THE FAIR DATA COLLECTION PROBLEM
IN WIRELESS SENSOR NETWORKS

by

KA-LOK HUNG

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science

August 2006, Hong Kong

Copyright © by Ka-Lok Hung 2006
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

[Signature]
KA-LOK HUNG
THE FAIR DATA COLLECTION PROBLEM IN WIRELESS SENSOR NETWORKS

by

KA-LOK HUNG

This is to certify that I have examined the above M.Phil. thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

DR. BRAHIM Bensaou, THESIS SUPERVISOR

PROF. LIONEL NI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering

11 August 2006
DEDICATION

To the memory of my grandmother

and

To my family

iv
ACKNOWLEDGMENTS

I would like to sincerely express my gratitude to my supervisor, Prof. Brahim Bensaou, for his guidance and patience throughout my research. Without his support, it is impossible for me to have a chance to be a MPhil Student.

I would like to thank the love and support from my parents and brother. They have provided a comfortable and happy environment for me to grow up.
# TABLE OF CONTENTS

Title Page i
Authorization Page ii
Signature Page iii
Dedication iv
Acknowledgments v
Table of Contents vi
List of Figures viii
List of Tables x
Abstract xi

## Chapter 1 Introduction 1
1.1 Energy Aware Routing for WSN 3
   1.1.1 Optimization Based Protocol 3
   1.1.2 Data Centric Protocol 5
   1.1.3 Hybrid Protocol 7
1.2 Motivation 7
1.3 Fair Data Collection Problem 8
1.4 How to Achieve Fairness 9
   1.4.1 Max-min Fairness 9
   1.4.2 Proportional Fairness 10
1.5 Organization of the Thesis 10

## Chapter 2 Fair data collection based on Sub-gradient optimization method 11
2.1 Introduction 11
2.2 The System Model 12
2.3 The Problem Formulation 12
2.4 Sub-gradient Routing Algorithm 15

LIST OF FIGURES

1.1 A topology example 8
2.1 Ring network topology with 4 nodes 22
2.2 Lollipop topology with 5 nodes 25
2.3 Individual flow value under different utility functions (Simple example, routing model I) 28
2.4 Aggregate flow value under different utility functions (Simple example, routing model I) 28
2.5 Individual flow value under different weight adjustments (Simple example, routing model I) 29
2.6 Aggregate flow value under different weight adjustments (Simple example, routing model I) 29
2.7 Comparison of Individual flow value for different utility functions (Simple example, routing model II) 31
2.8 Aggregate flow value for different utility functions (Simple example, routing model II) 31
2.9 Comparison of Individual flow value for different weight adjustments (Simple example, routing model II) 32
2.10 Aggregate flow value for different weight adjustments (Simple example, routing model II) 32
2.11 The network topology (Random example) 33
2.12 Comparison of Individual flow value (Random example) 34
2.13 Number of routes (Random example) 34
2.14 Aggregate flow value (Random example, routing model 1) 35
2.15 Aggregate flow value (Random example, routing model 2) 35
3.1 Individual flow values under different utility functions (Simple example) 59
3.2 Aggregate flow values under different utility functions (Simple example) 60
3.3 Aggregate flow value when $k = 1$ and $\#step2 = 0$ (Simple example) 61
3.4 Aggregate flow value when $k = 2$ and $\#step2 = 0$ (Simple example) 61
3.5 Aggregate flow value when $k = 3$ and $\#step2 = 0$ (Simple example) 62
3.6 Aggregate flow value when $k = 3$ and $\#step2 = 1$ (Simple example) 62
3.7 Aggregate flow value when $k = 3$ and $\#step2 = 2$ (Simple example) 63
3.8 Comparison of Individual flow value for different weight adjustments (Simple example) 63
3.9 Aggregate flow value for different weight adjustments (Simple example) 64
3.10 Aggregate flow value for adjustment1 and \(\#step2 = 0\) (Simple example) 64
3.11 Aggregate flow value for adjustment2 and \(\#step2 = 0\) (Simple example) 65
3.12 Aggregate flow value for adjustment3 and \(\#step2 = 0\) (Simple example) 65
3.13 Random Network Topology 66
3.14 Aggregate flow values for the Random Network 67
3.15 Aggregate flow value when \(\#step2 = 0\) (Random example) 68
3.16 Aggregate flow value when \(\#step2 = 1\) (Random example) 68
3.17 Aggregate flow value when \(\#step2 = 2\) (Random example) 69
3.18 Aggregate flow value when \(\#step2 = 3\) (Random example) 69
3.19 Individual flow values for the Random Network 70
3.20 Convergence of the asynchronous algorithm 73
3.21 Energy overhead of the asynchronous algorithm 73
LIST OF TABLES

3.1 Bottleneck of each node of the random network 70
3.2 The data collection of the maximum lifetime approach 72
THE FAIR DATA COLLECTION PROBLEM IN WIRELESS SENSOR NETWORKS

by

KA-LOK HUNG

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

ABSTRACT

Sensor networks are deployed to gather some useful data from a field and forward it toward a set of base stations or sinks for data analysis and decision making. Each sensor node is endowed with a finite amount of energy, and each byte transmission or reception costs a certain fixed fraction of energy as well as a variable fraction that depends on the distance between sender and receiver for transmissions. The traditional approach of designing energy-aware routing algorithms for sensor networks by maximizing the so-called lifetime; the time until the first node exhausts its limited battery; is not a panacea because of the expected high level of node redundancy in sensor networks. Instead, maximizing the volume of data collected at the sinks until some particular set of nodes exhaust their battery and partition the network is more desirable. However, doing so, results in an optimized network throughput at the expense of fairness, some nodes may suffer from starvation. Therefore, data collection maximization and fairness should be always considered together. We call this problem the fair data collection problem.

In this thesis, we formulate the problem of fair data collection for sensor networks as a utility maximization problem subject to energy constraints, and invoke Lagrange relaxation, duality and sub-gradient technique to solve the problem. Different performance aspects of the obtained algorithms are considered, particularly the problem of
path oscillation, which is well known to happen in any routing algorithm where the link costs are functions of the traffic load. In order to derive an asynchronous algorithm, the proximal optimization method is also considered. Both convergence and energy overhead of the asynchronous algorithm are studied.
CHAPTER 1

INTRODUCTION

Recent cost reductions in micro-sensor technology and advances in wireless communication technology have spun off a new kind of networks where, typically a large number of unattended sensors, deployed in an ad hoc fashion, collaborate through multi-hop wireless communication channels to perform some common tasks. Such a network is known as a Wireless Sensor Network (WSN). Each sensor can sense the environment, collect data and transmit such data to a sink node that is assumed to have an unlimited power source. Due to their flexibility and low cost of deployment, WSNs have a wide range of applications such as environmental monitoring [33], military applications [1] and human imaging and tracking [8].

Although WSNs bring about many new applications and benefits, they may also come up with many challenging design factors such as:

1. **Distributed architecture**: A large number of sensor nodes are deployed randomly over the region of interest. There is no infrastructure to coordinate the communication.

2. **Multihop communication**: Each sensor node is equipped with a low power wireless radio module having limited communication range. Multi-hop communication is therefore required to forward the collected data from the area of interest to the data collection point (sink node).

3. **Limited energy**: Each sensor node is endowed with a finite amount of energy, and each byte’s reception and transmission cost a certain fixed fraction of energy for reception as well as a variable fraction that depends on the distance between sender and receiver.

4. **Fault Tolerance**: Sensors may be exposed in the physical environment for a very long period of time without replacement and may be fail due to physical damage or interference. The WSN should be able to sustain the network function even if there are sensor failures.
5. **Scalability**: The number of sensors deployed in the area of interest may be in the order of hundreds or thousands. The proposed working scheme for a WSN should be able to fully utilize such dense deployment.

Compared with all the design factors, limited energy is the primary consideration. Since a sensor is a small device which can only be equipped with limited power supply, for example, an AA size battery. In most applications, unlike ad-hoc networks, power source replacement is not possible. Also, each sensor in the network serves as both data originator and router. If some of the sensors are running out of energy, the network topology will change. In the extreme case, the network will be partitioned and some sensors can never forward data to the sinks. Therefore, energy conservation plays a very important role when designing a working scheme for WSN.

The major duties of each sensor are to sense events in the area of interest, process the collected data and forward the data to the sink. The power consumption hence can be divided into three aspects: sensing, data processing and wireless communication. Among these three aspects, wireless communication is well known to consume the most energy. As such, when designing communication protocols for WSNs, energy conservation should be the primary concern.

Many recent research studies have addressed the different problems of energy saving communication protocols in all the layers of the protocol stack. At the physical layer, some of the researchers try to limit the transmission power and enhance spatial reuse without losing the network connectivity [25, 27, 35]. At the Medium Access Control (MAC) layer, the authors of [36] put the sensors to sleep periodically and hence reduce the energy consumption of idle listening, overhearing others traffic and packets collisions. They propose a scheme to synchronize the sleeping schedule among the sensors. At the application layer, depending on different application requirements, some approaches [22, 28] have shown significant energy conservation.

In this thesis, we focus on the energy awareness of the routing protocol at the network layer of WSN. We consider the analysis and design of routing protocols for energy-constrained wireless sensor networks with a primary focus on maximizing the amount of data collected with fairness consideration at some collection points (sinks).
1.1 Energy Aware Routing for WSN

A number of energy aware routing protocols have been studied in recent years for WSN. The routing protocols can be divided into three classes: optimization based protocols, data centric protocols and hybrid protocols.

1.1.1 Optimization Based Protocol

In optimization based protocols, the energy aware routing problem is formulated as a linear or non-linear optimization problem. Iterative optimization algorithms or heuristics are applied to solve the problem.

In the early ad-hoc routing literature, a routing algorithm for minimizing energy consumption is proposed in [32]. The protocol is in fact an energy-aware MAC protocol that enables choices of hops in the routing, based on different cost metrics related to the energy consumption. Among the possible disadvantages of this algorithm is that the battery energy of the nodes along the smallest energy consumption path are drained quicker than other nodes. Instead of minimizing the energy consumption, maximum lifetime routing is proposed in [5]. The lifetime of the network is defined as the time until the first node runs out of energy, and based on the underlying assumption that the sensor nodes generate traffic periodically at a predefined rate, a linear programming formulation is presented and a heuristic algorithm is developed to solve the problem. Each link is associated with a link cost which is defined as a function of the nodes’ remaining energy, initial energy and required transmission and reception energy when using a given link. The link cost of link(i, j) is proposed as

\[ \text{cost}_{ij} = (e^t_{ij})^{x_1} E_i^{-x_2} E_i^{x_3} + (e^r_{ij})^{x_1} E_j^{-x_2} E_j^{x_3}, \]

where the transmission energy consumed at node i to transmit a data unit to its neighboring node j is denoted by \( e^t_{ij} \), the energy consumed by the receiver j is denoted by \( e^r_{ij} \), \( E_i \) and \( E_j \) are the initial energy of node i and j, \( E_i \) and \( E_j \) are the remaining energy of node i and j, \( x_1, x_2 \) and \( x_3 \) are the parameters of the algorithm. The objective of the heuristic is to maximize the network lifetime by routing the packet through the least cost path. The algorithm can be implemented with any existing shortest path algorithms such as the Bellman–Ford algorithm. By tuning the parameters through simulation, the proposed algorithm shows significant improvement in term of network.
lifetime when compared with some existing routing algorithms like minimum total energy routing and and max-min residual energy routing. In [39], an iterative max-flow algorithm is presented to address a similar linear programming problem. In [29], the problem is treated as a maximum concurrent flow problem and a distributed routing algorithm is presented to maximize the network lifetime. In [21], a similar maximum lifetime routing problem is defined and a sub-gradient algorithm is presented to solve the problem distributively. Arguably, the main limitation of maximum lifetime routing is the definition of lifetime itself. Indeed, in many sensor networks applications, nodes are redundant – this is the case in unplanned networks where sensor nodes are dropped randomly and in such large numbers as to increase the probability of total coverage. In this case, the exhaustion of one node’s battery does not mean that the network capability is diminished. By solving the maximum data collection problem, the network can operate until some critical nodes exhaust their battery and partition the sources from their respective sinks.

In some applications, the traffic generation rate may be variable rather than fixed and predictable. Online routing algorithms are proposed in [15, 18] to address this issue. In [18], a max-min $zP_{\text{min}}$ algorithm is proposed. A message takes a path which maximizes the minimum residual energy fraction with the constraint that this path consumes at most $z$ times more energy than the minimum energy consumption path. In [15], a shortest path routing algorithm is proposed, where the link cost is based on the energy cost of transmission and the residual energy fraction of the sender. A logarithmic competitive ratio is achieved if admission control of messages is allowed. Both of the proposed algorithms do not include the energy cost of receiving data which in real world applications is known to be of the same order of magnitude as the cost of sending data.

In [37], the maximum data extraction problem is presented and a sub-gradient algorithm to solve the problem is proposed. This formulation is closely related to ours with the exception of two significant differences i) A linear programming formulation is used for maximizing the total amount of data extracted and thus the fairness issue among different nodes is not considered; ii) The work in [37] considers only one sink in the network whereas in our work, the system model includes multiple possible sinks and each node is associated with one sink. This is an important issue because of the issues of complexity and also stability of the algorithm.
1.1.2 Data Centric Protocol

In a data centric protocol, the energy saving techniques depend on the properties of the sensor network. One of the most common techniques is data aggregation. Since sensors may be densely deployed in the area of interest, the data collected from some sensor may be redundant or correlated. By aggregating or eliminating redundant data, fewer packet transmissions and receptions are needed and hence reduces the energy consumption is reduced.

In Directed diffusion [13], all data generated by the sensor is labelled by a attribute-value pair such as name of objects, interval, duration, geographical area and the like. The data is collected on demand and is triggered by the sink. The sink creates a query of interest by the attribute-value and broadcasts it through the neighboring sensors. The interest is cached in the nodes for later use. During the broadcasting of the interests among the network, paths are also established for sensors to forward data to the sink. The routing algorithm is data-centric in which data is combined according to the label. The number of transmissions is minimized which results in saving energy and prolonging network lifetime. Unlike traditional routing algorithms which are mainly designed for end-to-end flows, the routing algorithm for directed diffusion generates paths that collect data from multiple sensors to a single sink. Since each time a broadcast of interest is needed before data collection can be performed, direct diffusion is not suitable for the situation if most of the queries are just one-time as the overhead is too large.

In LEACH [9], the authors propose a hierarchical clustering algorithm which is called the Low Energy Adaptive Clustering Hierarchy(LEACH) for WSNs. Some cluster heads are selected randomly from the network. Each sensor is associated with a cluster head. The sensors forward the data to their corresponding cluster heads. The role of the cluster heads is to aggregate the data and forward it to the sink. To achieve better load balancing, the cluster heads are reselected periodically. The cluster head selection is based on the following rule: Each sensor node \( n \) selects a random number \( r \) between 0 and 1. If \( r \) is smaller than a number \( T(n) \), the sensor will be selected as the cluster head. \( T(n) \) is defined as

\[
T(n) = \frac{P}{1 - p(r - \text{mod} (1/p))} \quad \forall n \in G
\]

where \( p \) is the desired percentage of cluster heads and \( G \) is the set of nodes that have
not been cluster heads in the last $1/p$ rounds. If one node’s random number $r$ is smaller than the value of $T(n)$, that node will become the cluster head in the next round. The newly selected cluster heads broadcast advertisement messages to the remaining sensors. Some sensors may receive multiple advertisements from different cluster heads. The association of the cluster heads is then based on the signal strength of the advertisement messages. Time division multiple access (TDMA) and code-division multiple access (CDMA) medium access control are used to reduce the packet collisions inter-cluster and intracluster. LEACH assumes that each node can transmit directly to the cluster head and the sink. Hence, the protocol is not applicable when the sensors are deployed over a large area as multi-hop communication is needed. It also assumes that there is always traffic for a node to send.

An improved version of LEACH which is called Power-Efficient GAthering in Sensor Information Systems (PEGASIS) is proposed in [18]. Unlike LEACH which is a cluster-based protocol, PEGASIS is a chain-based protocol. Each sensor only needs to transmit and receive to a neighbor. A communication chain is formed. A sensor of the chain is then selected to forward the data to the sink. The data is aggregated when the traffic is passing through the chain which results in energy saving. Simulation shows that it outperforms LEACH by about 100 to 300% as it does not include a dynamic cluster formation overhead and again reduces energy consumption by data aggregation. However, there may be a huge delay when packets travel through a "long" chain. An extension to PEGASIS which is called the Hierarchical PEGASIS [30] is proposed to address this issue. To reduce the transmission delay, simultaneous transmission techniques of data such as signal coding and spatial transmission are studied.

