SECURE MINIMUM TIME DATA COLLECTION (SMTDC) PROTOCOL FOR WIRELESS SENSOR NETWORKS

BY

HE HUANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Advisor:

Professor Klara Nahrstedt

ABSTRACT

Recent work has shown that a mobile data collector moving along a predefined trajectory can improve the real-time data collection duration and efficiency in wireless sensor networks (WSN). Due to the fixed trajectory and limited communication range, data collection is conducted using a many-to-one communication pattern known as convergecast. However, because of the confidentiality concern of data being transmitted, security issues such as security key leakage, eavesdropping, and malicious attack raise significant challenges in minimizing the data collection time. To address this issue, we present the design and implementation of the Secure Minimum Time Data Collection (SMTDC) protocol, a tree formulated, and time-scheduled protocol for large scale, stationary, hardware-limited WSN. SMTDC can cooperate with many existing security communication frameworks. During the tree formation phase of SMTDC, we build well-balanced optimized trees that have the potential for minimum data collection time. We formulate our approach as an integer linear programming problem and solve it using linear relaxation based iterative rounding (LR-IR). During the time scheduling phase of SMTDC, we use a heuristic time-slot arrangement algorithm to solve the tree scheduling problem. The proposed algorithms and schemes are validated through simulation experiments using GUROBI solver and OMNET++ under realistic WSN topology. The result shows that SMTDC tree formation outperforms other algorithms in building a more effectively secure and load-balanced tree, and SMTDC scheduling significantly improves the data collection time over pre-generated tree topology.

Keywords—Wireless sensor networks; secure data collection; tree formation; time scheduling

To my mother, for her love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to express my special appreciation to my advisor, Dr. Klara Nahrstedt, for initially giving me the opportunity to become a MONET member three years ago, for continuing to support and guide me, and for all the valuable discussion and comments. Professor Nahrstedt's perseverance, enthusiasm, and wisdom about computer science, as well as her patience and kindness with students, have influenced me deeply and played an important role in my growth. I will be always grateful and remember her trust, support, and encouragement for the rest of my life.

Besides my advisor, I would like to thank to Dr. King-Shan Lui and Haiming Jin; they gave me great advice about my research, and encouragement and help whenever I needed them.

TABLE OF CONTENTS

Chapter 1 Introduction 1
Chapter 2 Related Work 5
2.1 Security threats and challenges in WSN 5
2.2 Tree formation mechanism in WSN 7
2.3 Scheduling mechanism in WSN 8
Chapter 3 System Model and Observation 10
3.1 Security communication framework 10
3.2 Messages lifecycle analysis 12
3.3 Data collection time model for tree with height one 14
3.4 Data collection time model for general trees 17
Chapter 4 SMTDC Design and Solution 22
4.1 SMTDC tree formation 22
4.2 SMTDC scheduling 32
Chapter 5 Performance Evaluation 37
5.1 Performance of SMTDC tree formation 37
5.2 Performance of SMTDC scheduling 43
Chapter 6 Conclusion 45
Appendix A Specifications of Raspberry Pi 46
Appendix B Deduction of Security Constraint 47
References 48

Chapter 1 INTRODUCTION

With wireless connectivity, advanced sensor networks, and machine-to-machine communications, the Internet of Things (IoT) has profound implications for industrial automation. As a key component of IoT, wireless sensor networks (WSNs) have been finding their application in diversified scenarios, such as smart grid monitoring, traffic monitoring, battlefield monitoring, etc.

WSN applications used in military and commercial fields require the data communication within the network to be secure. But on the other hand, thousands of tiny and inexpensive sensor nodes are hardware-limited devices, and they have boundaries in communication and data processing. The confidentiality of data decreases communication efficiency, and complicates the WSN topology [1]. Furthermore, an inefficient topology leads to a sacrifice of network lifetime [2], and most importantly, prolongs the data collection time of WSN [3], [4], [5]. Based on the above motivation and related works, in this thesis we focus on large-scale WSNs with vast amount of stationary measurement devices (MDs) deployed in the target area, and one mobile sink, or data collector (DC), gathering data from MDs. All MDs are hardware-limited devices and have limited power for long range message transmission. DC is unable to reach every MD to collect data because of spatial and temporal constraints. So the DC only connects directly to MDs that are in its proximity. We define the MDs in the direct communication area with DC as root MDs. The data collected by MDs, which are outside the DC's communication area, has to be forwarded to root MDs first in a multi-hop manner.

As shown in Figure 1.1, a DC is installed on a vehicle that moves along a predefined fixed trajectory, a road, or a certain street. When the DC is moving along its trajectory (the red line in Figure 1.1), it can only communicate with the MD nodes in the blue shaded area due to their closer proximity. All other MDs outside the blue shaded area must first forward their messages to the MDs in this area, and then complete the final data transmission to DC.

In the current large-scale WSN systems, as mentioned in [6], [7], [8], security of data communication between two telemetric devices is one essential condition. Rehana et al. use symmetric key encryption for both encryption and decryption of the packets when sending packets over the wireless network in [7]. Gong et al. introduce an asymmetric PKI model in

[8], where, in addition to a password, the user uses the public key of the server for message encryption and decryption.

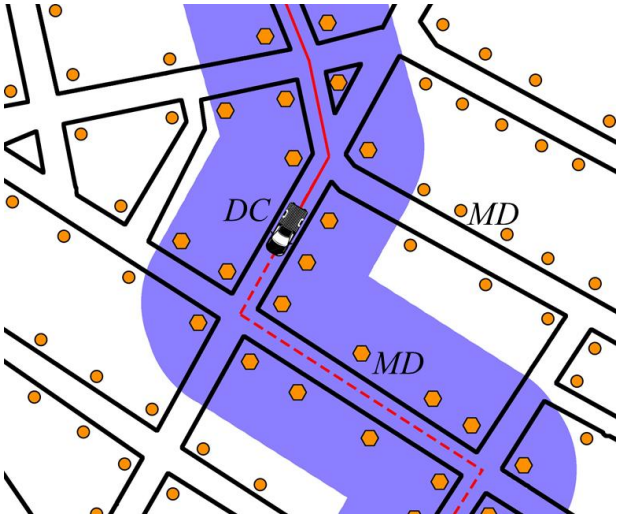


Figure 1.1: A real example of trajectory defined DC in WSN

In this thesis we aim at minimizing the data collection time in large-scale, secure WSNs. We design a data collection protocol to be used in a more general security scenario, which applies to many security communication frameworks, not to one specific security situation. We generalize the preexisting confidential data communication mechanism of one relay-node, MD or DC in our WSNs to three serial stages, encryption, transmutation, and then decryption. Besides, in secure data collection, a strong correlation exists between data, and hence, sequential data integration is needed. We generalize one round of data collection in a WSN to be a propagation in an aggregated convergecast tree, in which messages transmit first top-down then bottom-up. Specifically, when one root MD, or other MD nodes, receive a data collection command message from a DC or its parent MD nodes, it will forward this secure message to its child MDs, and wait for the feedback from all of its child MDs before sending the message back to its parent MD.

Based on this scenario, we start by researching the primary limiting factors of data collection time in WSNs, which are mentioned in [3], [5]. The limiting factors can be categorized as follows: (1) co-channel interference, (2) half-duplex communication model of each sensor node, (3) topology of the network, and (4) time scheduling of certain topology. As mentioned in [5], [9], constraint (1) can be easily eliminated using interference-aware TDMA or single-channel, CSMA-based protocols. Constraint (2) is inevitable due to the legacy telemetric devices widely deployed in real industry. To achieve further improvement, we focus on the limiting factors (3)

and (4), and we are building an optimized tree topology and a time scheduling arrangement method that minimizes the data collection time. At the same time we aim to achieve the balance between workload and security issues, such as leakage of confidential messages.

Although significant effort has been expended in recent years on building optimized trees and designing scheduling methods for maximum data throughput and minimum data collection time in general wireless networks, the security issue in the process of tree formation has not been considered. Besides, as we will discuss in Chapter 2, almost all of these works [3], [5] choose to use shortest path derived algorithms when building trees. While these mechanisms are relatively easy to implement, they do not consider balancing the workload of each sensor node. Besides, none of the previous work discussed the impact of routing tree formation on time slot scheduling under the influence of security constraints and load balancing limitation. These shortcomings motivate us to design and implement a secure, load-balanced tree formation mechanism for fast data collection.

We first undertake a series of experiments on a state-of-the-art wireless sensor network hardware, Raspberry Pi. Under generalized security protocol, where every MD or DC has to execute encryption, decryption, and then transmutation for every message between them, we study how the data collection time for one single MD is influenced by the number of child MDs. We also investigate how the data collection time of the entire WSN is influenced by different topologies of WSN. Our experiments show that given a certain number of MDs in a WSN, minimum data collection time varies significantly with the average number of child nodes of all MDs. Therefore, we introduce our Secure Minimum Time Data Collection (SMTDC) protocol for secure WSN.

In the SMTDC tree formation part, the expected data collection time of WSNs is formalized as finding convergecast trees. And in this optimization problem our objective is to find the minimum summation of time spent in each level of the tree. The time spent in each level of the tree depends on the average number of child MDs for one parent MD. This optimization problem is NP-hard, so we propose an approximation algorithm, algorithm 1, to solve it. In addition, under the uncertain load balancing conditions (undecided average number of child MDs before tree formation), we design a heuristic iterative algorithm, algorithm 2, to solve this nonlinear optimization problem. After SMTDC tree formation builds a convergecast tree, in the SMTDC scheduling part, we use a message priority algorithm, algorithm 3, to further reduce the data collection time.

The following lists our key findings and contributions. We:

- Generalize the current prevailing security protocol of message transmission into three-stage model, systematic analysis and the impact of load balance on the data collection time.
- Formulate the SMTDC tree topology as a 0-1 integer linear programing problem which aims to build tree topologies that potentially have minimum data collection time. We propose an approximation algorithm with the help of linear relaxation based iterative rounding to solve this NP-hard problem.
- Propose a heuristic iterative algorithm to solve a nonlinear optimization problem, in which the load balancing condition is undecided before SMTDC tree formation.
- Design a heuristic SMTDC scheduling algorithm that achieves the minimum data collection time for any convergecast tree topology in which messages propagate using top-down bottom-up models.
- Conduct simulation experiments using GUROBI solver and OMNET++ under realistic WSN topology, the results of which validate the proposed algorithms and prove that SMTDC is a scalable, load-balancing, minimum data collection protocol

The rest of the thesis is organized as follows: in Chapter 2, we discuss related work and briefly introduce our previous security protocol. Chapter 3 gives an overview of the network and transmission model of SMTDC, and discusses our load balancing observation. In Chapter 4, we present two phases of SMTDC, the SMTDC tree formation scheme and the SMTDC scheduling scheme. Chapter 5 evaluates the performance of proposed schemes and Chapter 6 concludes the thesis.

Chapter 2 RELATED WORK

In this chapter, we first briefly investigate security challenges occurring in WSNs, which can be generalized to a probability problem. As mentioned earlier, we designed our SMTDC to cooperate with prevailing security protocols to handle security attacks. Then we review the state-of-the-art tree formation mechanism and scheduling mechanism for data collection in WSN.

2.1 Security threats and challenges in WSN

In reality, WSNs tend to be more vulnerable to various security attacks than wired networks as the unbounded transmission medium is more accessible to security attacks than those of the bounded medium. Nearly all security schemas believe that the adversary can achieve full control over a sensor node in a WSN by direct or close-range physical access. This characteristic of WSNs posed various challenges to researchers. Besides, sensor nodes are normally deployed randomly in expansive unprotected areas, such as enemy territory. Even if each sensor has advanced equipment, they are hard to protect from being compromised.

