Evaluation

We used MATLAB and developed a discrete-event RFID simulator to validate the analytical models and to evaluate the performance of the exclusive estimators and the MRCE algorithms.

6.4.1 Performance Evaluation for Exclusive Estimators

In this section, we investigate the performance of the exclusive estimators, validate the models, and compare them in terms of the mean and variance of the estimation error. We first consider the topology given in Fig. 6.2 (a). The interrogation zone of each reader is 15 m. The tags are randomly deployed within the zone of readers. The number of tags within

---

Figure 6.3: Mean of estimation error for different number of interrogations \( M \), (a) asynchronous exclusive estimator, and (b) synchronous exclusive estimator.
the zone of all readers is equal. The number of tags shared between any two of readers is 15\% of \( n_r \). Also, 5\% of the tags are shared among three of them. We use asynchronous and synchronous exclusive estimators to estimate \( n_{r_3|W} \) where \( W = \{r_1, r_2\} \). The estimation of \( n_W \), is obtained as \( \hat{n}_W = \hat{n}_{r_1} + \hat{n}_{r_2|r_1} \), where \( \hat{n}_{r_1} \) and \( \hat{n}_{r_2|r_1} \) are obtained by using (6.2) and (6.3), respectively. We notice that \( n_{r|W} = 0.75n_r \).

First, we investigate the performance of the models by varying the number of interrogations \( M \) from 1 to 50. We set the frame size \( f \) to 500 and the persistence probability \( p \) to 1. We measure the mean and variance of the estimation error for both asynchronous and synchronous exclusive estimators. These errors contain the errors in estimation of \( \hat{n}_W \) and \( \hat{n}_3 \) as well. Figs. 6.3 (a) and (b) show that both estimators are asymptotically unbiased and the mean of the error approaches zero rapidly when \( M \) increases in the system. Figs. 6.4 and 6.5 show the standard deviation of error obtained from simulations and compare these values with analytical upper bounds obtained in equations (6.8) and (6.17). For both estimators, the variance of error approaches zero for large values of \( M \). As shown in Figs.
Figure 6.5: Standard deviation of estimation error for different number of interrogations \( M \) when using the \textit{synchronous} exclusive estimator.

In Figs. 6.4 and 6.5, the upper bounds are tight when the number of tags within the zone of readers is small.

Next, we investigate the behavior of these estimators for different number of tags within the range of readers. We use the notion of operational range for comparison. The operational range of the estimators is defined as the range of \( n_r \) such that the mean of estimation error is within a certain threshold of the actual value. We use \( \pm 1\% \) as the threshold. The lower bound of the range is always zero and the upper bound depends on frame size \( f \) and persistence probability \( p \). Lowering the persistence probability \( p \) can decrease the effective number of tags transmitting in an interrogation process and it can increase the operational range. The analytical models for the estimators are valid as long as \( n_r \) is within the operational range. We vary the number of tags within the zone of each reader from 100 to 10,000 by steps of 50. Fig. 6.6 shows the mean of estimated value for asynchronous and synchronous exclusive estimators for two values of persistence probability, \( p = 1 \) and \( p = 0.5 \). The mean of estimation for the asynchronous exclusive
Figure 6.6: The mean of estimation error for varying number of tags, asynchronous and synchronous exclusive estimators.

The operational range of the exclusive estimators is a function of the persistence probability. Fig. 6.7 shows the operational range of the asynchronous and synchronous exclusive estimators for various values of $p$. The operational range is extended when the probability is decreased. Although decreasing $p$ can increase the operational range, it may increase the estimation error under some circumstances, especially for the systems with a low number of tags. Figs. 6.8 (a) and (b) show the behavior of the standard deviation of the error versus the persistence probability for the asynchronous and synchronous exclusive estimators, respectively. We notice that the values of $n_r$ equal to 1000 and 2000 are not in the operational range of the asynchronous exclusive estimator. Therefore, Fig. 6.8 (a) does not include the curves for 1000 and 2000 nodes. Both estimators have similar behavior.
Figure 6.7: Operational range of asynchronous and synchronous exclusive estimators for varying persistence probabilities.