In addition to data aggregation, some routing protocols try to reduce the number of transmissions by trading data accuracy for energy efficiency. In [23], a routing protocol which is called the Threshold-Sensitive Energy Efficient Sensor Network Protocol(TEEN) is proposed. In TEEN, which is a cluster-based routing protocol, the area of interest is sensed continuously. In order to reduce the number of transmissions, the users can set a soft threshold and a hard threshold at the cluster heads. Both soft and hard thresholds are broadcast to the members of the clusters by the cluster heads. The hard threshold is used to define a range of interest of some attributes of the sensed event. The data transmission only occurs when the attributes are within the range and hence reduce the number of transmissions. The soft threshold defines how large the difference or change in value of the attributes will be trigger the transmission. A
smaller soft threshold allows higher accuracy of data collection.

1.1.3 Hybrid Protocol

In a hybrid protocol, both optimization formulation and data aggregation are considered for the energy aware routing problem. In [14], the authors consider the maximum lifetime data gathering problem with data aggregation. They formulate the problem as an integer program with linear constraints. A polynomial time cluster-based heuristics is proposed to solve the problem. They assume that the sensors can always combine their own data to the data received from the others which is known as perfect data aggregation assumption. In [38], the authors study the minimum energy data gathering problem with the consideration of data aggregation in WSNs. The goal is to find the flow value vector for a continuous sensing sensor application such that total energy consumed is minimized. The problem is formulated as a linear programming problem with linear constraints. They again assume that the perfect data aggregation is valid. An iterative sub-gradient algorithm is proposed to solve the problem.

1.2 Motivation

In the optimization based and hybrid routing protocol literature, the energy-aware routing problem in most of the work has been considered as a network lifetime maximization problem, which aims to maximize the time interval between the initiation of the operation and the death of the first sensor in the network with the assumption of constant data generation rates [5, 14, 21, 29, 39]. Most studies of this problem make some assumptions about the operation of sensor networks that are not general for all sensor networks, such as constant data rates generated by sensors. Sensor networks are usually application-centric and there are many applications that do not match such assumptions. For instance, in event-based sensor networks, sensors generate data and report it to sink nodes only when some phenomenon of interest occurs. Also, sensors are usually deployed with high redundancy, which means the death of one or more sensors makes no difference to the operation of sensor networks. Therefore, while satisfying certain coverage or other requirements, as long as some sensors can find routes to the sink nodes, the sensor network can still be regarded as functional. In [15, 18, 37, 38], the proposed routing model does not have constant data rates generation assumption
but all of them have some drawbacks like ignorance of reception energy consumption, prefect data aggregation and so on.

1.3 Fair Data Collection Problem

From this perspective, we study the problem of energy-efficient routing in sensor networks with the objective of maximizing the volume of data collected at the sinks. Each sensor can sense the environment, collect data and transmit this data to a sink node, which is assumed to have an unlimited energy supply. Due to the short transmission range of sensor nodes, multi-hop communication is required when forwarding data from sensors to sinks which have unlimited energy supply. The objective of our routing algorithm is to compute the flow value of each link such that the amount of data collected from different nodes is maximized. Instead of only maximizing the total data flow, we focus on maximizing the data collected by each sensor node fairly. To illustrate the importance of such fairness, consider the network shown in Fig. 1. We assume that nodes 1, 3 and 5 will send their data to sink 7 only, and nodes 2, 4 and 6 will only send their data to sink 8. For the sake of simplicity, we assume that the energy consumption cost of sending and receiving data are fixed and the same for all nodes which have the same initial battery energy. If the routing algorithm only maximizes the data collected without fairness, then node 5 and 6 will use all their energy to send their own traffic to the sinks, and other nodes are not able to send data to sinks. This is because nodes 5 and 6 are bottleneck nodes on all routes leading to the sinks, and data generated by nodes 5 and 6 costs less in terms of energy than any other sensors.
because the reception cost at the sinks is nil.

This simple illustration shows that fairness should be considered if the routing protocol is to ensure that each sensor is able to send a “reasonably” large amount of data to the sink. The fairness issue is application-dependant. For example, in an environmental monitoring system where the data from different sensors are of equal importance, max-min fairness is more appropriate as it maximizes the minimum amount of data that can be sent by sensors. On the other hand, in an enemy troop movement monitoring system, it is preferable to monitor such movements with continuously increasing accuracy when they get closer to a given target. Enemy troops movements that take place far away from this strategic target position, can be monitored less frequently than those that take place at a closer range. In this situation, sensors which are closer to the sinks (targets) should be allowed to send more traffic than those that are farther away. In this case, proportional fairness is more appropriate as it penalizes sensors based on their global use of resources rather than their use of resources on the bottleneck link.

1.4 How to Achieve Fairness

Different fairness principles define how the resources fairly are the resources shared in the network. It has been shown repeatedly in the literature that fairness can be achieved by maximizing a concave non-decreasing utility function. The shape of the utility function determines the targeted fairness objective. We comment and review hereafter some fairness objectives and the associated utility functions.

1.4.1 Max-min Fairness

One of the best known fairness criteria is the so-called Max-Min Fairness principle. A feasible flow volume vector $x$ is said to be max-min fair if and only if any component $x_i$ of the vector cannot be increased without decreasing another component $x_j$ whose value is lesser or equal to $x_i$. In [24], the authors show that when all the sources share a common utility function $U(x) = (1 - k)^{-1}x^{1-k}$ and $k \rightarrow \infty$, then maximizing the sum of the individual utilities leads to a flow volume allocation that is max-min fair.
1.4.2 Proportional Fairness

In [16], the authors proposed the so-called proportional fairness. A flow volume vector $\hat{x}$ is said to be proportional fair if and only if it is feasible, and for any other feasible vector $x$, the aggregate of proportional changes is non-positive, i.e.

$$\sum_i \frac{x_i - \hat{x}_i}{\hat{x}_i} \leq 0. \quad (1.1)$$

It has been shown that the optimal proportional fair flow volume allocation is achieved by maximizing the sum of the individual utility functions, and all the sources share the common utility function $log(.)$. The result comes from the optimality condition of the primal problem [17], i.e.

$$\sum_i U_i(\hat{x}_i)(x_i - \hat{x}_i) = \sum_i \frac{x_i - \hat{x}_i}{\hat{x}_i} \leq 0. \quad (1.2)$$

1.5 Organization of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2, a sub-gradient routing algorithm is proposed to address the fair data collection problem. In Chapter 3, a proximal optimization algorithm is presented for an asynchronous data collection algorithm. Finally, future work is discussed and conclusion is drawn in Chapter 4.
CHAPTER 2

FAIR DATA COLLECTION BASED ON SUB-GRADIENT OPTIMIZATION METHOD

2.1 Introduction

In this chapter, the fair data collection problem is modeled as a concave utility maximization problem subject to energy constraint and data flow constraints. Two possible models of routing are studied [10] [11].

1. **Routing model I** associates each link with a utility function that depends on the amount of data which originates at the ingress node of the link.

2. **Routing model II** associates each source node with a utility function that depends on the amount of data which originates at this source.

By invoking Lagrange relaxation and duality, sub-gradient routing algorithms are proposed to solve the optimization problem for both models. Routing model I, forces the source to split its traffic among as many routes as the number of its neighbors, leading to multiple instances of least cost path routing, whereas in routing model II the optimal solution leads to a single least cost path routing. From numerical experiments, we show that solving the routing model II results in higher individual data collection than routing model I. However, in the case of route failure, routing model I is preferable as the source is able to send data to the sink by using other functioning routes among its multiple paths. More importantly, we prove analytically that in both models, when there are multiple equal cost optimal paths (which is the case in most practical cases with topologies other than the tree topology), unless the sources split their traffic equally among these optimal paths, a stable equilibrium does not exist with the sub-gradient based approach which has proved to be popular recently for this kind
of problem. The conditions of stability of the algorithms are studied analytically and numerical results given to illustrate the convergence of the algorithms.

The remainder of this chapter is structured as follows. In Section 2.2, we present the system model. We formulate the problem in section 2.3, and present the sub-gradient routing algorithm in section 2.4.2. In section 2.5, the stability of the algorithms are studied. Numerical results are presented in section 2.6.

### 2.2 The System Model

We model the wireless sensor network as a directed graph $G(\mathcal{N}, L)$ where $\mathcal{N}$ is the set of all nodes and $L$ is the set of all directed links $(i, j), i, j \in \mathcal{N}$. Let $\text{out}(i)$ be the set of all nodes $j$ such that arc $(i, j) \in L$ and $\text{in}(i)$ be the set of nodes $j$ such that arc $(j, i) \in L$. The network topology is represented by a node-incidence matrix $A \in \mathbb{R}^{\vert \mathcal{N} \vert \times \vert L \vert}$ where each element $a_{il}$ is defined as

$$a_{il} = \begin{cases} 
1, & \text{if link } l \text{ is an outgoing link of node } i \\
-1, & \text{if link } l \text{ is an incoming link of node } i \\
0, & \text{otherwise}.
\end{cases}$$

Let $e_{ij}^s$ be the energy cost for node $i$ to send one unit of data (one byte) to node $j$ and let $e_{ji}^r$ be the energy cost for node $i$ to receive one byte from node $j$. Denote by $E_i$ the initial amount of battery energy contained in node $i$.

The network contains a set $T$ of sink nodes $T \subset \mathcal{N}$, that are assumed to only receive data without generating any, and that have an unlimited supply of power. Furthermore, each sensor node in the set $S = \mathcal{N} \setminus T$ is associated with exactly one corresponding sink in the set $T$. The set of sources associated with a sink $t \in T$ is denoted as $S(t)$. Denote $x_{ij}^k$ the flow generated by source $k$ transiting from node $i$ to node $j$. The total amount of traffic on link $(i, j)$ is thus $x_{ij} = \sum_{k \in S} x_{ij}^k$. The traffic generated by a source $k$ is called commodity $k$, and the flow vector of commodity $k$ is denoted $x^k = \{x_{ij}^k\}$ and the flow vector of all commodities is denoted as $x = \{x_{ij}\}$.

### 2.3 The Problem Formulation

A multicommodity flow model is used to model the fair data collection problem. The problem can be written as a convex programming problem where the objective is to
maximize a strictly increasing concave utility function of each commodity. There are several problem constraints. The first constraint is the energy constraint: The energy consumption for transmitting and receiving data for each sensor node $i$ is limited by its initial energy, i.e.,

$$\sum_{j \in \text{out}(i)} e_{ij}^s \sum_{k \in S} x_{kj}^i + \sum_{j \in \text{in}(i)} e_{ji}^r \sum_{k \in S} x_{jk}^i \leq E_i.$$  

(2.1)

Define an energy consumption matrix $P \in \mathbb{R}^{|N| \times |L|}$ whose elements $p_{il}$ are:

$$p_{il} = \begin{cases} e_{ij}^s, & \text{if link } l = (i, j) \text{ exists} \\ e_{ji}^r, & \text{if link } l = (j, i) \text{ exists} \\ 0, & \text{otherwise}; \end{cases}$$  

(2.2)

and let $E = \{E_i\}$ be the initial energy vector. The energy constraints for all the nodes (excluding the sinks) can be written compactly as

$$Px \leq E$$  

(2.3)

The second set of constraints stems from flow conservation considerations. That is, firstly, no source should route its own traffic on behalf of other sources. This writes mathematically as:

$$-\sum_{j \in \text{in}(i)} x_{ji}^i = 0, \quad \forall i \in N.$$  

(2.4)

Secondly, flow conservation holds at each node, for each commodity $i$, which translates into:

$$\sum_{j \in \text{out}(k)} x_{kj}^i - \sum_{j \in \text{in}(k)} x_{jk}^i = 0 \quad \forall k \in S, k \neq i.$$  

(2.5)

Finally, sinks accept all of their associated sources’ traffic and nothing else, which writes:

$$-\sum_{j \in \text{in}(k)} x_{jk}^i = 0 \quad \forall k \in T, i \notin S(k),$$  

$$-\sum_{j \in \text{in}(k)} x_{jk}^i = -\sum_{j \in \text{out}(i)} x_{ij}^i \quad \forall k \in T, i \in S(k).$$  

(2.6)

By defining a modified node-edge incidence matrix $A^i = \{a_{xl}^i\}$ for each commodity $i$ as:

$$a_{xl}^i = \begin{cases} 0, & \text{if } x = i, a_{xl} = 1 \\ a_{xl}, & \text{otherwise}, \end{cases}$$  

(2.7)
and flow vector $s^i = \{s^i_k\}$ as

$$s^i_k = \begin{cases} 
- \sum_{j \in \text{out}(i)} x^i_{ij}, & \text{if } k \in T, i \in S(k) \\
0, & \text{otherwise},
\end{cases} \quad (2.8)$$

we can write these flow constraints compactly as

$$A^i x^i = s^i, \quad i \in S. \quad (2.9)$$

Finally, the last constraint imposes that commodity flow vectors are non-negative:

$$x^k \succeq 0 \quad (2.10)$$

In general, maximizing concave non-decreasing utility function of a given variable (representing our context the volume of data) guarantees fairness [7, 31]. To illustrate this, consider a concave non-decreasing utility function $U(.)$ for $x$ and $y$ where $x > y$. We have $U'(y) > U'(x)$, which means, to maximize $U(x) + U(y)$ it is preferable to increase $y$ than $x$, as for the same increment, $U(y + \delta)$ is larger than that $U(x + \delta)$. Following this principle, the smaller flow value is always favored by concave utility functions, thus the system tends to be fair. The particular shape of the concave utility function defines the fairness principle in question. To ensure fairness among nodes we adopt this approach.

Consider routing problem I, where multiple paths are used to split the traffic of a given source and transfer it to the sink. Under this scenario, each outgoing link $(i, j)$ of a given node $i$ is associated with a utility function $U_{ij}$. Thus the network utility is written in this case $\sum_{i \in S} \sum_{j \in \text{out}(i)} w_{ij} U_{ij}(x^i_{ij})$; and the problem of fair routing with maximum data collection can be written as

$$P_1 : \max_x \sum_{i \in S} \sum_{j \in \text{out}(i)} w_{ij} U_{ij}(x^i_{ij}) \quad \text{subject to} \quad P x \leq E$$

$$A^i x^i = s^i \quad i \in S$$

$$x^i \succeq 0 \quad i \in S,$$  \quad (2.11)

where $w_{ij}$ is a positive constant. If a common utility function is used for all the links, the weights $w_{ij}$ can be treated as sensor $i$’s willingness to pay for its own data flow that passes through link $(i, j)$. The flow $x^i_{ij}$ with a larger value of $w_{ij}$ is favored.
Consider routing problem II, where the fairness applies to the total traffic generated by a source regardless of which outgoing link it uses. Under this scenario, each node \( i \) is associated with a utility function \( U_i \). The network utility function is written in this case 
\[
\sum_{i \in S} w_i U_i \left( \sum_{j \in \text{out}(i)} x_{ij} \right);
\]
and the problem of fair routing with maximum data collection can be written as

\[
\mathbf{P}_2 : \max_x \sum_{i \in S} w_i U_i \left( \sum_{j \in \text{out}(i)} x_{ij} \right) \quad \text{subject to} \quad \mathbf{Px} \leq \mathbf{E}
\]

where \( w_i \) is a positive constant. If a common utility function is used for all the flows, the weights \( w_i \) can be used to further differentiate the sensor nodes. The larger the value of \( w_i \) the more traffic can node \( i \) send.

Since the volume of traffic is formulated on each link in both routing models, the above problem formulations are known as the node-link formulation [26].

### 2.4 Sub-gradient Routing Algorithm

#### 2.4.1 The Dual Problem

To solve the above optimization problems directly global topology and flow information exchange between all the nodes in the network is needed. In order to avoid this, we consider Lagrange relaxation and duality. We relax the energy constraints and maintain the flow conservation constraints.