Attacks against WSNs can be classified according to different criteria. One is to separate them into attacking against the security mechanism and attacking against the basic routing communication mechanism. Also we can separate attacks into passive adversary and active adversary based on the types of attacks. The passive adversary only monitors the communication channel, stealing key material which threatens the confidentiality of data. The active adversary takes efforts to modify or alter the transmission system, which threatens authentication and breaks the behavior of the WSN. Here we point out the major types of attacks in WSNs:

- **Denial of service (DoS)**: The DoS attack [10], [11] tries to exhaust the resources of the victim. DoS prevents the legitimate network user from normally accessing the services and resources by flooding garbage packets to sensor nodes. DoS attacks can be performed in different layers of the network [12]. Jamming and malicious flooding are two representative example of DoS attack.
- **Sybil attack**: In a Sybil attack [13], [14], a node forges the identities of more than one node in the WSN, thus the distributed algorithm is unable to maintain data integrity.

Newsome et al. [13] show that they can use radio resource testing to discover the Sybil node in a WSN with certain probability.

- **Wormhole attack**: In a Wormhole attack [15], [16], an attacker learns or records the message packets at one location in the WSN and mimics or sends these messages at another location. The attack creates a tunnel in the WSN, and this is a significant threat because an attacker is unnecessary to compromise a node to accomplish the attack.
- **Blackhole attack**: Blackhole attack [17], [18] usually occurs in flooding-based protocols. An attacker maliciously manipulates a node to act as a black hole that attracts all the traffic in the WSN; it attracts the traffic with declarations such as: this node contains high-quality data or the shortest path to a sink node. Thus the malicious node is able to insert itself into the sink and sensor nodes, and all messages between sink and sensor node have to pass through it.
- **Hello Flood attack**: Hello Flood attack [19] uses 'hello' packets to convince the sensor nodes that the malicious node is their neighbor. The attacker usually uses a high transmission range and sends 'hello' packets in a large area of the WSN. Thus, when victim nodes try to send messages to the sink node, they go through the attacker as they treat it as their neighbor, and victim nodes are ultimately spoofed by the attacker.

Table 2.1. Summary of Security Threats and Countermeasures in WSN

Attack type	Countermeasure scheme	Characteristic
DoS attack, Jamming	enhanced detection protocols [20]	Consistency checking, signal strength measurements for poor packet delivery ratios, Location information as the consistency check.
Sybil attack	Newsome et al. [13]	Regularly changing of key, Resetting of device and changing of session keys, Position verification for detecting Sybil entity.
Wormhole attack	TIK [21]	Monitoring system using packet leach techniques, with symmetric cryptography and time synchronization.
Blackhole attack	Intrusion Detection System [17]	Local information collected by watch dogs and optimized into the global information, global decision made by cluster head to compensate vulnerability in communication pattern.
Hello Flood attack	Singh et al. [19]	Using signal strength and client puzzle method. Nodes classified based on the signal strength. Battery power used to check the validity of suspicious nodes.

In Table 2.1, we summarize a variety of types of threats and their corresponding countermeasures and security schemes.

2.2 Tree formation mechanism in WSN

The number of papers written on data collection, clustering, and tree formation is too vast to enumerate comprehensively. Instead, we focus our review on key works, particularly those that have aimed at data collection in large-scale WSNs with mobile sinks and stationary sensor nodes.

To the best of our knowledge, one major algorithm for tree formation in the data collection protocol is the shortest path algorithm (SPA) or similar. In tree formation using SPA, a node in the WSN chooses its parent node based on the shortest hop to the data collector (sink) among its neighbor nodes. An SPA like Dijkstra's algorithm uses the number of hops as its only criterion, rather than other important factors such as workload, speed, and cost. The result is that tree formation using SPA generates an unbalanced tree, in which some nodes tend to have large numbers of child nodes that exceed the resource constraints of the sensor devices.

For example, Gao et al. [3] use SPA, and their model is based on the fact that the mobile sink has to talk to every data collection node in its direct communication area (DCA), which equivalently means the mobile sink has to build a tree with every sensor with tree height at least one. In our protocol, a nearby sensor node in the DCA area can be formed into one single tree, obviously a smaller height summation than that of the algorithm in [3].

The ENergy Critical node Aware Spanning Tree (ENCAST) algorithm proposed by Zou et al. also uses the idea of SPA; ENCAST finds a shortest path tree by breadth-first traversal from the data collector (sink). The data path in this tree is guided by the minimum number of hops towards the data collector. One case that can happen in their work is that there might be multiple SPA trees in a sensor WSN, due to the fact that one node may have many neighbors with the same minimum-hop to the sink. The energy of a node is ENCAST's second criterion of path selection; however, certain nodes' energy may decrease faster than others, which leads to a rapid changing in the tree topology.

In [2], Chen et al. also use SPA to build a shortest-path tree, and they use an adjustment method to convert the data collection tree to a load balancing tree with the purpose of balancing the

work load of each no-left node. They assume each node in the tree has limited power, and data collection and forwarding messages consume a certain amount of power. The load-balancing tree in [2] is focused on saving the energy of each parent node, thus increasing the lifetime of the network.

Bandara et al. [22] present a Generic Top-down Cluster (GTC) and cluster tree formation algorithm that overcomes clustering tree problems such as large variations in tree, and large distance from sink to leaf node. They achieve this by varying parameters and properties in their GTC algorithm such as controlled breadth and depth, tree size, etc.

Erciyes et al. [23] propose two algorithms to form spanning trees in sensor networks. The first algorithm is a modification of the spanning tree formation algorithm and forms a hierarchical spanning tree with a given sink, and it also considers the energy consumption in the WSN. The second algorithm is a modification of the breadth-first search algorithm.

In [5], [24], they use the Capacitated Minimal Spanning (CMS) tree heuristic. They are all based on one greedy scheme presented by Dai and Han [25]. CMS is a centralized solution that assumes there is only one single root or base station with limited number of sensor nodes in the grid network, so that one centralized root sensor node has to be determined first. In addition, CMS focuses on constructing a top-balanced tree over the sensor network, which means CMS only guarantees that a roughly equal number of subtree nodes of the first level of sensor nodes are directly connected to the mobile sink.

2.3 Scheduling mechanism in WSN

In this section, we introduce the most representative papers on scheduling mechanisms in WSNs, particularly scheduling mechanisms under certain or predefined topologies. A scheduling mechanism is mainly used for eliminating interference and parallel transmission, thus lowering the bound on data collection time and reducing the energy consumption.

Incel et al. [5] combine scheduling with transmission power control; their concurrent transmission is conducted by using multiple frequency channels. They are the pioneers in discussing the effect of multichannel scheduling combined with the impact of tree topology and channel assignment mechanism. They design algorithms to achieve the lower bound on

scheduling length under the tree topology WSN in realistic settings; their algorithms are based on breadth first search.

In [26], Song et al. present an energy-efficient and time-optimal packet scheduling algorithm for a raw data convergecast tree. In their case, data is collected in periodic rounds from all the nodes to the sink, and the interference is assumed to be eliminated. In addition, they bring up a 3-coloring scheme for channel assignment, and they briefly mention that 3-coloring mitigates the interference on links in sensor trees. But it is not certain whether the color represents the frequency, codes, or other interference factors.

Here we have to mention an important work done by Florens et al. [27]. In their work, they propose an optimal centralized algorithm to obtain transmission schedules that achieve minimum delay in data collection for both omnidirectional and directional channels. They derive the lower bounds on the time for special network topologies, such as line, multiline, and tree. However, the effect of data fusion and integration for raw data, which is the major integration characteristic for WSN, is not considered. Besides, Florens et al. assume the same time slot length for the sink node and relay node.

Revah and Segal in [28] extend the work of Florens [27] by reducing not only the average delivery time but also the completion time. They argue that Florens' work [27] does not consider the idle time of messages transmission. They argue it is unrealistic that a message can be transmitted without any delay. In their algorithm, messages in different subsets can be transmitted together without being delayed by messages from other subsets. However, they did not test their algorithm in a tree topology WSN.

Chen et al. [24] also extend Florens et al. [27] in their time scheduling algorithm; they double the length of the time on the relay node and introduce a guard-time such that the sink can receive messages continuously.

Chapter 3 SYSTEM MODEL AND OBSERVATION

3.1 Security communication framework

In our convergecast data collection tree, all MD nodes, including root MD, can also be categorized, based on their communication roles, into two types: relay MD, and leaf MD. The relay MD sends the combined security information given by a higher level MD in the tree to each of its child MD nodes, and the relay MD waits for the response of the encrypted data collection message from its child MD nodes. The leaf MD nodes only respond to relay MD nodes, and they do not need to transmit the data collection message to other MD nodes. As mentioned before, one of the major differences between this thesis and most of the previous work is that we consider the security issue a top priority when we design our protocol. Hence, we choose three-serial-stage message transmission mechanisms such that message and data are encrypted in the WSN, which provides protection against eavesdropping, modification, and injection of packets. And this message transmission mechanisms is used in the top-down, and then bottom-up, data collection approach such that we can have the minimum number of messages in one round of data collection for the entire WSN. In other convergecast data collection protocols which do not consider the issue of security transmission, such as [2], [3], [5], every sensor node sends an unencrypted message through a path to the sink node in the tree as soon as the sensor node fetches some local data. Hence, a relay node has to send multiple messages to its parent node for all the sensors in its sub-branch. As mentioned in [29], the more the number of messages transmitted by the data collection protocol in the WSN has been increased, the more the risk to secure transmission over the WSN has increased.

In our convergecast tree for data collection, we regard one message between two MDs as an edge. The total number of messages equals the number of edges times two in the final tree topology. Every relay node in our framework does data compression and integration after it gets all the messages from its child nodes. Thus, each edge transmits two messages: one for top-down transmission, and one for bottom-up. Hence we lower both the number of messages needed for data collection and the security risk in message transmission.

SMTDC is designed to have built-in extensibility and scalability to support many security communication protocols. A representative secure communication protocol with packet integration that can be used in a top-down-bottom-up collection mechanism is [6]. In [6], the

confidentiality of the data between two nodes is ensured through encryption and decryption using different types of Diffie-Hellman (DH) keys, including a public group key of the entire tree and public/private key pair of MDs. When an MD receives one message, it has to decrypt the message received and verify necessary signatures first, then it prepares for forwarding. When an MD forwards one message, it will use its own DH half key and sign it. Then, it sends this encrypted message to its child node or parent MD node, maybe a leaf node, or root MD node.

Inspired by many security protocols such as [6], [7], here we define the secure top-down-bottom-up data collection mechanism that our SMTDC is based on. In one round of data collection, the top-down-bottom-up mechanism consists of two phases: the top-down phase and the bottom-up phase. And the secure message transmission in data collection consists of three serial stages: encryption, transmission, and decryption.

In the top-down phase of data collection, the messages are transmitted from the root node in the top of tree to leaf nodes in the bottom of tree. When a relay MD node receives a secure message from its parent node, it decrypts the message first; then the relay MD encrypts an individual secure message for each of its child MDs, and then transmits secure messages separately to its child MDs.

In the bottom-up phase of data collection, when a relay MD node receives a secure message from its child MD nodes, it decrypts the message first and then executes necessary data integration. After it receives messages from all its child MD nodes, it prepares and encrypts a new message and transmits to its parent MD.

We need to mention that a leaf MD in the top-down phase only receives a message from its parent MD and executes decryption; in the bottom-up phase, the leaf MD encrypts a new message and transmits to its parent node.

Figure 3.1 shows how the security protocol works with top-down-bottom-up collection mechanism in one round of data collection. There are four MD nodes (MD0~MD3) in the tree topology, and every message in the data collection (black solid number circle ❶~❻) consists of three serial stages: encryption, transmission, and decryption. When an MD node (e.g. MD1) receives a security message in the top-down data collection (message ❶ from MD0), it decrypts the message first, then encrypts and prepare for new messages, and then transmits to its child nodes and waits for a reply. (MD1 decrypts message ❶, then encrypts two messages,

❷ and ❸, then transmits to its child nodes MD2 and MD3.) One MD does not send a secure message back to its parent node in the bottom-up phase until it receives the bottom-up messages from all its child MDs. (MD1 does not encrypt and send message ❻ to MD0 until MD1 receives ❹ and ❺ from MD2 and MD3.)