In terms of variance of error. For the systems with a small number of tags compared to the frame size, the variance of the estimation error increases when $p$ decreases. Therefore, $p = 1$ is a suitable choice in terms estimation error. On the other hand, for large values of $n_r$ compared to the frame size, decreasing the persistence probability can improve the error to some extent. However, for very small persistence probabilities, the error start increasing again. There is a trade off between the operational range and the variance of error in choosing the design parameter $p$.

### 6.4.2 Performance Evaluation for A-MRCE and S-MRCE Algorithms

In this section, we first investigate the performance of the A-MRCE and S-MRCE algorithms for a system with ten readers shown in Fig. 6.2 (b). The interrogation zone of each reader has a range equal to 15 m. The tags are randomly deployed within the interrogation
Chapter 6. Cardinality Estimation in RFID Systems with Multiple Readers

Figure 6.8: Standard deviation of error for varying persistence probabilities: (a) asynchronous exclusive estimator, (b) synchronous exclusive estimator.

zone of the readers. We set the frame size $f$ to be 500 time slots, the persistence probability $p$ to 1, and $M$ to 50. Then, we increase the total number of tags in the system from 500 to 5000 with steps of 500. These numbers are chosen such that the single reader and exclusive estimators work in their operational range (i.e., the mean of estimation error is zero). Fig. 6.4 shows the standard deviation of error for the A-MRCE and S-MRCE algorithms and compares the analytical results obtained in (6.10) and (6.18) with the simulation results. The simulation results are averaged over 1000 iterations. As we expect, the analytical values provide upper bounds on the simulation results for both algorithms. In the worst case, the analytical results are 40% and 30% higher than the simulation results. As Fig. 6.4 shows, the standard deviation of error is small compared to the actual number of tags in the system, which proves the accuracy of the algorithms. Moreover, it can be seen that the S-MRCE algorithm outperforms the A-MRCE algorithm substantially in terms of estimation error. However, we notice that S-MRCE is not suitable for cases when readers cannot perform synchronously in the system.

Next, we compare the A-MRCE and S-MRCE algorithms with two other algorithms:
Chapter 6. Cardinality Estimation in RFID Systems with Multiple Readers

Figure 6.9: Comparison of A-MRCE and S-MRCE in terms of standard deviation of error.

the lottery frame (LoF)\cite{47} and EZB\cite{50} algorithms. Both EZB and LoF algorithms can also be extended to estimate the number of tags for systems with multiple readers. To achieve that, the interrogation vectors of all the readers in the system are merged using slot-wise and operator. The resulting vector is similar to a vector obtained by a single reader interrogating all the tags in the system. Therefore, this vector can be used as an input for the EZB or LoF algorithm to estimate the total number of tags in the system. We compare these algorithms for a system with three readers and ten readers separately.

First, we compare the mean and variance of estimation error of A-MRCE, S-MRCE, LoF, and EZB algorithms for various interrogation times for an RFID system with three readers shown in Fig 6.2 (a). The number of tags within the range of each reader is 750. Figs. 6.10 (a) and (b) show the mean and standard deviation of error in estimating the total number of tags versus the interrogation time, respectively. The interrogation time is defined as the number of time slots that each reader requires to perform interrogation. The frame size for A-MRCE, S-MRCE, and EZB algorithms is set to 500 while it is set to
Figure 6.10: Comparing different algorithms in terms of interrogation time (number of time slots), (a) mean of estimation error, and (b) standard deviation of estimation error.