Let the set \( \chi \) be the set of flow values that satisfy the flow constraints:

\[
\chi = \left\{ x^1 \in \mathbb{R}^{|L|} | \mathbf{A}^1 x^1 = s^1 \right\} \times \cdots \times \left\{ x^{|S|} \in \mathbb{R}^{|L|} | \mathbf{A}^{|S|} x^{|S|} = s^{|S|} \right\}.
\]

The partial Lagrangian \( L_1(x, p) \) and \( L_2(x, p) \) of the respective primal problems \( \mathbf{P}_1 \) and \( \mathbf{P}_2 \) can be obtained by introducing Lagrange multiplier \( p \in \mathbb{R}^{|S|} \) for the energy constraints:

\[
L_1(x, p) = \sum_{i \in S} \sum_{j \in \text{out}(i)} w_{ij} U_{ij}(x_{ij}) + \sum_{i \in S} p_i \left\{ E_i - \sum_{j \in \text{out}(i)} e^i_{ij} \sum_{k \in S} x^k_{ij} - \sum_{j \in \text{in}(i)} e^i_{ji} \sum_{k \in S} x^k_{ji} \right\},
\]

(2.14)
and,
\[ L_2(x, p) = \sum_{i \in S} w_i U_i \left( \sum_{j \in \text{out}(i)} x_{ij} \right) + \sum_{i \in S} p_i \left\{ E_i - \sum_{j \in \text{out}(i)} e_{ij}^{s} \sum_{k \in S} x_{kij} - \sum_{j \in \text{in}(i)} e_{ji}^{r} \sum_{k \in S} x_{kji} \right\}. \]  
(2.15)

The dual functions \( D_1(p) \) and \( D_2(p) \) of problems \( P_1 \) and \( P_2 \) are defined respectively as
\[
D_1(p) = \max_{x \in \mathcal{X}} L_1(x, p), \quad (2.16)
\]
\[
D_2(p) = \max_{x \in \mathcal{X}} L_2(x, p), \quad (2.17)
\]
and the corresponding dual problems \( D_1 \) and \( D_2 \) of the primal problems \( P_1 \) and \( P_2 \) can be written respectively
\[
D_1: \min_{p} \quad D_1(p) \quad (2.18)
\]
\[
\text{s.t.} \quad p \succeq 0. \quad (2.19)
\]
\[
D_2: \min_{p} \quad D_2(p) \quad (2.20)
\]
\[
\text{s.t.} \quad p \succeq 0. \quad (2.21)
\]

We can easily prove that Slater’s condition for the constraints qualification is satisfied: that is, there exists a feasible solution \((x, p)\) such that the strict inequalities for the energy constraints hold, that is,
\[
\sum_{j \in \text{out}(i)} e_{ij}^{s} \sum_{k \in S} x_{kij} + \sum_{j \in \text{in}(i)} e_{ji}^{r} \sum_{k \in S} x_{kji} < E_i \quad \forall i \in S. \quad (2.22)
\]

Thus, thanks to the convex nature of the optimization problem, the strong duality holds [4], and therefore, the optimal value of the primal problems and that of the corresponding dual problems are the same.

### 2.4.2 Sub-gradient Algorithm

The objective function in (3.4) and (2.12) are not strictly concave in all the variables \( x_{ij}^h \). The dual functions are non-differentiable in some points. Instead of using a traditional gradient method such as in [7], we invoke here the sub-gradient method [2]. The sub-gradient at \( x \in \mathbb{R}^n \) of a non-differentiable convex function \( f \) is a vector \( d \in \mathbb{R}^n \) such that:
\[
f(y) \geq f(x) + (y - x)^T d, \quad \forall y \in \mathbb{R}^n. \quad (2.23)
\]
If the convex function \( f \) is differentiable, the only choice of sub-gradient is the gradient of the function.

**Theorem 2.4.1** Let \( x(p) \) be the maximizer of problem (3.13) (respectively (2.17)). Then, the sub-gradient \( g \) corresponding to \( p \) is:

\[
g = E - Px(p) \quad (2.24)
\]

**Proof:** We concentrate hereafter on proving the Theorem for the case of (3.13). The proof for (2.17) is similar.

For any point \( \bar{p} \),

\[
D_1(\bar{p}) = \max_{x \in \chi} \sum_{i \in S} \sum_{j \in \text{out}(i)} w_{ij} U_{ij}(x_{ij}(\bar{p})) + \sum_{i \in S} \bar{p}_i \left\{ E_i - \sum_{j \in \text{out}(i)} e_{ij}^s \sum_{k \in S} x_{ij}^k(\bar{p}) - \sum_{j \in \text{in}(i)} e_{ji}^r \sum_{k \in S} x_{ji}^k(\bar{p}) \right\}. \quad (2.25)
\]

By substituting \( x_{ij}^k(p) \) for \( x_{ij}^k(\bar{p}) \), we have:

\[
D_1(\bar{p}) \geq \max_{x \in \chi} \sum_{i \in S} \sum_{j \in \text{out}(i)} w_{ij} U_{ij}(x_{ij}(p)) + \sum_{i \in S} \bar{p}_i \left\{ E_i - \sum_{j \in \text{out}(i)} e_{ij}^s \sum_{k \in S} x_{ij}^k(p) - \sum_{j \in \text{in}(i)} e_{ji}^r \sum_{k \in S} x_{ji}^k(p) \right\}, \quad (2.26)
\]

which can be rewritten as:

\[
D_1(\bar{p}) \geq D_1(p) + \sum_{i \in S} (\bar{p}_i - p_i) \left\{ E_i - \sum_{j \in \text{out}(i)} e_{ij}^s \sum_{k \in S} x_{ij}^k(p) - \sum_{j \in \text{in}(i)} e_{ji}^r \sum_{k \in S} x_{ji}^k(p) \right\}. \quad (2.27)
\]

In matrix form this writes,

\[
D_1(\bar{p}) \geq D_1(p) + (\bar{p} - p)^T (E - Px(p)), \quad (2.28)
\]

which according to the definition of a sub-gradient proves Theorem 2.4.1

In the sub-gradient method which is an iterative method, at each iteration step \( t = 1, 2, 3, ... \), the maximizer of problem (3.13) (respect. problem (2.17)) is found, then the sub-gradient \( g \) is evaluated, then the dual variable \( p \) is updated as follows:

\[
p_i(t + 1) = \max(p_i(t) - \gamma(t)g_i(t), 0) \quad (2.29)
\]
where $\gamma(t)$ is a positive scalar step size. If $\gamma(t)$ is a constant, it can be shown that if an optimal $\bar{p}$ exists, the sub-gradient method converges statistically to within $\gamma G^2/2$ of the optimal value where $G$ is a constant such that $||g(t)||_2 \leq G$ for all $t$, and $||.||_2$ is the Euclidian norm. Proof on a similar problem could eventually be found in [6]. Alternatively, $\gamma(t)$ can be obtained by using a diminishing update rule,

$$\gamma(t) = \frac{\gamma(0)}{\sqrt{t}}$$

(2.30)

where $\gamma(0)$ is a fixed constant. In this case, the sub-gradient algorithm converges to the optimum [2].

The remaining question is how can we find the optimal flow $\bar{x}$ given the price vector $p$. For this purpose, we assume the communication links are bi-directional, which is the case in most practical cases as the medium access control protocol is often based on CSMA/CA with acknowledgements, that is $out(i) = in(i)$ for all $i$. Note that although the sinks are supposed to have outgoing links they never generate outgoing traffic.

**Routing model I**

We can rewrite problem (3.13) as

$$D_1(p) = \max_{x \in \chi} \sum_{i \in S} \sum_{j \in out(i)} \sum_{t \in S} \sum_{v \in out(t)} \left( \sum_{t \in S} \sum_{v \in out(t)} \left( e^e_{tv} \sum_{i \in S} (e^e_{tv} \sum_{i \in S} x^i_{tv} + e^r_{vt} \sum_{i \in S} x^i_{vt}) \right) \right)$$

$$= \max_{x \in \chi} \sum_{i \in S} \sum_{j \in out(i)} \sum_{t \in S} \sum_{v \in out(t)} \left( \sum_{t \in S} \sum_{v \in out(t)} \left( e^e_{tv} \sum_{i \in S} x^i_{tv} + e^r_{vt} \sum_{i \in S} x^i_{vt} \right) \right) + \sum_{i \in S} p_i E_i$$

$$= \max_{x \in \chi} \sum_{i \in S} \sum_{j \in out(i)} \sum_{t \in S} \sum_{v \in out(t)} \left( \sum_{t \in S} \sum_{v \in out(t)} \left( e^e_{tv} \sum_{i \in S} x^i_{tv} + e^r_{vt} \sum_{i \in S} x^i_{vt} \right) \right) + \sum_{i \in S} p_i E_i$$

(2.31)

If we interpret the dual variable $p_i$ as the price for the energy unit at node $i$, then (2.31) can be regarded as a demand and supply problem. Given the price vector $p$, each source $i$ tries to maximize the revenue $\sum_{j \in out(i)} w_{ij} U_{ij}(x^i_{ij})$ with the cost $\sum_{i \in S} \sum_{j \in out(i)} (p_i e^e_{tv} + p_v e^r_{vt}) x^i_{tv}$ under the flow conservation constraint for commodity $i$.

Source $i$ has to find a least cost path for each of its outgoing links to send out its own
traffic so that the total cost is minimized. If there is more than one least cost path per outgoing link, source \( i \) can split the traffic among the least cost paths arbitrarily as they all lead to the same objective (notice that this requires a kind of source routing protocol). The cost of each link \((i, j)\) is given by

\[
p_i e_{ij}^c + p_j e_{ji}^c.
\]

We define \( SP_1(i, j) \) to be the set of least cost paths for node \( i \) to send data via node \( j \) to its sink. If there is no valid path to the sink, \( SP_1(i, j) = \emptyset \). \(|SP_1(i, j)|\) may be larger than 1. The path cost of the least cost path via link \((i, j)\) is denoted \( lc(SP_1(i, j)) \). Since \( U_{ij} \) is a strictly increasing concave function, the optimal value of \( \bar{x}_{ij}^i \) can be obtained by

\[
\bar{x}_{ij}^i = U_{ij}^{t-1} \left( \frac{lc(SP_1(i, j))}{w_{ij}} \right).
\]

Routing model II

In a similar manner, we can rewrite problem (2.17) as

\[
D_2(p) = \max_{x \in \mathcal{X}} \sum_{i \in S} \left\{ w_i U_i \left( \sum_{j \in \text{out}(i)} x_{ij}^i \right) - \sum_{t \in S} \sum_{v \in \text{out}(t)} (p_t e_{tv}^s + p_v e_{vt}^s) x_{tv}^i \right\} + \sum_{i \in S} p_i E_i.
\]

With the same approach, equation (2.34) can be also regarded as a demand and supply problem. Given the price vector \( p \), each source \( i \) tries to maximize the revenue \( w_i U_i(\sum_{j \in \text{out}(i)} x_{ij}^i) \) with the cost \( \sum_{t \in S} \sum_{v \in \text{out}(t)} (p_t e_{tv}^s + p_v e_{vt}^s) x_{tv}^i \) under the flow conservation constraint for commodity \( i \). Source \( i \) has to find a least cost path to send out its own traffic so that the total cost is minimized. If there is more than one least cost paths, source \( i \) can split its traffic among these least cost paths arbitrarily since they all lead to the same objective. Therefore, the above global optimization problem can be decomposed into as many instances of the least cost path problem as the number of sources. Define \( SP_2(i) \) to be the set of least cost paths from node \( i \) to its associated sink. If there is no valid path to the sink, \( SP_2(i) = \emptyset \). \(|SP_2(i)|\) may be larger than 1. The path cost of the least cost path is denoted as \( lc(SP_2(i)) \). Since \( U_i \) is a strictly increasing concave function, the optimal value of \( \bar{x}_{ij}^i \) can be obtained by

\[
\sum_{j \in \text{out}(i)} \bar{x}_{ij}^i = U_i^{t-1} \left( \frac{lc(SP_2(i))}{w_i} \right).
\]
2.4.3 Comments

In routing model I, since a node considers all of its neighbors as candidate first hops, routing model I always uses at least as many routes as the number of neighbors. In effect routing model I results in a multi-path routing algorithm. In routing model II, if there is a unique least cost path for a given source \( i \) in the optimum routing then source \( i \) will use single-path routing for forwarding its traffic.

Besides, when \( |SP_1(i,j)| > 1 \) or \( |SP_2(i)| > 1 \) the traffic between source-sink pairs can be split arbitrarily between paths as they all lead to the same objective mathematically. However, as we show in Section 2.5, a stable equilibrium does not exist if the traffic is not split evenly among multiple least cost paths. Therefore, to achieve even load balancing, we suggest to split the traffic equally among multiple least cost path in the routing algorithm.

In section 2.6, we show that the optimum solution of routing model II yields a higher individual throughput than that of routing model I. However, the multiple path optimum solution of routing model I is more robust in case of link-failure.

The complete algorithm (in its centralized synchronous representation) is shown in Algorithm 1 for routing model I and Algorithm 2 for routing model II, where convergence is reached when the price change is arbitrarily small from one iteration to the next.

**Algorithm 1** Sub-gradient Routing Algorithm for model I

1: Set \( t = 0 \); set the initial value of \( p_i(t) \) for each node \( i \).
2: repeat
3: \hspace{1em} for each neighbor node \( j \) of node \( i \) do
4: \hspace{2em} Compute the price of \((i, j)\): \( p_i(t)e_{ij}^* + p_j(t)e_{ji}^* \);
5: \hspace{2em} Find the least cost path(s) from node \( i \) to its sink via link \((i, j)\);
6: \hspace{2em} Calculate the new flow \( \bar{x}_{ij}^i(t) \) by (2.33)
7: \hspace{2em} Split the traffic among the least cost paths equally;
8: \hspace{2em} Broadcast the new flow value \( \bar{x}_{ij}^i(t) \) to the all the nodes in \( SP_1(i, j) \);
9: \hspace{2em} Update the price \( p_i(t + 1) \) by (2.29);
10: \hspace{2em} Broadcast the new price \( p_i(t + 1) \) to the neighbors;
11: \hspace{2em} if the diminishing step update rule is used then
12: \hspace{3em} Update \( \gamma(t + 1) \) by (2.30);
13: \hspace{2em} end if
14: \hspace{2em} Set \( t = t + 1 \) Synchronously // next iteration */t;
15: \hspace{1em} end for
16: until Convergence

Note that since the two algorithms are in essence least cost path algorithms, they
Algorithm 2 Sub-gradient Routing Algorithm for model II

1: Set $t = 0$; set the initial value of $p_i(t)$ for each node $i$.
2: repeat
3:   Compute the link $(i, j)$ price: $p_i(t)e_{ij}^s + p_j(t)e_{ji}^s$;
4:   Find the least cost path(s) from node $i$ to its sink;
5:   Calculate the new flow value $\sum_{j \in \text{out}(i)} x_{ij}^t$ by (2.35);
6:   Split the traffic among the least cost paths equally;
7:   Broadcast the new flow value to all the nodes in $SP_2(i)$;
8:   Update the price $p_i(t + 1)$ by (2.29);
9:   Broadcast the new price $p_i(t + 1)$ to the neighbors;
10: if the diminishing step update rule is used then
11:   update $\gamma(t + 1)$ by (2.30);
12: end if
13: Set $t = t + 1$ synchronously /* next iteration */;
14: until Convergence

can be implemented as fully distributed algorithms. The implementation is out of the scope of this paper and will be reported later. In a nutshell, the essential idea of the distributed algorithm is as follows. A bidirectional distance vector is propagated by the nodes. In one direction starting at the sinks, nodes broadcast upstream (in one single message) to their one hop neighbors messages (called price messages) that contain the cost of the least cost path to each sink. In the other direction, each node forwards to each neighbor $j$ a message (called traffic message) indicating for each sink, the level of aggregate traffic that will travel along neighbor $j$. When a node receives a price message it updates the price of the least cost path, calculates its new traffic according to (2.33) or (2.35) and broadcasts a price message and a traffic message. The details of the implementation and the study of the impact of this approach on the convergence of the practical algorithms are left for future work.

2.5 Stability of the Routing Algorithm

In this section, we study the relationship between traffic splitting among multiple least cost paths and the stability of the routing algorithm. In general it is hard to analyze this relationship for general networks, therefore, we consider a simple ring network topology to illustrate the necessity of splitting the traffic. Our analysis is related to that of [34] which discuss a similar formulation of the cross layer TCP/IP protocols design.

We consider a ring network with $N$ sources and 1 common sink. The sink is denoted as node 0 and the sources are indexed from 1 to $N$ in a clockwise direction. Fig. 2.1 shows an example of such network with $N = 3$. All the links are assumed to
be bidirectional. We assume that the sink never generates outgoing traffic and only receives data from node 1 and N. All source start initially with the same energy $E$. The energy consumption for sending and receiving a bit are fixed and are the same for all the nodes; denote them $e_s$ and $e_r$ respectively. We first discuss the routing stability and traffic splitting for routing model II.

**Theorem 2.5.1** In routing model II, if the traffic from a source is not split among multiple least cost paths then there is no stable equilibrium for the ring network when $N$ is odd.

**Proof:** Since there is no traffic splitting, the routing algorithm is a pure single path routing. Source $i$ only routes the data in a clockwise or counterclockwise direction. It can be shown easily that the routing can be characterized by a parameter $r \in \{0...N\}$ where the nodes $1...r$ route the data in the counterclockwise direction and nodes $r + 1...N$ route the data in the clockwise direction in the optimum solution. Since all routes pass through nodes 1 and N, then these two nodes are energy bottleneck nodes. Nodes 1 and N consume all their energy before nodes $2...N - 1$ (assuming that $N \geq 3$).