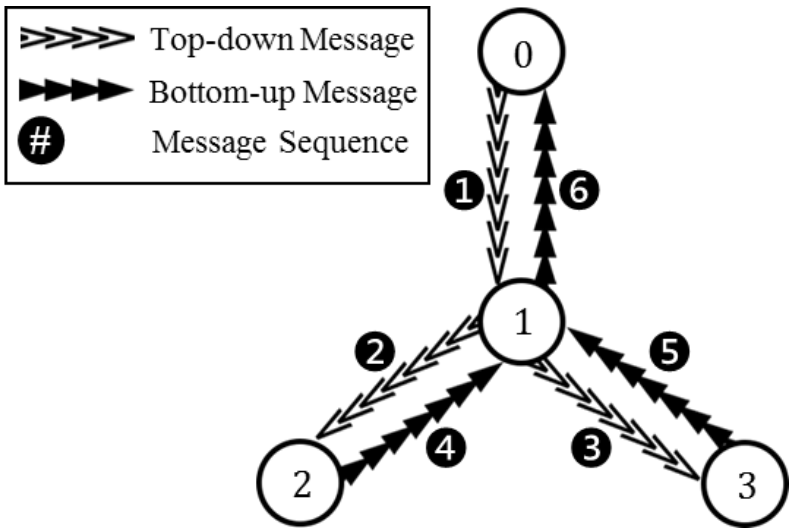


Figure 3.1: Top-down and bottom-up data collection mechanism

The top-down-bottom-up data collection mechanism can cooperate with many secure transmission protocols, such as [7] and [8], to ensure the confidentiality of messages. In this thesis, the cryptography is not our focus; we use a three-serial-stage message transmission model as a prototype secure message communication framework that cooperates with SMTDC for the simulations in the rest of the thesis.

3.2 Messages lifecycle analysis

As we mentioned before, the lifecycle of security messages for one half-duplex MD has three serial stages: encryption, transmission, and decryption. Let t_e be the time spent for encryption before sending one security message, t_t be the time spent for transmitting, and t_d be the time for decryption of one security message. So the time spent on one edge in the tree is equal to $t_e + t_t + t_d$. The specific values of t_e, t_t, and t_d can vary according to different security protocols and different hardware devices, but [30], [31], and [32] reach a consensus that the transmission time t_t is dominant compared to t_e, t_d.

To understand whether the transmission time is still a dominant portion in computationally-constrained devices, we conducted experiments in our test bed, where we use credit card sized Raspberry Pi [33] devices to emulate the wireless resource-constrained MDs. Raspberry Pi is a tiny, single-board computer (or embedded device) with limited computational, storage and communications capabilities. The CPU is 700 MHz and the memory available is 512MB. The Raspberry Pi is equipped with USB WiFi adapter, which has an internal chip antenna and is compatible with 802.11b/g/n; we use the mode of 150 Mbps 802.11n for our simulation. (Some of the specifications of Raspberry Pi are listed in Appendix A.)

Figure 3.2: Simple linear topology

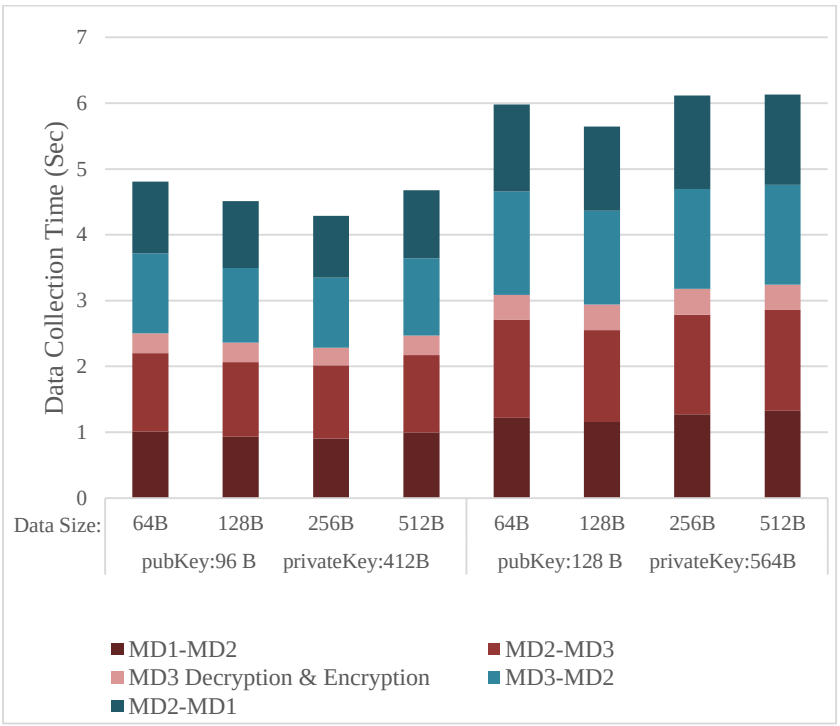


Figure 3.3: Total time line of linear topology

Under the simple topology shown in Figure 3.2, using our top-down-bottom-up mechanism with security protocol [6], we record the timeline of message passing from MD1 to MD2, to

MD3, then back to MD1 under different data sizes and key sizes using Raspberry Pi. Results are shown in Figure 3.3. 'MD1-MD2' means the time between MD1 receiving the message and MD2 receiving the message ($t_e + t_t$ of message from MD1, t_e for the encryption in MD1, and t_t for the transmission between MD1 and MD2). 'MD2-MD3' means the time from MD2 receiving the message until MD3 receives the message, which includes ($t_d + t_e + t_t$) as MD2 has to decrypt the message first ($t_d + t_e$ for the decryption and encryption in MD2). We find that transmission time t_t is about three times greater than encryption and decryption time together, $t_t \approx 3 * (t_e + t_d)$ in our protocol [6] under linear topology shown in Figure 3.2.

3.3 Data collection time model for tree with height one

In this section, based on previous observation of t_e, t_t, and t_d, we investigate the data collection time model for a tree with height equal to one, with only one parent node, and in which all leaf nodes directly connect to the parent node (tree topology shown in Figure 3.4). We define T_{round} as the total time needed for one parent MD node to gather data from its child MD nodes, in a tree with height equal to one in one round of data collection; $T_{top-down}$ is the time needed for the top-down phase of data collection, and $T_{bottom-up}$ is the time needed for the bottom-up phase of data collection in the above scenario.

We measure T_{round} with the number of child MD nodes in the range from one to ten, using the message encryption and decryption mechanism mentioned in security protocol [6] and assuming all MDs are half-duplex mode hardware-limited devices. We record the total time, comprising the data collection time of one parent MD with those of various numbers of child MDs.

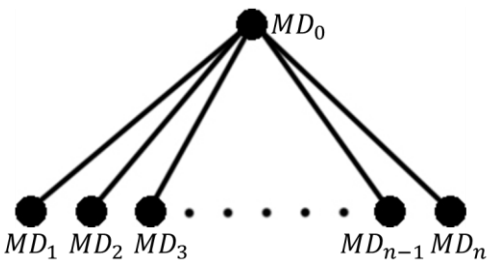


Figure 3.4: Parent-leafs topology

As shown in Figure 3.4, let us assume that in one round of data collection, one parent MD (MD_0) has n child MD nodes ($MD_1 \sim MD_n$). In hardware-limited devices, we find that one single process takes nearly 100% of CPU resources. In addition, they are all half-duplex devices, so the message from parent MD to child MD and the message from child MD to parent MD cannot be transmitted at the same time. As mentioned previously, one round of data collection in one level of the tree has two phases, top-down and bottom-up. For the topology in Figure 3.4, Figure 3.5 shows the top-down data collection timeline, where the x-axis represents the timeline for all MDs. We can see that in the top-down phase, the parent relay MD, MD_0, needs to first encrypt every message and send them to child MDs, $MD_1 \sim MD_n$, separately. The encryption and transmission of one MD have to be in serial order, so the parent MD_0 sends n secure messages to n child MDs, which takes $n * (t_e + t_t)$. And all child MD nodes can process the security messages in parallel; as shown in Figure 3.5, the decryption time, t_d, for $MD_1$and MD_2 overlaps with the encryption and transmission time of MD_0, but it still takes t_d time for the last MD node, MD_n, to finish decrypting the last top-down message. And we can see that $MD_1 \sim MD_{n-1}$ already finish decryption when MD_n receives a message from MD_0. Thus the total time for the top-down phase is $n * (t_e + t_t) + t_d$.

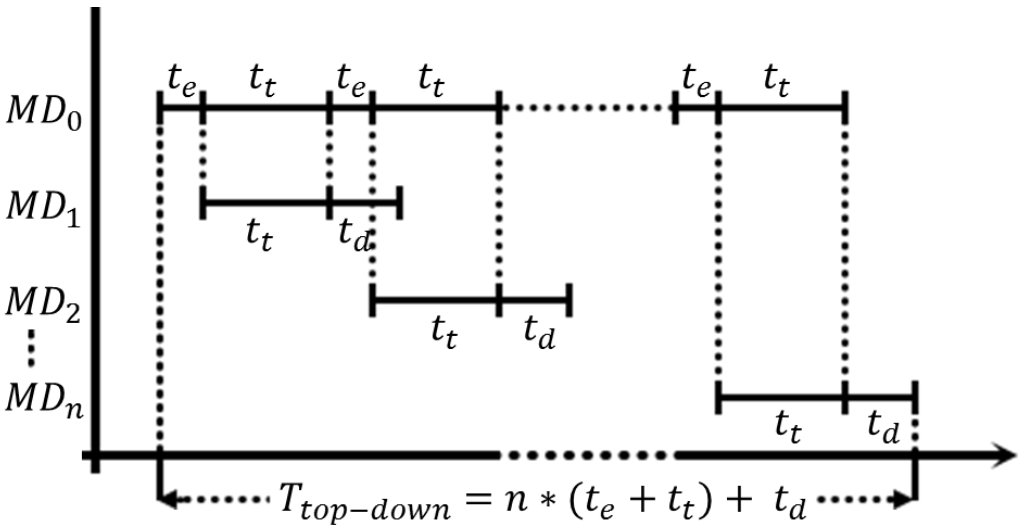


Figure 3.5: Top-down phase of data collection timeline for one parent MD and its n child MDs

Figure 3.6 shows the bottom-up data collection timeline for the topology in Figure 3.4. Similarly, in the bottom-up phase, all child MD nodes, $MD_1 \sim MD_n$, send secure messages to parent MD_0, and MD_0 takes $n * (t_d + t_t)$ to receive and decrypt all individual messages, but before the first child MD (MD_1 in the figure) transmits a message to MD_0, it has to encrypt the

message first. As shown in Figure 3.6, MD_1 takes t_e before MD_0 receives the first message, and we assume that $MD_2 \sim MD_n$ already finish encryption when MD_1 communicates with MD_0. As shown in Figure 3.5, all the t_e for MD_2~MD_n overlap with the timeline of MD_0; thus the total time for the bottom-up phase is $n * (t_d + t_t) + t_e$.