We vary the number of interrogations for various algorithms (change $M$ and the number of hash functions) and determine the mean and variance of estimation error. Fig. 6.10 (a) shows that LoF algorithm has a lower mean of estimation error compared to other algorithms for the same interrogation times. Then, LoF has a wider operational range compared to other schemes. However, as Fig. 6.10 (b) shows, the variance of estimation error is higher in LoF compared to other algorithms.

Next, we consider the system in Fig. 6.2 (b). The number of tags within the zone of different readers is $n_r = 500$. The frame size $f$ is set to 500 time slots for A-MRCE, S-MRCE, and EZB algorithms while the frame size of the LoF is set to 16. The number of interrogations $M$ is set to 20 for A-MRCE and EZB algorithms and it is set to 10 for S-MRCE algorithm. We use 600 various hash functions for the LoF algorithm. We notice that under this setting, the readers in all the algorithms have the same interrogation time. We also mention that the mean of estimation error is close to zero for all the estimators under this setting. At the beginning, we only consider reader 1 for the simulations. Then, we add the readers one by one to the system and investigate the performance of the algorithms.
Figure 6.11: Comparing the standard deviation of estimation error for A-MRCE, S-MRCE, EZB [50], and LoF [47] algorithms.

in the presence of different number of readers. Fig. 6.11 shows the standard deviation of the estimation error for the total number of tags for various number of readers. It can be seen that the standard deviation of error in the A-MRCE and S-MRCE algorithms seems to grow linearly with the number of readers while it seems to increase exponentially in LoF and EZB algorithms. In the presence of multiple readers, EZB and LoF algorithms merge the interrogation vectors of several readers to obtain one interrogation vector. For a fixed number of tags within the range of the readers, increasing the number of readers would linearly increase the total number of tags in the system. Since both algorithms use the merged interrogation vector for estimation and the variance of the estimation error for either EZB and LoF algorithms increases exponentially with the number of tags, the variance of EZB and LoF algorithms increases exponentially as the number of readers linearly increases. However, for the A-MRCE and S-MRCE algorithms, whenever a reader is added to the system, only the error of the exclusive estimator which is used to estimate the tags within range of that reader is added to the total error. Therefore, the increase in
the error is linear as the number of readers increases linearly.

6.5 Summary

In this chapter, we studied the problem of anonymous cardinality estimation in RFID systems with multiple readers. We proposed two ML estimators, namely an asynchronous exclusive estimator and a synchronous exclusive estimator, to estimate the number of tags which are exclusively located within the zone of a reader. We showed that these estimators are asymptotically unbiased and we derived upper bounds on the variance of estimation error. We proposed the A-MRCE and S-MRCE algorithms which can accurately estimate the tag population anonymously using the query replies of different readers and exclusive estimators. We derived the probability density function of the estimation error and showed that it can be approximated as a normal distribution with zero mean. The accuracy of the model and the approximations are validated via simulations. For performance comparisons, results showed that the variance of estimation error for A-MRCE and S-MRCE algorithms increase linearly with the number of readers while it increases exponentially for EZB and LoF algorithms.

6.6 Analytical Proofs

6.6.1 Proof of Theorem 5

Let \( \phi_z \) and \( \psi_z \) denote the event that \( z \) predetermined slots are nonempty in \( v_r \) and empty in \( v_w, \forall w \in \mathcal{W} \), respectively. Let \( \theta_z \) denote the event that both events \( \phi_z \) and \( \psi_z \) occur.
We have \( P(\theta_z) = P(\phi_z \cap \psi_z) = P(\phi_z | \psi_z)P(\psi_z) \), where

\[
P(\phi_z | \psi_z) = \sum_{t=0}^{n_r \backslash W} P(\phi_z | t \text{ tags pick these } z \text{ slots}) \times P(t \text{ tags pick these } z \text{ slots} | \psi_z).
\]

The condition on observed \( \psi_z \) affects the upper bound of the summation. The probability that \( z \) time slots are non-empty if \( t \) tags choose them is as follows [139, p. 92]:

\[
P(\phi_z | t \text{ tags pick these } z \text{ slots}) = \sum_{k=0}^{z} (-1)^k \binom{z}{k} \left(1 - \frac{z}{f}\right)^t.
\]