When the weights and utility functions are the same, the optimization problem can be simplified to:

$$\max_{r \in \{0...N\}} \max_{x} \sum_{i=1}^{N} U(x_i)$$

subject to

$$\sum_{i=2}^{N-1} (e_r + e_s)x_i + e_sx_1\zeta(r) \leq E$$

$$\sum_{i=r+1}^{N-1} (e_r + e_s)x_i + e_sx_N\zeta(N - r) \leq E$$

(2.36)
where

\[ \zeta(r) = \begin{cases} 
0, & \text{if } r = 0 \\
1, & \text{otherwise}
\end{cases} \]  

(2.37)

We apply Lagrange multipliers \( p_1 \) to the first constraint and \( p_N \) to the second constraint and form the Lagrangian:

\[
L_{sp}(r, x, p_1, p_N) = \sum_{i=1}^{N} U(x_i) \\
+ p_1 \left\{ E - \sum_{i=2}^{N-1} (e_r + e_s)x_i - e_s x_1 \zeta(r) \right\} \\
+ p_N \left\{ E - \sum_{i=r+1}^{N-1} (e_r + e_s)x_i - e_s x_N \zeta(N - r) \right\}.
\]

(2.38)

The dual function is \( D_{sp}(p_1, p_N) = \max_{r,x} L_{sp}(r, x, p_1, p_N) \), and the dual problem writes:

\[
\begin{align*}
\text{minimize} & \quad D_{sp}(p_1, p_N) \\
\text{subject to} & \quad p_1 \geq 0, \quad p_N \geq 0.
\end{align*}
\]

(2.39)

For given \( r, p_1 \) and \( p_N \), the optimum \( x \) can be found by:

1. If \( r = 0 \)
   \[
   \begin{align*}
x_i & = U'(p_N(e_s + e_r)) \quad \forall i = 1...N - 1 \\
x_N & = U'(p_N e_s)
   \end{align*}
   \]
   (2.40)

2. If \( 1 \leq r \leq N - 1 \),
   \[
   \begin{align*}
x_1 & = U'(p_1 e_s) \\
x_i & = U'(p_1(e_s + e_r)) \quad \forall i = 2...r \\
x_i & = U'(p_N(e_s + e_r)) \quad \forall i = r + 1...N - 1 \\
x_N & = U'(p_N e_s)
   \end{align*}
   \]
   (2.41)

3. If \( r = N \),
   \[
   \begin{align*}
x_1 & = U'(p_1 e_s) \\
x_i & = U'(p_1(e_s + e_r)) \quad \forall i = 2...N
   \end{align*}
   \]
   (2.42)

Since \( U \) is strictly concave increasing and differentiable, to maximize \( x \) according to the above results, \( r \) can be evaluated by:

1. If \( p_N > p_1 \) and \( p_N e_s \geq p_1(e_s + e_r) \), then \( r = N \);
2. If \( p_N < p_1 \) and \( p_1 e_s \geq p_N(e_s + e_r) \), then \( r = 0 \);

3. If \( p_N > p_1 \) and \( p_N e_s \leq p_1(e_s + e_r) \), then \( r = N - 1 \);

4. If \( p_N < p_1 \) and \( p_1 e_s \leq p_N(e_s + e_r) \), then \( r = 1 \);

5. If \( p_N = p_1 \), then \( r \in 1...N - 1 \).

Given these optimal solutions, we can now show that stable equilibrium does not exist in all the above cases of \( p_1 \) and \( p_N \).

For the first and second cases, it can be proved easily that the optimum \( x \) does not exist when \( r = N \) or \( r = 0 \) since at least \( x_N \) or \( x_1 \) can be improved without decreasing other components of the vector \( x \) by changing the routing direction.

For the third case, we show the absence of stable equilibrium by contradiction. Assume that stable equilibrium exists, at optimality both constraints in (3.5) should be active (all the energy of the bottlenecks must be completely consumed). That is:

\[
(N - 2)(e_s + e_r)U'(p_1(e_s + e_r)) + e_s U''(p_1 e_s) = E \\
\]

which implies that

\[
U''(p_N e_s) - U''(p_1 e_s) = \{(N - 2)(e_s + e_r)U''(p_1 e_s)\} / e_s > 0 \tag{2.44}
\]

However, as \( U \) is strictly concave increasing and differentiable, \( U''(p_N e_s) \) must be smaller than \( U''(p_1 e_s) \) if \( p_N > p_1 \). Contradiction occurs and thus stable equilibrium does not exist.

For case four the same approach can be used.

Finally, in case five, a stable equilibrium does not exist because the volumes of traffic flow through the two bottlenecks will never be the same because \( N \) is odd. When the prices are updated, \( p_1 \) and \( p_N \) will be different and we will fall back into cases (1) to (4).

Consider now routing model I. Despite, the fact that the source nodes split their traffic among their neighbors, the same problem of instability still occurs, though not
in the simple ring topology. Indeed in the ring scenario, each source will use both the clockwise and the anticlockwise directions simultaneously. However, we can easily construct a scenario where the instability exists. For this purpose, we consider the previous ring network with 3 sources and one sink and add one more source (node 4) to this ring network and anchor it to source 2 to form a lollipop network topology as shown in Fig. 2.2.

**Theorem 2.5.2** If the source nodes do not split their traffic among multiple least cost paths in the lollipop network topology of Fig. 2.2, then a stable equilibrium does not exist for the optimization problem of routing model I.

**Proof:** Assume $p_i(t)$ is the price of node $i$ at time $t$. We now show that stable equilibrium does not exist in the following cases:

1. $p_1(t) = p_3(t)$;
2. $p_1(t) > p_3(t)$;
3. $p_1(t) < p_3(t)$.

For the first case, when $p_1(t) = p_3(t)$, there are two equal least cost paths for source 4. Without losing generality, we assume that source 4 chooses a counterclockwise
direction to route the traffic, the traffic that passes through node 3 will be larger than node 1. At time \( t + 1 \), \( p_3(t + 1) > p_1(t + 1) \). Source 4 will choose the clockwise direction to route the data as the path cost of the clockwise direction is less than that of the counterclockwise direction. Therefore, there is no stable equilibrium when the price is the same.

For the second case \( p_1(t) > p_3(t) \), source 4 routes its data through node 3 to the sink. If stable equilibrium exists, it is easy to see that both node 1 and node 3 consume all their energy at optimality and have a strictly positive price. We denote \( x_i^+ \) as the volume of traffic source \( i \) generates in a clockwise direction and \( x_i^- \) as the volume of traffic of source \( i \) generates in a counterclockwise direction. The following condition must be satisfied:

\[
(x_1^+ + x_2^+ + x_4^+)(e_s + e_r) + x_3^+ e_s = E \\
(x_2^- + x_3^-)(e_s + e_r) + x_1^- e_s = E
\]  

(2.45)

where the first condition means all the energy in node 3 must be consumed and the second condition means all the energy in node 1 must be consumed at optimality. Similar to the analysis for routing model two, \( x_i^+ \) and \( x_i^- \) can be evaluated by the formula below:

\[
x_1^+ = U'(p_1 e_s + p_2 e_r + p_2 e_s + p_3 e_r + p_3 e_s) \\
x_1^- = U'(p_1 e_s) \\
x_2^+ = U'(p_2 e_s + p_3 e_r + p_3 e_s) \\
x_2^- = U'(p_2 e_s + p_1 e_r + p_1 e_s) \\
x_3^+ = U'(p_3 e_s) \\
x_3^- = U'(p_3 e_s + p_2 e_r + p_2 e_s + p_1 e_r + p_1 e_s) \\
x_4^+ = U'(p_4 e_s + p_2 e_r + p_2 e_s + p_3 e_r + p_3 e_s) \\
x_4^- = 0
\]  

(2.46)

If \( p_1(t) > p_3(t) \) and \( U \) is strictly concave increasing and differentiable, we have the following relations:

\[
x_1^+ > x_3^- \\
x_2^+ > x_2^- \\
x_3^+ > x_1^-
\]  

(2.47)

which implies

\[
(x_1^+ + x_2^+ + x_4^+)(e_s + e_r) + x_3^+ e_s > (x_2^- + x_3^-)(e_s + e_r) + x_1^- e_s
\]  

(2.48)

This contradicts condition (2.45) and completes the proof of case two.

For the last case, it can be proved similarly by making some simple modifications of the proof for case two. We skip the details here.
2.6 Numerical Results

To illustrate the behavior of our algorithms, we first start with the simple example of Fig. 1.1 where the topology is simple and predictable enough to allow us to study and understand the effect of applying different utility functions and adjusting the weights of the links to the data flow. In addition to this simple topology, we also use a more realistic random topology to study the scalability of the algorithm.

2.6.1 Simple Network Example

The topology in Fig. 1.1 is assumed to be fixed and the sending and receiving costs are uniform and equal to 2\,J/\text{byte} and 1\,J/\text{byte} respectively. Each source node starts with 10000\,J initial energy. A constant step size is used. The following known [24] concave utility functions are used:

Routing Model I:

\[
U_{ij}(x_{ij}^i) = \begin{cases} 
\log x_{ij}^i, & \text{if } k = 1 \\
(1 - k)^{-1}x_{ij}^{1-k}, & \text{otherwise}
\end{cases}
\] (2.49)

Routing Model II:

\[
U_i(\sum_{j \in \text{out}(i)} x_{ij}^i) = \begin{cases} 
\log(\sum_{j \in \text{out}(i)} x_{ij}^i), & \text{if } k = 1 \\
(1 - k)^{-1}(\sum_{j \in \text{out}(i)} x_{ij}^{1-k}, & \text{otherwise}
\end{cases}
\] (2.50)

1. Routing model I:

Fig. 2.3 shows the individual flow value under different utility functions. We can see that nodes 4 and 6 obtain the highest flow values in all cases. This is because i) both nodes 4 and 6 have two routes to their sink and ii) node 6 has one direct link to its sink. The figure also shows that larger values of \(k\) lead to smaller flow values for bottleneck nodes 5 and 6 and more fairness. In addition, nodes 2 and 4 gain a larger portion of energy in the bottleneck nodes. Fig. 2.4 shows the aggregate flow value under different utility functions. The flow value drops when \(k\) increases and the algorithm converges within 40 iterations in all cases. Fig. 2.5 and Fig. 2.6 show the impact of changing the weights \(w_{ij}\) on the individual flow and the aggregate flow respectively. We compare three
Figure 2.3: Individual flow value under different utility functions (Simple example, routing model I)

Figure 2.4: Aggregate flow value under different utility functions (Simple example, routing model I)
Figure 2.5: Individual flow value under different weight adjustments (Simple example, routing model I)

Figure 2.6: Aggregate flow value under different weight adjustments (Simple example, routing model I)
weight assignments, in which we bias the flow assignment against nodes that have direct links to their sinks. This is mainly because these nodes’ data do not incur the energy cost of reception at the sinks and thus often the nodes use up a large portion of their own energy for their own data. In the first setting, all the weights are equal to 1; in the second setting, the weights of the direct links from sources to their corresponding sinks are equal to 0.6 and the rest of the links are equal to 1; and, finally in the third setting, the weights of the direct links from sources to their corresponding sinks are equal to 0.3 and the rest of the links are equal to 1. The figures show that the smaller the weights of the direct links, the smaller the flow value of the bottleneck nodes and the larger the flow value of other nodes (i.e., better fairness). Naturally, the total flow value decreases when the weights of the direct links decrease.

2. Routing model II:

Fig. 2.7 shows the individual flow value under different utility functions. We can see that nodes 5 and 6 get the highest flow values in all cases. This is because nodes 5 and 6 are closer to their sinks than nodes 1 to 4 and they only incur the cost of sending since they are directly connected to the sinks. The figure also shows that larger values of $k$ lead to (more fairness) smaller flow values for nodes 5 and 6. In addition, nodes 1 to 4 gain a larger portion of energy in the bottleneck nodes. Fig. 2.8 shows the aggregate flow value under different utility functions. As we can see, the flow values are nearly the same and the algorithm converges within 60 iterations in all cases. Fig. 2.9 and Fig. 2.10 show the impact of changing the weights $w_i$ on the individual flow and the aggregate flow respectively. We compare the three previous weight assignments, and draw the same conclusions as for routing model I.

2.6.2 Random Network Example

To illustrate the scalability of the algorithm, we use the topology of Fig. 2.11 where 15 nodes randomly distributed over a square area of 50m x 50m. There are 13 sources and 2 sinks. Each source forwards its traffic to the closest sink (in number of hops, ties are broken randomly). Each source has 10000J initial energy and a maximum communication range of 15m. The energy consumption model of [37] is used: sending
Figure 2.7: Comparison of Individual flow value for different utility functions (Simple example, routing model II)

Figure 2.8: Aggregate flow value for different utility functions (Simple example, routing model II)
Figure 2.9: Comparison of Individual flow value for different weight adjustments (Simple example, routing model II)

Figure 2.10: Aggregate flow value for different weight adjustments (Simple example, routing model II)
1 byte from a node $i$ to a node $j$ separated by a distance $d$ meters costs $e_{ij}^s = c_1 + c_2d^2$ and receiving 1 byte costs $e_{ji}^r = c_1$, with $c_1 = 400\,\text{nJ/byte}$ and $c_2 = 800\,\text{pJ/byte/m}^2$.

The diminishing update rule for the step size is used. In routing model I, $\gamma(0)$ is set to $10^{-7}$ and the initial price $p$ for all sources is 0.0005. In routing model II, $\gamma(0)$ is set to $0.5 \times 10^{-8}$ and the initial price $p$ for all sources are 0.0001. Fig. 2.12 compares the individual flow values between the two routing models. It is clear that the throughput achieved with single path routing is much higher than that achieved with multi-path routing. Fig. 2.13 shows the number of routes of each node in the optimal solutions. As we mentioned before, in the case of link failure, routing model II is preferable as it always ends up with multiple routes if possible. Fig. 2.14 and Fig. 2.15 shows the total flow value of both models. The total flow value of routing model II is nearly twice that of routing model I. Both figures confirm the convergence of the algorithm within a small number of iterations (50 iterations).
Figure 2.12: Comparison of Individual flow value (Random example)

Figure 2.13: Number of routes (Random example)
Figure 2.14: Aggregate flow value (Random example, routing model 1)

Figure 2.15: Aggregate flow value (Random example, routing model 2)
CHAPTER 3

FAIR DATA COLLECTION BASED ON PROXIMAL OPTIMIZATION METHOD

3.1 Introduction

In the last chapter, we have proposed a sub-gradient routing algorithm. The algorithm requires synchronization among different sensors such that the optimal operation point is reached. However, such synchronization is hard to achieve in a large scale WSN. It may also induce a huge overhead.

In order to derive an asynchronous algorithm, we consider another optimization method which is called the proximal optimization method. Similar to the sub-gradient optimization method, the proximal optimization method is designed to solve the non-differentiable optimization problem. The core of the proposed method is indeed a gradient descent method which is well studied for asynchronous implementation. In this chapter, we re-formulate the optimization problem and derive a synchronization algorithm based on the proximal optimization method. By considering the asynchronous properties of the algorithm, we derive an asynchronous algorithm which results in a simpler implementation than the sub-gradient algorithm for practical real system. Numerical and simulation results are used to illustrate the convergence of both versions of the algorithms [12].

The remainder of this chapter is structured as follows. In Section 3.2, we present the system model. We formulate the problem in section 3.3, and present the proximal optimization algorithm in section 3.4. In section 3.5, we extend the synchronous algorithm into an asynchronous algorithm. Numerical and simulation results are presented in section 3.6.

3.2 The System Model

We model the wireless sensor network as a directed graph $G(\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ is the set of all nodes and $\mathcal{L}$ is the set of all directed links $(i, j), i, j \in \mathcal{N}$. The network contains
a set \( T \) of sink nodes \( T \subset N \), that are assumed to only receive data without generating any, and that have an unlimited supply of power. Furthermore, each sensor node in the set \( S = N \setminus T \) is associated with exactly one corresponding sink in the set \( T \). The initial amount of battery energy contained in node \( i \) is denoted by \( E_i \). A path or a route is a subset of nodes. In most WSNs applications, the sensors are static and the network topology is fixed. A source can find the set of all possible routes towards sink by source routing. The set of paths that connect source \( s \) and its corresponding sink is defined as \( R(s) \). \( R \) is the union of the set \( R(s) \) for all the \( s \in N \). The source of the route \( r \) is defined as \( src(r) \).

Given a node \( i \), let \( f(i) \) be the set of paths that contain node \( i \) and \( f(i) \subseteq R \). Given a route \( r \), we denote the set of nodes that \( r \) contains be \( N(r) \).