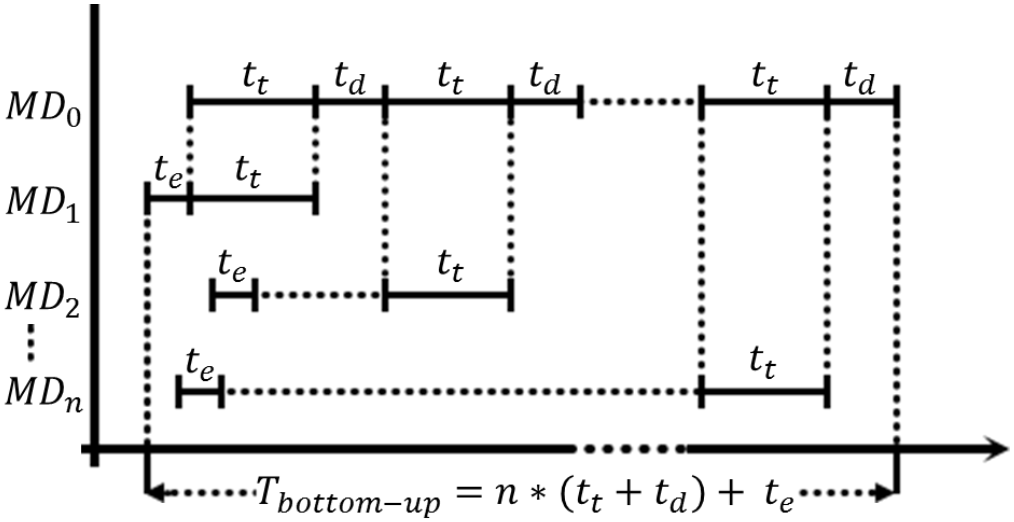


Figure 3.6: Bottom-up stage of data collection timeline for one parent MD with its n child MDs

Thus we can say that the time in which a parent MD in a certain level of the convergecast tree can finish one round of data collection, T_{round}, equals the time for both the top-bottom and bottom-up stages. Thus:

$$T_{round} = T_{top-down} + T_{bottom-up}$$

$$T_{round} = n * (t_e + t_t) + t_d + n * (t_d + t_t) + t_e$$

$$T_{round} = (n + 1) * (t_e + t_d) + 2n * t_t \quad (3.1)$$

And in one certain level, the average time spent for one child MD is:

$$\frac{T_{round}}{Num\ of\ child\ MD} = \frac{(n + 1) * (t_e + t_d) + 2n * t_t}{n} \quad (3.2)$$

We measure the T_{round} with child MD range from 1 to 10. The simulation result are shown in Figure 3.7, where the x-axis is the number of child MD nodes, n. The measured total time,

T_{round} (the red round-dots line), maps the primary vertical axis (the y-axis on the left). And the prediction time per child MD (the green dashed line) and the average time per child MD (the blue triangle-dots line) map the secondary vertical axis (the y-axis on the right). The prediction time per child MD equals the predicted T_{round} (equation (3.1), with given n, t_e, t_d, and t_d) divided by the number of child MD. The average time per child MD equals the actual measured time (T_{round} from simulations) divided by the number of child MDs. In Figure 3.7, we can see that our average time per child MD satisfies our prediction time model.

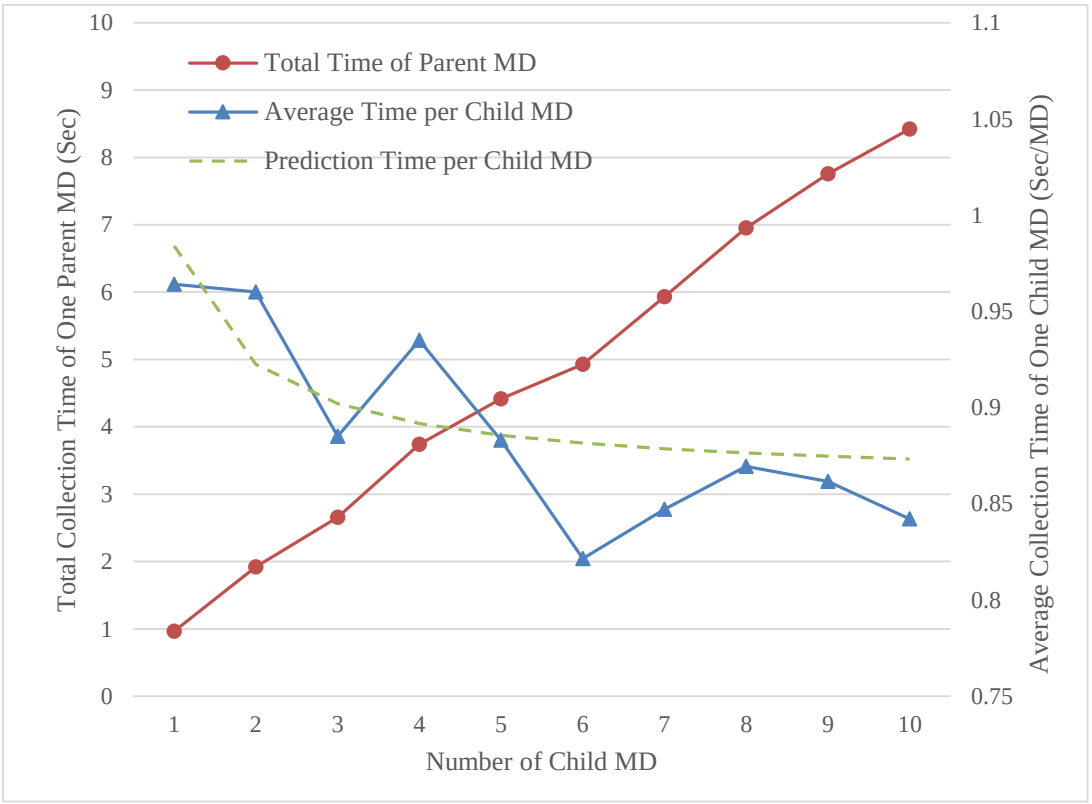


Figure 3.7: Simulation result of parent-leafs topology

3.4 Data collection time model for general trees

In this section we generalize our data collection time model to apply to all general convergecast trees, which have greater heights than the tree discussed in the previous section. And one of the major differences between this thesis and most of the previous work is that we integrate the load balancing function of the data collection time in one single level of the tree, denoted as $T_{level}(N_{sub})$, into our security data collection mechanisms.

Total data collection time of one convergecast tree is calculated as the summation of time spent in each level of the tree; the 'level' is the distance from one edge to the root MD. As mentioned in section 3.1, our security protocol collects the data initially from top to bottom, then waits for responses from bottom back to top, and we assume that different relay MD nodes in the same level can communicate with their child MD nodes at the same time. And one MD node must collect all data from its child MD node before sending back to its parent node. Thus, the total time in our data collection model is defined as follows:

$$T_{total} = \sum_{hi=1}^{H} (T_{top-down}^{hi} + T_{bottom-up}^{hi}) \tag{3.3}$$

Here, T_{total} is the total data collection time of an entire tree; H is the total level of the tree; $T_{top-down}^{hi}$, $T_{bottom-up}^{hi}$ are the top-down and bottom-up data collection time period in a given level, hi. As we mentioned before, for two different edges in the tree, if the parent relay MDs of these edges are different, then $t_e, t_t,$ and t_d of two different MDs can overlap in time. From Figure 3.5 and 3.6, we know that $T_{top-down}^{hi}$ and $T_{bottom-up}^{hi}$ largely depend on the maximum time of one parent relay MD in that specific level.

For example, in Figure 3.2, edge MD1-MD2 belongs to level 1, and edge MD2-MD3 belongs to level 2. The total time for data collection equals four continuous time periods: $T_{top-down}^{1}$ for transmission between MD1-MD2, $T_{top-down}^{2}$ for MD2-MD3, $T_{bottom-up}^{2}$ for MD3-MD2, and $T_{bottom-up}^{1}$ for MD2-MD1. In Figure 3.8, edges MD1-MD2, MD3-MD4, and MD3-MD5 are all in the second level. From the transmission model defined before, time spent in the second level depends largely on the collection time of subtree MD3, because time spent in subtree MD3 is longer than time in subtree MD1, and both times can overlap with each other.

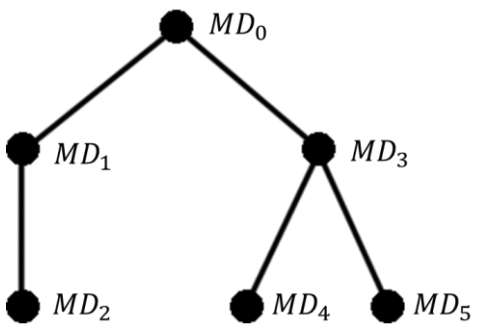


Figure 3.8: Example of tree structure

Constructing a convergecast tree and calculating the total data collection time under the best scheduling scenario with a single channel is shown to be NP-complete on general graphs by Choi et al. [34] and many other related works. We provide a heuristic method and generalize the total time in our data collection model as:

$$T_{total} = \sum_{hi=1}^{H} T_{level}^{hi}(N_{sub\_hi}) \tag{3.4}$$

where H is the total height of the tree, $T_{level}^{hi}(N_{sub})$ is the data collection time function for level hi, N_{sub_hi} is the average number of child MDs for the relay parent MDs in level hi. In Figure 3.8, MD0, MD1, MD3 are three relay MD nodes, and each of them has 2, 1, 2 MD nodes, so N_{sub_1} =2, $N_{sub\_2} = \frac{1+2}{2} = 1.5$.

Previously, we calculated the data collection time for one relay MD node with n child leaf MDs as T_{round}, $T_{round} = (n+1) * (t_e + t_d) + 2n * t_t$. We use T_{round} to simulate $T_{level}^{hi}(N_{sub\_hi})$; thus, we define:

$$T_{level}^{hi}(N_{sub\_hi}) = \left(N_{sub\_hi} + 1\right) * (t_e + t_d) + 2N_{sub\_hi} * t_t \tag{3.5}$$

We further simplify the definition of total data collection time of one tree, T_k, to:

$$T_{total}^{k} = T_{level}(N_{sub}) * y_k \tag{3.6}$$

$$T_{level}(N_{sub}) = (N_{sub} + 1) * (t_e + t_d) + 2N_{sub} * t_t \tag{3.7}$$

where N_{sub} is the average number of child MDs for all the relay parent MDs in the entire tree, and y_k is the height of tree T_k, which equals the number of edges from the farthest leaf MD to the root MD.

Now we simplify $T_{level}^{hi}(N_{sub\_hi})$ to $T_{level}(N_{sub})$; we define $T_{level}(N_{sub})$ as the data collection time for each level in the tree, and if the average number of child MDs for all the relay MDs equals to N_{sub}, we assume the data collection time for each level is the same. This simplification is based on the two following observations: (1) Equation (3.5) is approximately proportional to N_{sub}: In equation (3.5) we first calculate the specific time spent in each level, then calculate the summation to get the entire time for the tree. In equation (3.6) we first calculate the average time spent in each level of tree ($T_{level}(N_{sub})$) based on N_{sub}, then we

time it to the height of tree y_k. The difference between the two approaches is negligible. (2) In our SMTDC tree formation part introduced in the next chapter, we intend to build a load balancing tree in all levels of the tree, which has benefits in time scheduling and thus in minimizing the data collection time, as well as benefits in bearing unbalanced energy consumption and extending network lifetime as mentioned in Chen et al. [2].