We also have

\[
P(t \text{ tags pick these } z \text{ slots} | \psi_z) = \binom{n_r \backslash W}{t} \left(\frac{pz}{f}\right)^t \left(1 - \frac{pz}{f}\right)^{(n_r \backslash W-t)}.
\]

Hence, we have

\[
P(\phi_z) = \sum_{t=0}^{n_r \backslash W} \sum_{k=0}^{z} (-1)^k \binom{z}{k} \left(1 - \frac{k}{f}\right)^t \binom{n_r \backslash W}{t} \left(\frac{pz}{f}\right)^t \left(1 - \frac{pz}{f}\right)^{(n_r \backslash W-t)}
\]

\[
= \sum_{k=0}^{z} (-1)^k \binom{z}{k} \left(1 - \frac{pk}{f}\right)^{n_r \backslash W}.
\]

Therefore, we can write \( P(\theta_z) \) as

\[
P(\theta_z) = \left(1 - \frac{pz}{f}\right)^{n_w} \sum_{k=0}^{z} (-1)^k \binom{z}{k} \left(1 - \frac{pk}{f}\right)^{n_r \backslash W}.
\]

Let \( S_z \) denote the summation of probability of all possible occurrences of \( z \) events in a frame. This is equal to \( S_z = \binom{f}{z} P(\theta_z) \). Let \( P_z \) denote the probability of having exactly \( z \) slots nonempty in \( v_r \) and empty in \( v_w, \forall w \in \mathcal{W} \). This probability can be calculated using
Equation (6.19) shows that this problem is similar to the occupancy problem investigated in [130]. It is shown in [130,136] that the distribution for the occupancy problem is asymptotically normal. Using a similar approach, we can show that $Z$ has a normal distribution for large values of $f$ and $n_r$. To calculate the mean and variance of $Z$, we use the approach presented in [136]. We define an auxiliary random variable $X_i$ which takes values from $\{0, 1\}$. $X_i$ is equal to one if the $i^{th}$ time slot is non-empty in $v_r$ and empty in $v_w$, $\forall w \in \mathcal{W}$. Therefore, we have $Z = \sum_{i=1}^{f} X_i$. The mean and variance of variable $Z$ can be obtained using this auxiliary variable as follows:

$$
\mu_z = E[Z] = E\left[\sum_{i=1}^{f} X_i\right] = f P_1
$$

$$
\approx f \left(1 - \exp\left(-\frac{pm_{r \setminus W}}{f}\right)\right) \exp\left(-\frac{pn_{W}}{f}\right).
$$

$$
\sigma^2_z = E\left[\left(\sum_{i=1}^{f} X_i\right)^2\right] - \mu^2_z
$$

$$
= \sum_{i=1}^{f} E\left[X_i^2\right] + 2E\left[\sum_{i=1}^{f} \sum_{j=i+1}^{f} X_i X_j\right]
$$

$$
= f P_1 + f(f - 1)P_2 - \mu^2_z
$$

$$
\approx \mu_z \left(1 - \frac{\mu_z}{f} \left(1 + \frac{p^2 n_{W}}{f}\right)\right) - p^2 n_{r \setminus W} \exp\left(-\frac{2p}{f} (n_{r \setminus W} + n_{W})\right).
$$
6.6.2 Proof of Theorem