Given a route \( r \) and a node \( i \), let \( su(r,i) \) be the successor of node \( i \) and \( pd(r,i) \) be the predecessor of node \( i \). If node \( i \) is the starting node of path \( r \), then \( pd(r,i) \) is -1. Let \( e_{i,j}^s \) be the energy cost for node \( i \) to send one byte to node \( j \) and let \( e_{i,j}^r \) be the energy cost for node \( i \) to receive one byte from node \( j \). We define \( e_{i,-1}^r \) as 0. The flow volume or the flow value which of the path \( r \) is denoted as \( x_r \) and let the vector \( x = \{ x_r \} \) be the set of flow volumes. In the rest of the paper, we use flow volume and flow value interchangeably.

### 3.3 The Problem Formulation

The energy consumption for transmitting and receiving data for each sensor node \( i \) is limited by its initial amount of energy, i.e.,

\[
\sum_{r \in f(i)} e_{i,su(r,i)}^s x_r + \sum_{r \in f(i)} e_{pd(r,i),i}^r x_r \leq E_i, \quad \forall i \in S. \tag{3.1}
\]

Define an energy consumption matrix \( P \in \mathbb{R}^{\vert S \vert \times \vert R \vert} \) whose elements \( p_{ir} \) are

\[
p_{ir} = \begin{cases} e_{i,su(r,i)}^s + e_{pd(r,i),i}^r, & \text{if } r \in f(i) \\ 0, & \text{otherwise}, \end{cases} \tag{3.2}
\]

and let \( E = \{ E_i \} \) be the initial energy vector. The energy constraints for all the nodes (excluding the sinks) can be written compactly as

\[
P x \leq E. \tag{3.3}
\]
The fair data collection problem can be formulated as:

$$\mathbf{P}_1 : \max_x \sum_{s \in S} w_s U_s \left( \sum_{r \in R(s)} x_r \right)$$

subject to

$$\mathbf{P} x \leq E,$$

$$x_r \geq 0 \quad r \in R$$  \hspace{1cm} (3.4)

where $w_s$ is a positive constant. If a common utility function is used for all the flows, the weights $w_s$ can be used to further differentiate the sensor nodes. The larger the value of $w_s$ the more traffic node $s$ can send.

In this chapter, there are several assumptions for the utility functions and the primal problem:

A1: We assume that the utility functions $U_s$ are increasing, strictly concave and twice continuously differentiable in the interval $[0, \infty]$.

A2: $-U''_s(x) \geq \alpha > 0$ for $x \in [0, \infty]$.

A3: There exists a feasible vector $\bar{x}$ such that $\mathbf{P} \bar{x} \leq E$ and $\|\bar{x}\|_n < \infty$.

In the literature of bandwidth sharing in Internet [20] [24], a similar optimization problem which is based on fixed single path routing is studied and some fairness principles such as max-min fairness and proportional fairness are discussed. However, the traffic pattern between bandwidth sharing in Internet and fair data collection in WSNs is different. In Internet, the traffic between hosts is usually one to one and the traffic direction is most likely non-uniform. The bandwidth bottlenecks are located randomly over the network. In WSNs, the traffic pattern is many to one. Most of the sensors share one common destination. If the sensors have the same initial amount of energy, the bottlenecks are expected to be located near the sinks.

The simplest example is a linear topology which contains $N + 1$ nodes, i.e. $1 \rightarrow 2 \rightarrow \ldots \rightarrow N \rightarrow N + 1$. Nodes 1 to $N$ are the sources with the same initial energy $E$ and share the common the sink $N + 1$. We assume the transmission and reception costs are fixed and denoted as $e_s$ and $e_r$ respectively. The traffic generated by node $i$ is denoted as $x_i$. Since all the traffic has to pass through node $N$, the fair data collection problem can be simplified as

$$\max \sum_{i=1}^{N} w_i U(x_i)$$

subject to

$$\sum_{i=1}^{N-1} (e_r + e_s) x_i + e_s x_N \leq E$$  \hspace{1cm} (3.5)
We apply the lagrange multiplier $\lambda$ to the constraint and form the the Lagrangian:

$$L(x, \lambda) = \sum_{i=1}^{N} w_i U(x_i) + \lambda \left\{ E - \sum_{i=1}^{N-1} (e_r + e_s) x_i + e_s x_N \right\}. \tag{3.6}$$

The dual function is $D(\lambda) = \max_x L(x, \lambda)$, and the dual problem is

$$\minimize \quad D(\lambda)$$
$$\text{subject to} \quad \lambda \geq 0. \tag{3.7}$$

It is easy to prove that the primal and dual optimum $(x^*, \lambda^*)$ is related by

$$x_i^* = U'^{-1}(\lambda^*(e_s + e_r)) \quad \forall i = 1...N - 1$$
$$x_N^* = U'^{-1}(\lambda^* e_s) \tag{3.8}$$

If a common utility function $\log(.)$ is used and $w_s$ are the same, all the nodes will share the same amount of energy. It implies that all the non-bottleneck nodes get equal flow volume. Since reception cost at the sinks is nil, the bottleneck node is able to send more than the others. If the following known concave utility functions [24] are used

$$U_s\left( \sum_{r \in R(s)} x_r \right) = (1 - k)^{-1} \left( \sum_{r \in R(s)} x_r \right)^{1-k}, k = 2, 3.. \tag{3.9}$$

The flow value of the non-bottleneck nodes will increase and the bottleneck nodes will decrease when $k$ increases.

In last chapter, the mathematical formulation is described with the notation of link and node to describe the optimization problem. In this chapter, the formulation is based on the path and node to describe the problem. Although the notation is different, it is easy to see that our problem formulation for the proximal optimization method is equal to the problem formulation of the routing model II of last chapter. That just shows that how the same problem can be represented in different ways. Therefore, if both models are used for the same network topology, we expect that optimal solutions for both models will be the same.

### 3.4 Proximal Optimization Algorithm

To solve the optimization problem, it requires global information exchange between all the nodes in the network. In order to avoid this, we consider the Lagrange relaxation
and the dual problem. However, the variable $x_r$ in the problem (3.4) is not strictly concave. The dual function is non-differentiable and the gradient descent method cannot be applied. To address the non-differentiable issue, we use the Proximal Optimization Algorithm in [3]. We add a concave term for each variable to the objective function. That is:

$$
P'_1 : \max_x \sum_{s \in S} w_s U_s \left( \sum_{r \in R(s)} x_r \right) - \frac{1}{2\beta} \| x - y \|_2^2$$

subject to $Px \leq E$, $x_r \geq 0, r \in R$, $y \in \mathbb{R}^{|R|}$

(3.10)

where $\beta$ is any constant that larger than 0. We will discuss the role of $\beta$ in the later subsection.

### 3.4.1 Proximal Optimization Algorithm

For $t = 1, 2, \ldots$ do:

*Step1* Fix $y = y(t)$ and solve $P'_1$, obtain the new $x(t + 1)$. Since the objective function in $P'_1$ is strictly concave, such $x(t + 1)$ exists and is unique.

*Step2* Set $y(t + 1) = x(t + 1)$

**Remark:** We assume that $y(1)$ is any feasible point, i.e. $Py(1) \leq E$.

The proof of the convergence of the algorithm can be found in [3]. From the description, we know that the proximal optimization algorithm is a two-tier algorithm. Step2 will be executed only when the optimal solution for Step1 has been found. Therefore, we can treat Step1 as the inner tier and Step2 as the outer tier of the algorithm.

### 3.4.2 The Dual Problem

In order to solve Step1, we relax the constraint optimization problem into an unconstrained optimization problem by Lagrange relaxation and solve it by the gradient pro-
jection method [3]. The Lagrangian \( L(\mathbf{x}, \lambda, \mathbf{y}) \) is given by

\[
L(\mathbf{x}, \lambda, \mathbf{y}) = \sum_{s \in S} w_s U_s \left( \sum_{r \in R(s)} x_r \right) - \frac{1}{2\beta} \| \mathbf{x} - \mathbf{y} \|_2^2 + \sum_{i \in S} \lambda_i \left\{ E_i - \sum_{r \in f(i)} p_{ir} x_r \right\}
\]

\[
= \sum_{s \in S} \left\{ w_s U_s \left( \sum_{r \in R(s)} x_r \right) - \sum_{r \in R(s)} x_r \left( \sum_{i \in N(r)} \lambda_i p_{ir} \right) \right\} - \frac{1}{2\beta} \| \mathbf{x} - \mathbf{y} \|_2^2 + \sum_{i \in S} \lambda_i E_i
\]

(3.11)

and we define \( L_s(\mathbf{x}, \lambda, \mathbf{y}) \) as

\[
L_s(\mathbf{x}, \lambda, \mathbf{y}) = w_s U_s \left( \sum_{r \in R(s)} x_r \right) - \sum_{r \in R(s)} x_r \left( \sum_{i \in N(r)} \lambda_i p_{ir} \right) - \frac{1}{2\beta} \sum_{r \in R(s)} (x_r - y_r)^2 + \lambda_s E_s
\]

(3.12)

and \( L(\mathbf{x}, \lambda, \mathbf{y}) = \sum_{s \in S} L_s(\mathbf{x}, \lambda, \mathbf{y}) \). The dual function \( D(\lambda, \mathbf{y}) \) of problem \( P'_1 \) is defined as

\[
D(\lambda, \mathbf{y}) = \max_{\mathbf{x} \succeq 0} L(\mathbf{x}, \lambda, \mathbf{y})
\]

(3.13)

and the corresponding dual problem \( \mathcal{D} \) is

\[
\mathcal{D} : \quad \min_{\lambda} \quad D(\lambda, \mathbf{y})
\]

s.t. \( \lambda \succeq 0 \).

(3.14)

(3.15)

We sometimes overload the notation by substituting \( D(\lambda) \) to \( D(\lambda, \mathbf{y}) \) as \( \mathbf{y} \) can be treated as a constant during the execution of the gradient projection algorithm. Since the dual function now is differentiable, the gradient of the price \( \lambda_i \) can be computed by

\[
\frac{\partial D(\lambda, \mathbf{y})}{\partial \lambda_i} = E_i - \sum_{r \in f(i)} p_{ir} x_r.
\]

(3.16)

The optimal dual variable \( \lambda^* \) can be found by the gradient descent method. In each iteration, the dual variable is updated as below

\[
\lambda_i(t + 1) = \max(\lambda_i(t) - \gamma \nabla D_i(\lambda, \mathbf{y}), 0)
\]

(3.17)

where \( \gamma \) is a positive scalar step size.
Note that when optimality is reached, the vector $y$ is equal to the optimal flow volume $x^*$. Therefore, the quadratic terms in the Lagrangian vanish. Given the optimal price vector $\lambda^*$, the Lagrangian becomes

$$
\sum_{s \in S} \left\{ w_s U_s \left( \sum_{r \in R(s)} x^*_r - \sum_{r \in R(s)} \sum_{i \in N(r)} \lambda^*_i p_{ir} \right) + \lambda_s E_s \right\}.
$$

(3.18)

Equation (3.18) can be regarded as a demand and supply problem. Each source $s$ tries to maximize the revenue $w_s U_s \left( \sum_{r \in R(s)} x^*_r \right)$ with the cost $\sum_{r \in R(s)} \sum_{i \in N(r)} \lambda^*_i p_{ir}$. Since the path cost is linear in term of the flow volume, only the set of least cost paths will be consider. For those non-least cost paths, their flow values are set to zero.

### 3.4.3 Individual Flow Volume Calculation

Given a fixed vector $\lambda$ and $y$, each source has to solve the problem

$$
\arg\max_{x \geq 0} L_s(x, \lambda, y).
$$

(3.19)

The maximum point $\bar{x}$ should fulfill the below condition

$$
\frac{\partial L_s(\bar{x}, \lambda, y)}{\partial x_l} = 0
$$

(3.20)

where

$$
\frac{\partial L_s(x, \lambda, y)}{\partial x_l} = w_s U'_s \left( \sum_{r \in R(s)} x_r \right) - \sum_{i \in N(l)} \lambda_i p_{il} - \frac{1}{\beta} (x_l - y_l).
$$

(3.21)

Therefore, by summing over the equation (3.20) of all the outgoing traffic of the source $s$, we have

$$
0 = |R(s)| w_s U'_s \left( \sum_{r \in R(s)} \bar{x}_r \right) - \sum_{r \in R(s)} \sum_{i \in N(r)} \lambda_i p_{ir}
$$

$$
- \frac{1}{\beta} \sum_{r \in R(s)} (\bar{x}_r - y_r).
$$

(3.22)

If we substitute $\bar{X}_s = \sum_{r \in R(s)} \bar{x}_r$, we get the following differential equation

$$
\frac{|R(s)| w_s U'_s (\bar{X}_s)}{\beta} - \frac{\bar{X}_s}{\beta} = \sum_{r \in R(s)} \sum_{i \in N(r)} \lambda_i p_{ir} - \sum_{r \in R(s)} \frac{y_r}{\beta}.
$$

(3.23)
Through solving the differential equation, we can obtain the $\bar{X}_s$. Given the price vector $\lambda$ and vector $y$, we denote $\bar{X}_s(\lambda, y)$ as the solution of the equation (3.23). Injecting this back into $\nabla L_s(\bar{x}, \lambda, y)=0$, we can get the individual $\bar{x}_r$ for all $r \in R(s)$ as

$$\bar{x}_r(\lambda, y) = \beta \{ w_s U'(\bar{X}_s(\lambda, y)) - \sum_{i \in N(r)} \lambda_i p_{ir} \} + y_r. \quad (3.24)$$

The computed $\bar{x}_r$ can be negative. If some paths get negative flow values, we exclude the path which has the smallest flow value from the set $R(s)$ temporarily and sets its flow value to be zero. Then, we recalculate the total flow value $\bar{X}_s$ and individual $\bar{x}_r$ for the remaining paths. The algorithm keeps repeating until all the individual flow values are non-negative or no route is available. The algorithm is summarized in algorithm (3) and the correctness of the algorithm can be proved by some simple modifications of the proof in [19]. We sometimes overload the notation by substituting $x_r(\lambda)$ to $x_r(\lambda, y)$ and $X_s(\lambda)$ to $X_s(\lambda, y)$ due to the same reason as $D(x)$.

**Algorithm 3** Individual Flow Volume Calculation

1: Save $R(s)$: $R'(s) = R(s)$
2: repeat
3: Set $flag = false$
4: Solve the differential equation (3.23) and obtain $X^*_s$;
5: Get the individual flow volume $x_l$ by (3.24) for all $l \in R(s)$;
6: Sort the flow volumes;
7: if the smallest flow volume $x_{l'} < 0$ then
8: $R(s) = R(s) \setminus l'$;
9: $x_{l'} = 0$;
10: $flag = true$;
11: end if
12: until $flag == true$
13: Restore $R(s)$: $R(s) = R'(s)$

**Algorithm 4** The Gradient Projection Algorithm

1: Set $t = 0$; set the initial value of $x(0)$ and $\lambda(0)$.
2: for each node $s$ do
3: repeat
4: Communicate flow volumes $x_r(t)$ to nodes $n \in N(r)$ for all $r \in R(s)$;
5: Evaluate the price $\lambda_s(t+1)$ by (3.17);
6: Compute the cost of the link $(s, j)$: $\lambda_s(t)e_{ij} + \lambda_j(t)e_{ji}$ for all $(s, j) \in L$;
7: Communicate the cost of the link $(s, j)$ to all nodes that use link $(s, j)$
8: Evaluate new flow volumes $x_r(t+1)$ for all $r \in R(s)$;
9: Set $t = t + 1$ synchronously $/^{th}$next iteration$/$;
10: until Convergence
11: end for
3.4.4 Convergence Analysis

In this section, we show that the gradient projection method for step 1 in the previous section can converge if the step size is within certain range. We first summarize the gradient projection algorithm in algorithm (4).

Define $\bar{R}$ as the number of routes that pass through the most congested node, i.e. $\bar{R} = \max \{|f(i)|\}$, and $\bar{N}$ as the length of the longest route, i.e. $\bar{N} = \max \{|N(r)|\}$. Also, we define $e_{s_{max}}$ as the maximum transmitting energy consumption per byte, $e_{r_{max}}$ as the maximum receiving energy consumption per byte, and their sum $\bar{p} = e_{s_{max}} + e_{r_{max}}$. The maximum weight is defined as $\bar{w}$ which $\bar{w} = \max \{ w_s \}$.

**Theorem 3.4.1** For a fixed $y(t)$, consider the gradient projection method for step 1, if the step size satisfies $0 < \gamma < \frac{2(\beta^{-1} - \bar{w} \bar{\alpha})}{\bar{p} \bar{R} \bar{N}}$, the sequence of vector $x(t)$ and $\lambda(t)$ converges to the unique optimal points respectively.