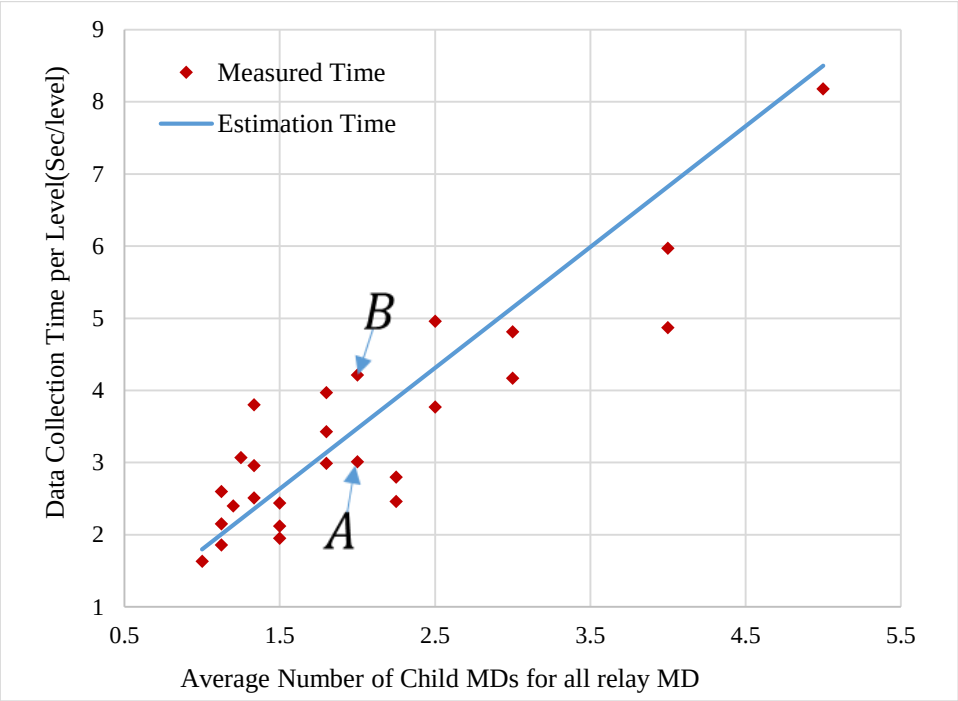


Figure 3.9: Average number of child MDs, N_{sub} vs. data collection time per level

To test the model of T_{total}^{k} (equation (3.6)), we measured data collection time obtained with 10~20 MDs, under various tree topologies. In Figure 3.9, the x-axis is N_{sub}, and the y-axis is the data collection time per level, which equals $\frac{T_{total}^{k}}{y_k}$. The solid line is our estimation of $T_{level}(N_{sub})$ using equation (3.7). One dot represents the total time per level for one specific tree topology; we can see that the scatter points are very close to our estimation line. Trees with the same N_{sub} have different data collection times per level, because the tree topology influences the total time. Dot A is the data of tree $\mathbb{T}_1$'s topology, which is shown in Figure 3.10 and dot B is tree $\mathbb{T}_2$'s topology shown in Figure 3.11.

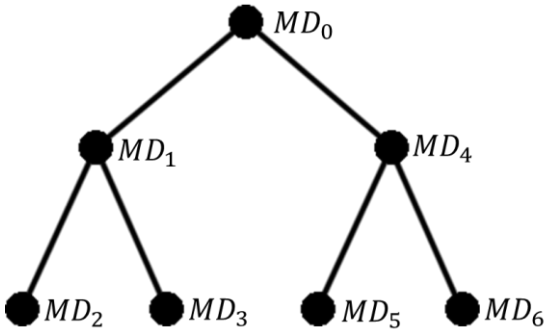


Figure 3.10: Tree $\mathbb{T}_1's$ topology

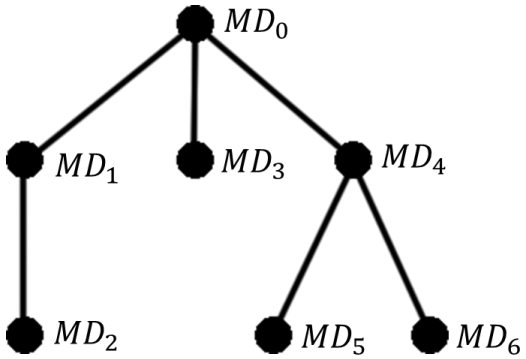


Figure 3.11: Tree $\mathbb{T}_2's$ topology

In a balancing tree, the load is equivalently distributed between all relay MDs to avoid concentrating all work in a small subset of relay MDs. For example, $\mathbb{T}_1$ and $\mathbb{T}_2$ in Figure 3.10 and 3.11 both have heights equal to 2 and N_{sub} equals 2. However, in$\mathbb{T}_2$, MD0 bears a higher forwarding load than MD1 because MD0 has three child MDs, and MD1 only has one. By contrast, $\mathbb{T}_1$ is more balanced in energy consumption, as all relay MDs, MD0, MD1, and MD4 in $\mathbb{T}_1$have equal numbers of child MDs.

Chapter 4 SMTDC DESIGN AND SOLUTION

Based on the previous chapter, we find that for a tree convergecast data collection model using a general security communication framework, every message needs three serial stages: encryption, transmission, and then decryption. Conceptually, there are three possible bottlenecks on the time of data collection: (1) the height of tree, (2) in each level of tree, the time spent for relay MD nodes to process each forwarding message and communicate with child MD nodes, (3) time spent on fetching local data. We can assume that (3) is a slight amount of time compared to message transmission and it can be eliminated by wisely using the interval between different rounds of data collection. Therefore, (1) and (2) become the fundamental obstacles to minimizing the data collection time.

In our SMTDC, we ask and answer two questions based on observations and experiments with hardware-limited devices in WSNs: (1) How can we build an optimized routing tree topology in a WSN which has relatively minimal height and is also well balanced, such that this tree topology potentially minimizes data collection time? (2) If this tree is built by following our intentions and constraints, how can we improve the data collection time even further by wisely scheduling the message forwarding sequence of MDs? At the same time, we also consider security issues such as security key leakage, eavesdropping, and DoS attack.

In this chapter we will introduce the processes of SMTDC protocol in detail, including SMTDC tree formation and SMTDC scheduling.

4.1 SMTDC tree formation

The first phase of SMTDC is the tree formation phase. In this phase, based on previous observation, we generalize the data collection time of a tree to the multiplication result of the total height of trees and time spent in each level of the tree. We first develop a linear optimization problem for minimum data collection time with fixed average number of child MD nodes for all relay MDs (N_{sub}), which is an NP-hard problem; then we propose an approximation algorithm (Algorithm 1) to solve this. Furthermore, we present a heuristic iterative method that uses Algorithm 1 to generate the tree topology for practical implementation when N_{sub} are unknown.

4.1.1 Notation definition of tree formation

Before we introduce our SMTDC tree formation, we need to define notations that are utilized in this work. Table 4.1 lists the notations. For a simple initial deployment shown in Figure 4.1, we plan to build a desired tree topology shown in Figure 4.2 based on this initial topology.
Let G be the background sensor nodes topology, and let $\mathcal{M}$ be the set for all MD nodes, $\mathcal{M} = \{1,2,a,b,c,d,e,f\}$. The data collector (DC), which is one mobile sink node, moves along a fixed predefined path with constant speed to collect data (the black vehicle in Figures 4.1 and 4.2). We assume that due to the limitation of transmission distance between two sensor devices (DC and MD, MD and MD), only a small number of MD nodes can directly communicate with the DC. We define these MD nodes with close proximity as potential root MD set, $\mathcal{R}$, in Figure 4.1; $\mathcal{R} = \{1,2\}$, k is potential root MD node, if $k \in \mathcal{R}$. And $\mathcal{R} \subseteq \mathcal{M}$.

For MD node $i \in \mathcal{M}$, it has set of neighbors $\mathbb{S}(i)$, as the potential parent node set. For example in Figure 4.1, $\mathbb{S}(a) = \{NULL, b, c, d, 1\}$, the $'NULL'$ element only exists in $\mathbb{S}(i)$, where i is a potential root MD node and i has no parent MD.

The term h is the distance (number of hops) from node i to node k (root MD node) in the final deployment. H_{max} is the max height of each tree and can be predefined as limiting the maximum height of one tree topology.
$X_{i,j,h}^{k}$ means that, in the final deployment, node i belongs to tree T_k and its parent node is $j, (j \in \mathbb{S}(i))$. And the distance from node i to root node k is h.

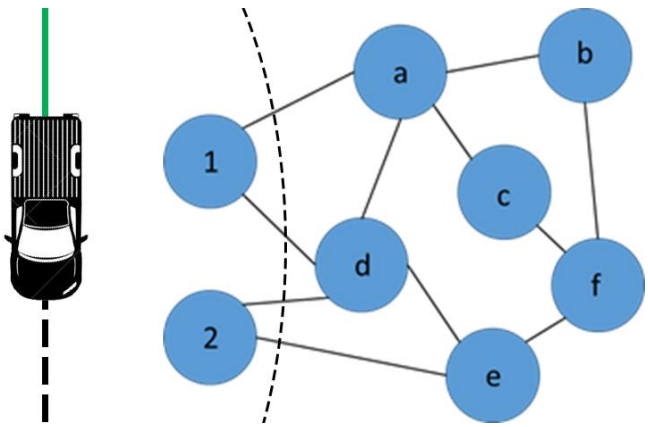


Figure 4.1: Initial deployment

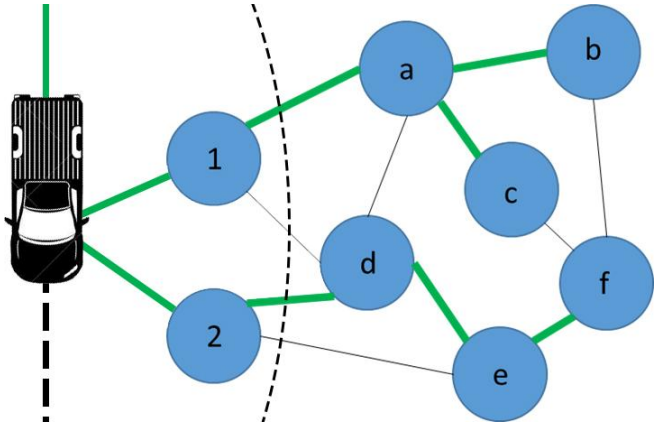


Figure 4.2: Final deployment

$$X_{i,j,h}^{k} = \begin{cases} 1, & if\ MD_i's\ data\ is\ collected\ from\ MD_k \\ & via\ parent\ node\ MD_j, and\ the\ distance \\ & from\ MD_i\ to\ MD_k\ is\ h \\ 0, & if\ MD_i's\ data\ is\ not\ collected\ from\ MD_k \\ & via\ parent\ node\ MD_j, and\ the\ distance \\ & from\ MD_i\ to\ MD_k\ is\ h \end{cases}$$

Thus, for all nodes in $\mathcal{M}$, we have $X_{i,NULL,0}^{i} = 1$, if node i is selected as a root MD node in the final deployment; otherwise, $X_{i,NULL,0}^{i} = 0$. We should mention that $NULL \in \mathbb{S}(i)$, for all $i \in \mathcal{R}$. We use y_k to represent the height of tree T_k, and y_k depends on the farthest node from node k in the tree; hence y_k satisfies:

$$y_k = max(h \bullet X_{i,j,h}^{k}), \qquad \forall i \in \mathcal{M}, \forall h \leq H_{max}, j \in \mathbb{S}(i)$$

or in problem definition we say:

$$y_k \geq h \bullet X_{i,j,h}^{k}, \qquad \forall i \in \mathcal{M}, \forall h \leq H_{max}, j \in \mathbb{S}(i)$$

For example, in Figure 4.2, in the final deployment node f belongs to tree T_2, and the distance from node f to node 2 is 3, and $X_{f,e,3}^{2}$ equals to 1, so $y_2 \geq 3 * X_{f,e,3}^{2}$, thus $y_2 \geq 3$.

As we mentioned earlier, we design SMTDC to be adapted to multiple security communication frameworks, not to some specific security protocol. We generalize different security challenges, such as Sybil attack, denial of service (DoS) attack, information spoofing, and key leakage, mentioned in [29], [35], to a simple probability problem. We assume that every MD node $i, (i \in \mathcal{M})$ encounters a security issue, with probability p_i (not identical). The probability that

the tree k encounters a security issue because of individual MD is $P_{attacked}(T_k)$ (the probability that T_k is attacked). We limit $P_{attacked}(T_k)$ to be no larger than a predefined threshold $P_{threshold}$.