We replace $\sum_{m} z_m^m$ and $\tilde{n}_W$ by $\mu_z + e_z^M$ and $n_W + e_n^M$, respectively while $e_z^M$ and $e_n^M$ are random variables. Since variable $Z$ has a normal distribution, variable $e_z^M$ is a normal random variable with zero mean and variance $\sigma_z^2 / M$. Moreover, it is assumed that $\tilde{n}_W$ is a normal random variable and the estimation is asymptotically unbiased. Therefore, $e_n^M$ is a normal random variable with zero mean. For large values of $f$ and $M$, the variance of $e_z^M$ and $e_n^M$ approaches zero and we have

$$
\tilde{n}_{r \backslash W} = -\frac{f}{p} \ln \left( 1 - \frac{\sum_{m} z_m^m}{M f} \exp \left( \frac{p \tilde{n}_W}{f} \right) \right)
\approx -\frac{f}{p} \ln \left( 1 - \frac{\mu_z + e_z^M}{f} \exp \left( \frac{p (n_W + e_n^M)}{f} \right) \right)
\approx -\frac{f}{p} \ln \left( 1 - \frac{\mu_z + e_z^M}{f} \exp \left( \frac{p n_W}{f} \right) \right)
\approx -\frac{f}{p} \ln \left( 1 - \frac{\mu_z}{f} \exp \left( \frac{p n_W}{f} \right) \right)
\approx n_{r \backslash W} - \frac{e_z^M}{p} \exp \left( \frac{p (n_W + n_{r \backslash W})}{f} \right) - e_n^M \left( \exp \left( \frac{p n_{r \backslash W}}{f} \right) - 1 \right).
$$

Since both variables $e_z^M$ and $e_n^M$ are normal random variables, variable $\tilde{n}_W$ is a normal random variable for large values of $M$. Moreover, the mean of error approaches zero for large values of $M$. Variables $e_z^M$ and $e_n^M$ are dependent in general since they are both derived from the same vectors $v_r$ and $v_w$. $e_n^M$ is the error in estimation of $n_W$. If $e_n^M$ is positive, it means that the average number of empty slots in vectors $v_w$ in different interrogations is greater than the expected value of the empty slots. Since $z$ is the number of slots which are non-empty in $v_r$ and empty in $v_w$, the average of $z_m^m$ for various interrogations is less
than \( \mu_z \) which means \( e_z^M \) is negative. Therefore, the errors \( e_w^M \) and \( e_r^M \) always cancel out each other. Consequently, the summation of variances for these two terms gives an upper bound for the variance of error as follows:

\[
\sigma^2_{n_r \backslash n_W} = \frac{1}{p^2} \exp \left( \frac{2p (n_{r \backslash n_W} + n_W)}{f} \right) \frac{\sigma^2_z}{M} + \left( \exp \left( \frac{pn_{r \backslash n_W}}{f} \right) - 1 \right)^2 \sigma^2_{n_W}.
\]

\[\blacksquare\]
Chapter 7

Conclusions and Future Work

In this chapter, we conclude the thesis by summarizing our results and highlighting the contributions of this dissertation. We also suggest several topics for further research.

7.1 Research Contributions

In this thesis, we proposed several power-efficient algorithms for WSNs using optimization theory and network coding in Chapters 2, 3, 4, and 5. We have also considered the problem of anonymous cardinality estimation in RFID systems with multiple readers in Chapter 6.

• In Chapter 2, we studied the problem of WSNs with multiple sinks and we formulated two algorithms to fairly share the network resources among various commodities and nodes in the system using the concept of lexicographical fairness. We first proposed the centralized LOCL algorithm to obtain the exact lexicographically optimal solution. LOCL is a stepwise algorithm. In each step, a linear mixed-integer programming problem is being solved. Our second algorithm, called the LONL algorithm, is a distributed algorithm. It can obtain the optimal solution under certain assumptions, and the sub-optimal solutions in general. The LONL algorithm is easier for implementation and more practical for deployments.

• In Chapter 3, we studied the tradeoff between network lifetime and system resources utilized to perform network coding for multicast traffic in WSNs. We proposed a
set of new variables called the coding flow variables. Using coding flow variables, we can determine the type of the operation and the rate at which the operation is performed in each sensor node. Then, we formulated the joint problem of constructing maximum-lifetime minimum-resource coding subgraph as a linear programming problem. The complexity of our algorithm increases linearly by the number of nodes and number of sessions, and it increases exponentially by the number of sinks. Hence, the algorithm is suitable for networks with a large number of sessions while each session has a small number of sinks.