**Proof:** In [3], the author shows that the gradient projection method for the dual problem converges if the dual function is lower bounded and fulfills the Lipschitz continuity property with a constant $K > 0$ which is

$$\|\nabla D(\lambda_1) - \nabla D(\lambda_2)\|_2 \leq K\|\lambda_1 - \lambda_2\|_2$$

(3.25)

and the step size $\gamma$ is within the range: $0 < \gamma < 2/K$. By the mean value theorem, there exists some $\xi \in [\lambda_1, \lambda_2]$, such that

$$\nabla D(\lambda_1) - \nabla D(\lambda_2) = \nabla^2 D(\xi)(\lambda_1 - \lambda_2)$$

(3.26)

and by the Cauchy Schwarz inequality, we know that

$$\|\nabla D(\lambda_1) - \nabla D(\lambda_2)\|_2 \leq \|\nabla^2 D(\xi)\|_2\|\lambda_1 - \lambda_2\|_2.$$ (3.27)

Therefore, if the second norm of the hessian matrix $H(\xi)$ is upper bounded, then $D(\lambda)$ has the Lipschitz continuity property. We first show how to find $H(\xi)$. From (3.16), $\nabla D(\xi) = E - P x(\xi)$ and

$$\nabla^2 D(\xi) = -P \frac{\partial x(\xi)}{\partial \xi}$$

(3.28)

where $\frac{\partial x(\xi)}{\partial \xi}$ is a $|R| \times |S|$ Jacobian matrix whose $(i, j)$ element is defined as

$$\frac{\partial x_i(\xi)}{\partial \xi_j} = \begin{cases} \frac{(e_{s_{su}(i,j)} + e_{rd(i,j)})(\sum_{l \in R(src(i))} x_l(\xi) - \frac{1}{\bar{p}})}{\bar{w} U_{src(i)}}, & \text{if } i \in f(j) \\ 0, & \text{otherwise}. \end{cases}$$

(3.29)
The hessian $H(\xi) \in \mathbb{R}^{|S| \times |S|}$ which element $h_{ij}(\xi)$ is equal to

$$
\begin{cases}
0, & \text{if } f(i) \cap f(j) = \emptyset \\
- \sum_{r \in f(i) \cap f(j)} \frac{1}{\bar{w}_{src(r)} \left( \sum_{l \in R(src(r))} x_l(\xi) \right) - \frac{1}{\beta}}, & \text{otherwise.}
\end{cases}
$$

(3.30)
i.e., $H(\xi) = PF(\xi)P^T$ where

$$
F(\xi) = -diag\left( \frac{1}{\bar{w}_{src(r)} \left( \sum_{l \in R(src(r))} x_l(\xi) \right) - \frac{1}{\beta}} \right), r \in \mathcal{R}).
$$

(3.31)

Since we assume that the $-U''_s(x) \geq \alpha > 0$, $\|H(\xi)\|_2$ is upper bounded and symmetric. The upper bound can be found by the inequality below [3]

$$
\|H(\xi)\|_2 \leq \sqrt{\|H(\xi)\|_1 \|H(\xi)\|_\infty} = \|H(\xi)\|_1 \text{ or } \|H(\xi)\|_\infty.
$$

(3.32)
as $H(\xi)$ is symmetric where

$$
\|H(\xi)\|_1 = \max_j \sum_{i=1}^{|S|} h_{ij}(\xi) \quad \text{and}
$$

$$
\|H(\xi)\|_\infty = \max_i \sum_{j=1}^{|S|} h_{ij}(\xi).
$$

(3.33)

It can be proved easily that

$$
\|H(\xi)\|_1 \text{ or } \|H(\xi)\|_\infty \leq \frac{\bar{R} \bar{N} \bar{p}^2}{\beta - 1 - \bar{w}\alpha}.
$$

(3.34)

By the assumption of the utility functions and the proof above, the objective function is lower bounded with the Lipschitz continuous property. According to the proposition 3.4 of [3], if the step size $\gamma$ is within the range $0 < \gamma < \frac{2(\beta^{-1} - \bar{w}\alpha)}{\bar{R} \bar{N} \bar{p}^2}$, then every limit point of the sequence $\{\lambda(t)\}$ is a dual optimal solution.

The rest of the proof are similar to the one in [20]. Due to the non-negativity constraints of $\lambda$ and the feasibility assumptions A3 of the primal problem, the set $\Omega^*$ of the optimal price is bounded and the level set $\Omega = \{\lambda \geq 0, D(\lambda) \leq D(0)\}$ is compact. The sequence of $\{D(\lambda(t))\}$ is decreasing and $\lambda(t)$ is in the set $\Omega$ if the step size is
within the range $0 < \gamma < \frac{2(\beta^{-1} - \bar{w} \alpha)}{R \bar{N} \bar{p}^2}$, the sequence $\{\lambda(t)\}$ must tend to at least one limit point. Since $x(\lambda(t))$ is a continuous function of $\lambda$. We have the following result

$$\lim_{t \to \infty} x(\lambda(t)) = x(\lim_{t \to \infty} \lambda(t)) = x(\lambda^*) = x^*. \quad (3.35)$$

This completes the proof.

3.4.5 The Role of the Parameter $\beta$

In general, a large $\beta$ implies that a smaller number of iterations of the proximal algorithm (i.e. the number of step1 and step2 executed) are needed to reach the optimal solution. The intuition is that a large $\beta$ makes the quadratic term more "blunt" and the algorithm advances at a faster speed to the optimal [3]. However, as we have proved, a large $\beta$ requires a smaller step size $\gamma$ in the gradient projection method for step2 such that the convergence is guaranteed and thus the speed of convergence of the projection method is reduced.

3.4.6 Enhancement of the Rate of Convergence

The rate of the convergence of the gradient project method can be speeded up by making use of the hessian matrix $H$. The Newton method which the the gradient is scaled up by the hessian matrix converges faster than the gradient projection method [2]. The price updating rule is (in matrix form)

$$\lambda(t + 1) = \max(\lambda(t) - \gamma[\nabla^2 D(\lambda, y)]^{-1} \nabla D(\lambda, y), 0). \quad (3.36)$$

However, the computation of the hessian matrix requires global information exchange. We then approximate the scale matrix by only keeping the diagonal terms of the hessian matrix. For each node $i$, it updates the price by

$$\lambda_i(t + 1) = \max(\lambda_i(t) - \gamma[H(\lambda)_i]^{-1} \nabla D(\lambda, y), 0). \quad (3.37)$$

3.5 The Asynchronous Algorithm

In the last section, we have described the proximal optimization algorithm for the fair data collection problem. It assumes that all the sensor are synchronized. The
updates of prices and flow volume at different sensors occur at times \( t = 1, 2, \ldots \). Such synchronization is hard to achieve in real WSNs. In this section, we propose an asynchronous algorithm where the prices and flow volumes are updated at different times at different sensors. The synchronous proximal optimization algorithm is a two-tier algorithm. We first show a partial asynchronous model for the inner tier of the algorithm (i.e. Step1). We next show how to relax the synchronization requirement of the outer layer of the algorithm (i.e. Step2). In order to evaluate the flow volumes given the prices vector, the utility function in the differential equation (3.23) has to be specified. In the rest of this section, we assume all the sensors share the common utility function \( \log(\cdot) \) as an example. We also assume that only the paths in the set \( R'(s) \) carry non-zero flow value and the paths in the set \( R(s) \setminus R'(s) \) carry zero flow. The set \( R'(s) \) is obtained by the individual flow value calculation algorithm in section 3.4.3. The differential equation can then be simplified to a quadratic equation

\[
X_s^2(t) + \left\{ \sum_{l \in R'(s)} \sum_{i \in N(r)} \lambda_i(t)p_{ir}\beta \right. \\
- \left. \sum_{r \in R'(s)} y_r \right\} X_s(t) - |R'(s)|\beta w_s = 0. 
\]  

(3.38)

The roots of the above quadratic equation can be found by

\[
X_s(t) = \frac{-b_s(t) \pm \sqrt{b_s^2(t) - 4c_s}}{2}
\]

(3.39)

where

\[
b_s(t) = \sum_{r \in R'(s)} \sum_{i \in N(r)} \lambda_i(t)p_{ir}\beta - \sum_{r \in R'(s)} y_r,
\]

\[
c_s = -|R'(s)|\beta w_s.
\]

(3.40)

Since \(|R'(s)|, \beta \) and \( w_s \) are always positive, \( \sqrt{b_s^2(t) - 4c_s} > 0 \) and \( \sqrt{b_s^2(t) - 4c_s} > b_s(t) \), two real distinct roots exist of which one is always positive and other is always negative. Due to the non-negative constraint of the total outing traffic of a source, \( X_s \) is equal to

\[
X_s(t) = \frac{-b_s(t) + \sqrt{b_s^2(t) - 4c_s}}{2}
\]

(3.41)
3.5.1 The Asynchronous Algorithm for Step1

In step1 where the vector $y(t)$ is fixed, each sensor has to solve the dual problem by the gradient projection method. In order to achieve asynchronism, we adopted the partial asynchronous model in [20] and derive an asynchronous gradient projection algorithm.

Let $T_s^λ = 1, 2, ...$ be the set of time instances which sensor $s$ updates its price. We assume that at times $t ∉ T_s^λ$, the price of sensor $s$ is unchanged. Also, let $T_s^x = 1, 2, ...$ be the set of time instances which sensor $s$ updates the flow volumes of its outgoing routes. At time $t ∉ T_s^x$, $x_r$ is unchanged $∀r ∈ R(s)$. We further assume that:

A4: The consecutive updates of prices and flow volumes are both bounded by $t_d$, and the one-way communication delay between two nodes is at most $t_d$.

A5: The constant $β$ is lower bounded by $\frac{1}{4}$, i.e. $β ≥ \frac{1}{4}$.

In the asynchronous environment, a node $i$ may not be able to get the exact flow volume information of other nodes which passes through node $i$. It estimates the gradient $g_i(t)$ by a weighted average of recent flow volume information

$$g_i(t) = E_i - \sum_{r ∈ f(i)} p_r \hat{x}_r(t)$$  \hspace{1cm} (3.42)

and updates its price by

$$\lambda_i(t + 1) = \max\{\lambda_i(t) - \gamma g_i(t), 0\}$$  \hspace{1cm} (3.43)

and at time $t ∉ T_s^λ$, $\lambda_i(t)$ is unchanged. The average flow volume $\hat{x}_r(t)$ is defined as

$$\hat{x}_r(t) = \sum_{τ = t - t_d}^{t} k(t, τ) \hat{x}_r(τ) \text{ where } \sum_{τ = t - t_d}^{t} k(t, τ) = 1.$$  \hspace{1cm} (3.44)

Similarly, a node $s$ may not be able to get the exact prices of other nodes that the traffic of node $s$ passes through. The individual $\hat{x}_r(t), ∀r ∈ R(s)$ is computed by estimated prices of other nodes. This price estimation $\hat{\lambda}_i(t)$ of a node $i$ is again a weighted average over the recent price information

$$\hat{\lambda}_i(t) = \sum_{τ = t - t_d}^{t} h(t, τ) \lambda_i(τ) \text{ where } \sum_{τ = t - t_d}^{t} h(t, τ) = 1.$$  \hspace{1cm} (3.45)
The $\tilde{x}_r(t)$ is defined as

$$
\tilde{x}_r(t) = \beta \left\{ w_s U'_s(\hat{X}_s(t)) - \sum_{i \in N(r)} \hat{\lambda}_i(t)p_{ir} \right\} + y_r,
$$

$$
\hat{X}_s(t) = -\hat{b}_s(t) + \sqrt{\hat{b}_s^2(t) - 4c_s},
$$

$$
\hat{b}_s(t) = \sum_{r \in R(s)} \sum_{i \in N(r)} \hat{\lambda}_i(t)p_{ir}\beta - \sum_{r \in R(s)} y_r.
$$

At time $t \not\in T^x_s$, $\tilde{x}_r(t)$ is unchanged $\forall r \in R(s)$.

Since the time elements in the set $T^\lambda_i$ and $T^x_i$ are not known to other nodes in the WSNs, no time synchronization is needed among different sensors. Moreover, when a node receives a sequence of prices or flow volumes update from other sensors, it does not need to know the exact order of the sequence. It only knows that all elements of the update sequence is within the delay bound $t_d$. The updates which are beyond the bound $t_d$ are discarded during the averaging operation.

We now present the convergence proof of the asynchronous gradient projection algorithm. We define $x_r(t)$ (of which the origin is sensor $s$, i.e. $src(r) = s$) to be an exact rate if it is calculated by the exact prices instead of the average of recent prices. That is

$$
x_r(t) = \beta \left\{ w_s U'_s(X_s(t)) - \sum_{i \in N(r)} \lambda_i(t)p_{ir} \right\} + y_r
$$

$$
X_s(t) = -b_s(t) + \sqrt{b_s^2(t) - 4c_s}
$$

$$
b_s(t) = \left\{ \sum_{r \in R(s)} \sum_{i \in N(r)} \lambda_i(t)p_{il}\beta - \sum_{r \in R(s)} y_r \right\}.
$$

The core of the proof is to show that the difference between the exact rate $x(t)$ and the averaging estimation $\hat{x}(t)$ will converge to zero when the time tends to infinity under the assumption A1 to A5. We then show that the exact rate is indeed the optimal solution.

**Theorem 3.5.1** Suppose the step size is small enough, under the assumption A1-A4, then starting from any initial flow volume $x(0)$ and $\lambda(0) \geq 0$, every limit point $(x^*, \lambda^*)$ of the sequence $(x(t), \lambda(t))$ generated by the asynchronous gradient projection method is primal dual optimal.
Proof: We first summarize the price and flow volume updating rules for the asynchronous gradient projection algorithm.

\[ \lambda_i(t + 1) = \max\{\lambda_i(t) - \gamma g_i(t), 0\} \]
\[ g_i(t) = E_i - \sum_{r \in f(i)} p_{ir} \hat{\lambda}_r(t) \]
\[ \hat{\lambda}_i(t) = \sum_{\tau = t - t_d}^{t} h(t, \tau) \lambda_i(t) \text{ where } \sum_{\tau = t - t_d}^{t} h(t, \tau) = 1 \]
\[ \hat{b}_s(t) = \sum_{r \in R(s)} \sum_{i \in N(r)} \hat{\lambda}_i(t)p_{ir}\beta - \sum_{r \in R(s)} y_r \]
\[ b_s(t) = \sum_{r \in R(s)} \sum_{i \in N(r)} \lambda_i(t)p_{ir}\beta - \sum_{r \in R(s)} y_r \]
\[ \hat{X}_s(t) = \frac{-\hat{b}_s(t) + \sqrt{\hat{b}_s^2(t) - 4c_s}}{2} \]
\[ \hat{X}_s(t) = \frac{-\hat{b}_s(t) + \sqrt{\hat{b}_s^2(t) - 4c_s}}{2} \]
\[ X_s(t) = \frac{-b_s(t) + \sqrt{b_s^2(t) - 4c_s}}{2} \]
\[ \bar{x}_r(t) = \beta \left\{ w_s U_s'(\hat{X}_s(t)) - \sum_{i \in N(r)} \hat{\lambda}_i(t)p_{ir} \right\} + y_r \]
\[ \bar{x}_r(t) = \sum_{\tau = t - t_d}^{t} k(t, \tau) \bar{x}_r(t) \text{ where } \sum_{\tau = t - t_d}^{t} k(t, \tau) = 1 \]
\[ x_r(t) = \beta \left\{ w_s U_s'(X_s(t)) - \sum_{i \in N(r)} \lambda_i(t)p_{ir} \right\} + y_r \]

We start the proof with some useful lemmas.