TABLE 4.1. Notations

t_e	Encryption time of one message
t_d	Decryption time of one message
t_t	Transmission time of one message
G	Background sensor node topology
T_{round}	Time that one parent MD in a certain level takes to finish one round of data collection (formula 3.1)
$T_{top-down}$	Time that one parent MD in a certain level takes to finish top-down stage in one round of data collection
$T_{bottom-up}$	Time that one parent MD in a certain level takes to finish bottom-up stage in one round of data collection
T_{total}	Total time of data collection
$\mathcal{M}$	MD set
$\mathcal{R}$	Potential root MD set
$\mathbb{S}(i)$	Potential parent node set
h	Distance (number of hops) from one node to one root MD in the final deployment
H_{max}	Predefined maximum value of tree height
T_k	Tree with root MD_k
$X_{i,j,h}^k$	In the final deployment, MD_i belongs to tree T_k, its parent node is MD_j, and the distance from MD_i to root node k is h
p_i	Probability that MD_k encounters a security issue
$P_{attacked}(T_k)$	Probability that the tree k encounters a security issue because of individual MD
$P_{threshold}$	Threshold value for tree being attacked or key leakage
N_{min}	Minimum number of the balancing constraint
N_{max}	Maximum number of the balancing constraint
N_{sub}	Average number of child MD nodes for all relay MD nodes in the tree
dt_i	Data collection time needed for a subtree branch, whose root is MD_i
rb_i	The time from when an MD receives a top-down message from higher level until a subtree branch of this MD, MD_i, readies to send bottom-up message back to this MD.
Seq	Transmission sequence, for one relay MD

4.1.2 Tree formation with predefined N_{sub}

We define our SMTDC tree formation as follows:

Objective function:

$$\min T_{level}(N_{sub}) \sum_{k \in \mathcal{R}} y_k \tag{4.1}$$

$$s.t.\ y_k \geq h \bullet\ X^k_{i,j,h}, \qquad \forall i \in \mathcal{M}, \forall h \leq H_{max}, j \in\ \mathbb{S}(i) \tag{4.2}$$

where $T_{level}(N_{sub})$ is the time spent on each level of a tree, according to a certain N_{sub} (details in equation (3.7)); and $\sum_{k \in R} y_k$ is the total height of all trees in our final tree formations.

(1) Security constraint:

Assume that every MD node $i, (i \in \mathcal{M})$ encounters a security issue, such as key leakage or eavesdropping, with probability p_i (not identical). The probability that the tree k encounters a security issue because of an individual MD is $P_{attacked}(T_k)$. We limit $P_{attacked}(T_k)$ to be no larger than some predefined threshold $P_{threshold}$; then leakage constraint is as follows (details in Appendix B):

$$\sum_{i \in \mathcal{M}} \sum_{j\ \in \mathbb{S}(i)} \sum_{h=1}^{H} X^k_{i,j,h} \bullet \log(\frac{1}{1-p_i}) \leq log\left(\frac{1}{1-P_{threshold}}\right), \qquad \forall k \in \mathcal{R} \tag{4.3}$$

(2) Tree constraint:

Each node, $i \in \mathcal{M}$, is contained in only 1 tree, and it can only have one height, and only one parent:

$$\sum_{k \in \mathcal{R}} \sum_{j\ \in \mathbb{S}(i)} \sum_{h=1}^{H} X^k_{i,j,h} = 1, \qquad \forall i \in \mathcal{M} \tag{4.4}$$

Each node and its parent have to be in the same tree, and the parent's height, h, is one less than the node's height:

$$X_{i,j,h}^{k} \leq \sum_{l \in \mathbb{S}(j)} X_{j,l,h-1}^{k}\ , \quad \forall i \in \mathcal{M},\ k \in \mathcal{R}, j \in \mathbb{S}(i), \qquad l \in \mathbb{S}(j) \tag{4.5}$$

Constraints (4.4) and (4.5) prevent the redundant loop in the tree topology.

(3) Load balance constraint:

As all MD nodes are resource-constrained in terms of computation power, communication bandwidth, and storage capacity, we define the maximum number of child MD nodes as N_{max}, and the minimum number of child MD nodes as N_{min}.

$$\begin{gathered} N_{min} \leq \sum_{i \in \mathbb{S}(j)} X_{i,j,h}^{k} \leq N_{max}, \quad \forall j \in \mathcal{M}, k \in \mathcal{R}, h \leq H_{max} \\ X_{i,j,h}^{k} \in \{0,1\}, \qquad \forall i \in \mathcal{M}, \qquad k \in \mathcal{R}, \qquad j \in \mathbb{S}(i) \end{gathered} \tag{4.6}$$

Since adding N_{sub} to the load balance constraint will generate an exponential number of constraints to our problem definition, which takes exponential time to solve using a current optimization problem solver like GUROBI, we use a compromise solution, and use N_{max} and N_{min} to simulate N_{sub}.

$$\begin{gathered} N_{max} = N_{sub} * (1 + \alpha), \qquad N_{min} = N_{sub} * (1 - \alpha), \\ \alpha \in [0,1) \end{gathered} \tag{4.7}$$

where α is a predefined constant value. We intend to build a balanced tree, in which all relay nodes tend to have similar numbers of child MD nodes; thus, we use N_{max} and N_{min} to reduce the complexity of our calculation.

As we know, the SMTDC tree formation is a mixed-integer programming problem, which is also an NP-hard problem. Thus, we use linear relaxation-based iteration rounding (LR-IR) techniques in an approximation algorithm to solve this problem.

Algorithm 1: LR-IR for SMTDC tree formation

Input: $G, p_i, P_{threshold}, \mathcal{R}, \mathcal{M}, H_{max}, N_{sub}, t_e,\ t_d,\ t_t, \alpha$

Output: SMTDC{ $X_{i,j,h}^{k}$}

1: Preprocessing to determine the $N_{min}, N_{max}, T_{level}(N_{sub})$ and reduce the number of undetermined variables in solution set { $X_{i,j,h}^{k}$}.

a. Calculate $N_{min}, and\ N_{max}$ based on N_{sub} and α
b. Calculate $T_{level}(N_{sub})$ based on $N_{sub}, t_e,\ t_d,\ t_t$
c. Run Dijkstra's algorithm to get the shortest hops $dist_{i,k}$ for each MD node $i \in \mathcal{M}$, and Eliminate variable $X_{i,j,h}^{k}$, $for\ all\ \ h < dist_{i,k},\ \forall i \in \mathcal{M}$
d. Eliminate variable $X_{i,j,h}^{k}$, $for\ all\ \forall i \in \mathcal{M}, k = j, but\ h \neq 1, and\ h = 1, but\ \ k \neq j$
e. Eliminate variable $X_{i,j,h}^{k}$, $for\ all\ \forall i \in \mathcal{M}, k = i, but\ h \neq 0, or\ j \neq NULL$

2: **while** True do

3: solve the LP relaxation of the SMTDC with $X_{i,j,h}^{k} \in [0,1]$, and get the optimal fractional solution set { $X_{i,j,h}^{k*}$}.

4: round the largest fractional variable $X_{i,j,h}^{k*}$ in the solution set { $X_{i,j,h}^{k*}$} to 1 ($X_{i,j,h}^{k*} \in (0,1)$, and $X_{i,j,h}^{k*} \in$ { $X_{i,j,h}^{k*}$}).

5: check constraint (4.4), then set $X_{i,j,h}^{k} = 0$, for the same i in step 4.

6: **if** $\forall\ X_{i,j,h}^{k*} \in \{X_{i,j,h}^{k*}\},\ \ X_{i,j,h}^{k*} = 0\ or\ \ X_{i,j,h}^{k*} =\ 1\ (\forall\ i\ and\ j \in \mathcal{M}, k \in \mathcal{R},\ \ h \leq H_{max})$ **then:**

7: **return** solution set { $X_{i,j,h}^{k*}$}.

8: **end**

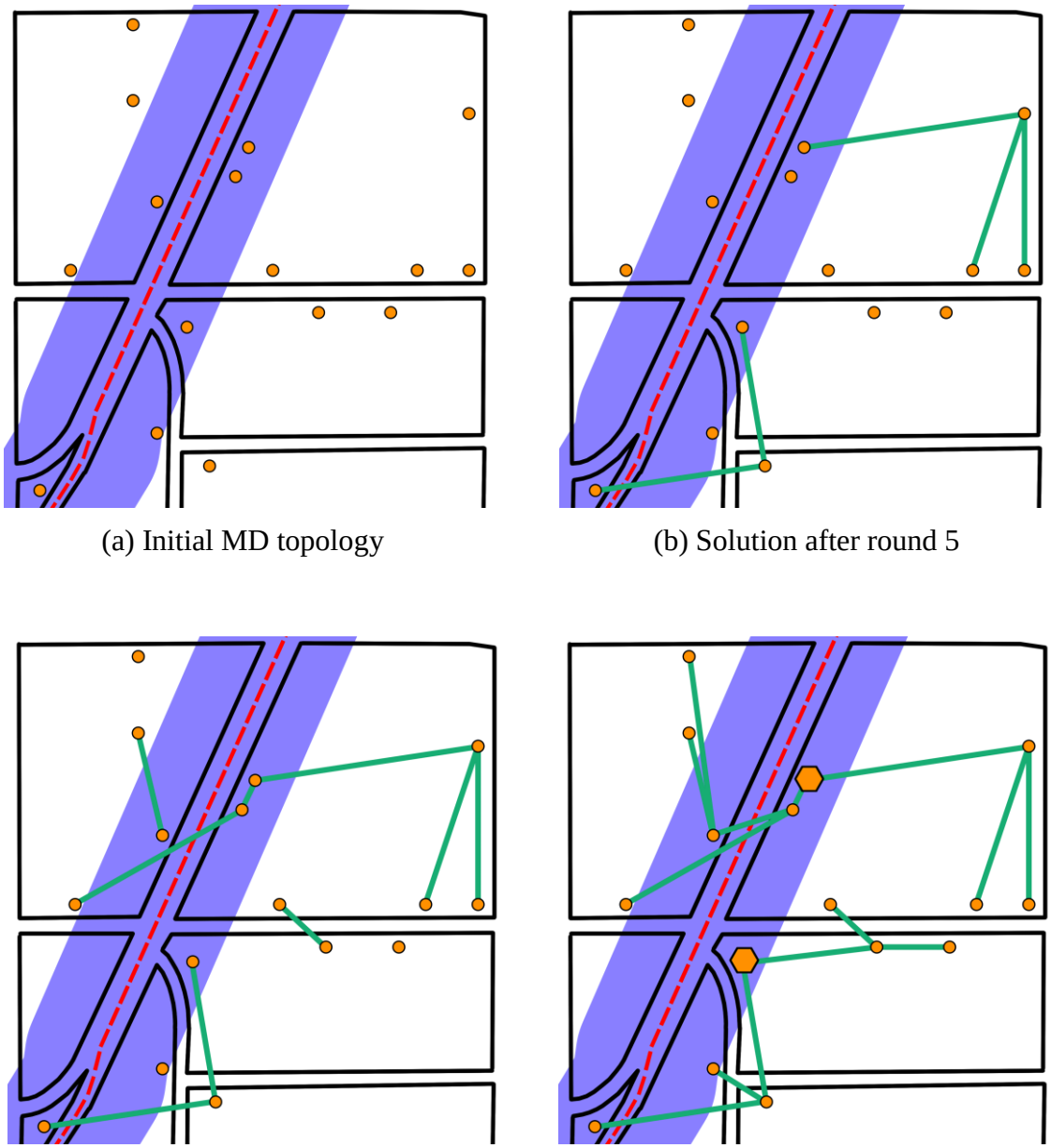
9: **end**

10: Root MD node selection: MD_i is root MD node if $X_{i,NULL,0}^{i} = 1$, for all $X_{i,j,h}^{k*}$ in solution set { $X_{i,j,h}^{k*}$}.