• In Chapter 4, we studied the use of RLNC with buffer sharing in WSNs. Through analytical models, we showed when an intermediate node can stop transmission of the packets for a particular flow. Based on this model, we designed a link-by-link and an end-to-end feedback mechanism for RLNC with buffer sharing. In link-by-link feedback mechanism, an intermediate node transmits feedback control packet to its upstream neighboring node and acknowledges each flow individually. The upstream node reduces its transmission rate when it receives a feedback packet. In end-to-end feedback mechanism, the sink is responsible of transmitting the feedback. We investigated the power efficiency of both feedback mechanisms and compared the case with and without buffer sharing, separately.

• In Chapter 5, we studied the loss inference problem in wireless sensor networks while nodes performs randomized linear network coding. By using the subspace property of network coding, we proposed estimators to determine the path loss rate between a sink and a source. We showed that the intermediate nodes with multiple incoming links can be considered as virtual sources and we are able to determine the path loss rates from the virtual sources to the sink as well. We considered the assumption that there is no link with poor condition in the network in deriving the estimators. We
characterized the conditions for a link to be either identifiable, possibly identifiable, or not identifiable. We proposed a passive loss inference with random linear network coding (PLI-RLC) algorithm to determine the link loss rates.

• In Chapter 6, we studied the problem of anonymous cardinality estimation in RFID systems with multiple readers. The goal was to estimate the number of tags within range of multiple readers while tags do not reveal their IDs. We proposed an asynchronous exclusive estimator and a synchronous exclusive estimator, to estimate the number of tags which are exclusively located within the zone of a reader. We derived upper bounds on the variance of estimation error for these estimators. Using these estimators, we proposed the A-MRCE and S-MRCE algorithms which can accurately estimate the tag population anonymously. We showed that these algorithms result in asymptotically unbiased estimations. We derived the probability density function of the estimation error and showed that it can be approximated as a normal distribution with zero mean. The accuracy of the model and the approximations are validated via simulations.

7.2 Suggestions for Future Work

In the following, we consider several interesting possibilities for extension of the current work.

1. Lexicographically Optimal Routing for WSNs with Multiple Sinks. Both LOCL and LONL algorithms can be extended to enable the source nodes to select the appropriate sink. Alternatively, the problem can be formulated such that the optimal location for the sinks is determined while the network lifetime is being maximized.

2. Lifetime-Resource Tradeoff for Multicast Traffic in WSNs. One can extend
the model by considering the problem of determining the optimal location of coding
devices while the objective is to maximize the network lifetime. The lifetime-resource
tradeoff can also be studied for networks such that the multicast rate cannot be
achieved without using network coding.

3. **Feedback Mechanism for RLNC with Buffer Sharing in Lossy WSN.** One
can study the effect of limited buffers on operation of RLNC with buffer sharing.
Another direction is to consider the effect of lossy feedback on the performance of
RLNC with buffer sharing.

4. **Link Loss Inference in WSNs with RLNC.** For PLI-RLNC algorithm, the inter-
mediate network nodes keep separate buffers for different flows. The algorithm can
be extended if the intermediate nodes can keep shared buffers for separate flows. The
accuracy of the algorithm can be improved in this case since more information can
be revealed from the network. Another direction of problem extension is to consider
a carrier sense multiple access MAC protocol for the purpose of link loss inference.

5. **Cardinality Estimation in RFID Systems with Multiple Readers.** One can
study the problem of choosing the design parameters (e.g., frame size \(f\), persistence
probability \(p\)) based on the initial estimation of the number of tags and study the
tradeoff between the variance of estimation error and interrogation time.
Bibliography


Bibliography