Lemma 3.5.1 Let \( \psi(t) = \lambda(t + 1) - \lambda(t) \) and

(a) For some constant \( C_1 \) and all \( t \),
\[ |\hat{X}_s(t) - X_s(t)| \leq C_1 \sum_{r \in R(s)} \sum_{i \in N(r)} \sum_{\tau' = t - t_d}^{t-1} |\psi_i(\tau')|, \]
(b) For some constant \( C_2 \) and all \( t \),
\[ |X_s(t + 1) - X_s(t)| \leq C_2 \sum_{r \in R(s)} \sum_{i \in N(r)} |\psi_i(t)|, \]
(c) For some constant \( C_3 \), and all \( t \),
\[ |U_s'(\hat{X}_s(t)) - U_s'(X_s(t))| \leq C_3 \sum_{r \in R(s)} \sum_{i \in N(r)} \sum_{\tau' = t - t_d}^{t-1} |\psi_i(\tau')|, \]
(d) For some constant $C_4$, and all $t$,
$$|U'_s(X_s(t + 1)) - U'_s(X_s(t))| \leq C_4 \sum_{r \in R(s)} \sum_{i \in N(r)} |\psi_i(t)|.$$  
(e) For some constant $C_5$, and all $t$,
$$|\hat{x}_r(t) - x_r(t)| \leq C_5 \sum_{l \in R(\text{src}(r))} \sum_{i \in N(l)} \sum_{\tau' = t - t_d}^{t - 1} |\psi_i(\tau')|.$$  
(f) For some constant $C_6$, and all $t$,
$$|x_r(t + 1) - x_r(t)| \leq C_6 \sum_{l \in R(\text{src}(r))} \sum_{i \in N(l)} |\psi_i(t)|.$$  
(g) For some constant $C_7$, and all $t$,
$$|\hat{x}_r(t) - x_r(t)| \leq C_7 \sum_{l \in R(\text{src}(r))} \sum_{i \in N(l)} \sum_{\tau' = t - 2t_d}^{t - 1} |\psi_i(\tau')|.$$

**Proof:**

(a)

$$\frac{|\hat{X}_s(t) - X_s(t)|}{\beta_s(t) + \sqrt{\hat{b}_s^2(t) - 4c_s} - \beta_s(t) + \sqrt{b_s^2(t) - 4c_s}} \leq \frac{\hat{b}_s(t) - b_s(t)}{2} + \frac{\sqrt{\hat{b}_s^2(t) - 4c_s} - \beta_s(t) + \sqrt{b_s^2(t) - 4c_s}}{2} \leq \frac{|\hat{b}_s(t) - b_s(t)|}{2} + \frac{|\hat{b}_s^2(t) - 4c_s - \beta_s^2(t) + 4c_s|}{2} \leq \left| \frac{\hat{b}_s(t) - b_s(t)}{2} + \frac{|\hat{b}_s^2(t) - 4c_s - \beta_s^2(t) + 4c_s|}{2} \right|$$  

where the last inequality comes from the fact that $|a^2 - b^2| - |a - b| \geq 0$ for some $a, b > 0$ if $|a + b| - 1 \geq 0$. Since we assume $\beta \geq \frac{1}{4}$, the condition is fulfilled. By this, we have,

$$\frac{|\hat{X}_s(t) - X_s(t)|}{\beta_s(t) + \sqrt{\hat{b}_s^2(t) - 4c_s} - \beta_s(t) + \sqrt{b_s^2(t) - 4c_s}} \leq \frac{|\hat{b}_s(t) - b_s(t)|}{2} \leq \frac{\hat{b}_s(t) - b_s(t)}{2} \leq \frac{\beta_s(t) + b_s(t)}{2} \leq \frac{1}{2}$$  

Due to the energy constraint of each node and feasibility assumption of the primal problem, price $\lambda$ is bounded and $\frac{\hat{b}_s(t) + b_s(t)}{2}$ is finite. Therefore, we can always find
a constant $C'_1 \geq \frac{1}{2} + \frac{\dot{b}_s(t) + b_s(t)}{2}$, such that

$$|\hat{X}_s(t) - X_s(t)| \leq C'_1|\dot{b}_s(t) - b_s(t)| = \beta C'_1 \sum_{r \in R(s)} \sum_{i \in N(r)} p_{ir} \left\{ \hat{\lambda}_i(t) - \lambda_i(t) \right\}$$

$$\leq \bar{p}\beta C'_1 \sum_{r \in R(s)} \sum_{i \in N(r)} \left\{ \max_{t-t_d \leq \tau \leq t} \lambda_i(\tau) - \lambda_i(t) \right\} \leq \bar{p}\beta C'_1 \sum_{r \in R(s)} \sum_{i \in N(r)} \sum_{\tau' = t-t_d}^{t-1} |\psi_i(\tau')|$$

(3.51)

(b)

$$|X_s(t+1) - X_s(t)|$$

$$= |\frac{-b_s(t+1) + \sqrt{b_s^2(t+1) - 4c_s}}{2} - \frac{-b_s(t) + \sqrt{b_s^2(t) - 4c_s}}{2}|$$

$$\leq |\frac{b_s(t+1) - b_s(t)}{2}| + |\frac{b_s^2(t+1) - 4c_s}{2} - \frac{b_s^2(t) - 4c_s}{2}|$$

$$\leq |\frac{b_s(t+1) - b_s(t)}{2}| + |\frac{b_s^2(t+1) - b_s^2(t)}{2}|$$

$$\leq b_s(t+1) - b_s(t)$$

where $C'_2 \geq \frac{1}{2} + \frac{b_s(t+1) + b_s(t)}{2}$

$$\leq C'_2 b_s(t+1) - b_s(t)$$

$$= \beta C'_2 \sum_{r \in R(s)} \sum_{i \in N(r)} p_{ir} \left\{ \lambda_i(t+1) - \lambda_i(t) \right\}$$

$$\leq \beta C'_2 \sum_{r \in R(s)} \sum_{i \in N(r)} |\psi_i(t)|$$

(3.52)

(c)

$$|U'_s(\hat{X}_s(t)) - U'_s(X_s(t))|$$

$$= \frac{1}{|R(s)|\beta} \left\{ X_s(t) - X_s(t) + \sum_{r \in R(s)} \sum_{i \in N(r)} p_{ir} \beta \left( \hat{\lambda}_i(t) - \lambda_i(t) \right) \right\}$$

$$\leq \beta C'_1 + \beta \frac{1}{|R(s)|\beta} \sum_{r \in R(s)} \sum_{i \in N(r)} \sum_{\tau' = t-t_d}^{t-1} p_{ir} |\psi_i(\tau')|$$

$$\leq \frac{(\beta C'_1 + \beta)\bar{p}}{|R(s)|\beta} \sum_{r \in R(s)} \sum_{i \in N(r)} \sum_{\tau' = t-t_d}^{t-1} |\psi_i(\tau')|$$

(3.53)
(d)

\[
|U'_s(X_s(t + 1)) - U'_s(X_s(t))| \\
= \frac{1}{|R(s)|\beta} \left\{ X_s(t + 1) - X_s(t) + \sum_{r \in R(s)} \sum_{i \in N(r)} p_{ir} \beta (\lambda_i(t + 1) - \lambda_i(t)) \right\} \\
\leq \frac{\beta C'_2 + \beta}{|R(s)|\beta} \sum_{r \in R(s)} \sum_{i \in N(r)} p_{ir} |\psi_i(t)| \\
\leq \frac{(\beta C'_2 + \beta) p}{|R(s)|\beta} \sum_{r \in R(s)} \sum_{i \in N(r)} |\psi_i(t)| \\
\tag{3.54}
\]

(e)

\[
|\tilde{x}_r(t) - x_r(t)| \\
= \beta \left\{ w_{src(r)} \left( U'_{src(r)}(\hat{X}_{src(r)}(t)) - U'_{src(r)}(X_{src(r)}(t)) \right) + \sum_{i \in N(r)} p_{ir} (\hat{\lambda}_i(t) - \hat{\lambda}_i(t)) \right\} \\
\leq \beta \left\{ w_{src(r)} |U'_{src(r)}(\hat{X}_{src(r)}(t)) - U'_{src(r)}(X_{src(r)}(t))| + \sum_{i \in N(r)} p_{ir} |(\hat{\lambda}_i(t) - \lambda_i(t))| \right\} \\
\leq (C'_1 \beta + \beta) \tilde{p} w_{src(r)} \sum_{i \in R(src(r))} \sum_{i \in N(t)} \sum_{t' = t - t_d}^{t-1} |\psi_i(t')| \\
\tag{3.55}
\]

(f)

\[
|x_r(t + 1) - x_r(t)| \\
= \beta \left\{ w_{src(r)} \left( U'_{src(r)}(X_{src(r)}(t + 1)) - U'_{src(r)}(X_{src(r)}(t)) \right) + \sum_{i \in N(r)} p_{ir} (\lambda_i(t) - \lambda_i(t + 1)) \right\} \\
\leq \beta \left\{ w_{src(r)} |U'_{src(r)}(X_{src(r)}(t + 1)) - U'_{src(r)}(X_{src(r)}(t))| + \sum_{i \in N(r)} p_{ir} |(\lambda_i(t + 1) - \lambda_i(t))| \right\} \\
\leq (C'_2 \beta + \beta) \tilde{p} w_{src(r)} \sum_{i \in R(src(r))} \sum_{i \in N(t)} |\psi_i(t)| \\
\tag{3.56}
\]
(g) 

\[
|\hat{x}_r(t) - x_r(t)| \\
= \left| \sum_{\tau = t-t_d}^{r} k(t, \tau) \hat{x}_r(t) - x_r(t) \right| \\
\leq \max_{t-t_d \leq \tau \leq t} |\hat{x}_r(\tau) - x_r(t)| \\
= \max_{t-t_d \leq \tau \leq t} \left| \hat{x}_r(\tau) - x_r(\tau) + x_r(\tau) - x_r(t) \right| \\
\leq \max_{t-t_d \leq \tau \leq t} \left| \hat{x}_r(\tau) - x_r(\tau) \right| + |x_r(\tau) - x_r(t)| \\
\leq \max_{t-t_d \leq \tau \leq t} \left| \hat{x}_r(\tau) - x_r(\tau) \right| + \sum_{\tau' = \tau}^{t-1} |x_r(\tau') - x_r(\tau' + 1)| \\
\leq \max_{t-t_d \leq \tau \leq t} \left| \hat{x}_r(\tau) - x_r(\tau) \right| + \sum_{\tau' = \tau}^{t-1} |x_r(\tau' + 1) - x_r(\tau')| \\
\leq \max_{t-t_d \leq \tau \leq t} C_5 \sum_{l \in R(src(r))} \sum_{i \in N(l)} \sum_{\tau' = \tau-t_d}^{t-1} |\psi_i(\tau')| + \sum_{\tau' = \tau-t_d}^{t-1} C_6 \sum_{l \in R(src(r))} \sum_{i \in N(l)} |\psi_i(\tau')| \\
\leq C_5 \sum_{l \in R(src(r))} \sum_{i \in N(l)} \sum_{\tau' = t-2t_d}^{t-1} |\psi_i(\tau')| + C_6 \sum_{l \in R(src(r))} \sum_{i \in N(l)} \sum_{\tau' = t-2t_d}^{t-1} |\psi_i(\tau')| \\
\leq C_7 \sum_{l \in R(src(r))} \sum_{i \in N(l)} \sum_{\tau' = t-2t_d}^{t-1} |\psi_i(\tau')| \\
\tag{3.57}
\]

Lemma 3.5.2

(a) \( g^T(t) \psi(t) \leq - \frac{1}{\gamma} \| \psi(t) \|_2^2 \) \( \forall t \in \mathbb{N}^+ \)

(b) For some constant \( C_8 \), \( z^T \nabla^2 D(\xi) z < 2C_8 \| z \|_n^2 \) \( \forall z \geq 0, \forall \xi \in \mathbb{R} \)

(c) \( \| \nabla D(\lambda(t)) - g(t) \|_n \leq C_9 \sum_{\tau' = t-2t_d}^{t-1} \| \psi(\tau') \|_n \) for some constant \( C_9 \)

(d) \( \| \psi(t) \|_n \to 0 \) as \( t \to 0 \) and \( D(\lambda(t + 1)) \leq D(\lambda(0)) \) for all \( t \) if \( C_8 + t_dC_9 > \gamma > 0 \)

Proof:

(a) It is the same as the proof for lemma 4(a) in [20].

(b) The proof is similar to the one for lemma 4(b) in [20]. As the dual function is always convex, the hessian matrix is positive semidefinite [2]. The eigenvalues of the hessian matrix are all nonnegative and \( z^T \nabla^2 D(\xi) z \leq \rho(\nabla^2 D(\xi)) \| z \|_n^2 \) where \( \rho(\nabla^2 D(\xi)) \) is the largest eigenvalue. Hence

\[
\rho(\nabla^2 D(\xi)) \leq \frac{\text{trace}(\nabla^2 D(\xi))}{R N \bar{p}^2} < \frac{\beta^{-1} - \alpha}{\beta^{-1} - 1} \tag{3.58}
\]
(c)
\[
\| \nabla D(\lambda(t)) - g(t) \|_n = \left\| \sum_{s \in S} \sum_{r \in f(s)} p_{sr}(\hat{x}_r(t) - x_r(t)) \right\|_n
\]
\[
\leq \bar{p} \sum_{s \in S} \sum_{r \in f(s)} |(\hat{x}_r(t) - x_r(t))|
\]
\[
\leq \bar{p} C_7 \sum_{s \in S} \sum_{r \in f(s)} \left\{ \sum_{i \in R(s)(r)} \sum_{i \in N(l)} \sum_{\tau' = t - 2td}^{t-1} |\psi_i(\tau')| \right\}
\]
\[
\leq \bar{p} C_7 |S| \bar{R}_n \sum_{\tau' = t - 2td}^{t-1} |\psi(\tau')|
\]
\[
\leq C_9 \sum_{\tau' = t - 2td}^{t-1} \|\psi(\tau')\|_n \quad \text{p-norms in finite dimensional vector space are equivalent}
\]
(3.59)

(d) It is the same as the proof of lemma 6 in [20].

Lemma 3.5.3 The derivation between the estimated values and the exact values of the prices, gradients and flow volume trend to zero, i.e.

If \( t \to 0 \),

(a) \( \| \sum_{i \in N(r)} p_{ir}(\hat{\lambda}_i(t) - \lambda_i(t)) \|_n \to 0 \), \( \forall r \in R(s) \)

(b) \( \| g(t) - \nabla D(\lambda(t)) \|_n \to 0 \), \( \forall i \in S \)

(c) \( \| \hat{x}_r(t) - x_r(t) \|_n \to 0 \), \( \forall r \in R \)

Proof: (a)

\[
\| \sum_{i \in N(r)} p_{ir}(\hat{\lambda}_i(t) - \lambda_i(t)) \|_n \leq \sum_{i \in N(r)} p_{ir} \left\{ \max_{t - td \leq \tau \leq t} \sum_{\tau' = \tau}^{t-1} |\psi_i(\tau')| \right\}
\]
\[
\leq \bar{p} \sum_{i \in N(r)} \sum_{\tau' = t - td}^{t-1} |\psi_i(\tau')|
\]
\[
\leq \bar{p} \sum_{\tau' = t - td}^{t-1} \|\psi(\tau')\|_1
\]
(3.60)

(b) By lemma 3.5.2(c) and (d), it converges to zero when \( t \) tends to infinity.
(c) From lemma 3.5.1(g),

\[
| \hat{x}_r(t) - x_r(t) | \leq C_7 \sum_{l \in R(r)} \sum_{i \in N(l)} \sum_{\tau' = t - 2t_d}^{t-1} | \psi_i(\tau') | \\
\leq C_7 \sum_{l \in R(r)} \sum_{\tau' = t - 2t_d}^{t-1} \| \psi(\tau') \|_1
\]

By lemma 3.5.2(d), it converges to zero when \( t \) tends to infinity.

We then finish the proof by recapturing the argument of the proof of theorem 2 in [20]. As \( \lambda \) is always non-negative and we assume that a finite feasible flow vector always exists (assumption A3), the set \( \Omega^* \) of the optimal prices is bounded and the level set \( \Omega = \{ \lambda \geq 0, D(\lambda) \leq D(0) \} \) of \( D \) is compact. If \( C_8 + t_d C_9 > \gamma > 0 \), the sequence \( D(\lambda(t)) \) is decreasing with \( t \) and \( \lambda \) is within the level set and thus a limit point \( \lambda^* \) of the sequence \( \lambda(t) \) must exist.

As we assume that the difference between consecutive elements in \( T^i_\lambda \) is bounded for all \( i \in S \), the sequence \( \{ \lambda(t), t \in \cap T^i_\lambda \} \) also converges to the limit point \( \lambda^* \). Let \( \{ t_k \} \) be a subsequence that \( \{ \lambda(t_k) \} \) converges to the same limit point \( \lambda^* \). As the difference between the estimated gradient and the exact gradient will converge to zero when \( t \) tends to zero (lemma 3a),

\[
\lim_{k \to \infty} g(t_k) = \lim_{k \to \infty} \nabla D(\lambda(t_k)) = \nabla D( \lim_{k \to \infty} \lambda(t_k)) = \nabla D(\lambda^*)
\]

where the second equality comes from the fact that \( \nabla D \) is continuous.

We next show that the projection of \( \lambda^* - \gamma \nabla D(\lambda^*) \) is equal to \( \lambda^* \).

\[
[\lambda^* - \gamma \nabla D(\lambda^*)]^+ = \lim_{k \to \infty} [\lambda(t_k) - \gamma \nabla D(\lambda(t_k))]^+ \\
= \lim_{k \to \infty} [\lambda(t_k) - \gamma g(t_k)]^+ \\
= \lim_{k \to \infty} \lambda(t_{k+1}) \\
= \lambda^*
\]

As \( \lambda^* = [\lambda^* - \gamma \nabla D(\lambda^*)]^+ \), by the projection theorem [20],

\[
(\lambda^* - \gamma \nabla D(\lambda^*) - \lambda^*)(\lambda - \lambda^*) \leq 0 \\
\gamma \nabla D(\lambda^*)^T (\lambda - \lambda^*) \geq 0
\]
As the dual function is strictly concave, the above inequality is a sufficient and necessary condition [2] that \( \lambda^* \) minimizes the dual function \( D \). We denote the optimal flow volume vector \( \lim_{k \to \infty} x(t_k) \) as \( x^* \).