In the first step of Algorithm 1 (line 1), we first run Dijkstra's or a similar algorithm to get the shortest hops $h_{i,k}$ from all MD nodes to root MD nodes. Here $i \in \mathcal{M}$, $k \in \mathcal{R}$; as we cannot find a smaller distance than $dist_{i,k}$, we set all $X_{i,j,h}^{k} = 0$, for all $h < h_{i,k},\ \forall i \in \mathcal{M}$. In each round

of LP relaxation of Algorithm 1, we round the largest fraction solution $\{X_{i,j,h}^{k*}\}$ to 1, $for\ all\ \forall i \in \mathcal{M}, k = j, but\ h \neq 1$.

In the preprocessing part of the algorithm in line 1.d, if one MD node itself is also treated as the root MD node of one tree, the parent index can only equal to $NULL$, and the height index can only equal to 0. For example, when $i = j$, only variables like $X_{i,NULL,0}^{i}$ exist. We eliminate variable $X_{i,j,h}^{k}$, for all $\forall i \in \mathcal{M}, k = i,$ but $h \neq 0,$ or $j \neq NULL$.



(a) Initial MD topology

(b) Solution after round 5

(c) Solution after round 9

(d) Solution after round 13

Figure 4.3: The procedure of SMTDC tree formation

We show a simple case of the procedure of SMTDC tree formation of Algorithm 1 in Figure 4.3. The background topology is extracted from the Smart Grid (SG) data set of Washington DC [36], the red line is the trajectory of a DC node, the MD nodes in the blue shaded area belong to the potential root MD set, $\mathcal{R}$. Figure 4.3 (a) shows the original MD deployment. Figure 4.3 (b), (c), and (d) show the formation of trees during rounds of iteration (while loop, line 2 to line 9), after the data preprocessing part (line 1). For convenience, we set N_{sub} to 2, $P_{threshold}$ to 0.8 and p_i to an identical value for all MDs. In each round of iteration, one edge (the green line) is determined after rounding the largest fractional $X_{i,j,h}^{k*}$ to 1 (line 4), which means the entire tree topologies are built step by step, and every round of iteration decides one edge in the final tree topologies. Figure 4.3 (b) shows the solution after round 5, as we can see only five edges are connected, because we only have set five $X_{i,j,h}^{k*}$ to 1 during the first 5 rounds. Figure 4.3 (c) shows the solution after round 9; we can see that new nodes join the tree, and new edges are connected. Figure 4.3 (d) shows the final solution after round 13; the orange hexagons represent the MDs that are selected as root MD in the final deployment.

4.1.3 Iterative SMTDC tree formation

In the previous section, when given the value of t_e, t_d, t_t, and N_{sub}, the $T_{level}(N_{sub})$ is a constant value with equation (3.7), and the objective function (4.1) is linear. As we know, if the objective function is linear and the constrained space is a polytope, then tree formation with predefined N_{sub} is a linear programming problem, which can be solved using well-known linear programming solutions. However, if N_{sub} is not given, the objective function (4.1) is nonlinear. And a nonlinear programming problem is hard to solve and takes unrealistic computation time as a whole because of its high dimension.

From the observation in Chapter 3 (equation (3.7)), we can find that $T_{level}(N_{sub})$ is approximately proportional to N_{sub}; they have the same monotonicity (equation (3.7)). Besides, when N_{sub} increases, according to our definition, each relay MD tends to have more child MDs; thus, the total height of the tree decreases with certain amount of MDs in that area. So we can say when N_{sub} increases, $\sum_{k\in\mathcal{R}} y_k$ decreases. Moreover, in our previous observation, we have shown that for cases of extreme N_{sub}, the total data collection time is very large. For example, when N_{sub} closes to one, in this case every y_k is very large, because if N_{sub} is small, then the number of child MDs allowed for one relay MD is small and every tree is likely a line topology.

Another case is when N_{sub} closes to $log\left(\frac{1}{1-P_{threshold}}\right)$, in which every y_k is close to one and $\sum_{k\in\mathcal{R}} y_k$ is small, because when an MD is selected as a root MD or relay MD, it requires many child MDs (constraint N_{sub}), and its child MD number is close to the maximum number of nodes allowed in a tree, which closes to $log\left(\frac{1}{1-P_{threshold}}\right)$, such that the height of the tree (y_k) is likely one or two and total heights of all trees $\sum_{k\in\mathcal{R}} y_k$ is small. The total data collection time is large in the above two cases because there is hardly any parallel transmission in either line topology or start topology.

Based on the above observation, we give the iterative procedure in Algorithm 2 for SMTDC tree formation when the desired average number of child N_{sub} is unknown.

Algorithm 2: Heuristic iterative SMTDC tree formation

Initialization: Set round number $i = 0, N_{sub}^{(0)} = N_{sub}^{init}, \alpha = 0.5$

1: Obtain $\{X_{i,j,h}^{k}\}$ with **Algorithm 1:** LR-IR for SMTDC tree formation with $N_{sub} = N_{sub}^{(i)}$

2: Obtain data collection time $T_{total}^{(i)} = T_{level}(N_{sub}) \sum_{k\in\mathcal{R}} y_k$

3: **if** absolute value $\left|N_{sub}^{(i)} - N_{sub}^{(i-1)}\right| < \delta_1 \; and \left(T_{total}^{(i)} - T_{total}^{(i-1)}\right) < \delta_2$ then:

4: **return** solution of $\{X_{i,j,h}^{k}\}$ in round i

5: **else**:

6: Compute next round estimation $N_{sub}^{(i+1)}$, $N_{sub}^{(i+1)} = \frac{Total\ number\ of\ child\ Node, |\mathcal{M}|}{Total\ number\ of\ parent\ Node, |P|}$ based on solution of $\{X_{i,j,h}^{k}\}$ in round i

7: Increment i and go to line 1.

In Algorithm 2, i is the round number, and δ_1 and δ_2 are two predefined values. We first initialize $N_{sub}^{(0)}$ to some predefined value; based on this $N_{sub}^{(0)}$, we can determine the load balance constraint (4.6). Then we run Algorithm 1 and generate the tree topologies, which satisfy

constraint (4.6), and we get an estimated minimum data collection time $T_{total}^{(i)}$. In line 3, we check whether the $N_{sub}^{(i)}$ and $T_{total}^{(i)}$ already converge to a certain value; if they converge and satisfy conditions in line 3, it means tree topologies generated in round i and tree topologies generated in round $(i-1)$ are nearly identical, so we stop the iteration and return a solution of $\{X_{i,j,h}^{k}\}$. If $N_{sub}^{(i)}$ and $T_{total}^{(i)}$ do not satisfy conditions in line 3, it means we can generate a better topology for smaller data collection time, so in the new topology, we calculate the $N_{sub}^{(i+1)}$ to reset the load balance constraint (4.6).

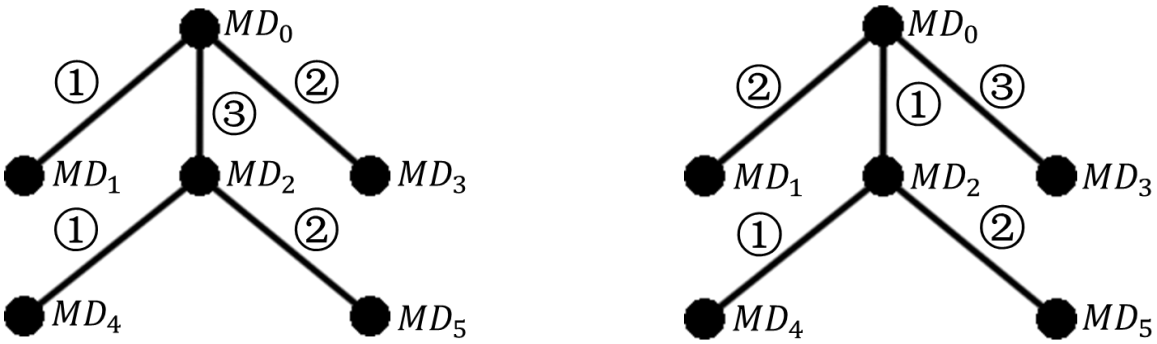
We will show in next chapter that this heuristic iterative algorithm works as designed and will converge within the limited number of rounds, and it will generate certain topology even with different initial values of $N_{sub}^{(0)}$.

4.2 SMTDC scheduling

We solve the second question that we brought up before: If the tree is built using the SMTDC tree formation approach, how can we improve the data collection time by solving the time scheduling problem? We take into account only the tree topology under the assumption that all interferences are eliminated, and each MD is in half-duplex model. For one single MD node, its transmission, encryption, and decryption times, t_t, t_e, t_d, have to be in different time slots. Also we consider the scheduling in the context of our top-down-bottom-up data collection model. Therefore, the SMTDC scheduling problem becomes: given one tree topology, decide the encryption and transmission sequence of child MDs in each relay MD node.

Figure 4.4 presents a quick example to illustrate the influence of different scheduling approaches on the total data collection time under the same tree topology. A circled number, such as ①, ②, ③, represents the priority that one parent relay MD node refers to, when the relay MD wants to send the initial top-down encryption message to its child MD nodes. In Figure 4.4 (a), when MD0 receives a message from a higher level and plans to forward the message to its child MD nodes, it first sends a security message to MD1 and MD3. After MD1 and MD3 receive the message, MD0 then sends a message to MD2. The timeline of the topology in Figure 4.4 (a) is shown in Figure 4.5. Another scheduling arrangement is shown in Figure 4.4 (b), where MD0 sends a message to MD2 first, then to MD1 and MD3. The timeline of the topology in Figure 4.4 (b) is shown in Figure 4.6. One obvious benefit is that as the

message arrives at MD2 earlier, the data collection time spent in the MD2 subtree branch can overlap with the time in which MD0 talks to MD1 and MD3. The $T_{overlap}$ in Figure 4.6 indicates that MD0 talks to MD1 and MD3 while MD2 talks to MD4 and MD5. So in the second scheduling arrangement, even in level 2 (MD4, MD5, N_{sub} is 2), it takes the same amount of time as the first approach, but in level 1 it can be considered that MD0 only talks to MD2.



(a) (b)

Figure 4.4: Example of scheduling arrangement for same topology

So the total time for the tree topology in Figure 4.4 (b) equals time spent in level two plus time spent in level one, $T_{level(2)}(N_{sub} = 2) + T_{level(1)}(N_{sub}=1)$. Based on equation (3.3) we can get the total time of the tree in Figure 4.4 (b), $T_b = 5(t_e + t_d) + 6t_t$. As shown in Figure 4.5, for tree topology in Figure 4.4 (a), it takes additional $2(t_e + t_t)$ because MD0 has to transmit the message to MD1 and MD3 first, so the total data collection time for the tree topology in Figure 4.4 (a) is $T_a = 5t_d + 7t_e + 8t_t$. Our experiment using Raspberry Pi shows that T_b roughly equals 72% of T_a, which proves that wisely scheduling the message transmission sequence of relay MDs can minimize the data collection even further.