Due to the non-negative and energy constraints, the feasible set of the flow volume vector \( x \) should be a compact set. As we again assume that the difference between consecutive elements in \( T^i \) is bounded for all \( i \in S \), there exists a sequence of \( \{ t_n \} \subseteq \{ t_k \} \cap (\cap_i T^i_k) \) that \( \tilde{x}(t_n) \) converges to a limit point. We have proved that the difference between the estimated flow volume and the exact flow volume will converge to zero when \( t \) tends to zero (lemma3c), therefore,

\[
\lim_{n \to \infty} \tilde{x}(t_n) = \lim_{n \to \infty} x(t_n) = x^*
\]

This completes the proof.

3.5.2 The Asynchronous Algorithm for Step2

In the synchronized version of our proximal optimization algorithm, Step2 is executed only when the gradient projection method is converged. The synchronization can be achieved if each sensor broadcasts periodically its convergence status to the others. However, such network flooding induces a huge overhead. Instead, we propose a flow monitoring technique which allows each sensor to execute step2 asynchronously. Each sensor keeps track of two variables. One is an integer variable \( outIt \) which represents the number of times Step2 has been executed. The other is a boolean variable \( isConverge \) which is true only if the asynchronous gradient projection algorithm has converged. When the sensor communicates the flow volumes information with other sensors, the values of these two variables are also included. Each sensor node \( i \) inspects the values of \( outIt \) and \( isConverge \) of those nodes which has some traffic that passes through node \( i \). To execute step2 asynchronously, each sensor runs algorithm (2) individually. We show via simulation that the asynchronous algorithm for Step2 together with the asynchronous algorithm for Step1 matches the global optimum of the synchronous proximal optimization method.
Algorithm 5 Asynchronous Algorithm for Step2

1: Set $i.outIt = 0$ and $i.isConverge = false$, $\forall i \in S$.
2: for each node $i$ do
3:   while true do
4:     boolean flag = true;
5:     if Step1 has converged then
6:       $i.isConverge = true$;
7:       for all $r \in f(i)$ do
8:         if ((src($r$).outIt < $i.outIt$) or (src($r$).isConverge == false and src($r$).outIt == $i.outIt$)) then
9:           flag = false;
10:          break;
11:       end if
12:     end for
13:     if flag == true then
14:       execute step2;
15:       $i.outIt++$;
16:       $i.isConverge = false$;
17:     end if
18:   end while
19: end for

3.6 Experimental Results

In this section, numerical experiments are used to show the performance of the proximal optimization algorithm. Since the proximal optimization algorithm is a two-tier algorithm. Step2 is executed only when the gradient projection algorithm of step1 is converge. We use ‘#step2’ to denote the current number of step2 has been executed when we are discussing the numerical results.

3.6.1 The Synchronous Algorithm

To illustrate the behavior of our synchronous algorithms, similar to the last chapter, we first start with the simple example of fig. 1.1 where the topology is simple and predictable enough to allow us to study and understand the effect of applying different utility functions and adjusting the weights of the links to the data flow. In addition to this simple topology, we also use a more realistic random topology to study the scalability of the algorithm.
Simple Network Example

The topology in fig. 1.1 is assumed to be fixed and the sending and receiving costs are uniform and equal to $2J/\text{byte}$ and $1J/\text{byte}$ respectively. Each source node starts with 10000J initial energy. A constant step size is used. The following known [24] concave utility functions are used:

$$U_s\left(\sum_{r \in R(s)} x_r\right) = \begin{cases} 
\log\left(\sum_{r \in R(s)} x_r\right), & \text{if } k = 1 \\
(1 - k)^{-1} \left(\sum_{r \in R(s)} x_r\right)^{1 - k}, & \text{otherwise}
\end{cases} \quad (3.66)$$

Fig. 3.1 shows the individual flow value under different utility functions. Each node execute step1 500 times under each step of step2. Step2 is executed for 10 times. Therefore, step1 is executed 5000 times in total. We can see that nodes 5 and 6 get the highest flow values in all the cases. This is because nodes 5 and 6 are closer to their sinks than nodes 1 to 4 and they only incur the cost of sending since they are directly connected to the sinks. The figure also shows that a larger values of $k$ lead to (more fairness) smaller flow values for nodes 5 and 6. In addition, nodes 1 to 4 gain a larger portion of energy in the bottleneck nodes.

Fig. 3.2 shows the aggregate flow value under different utility functions. The flow
value drops and the number of execution of step2 increases when $k$ increases. When $k$ is equal to 1 and 2, no execution of step2 is needed for the algorithm to converge. Fig. 3.10 and fig. 3.4 show that the gradient projection algorithm of the step1 converges within 300 iterations when $\#step2 = 0$. When $k$ is equal to 3, two executions of step2 are needed for the algorithm to converge. Fig. 3.5 shows that the gradient projection algorithm of the step1 converges within 500 iterations when $\#step2 = 0$. Fig. 3.6 and fig. 3.7 show that gradient projection algorithms converge within 10 iterations when $\#step2 = 1$ and $\#step2 = 2$.

Fig. 3.8 and Fig. 3.9 show the impact of changing the weights $w_i$ on the individual flow and the aggregate flow respectively. We compare three different weight assignments, i) all the weights are equal to 1; ii) the weights of the bottleneck nodes 5 and 6 are equal to 0.6 and the rest of the nodes are equal to 1; and, iii) the weights of the bottleneck nodes 5 and 6 are equal equal to 0.3 and the rest of the nodes are equal to 1. The figures show that the smaller the weights of the bottleneck nodes, the smaller the flow value of the bottleneck nodes and the larger the flow value of other nodes (i.e., better fairness). The total flow value decreases when the weights of the bottleneck nodes decrease. The adjustment of the weight does not affect the numbers of execution of step2 for convergence. In all cases, the step2 is not needed for convergence. Fig. 3.10, fig. 3.11 and fig. 3.12 show that the gradient projection algorithm of the step1 converges within 300 iterations when $\#step2 = 0$. 

Figure 3.2: Aggregate flow values under different utility functions (Simple example)
Figure 3.3: Aggregate flow value when $k = 1$ and $\#step2 = 0$ (Simple example)

Figure 3.4: Aggregate flow value when $k = 2$ and $\#step2 = 0$ (Simple example)
Figure 3.5: Aggregate flow value when $k = 3$ and $\#step2 = 0$ (Simple example)

Figure 3.6: Aggregate flow value when $k = 3$ and $\#step2 = 1$ (Simple example)
Figure 3.7: Aggregate flow value when $k = 3$ and $\#step2 = 2$ (Simple example)

Figure 3.8: Comparison of Individual flow value for different weight adjustments (Simple example)
Figure 3.9: Aggregate flow value for different weight adjustments (Simple example)

Figure 3.10: Aggregate flow value for adjustment 1 and \#step2 = 0 (Simple example)
Figure 3.11: Aggregate flow value for adjustment2 and $\#step2 = 0$ (Simple example)

Figure 3.12: Aggregate flow value for adjustment3 and $\#step2 = 0$ (Simple example)
Random Network Example

To illustrate the behavior and scalability of the algorithm, we use the random topology in Fig. 3.13. Each source forwards its traffic to the closest sink (in number of hops and ties are broken randomly). The number inside the parentheses is the corresponding sink of a node. The transmission range of a node is 15m. 22 nodes randomly distributed over a square area of 80m X 80m, of which 20 are sources and 2 are sinks. Most of the nodes do not have a direct link to their sinks (only nodes 6, 8, 12, 15 and 20 have a direct link to the sinks). Each source has 10KJ initial energy. The common utility function $\log(.)$ is used and each node has equal weight. The energy consumption model in [37] is used: sending 1 byte from node $i$ to node $j$ which are separated by a distance of $d$ meter costs $e_{ij}^s = c_1 + c_2 d^2$ and receiving 1 byte costs $e_{ji}^r = c_1$ where $c_1 = 400 nJ/byte$ and $c_2 = 800 pJ/byte/m^2$.

Routes Selection

Multiple routes between each source-destination pair are found, and we study the performance of the algorithm by varying the number of available routes for a source. There are three different settings: i) Only link-disjoint routes are used. Each sensor gets 2 outgoing routes on average in this example. ii) At most 5 not necessarily dis-
joint routes are used. iii) At most 10 not necessarily disjoint routes are used. In all the cases, the route with the smallest number of hops are selected first.

Convergence

Fig. 3.14 show that 4 executions of step2 are needed. In this example, 500 iterations are executed for the gradient projection method in order to reach the optimal under each step2. Fig. 3.14 also shows that the performance is the same for the three route selection methods. Three executions of step2 are needed for convergence. Fig. 3.15, fig. 3.16, fig. 3.17, and fig. 3.18 show that the gradient projection algorithm of the step1 converges within 400 iterations when \( \#step2 = 0 \), \( \#step2 = 1 \) and \( \#step2 = 2 \).

Individual Flow Value

Fig. 3.19 show the individual flow volumes of each node. Table 3.1 shows the bottleneck nodes on the path from each source node to its sink. Some of the nodes have two bottleneck as there are two optimal outgoing routes. Nodes 8, 12, 15, 16 and 20 are the bottlenecks and they are located nearer to the sinks than the other nodes. Node 8 gets the highest flow volume as it is the closest node to the sink and does not need to forward other nodes’ traffic. As we have explained in Section 3.3, nodes 1, 4, 7, 11, 14
Figure 3.15: Aggregate flow value when \#step2 = 0 (Random example)

Figure 3.16: Aggregate flow value when \#step2 = 1 (Random example)
Figure 3.17: Aggregate flow value when $\#step2 = 2$ (Random example)

Figure 3.18: Aggregate flow value when $\#step2 = 3$ (Random example)
and 19 get the same flow values because they share the same bottlenecks. Similarly, nodes 3, 5, 9, 10, 13, 17 and 18 share the same bottlenecks and have the same flow values. Nodes 2 and 6 which have a single bottleneck get nearly the same flow volumes as nodes 1, 4, 7, 11, 14 and 19.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Bottleneck</th>
<th>Nodes</th>
<th>Bottleneck</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16,20</td>
<td>11</td>
<td>16,20</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>12,15</td>
<td>13</td>
<td>12,15</td>
</tr>
<tr>
<td>4</td>
<td>16,20</td>
<td>14</td>
<td>16,20</td>
</tr>
<tr>
<td>5</td>
<td>12,15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>16,20</td>
<td>17</td>
<td>12,15</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>18</td>
<td>12,15</td>
</tr>
<tr>
<td>9</td>
<td>12,15</td>
<td>19</td>
<td>16,20</td>
</tr>
<tr>
<td>10</td>
<td>12,15</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 3.1: Bottleneck of each node of the random network

**Maximum Lifetime VS Fair Data Collection**

To compare the maximum lifetime approach and our approach, we implement the algorithm in [5] which is a shortest path algorithm for maximizing the lifetime. The link
cost of link \((i, j)\) is

\[
cost_{ij} = (e_t^{ij})^{x_1}E_i^{-x_2}E_i^{x_3} + (e_r^{ij})^{x_1}E_j^{-x_2}E_j^{x_3},
\]

where the transmission energy consumed at node \(i\) to transmit a byte to its neighboring node \(j\) is denoted by \(e_t^{ij}\), the energy consumed by the receiver \(j\) to receive a byte is denoted by \(e_r^{ij}\), \(E_i\) and \(E_j\) are the initial energy of node \(i\) and \(j\), \(E_i\) and \(E_j\) are the remaining energy of node \(i\) and \(j\). \(x_1, x_2\) and \(x_3\) are the parameters of the algorithm.

We set \(x_1 = 10, x_2 = x_3 = 30\) which is the same as the paper [5]. The link cost is updated when the sender \(i\) has sent \(\lambda Q_i^{(c)}\) amount of data where \(Q_i^{(c)}\) is the data generation rate of the sender \(i\) and \(\lambda\) is the augmentation step size. The algorithm stops when a sensor cannot find a path to its destination. We apply this algorithm to the random network example. The data generation rate is set to 1 for all the sources. We record the amount of data collected until the algorithm stopped and label it as data1 in the table 3.2. We also record the amount of data collected until the first node finishes all its energy and label it as data2 in table 3.2. Table 3.2 shows the results of different \(\lambda\).

We can observe from the table 3.2 that the volume collected according to both stopping conditions are the same. This is because given the the same initial energy for all the sensors, node 16 always finishes the energy first. As the traffic from node 2 and 6 has to always pass through node 16 to the destination which makes the data collected of both stopping condition be the same.

\(\text{From fig. 3.14, the total amount of data collection in the fair data collection approach is around 75 GB which is significantly larger than the data collected in the maximum lifetime approach. It is mainly due the fact that the fair data collection approach always fully utilize the energy of the bottlenecks. However, in the maximum lifetime approach, the algorithms stop when one of the nodes is unable to route the data to the destination or runs out of energy. In most WSNs application, large number of sensors with high redundancy are deployed over the area of interest while small amount of sensor nodes failure usually is not significant. For instance, in the random network example, there are 17 out of 20 nodes can be still functional when the lifetime algorithm stops. In this situation, depending on the coverage requirement, the fair data collection approach may be more preferable.}\)
<table>
<thead>
<tr>
<th>λ</th>
<th>Data1 (GB)</th>
<th>Data2 (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1MB</td>
<td>44.84</td>
<td>44.84</td>
</tr>
<tr>
<td>100KB</td>
<td>45.7</td>
<td>45.7</td>
</tr>
<tr>
<td>10KB</td>
<td>45.786</td>
<td>45.786</td>
</tr>
<tr>
<td>5KB</td>
<td>45.791</td>
<td>45.791</td>
</tr>
</tbody>
</table>

Table 3.2: The data collection of the maximum lifetime approach

### 3.6.2 The Asynchronous Algorithm

In this subsection, we evaluate the convergence behavior and the energy overhead of the asynchronous algorithm. We implement and simulate the algorithm in a multi-threading environment with the Java language. Each thread controls a sensor and communicate the price and flow volume information with the others periodically by message passing. We study the convergence and the energy overhead of the algorithm via simulation for two different scenarios: The first one assumes even energy distribution, which means that each sensor has the same initial 10KJ energy. The second one assumes uneven energy distribution where the initial energy of a sensor node is randomly selected from the set \( \{1, 2, 3, ..., 10\} \) KJ. The same initial price of 0.1 and a common constant step size are used for both scenarios.

#### Convergence

Each sensor updates the prices and flow volumes periodically and asynchronously for 2000 times for the random example. We sum up the total flow volume for each round of update. Fig. 3.20 shows the total flow volume at each round. We include the optimal solution of the synchronous algorithm for reference. The figure shows that the asynchronous algorithm converges to the optimum of the synchronous algorithm within 1500 rounds.

#### Energy Overhead

We then evaluate the overhead of our algorithm. We assume that the packet size is 100 bytes for each price or flow volume information. Fig. 3.21 plots the percentage of the initial energy consumed due to such communication overhead up to 1500 rounds. In both scenarios, the maximum energy consumption of a node is less than 0.1% of the initial energy which is negligible.
Figure 3.20: Convergence of the asynchronous algorithm

Figure 3.21: Energy overhead of the asynchronous algorithm
CHAPTER 4

CONCLUSIONS

In this thesis, we have studied the data collection problem in WSNs. Our objective is to design energy efficient algorithms that maximize the data collection with the consideration of fairness.

The fair data collection problem is modelled as a concave utility maximization problem. By considering the Lagrange relaxation and duality of the problem, a synchronous sub-gradient routing algorithm is derived. Two routing models are studied. The stability of the routing algorithm is discussed. Numerical results show that the algorithm converge fast and work well.

In order to derive an asynchronous algorithm which is more suitable for large scale WSNs, we consider an other non-differentiable optimization which is called proximal optimization method. We reformulate the convex optimization problem in terms of paths and nodes. A two-tier optimization synchronous algorithm is proposed to address the fair data collection problem. Based on an partial asynchronous model, we extend our algorithm to support asynchronous environment. Each sensor is allowed to update their prices and flow volumes asynchronously at different times with different frequencies. Numerical results shows that the asynchronous algorithm converge to the same optimum as the synchronous algorithm with negligible energy consumption overhead.

There are serval related issues can be investigated further in the future. Note that in both sub-gradient and proximal algorithm, sensors need to communicate to the other sensors with the flow volumes explicitly. It would be an interesting question to study if it is possible to reduce such communication overhead without losing the guarantee of the convergence. Another direction of future work is to study how the algorithms behave if it is implemented on top of a real MAC layer.
APPENDIX A

PUBLICATIONS


REFERENCES