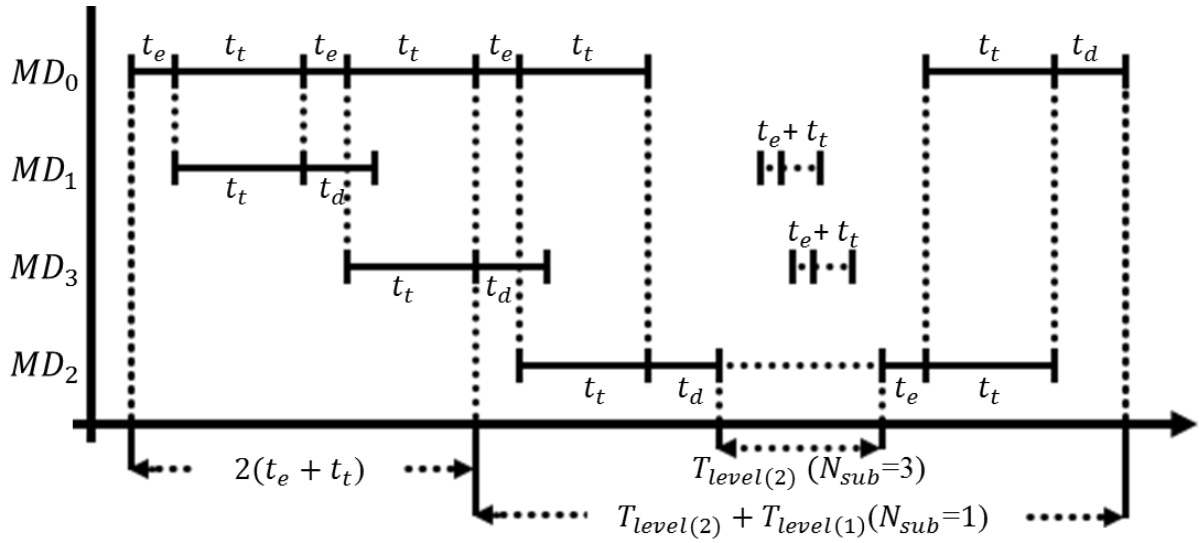


Figure 4.5: Timeline for the topology in Figure 4.4 (a)

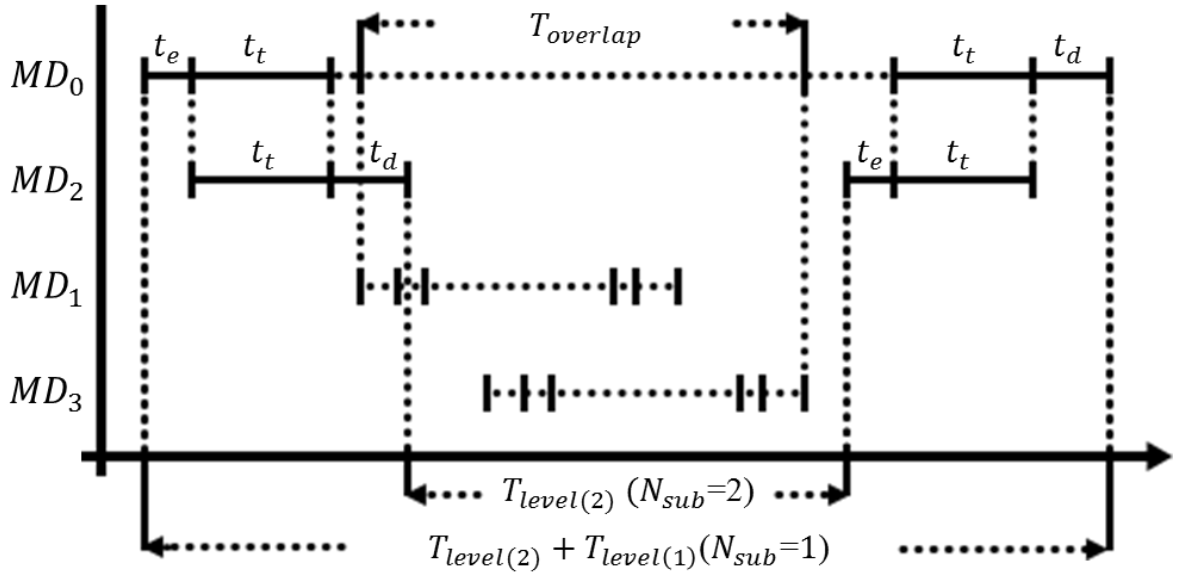


Figure 4.6: Timeline for the topology in Figure 4.4 (b)

We describe a time slot assignment schema in Algorithm 3, called the SMTDC -Scheduling arrangement, which is run locally by each MD. The key concepts of this algorithm are: (1) In the convergecast tree, let the transmission of different branches be as parallel as possible, such that the data collection time of subtrees can overlap in order to save the total time. (2) For every relay parent MD, it puts its child MD, whose subtree takes longer time in data collection, in top priority when the parent MD is forwarding messages to all of its child MDs. The reason one subtree branch takes longer than another is that it is higher, or there are more MD nodes in

the subtree, etc., so we distribute the task to this subtree first. We use dt_i to represent the data collection time needed for a branch subtree, whose root is MD_i.

Algorithm 3: SMTDC Scheduling Arrangement

After the tree topology is determined by SMTDC tree formation, each MD node already knows its parent and child MD nodes, based on $\{X_{i,j,h}^k\}$. Every MD node uses this algorithm to determine the message forwarding order.

Initialization:

a. Total branch data collection time, dt: 0

b. Children data time set CDT ← Ø,

c. Reply back time set RBT← Ø,

d. Child MD transmission sequence $Seq \leftarrow \emptyset$

1: **if** MD_i is a leaf MD node, set $dt = t_e + t_d$.

2: return dt to its parent MD

3: **done**

4: **else if** MD_i is a relay MD node:

5: wait until MD_i receives all dt messages from its j child MD nodes ($dt_1 \sim dt_j$)

6: child data time set: CDT ← $\{dt_1,\ dt_2,\ \ldots, dt_j\}$

7: sort CDT in **decreasing** order: $\{dt_1',\ dt_2',\ \ldots, dt_j'\}$

8: set child MD transmission sequence, Seq, based on CDT in decreasing order.

9: calculate reply back time for every child MD: rb_j

10: $rb_j \leftarrow dt_j' + j * (t_e + t_t)$

11: construct reply back time set RBT ← $\{rb_1,\ rb_2,\ \ldots, rb_j\}$

12: sort RBT in **increasing** order: $\{rb'_1,\ rb'_2,\ \ldots, rb'_j\}$

13: **for** each rb'_j in increasing order in RBT:

14: $dt = Max(dt, rb'_j) + (t_t + t_d)$

15: **end for**

16: return dt back to its parent MD node

17: **end if**

For every MD node running Algorithm 3, it first judges whether it is itself a leaf MD. If it is itself a leaf MD, then it does not need to consider the transmission sequence because it has no child MD. Thus it simply replies with the time needed for message encryption and decryption, $t_e + t_d$. If one MD is a relay node, it needs to decide the transmission sequence, Seq, for all of its child MD nodes. And it also needs to calculate and return the time needed for its own subbranch. The Seq is decided by the amount of time needed by its child subtrees, where the parent relay MD node transmits first to the subtree that takes longer time (line 8, Algorithm 3). The total time of one subtree branch is calculated in lines 11 to 15. In the child data time set, CDT, every element, dt_i, indicates the time needed for the MD_i subtree to finish the data collection. In the reply back time set, RBT (in line 12), every element rb_j is the time when a child MD node, MD_j, is ready to send a message back to the parent MD (but has not sent yet). So in line 10, rb_j equals dt'_j plus $j * (t_e + t_t)$, where $j * (t_e + t_t)$ is the time needed for MD_j to receive a message from its parent MD, as MD_j's subtree data collection time is ranked number j among all its parent MD's child nodes. And when we calculate dt, we consider that every bottom-up message back to the parent MD has to add one t_t for message transmission and one t_d for the parent MD to decrypt the message.

As in both line 7 and line 12, it takes $O(NlogN)$ time to sort the CDT and RBT, it is obvious that the worst-case running time of Algorithm 3 is $O(NlogN)$, where N is the number of child MD nodes in one relay MD.

Chapter 5 PERFORMANCE EVALUATION

In this chapter, we evaluate the performance of the proposed SMTDC protocol under different settings and topologies to see how factors such as the probability of encountering a security problem, load balancing limitation, size of tree, and scheduling arrangement impact the performance of data collection time for the SMTDC protocol.

5.1 Performance of SMTDC tree formation

We run experiments of SMTDC tree formation on the SG data set of Washington DC [36], which contains 8, 20, 35, 50, 100, 150 nodes, of all the smart grid poles in the city. Figure 4.1 contains 8 nodes, and two of these nodes are root MD nodes. We assume that MDs communicate with each other wirelessly with an identical communication range of 75 m. We use the GUROBI solver [37] for solving the LP relaxation in the main while loop of our Algorithm 1, and we use different Python scripts to do the data preprocessing part. Figure 4.2 and Figure 4.3 (d) show the simulation result of our algorithm on the 8-node and the 16-node sample case. For our simulation, the algorithm works as designed.

5.1.1 Simplified simulation for tree height

Figure 5.1 shows the comparison of total tree depth of the generated tree from our SMTDC, CTF in [6], and random tree algorithm. In CTF, it tries to build trees that have minimum height summation, and it assumes the link delay for one level in the tree, which is the time for one MD node to talk to any number of child MD nodes, is identical for hardware-limited devices. In the random tree algorithm, every MD randomly selects a neighbor and builds a random tree. Because CTF and random tree algorithm do not consider the load balancing factor and they assume p_i is identical for $i \in \mathcal{M}$. In SMTDC, we set $T_{level}(N_{sub})$ in our objective function (4.1) to be constant value '1', thus our objective function (4.1) becomes:

$$\min \sum_{k \in \mathcal{R}} y_k$$

We loosen the load balance constraint (4.6); we let N_{max} be 12, such that the load balance constraint will not be an primary constraint when we build the tree; and we set all p_i to 0.5%, $P_{threshold}$ to 20% to simply our algorithm.

In this simulation we prove two points: First, in simplified tree formation, where minimizing the total summation of heights is the only objective, our SMTDC tree formation and Algorithm 1 work as desired, and provide an even better solution compared to the CTF and the random tree algorithm. Second, SMTDC tree formation can be adapted to a more sophisticated and realistic scenario. We consider additional factors such as time for transmission and data procession, security issues, and load balancing issues, all of which are a huge leap from previous work.

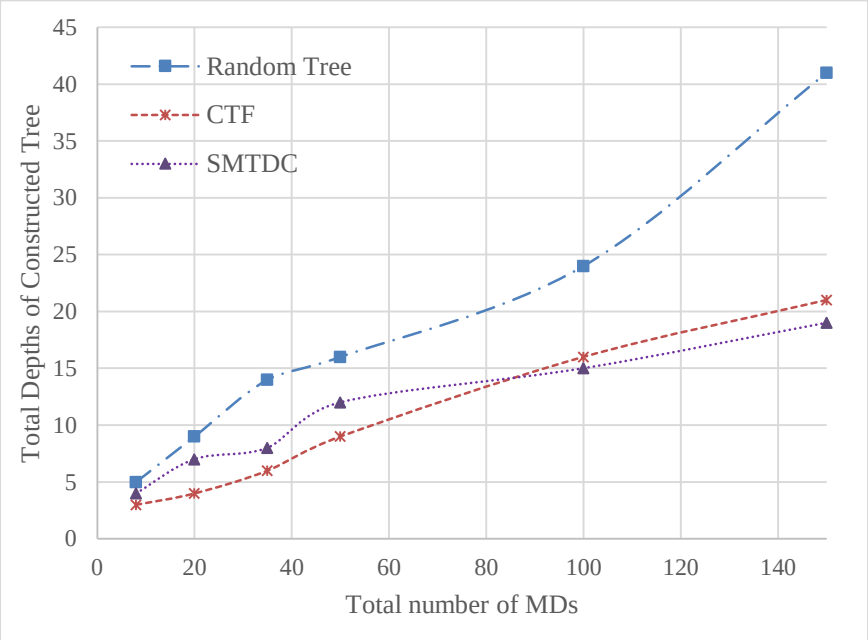


Figure 5.1: Comparison of the total tree depth of SMTDC, CTF, and random tree algorithm

5.1.2 Influence of $P_{threshold}$

To simulate the relationship between the probability threshold of encountering security problems of each tree, $P_{threshold}$, and the data collection time, T_{total}, we fix the number of MDs to be 100, and we choose $N_{sub}$to be 3 and 4, and use the data collection model mentioned in section 3.4. In addition, we use the data from the total time of parent MD vs. number of child MD nodes in Figure 3.7 to simulate the time spent on each level of the tree. We assume the probability p_i satisfies a roughly normal distribution, with average value equal to 0.005 and its standard deviation equal to 0 0.005, but all probabilities within the range of $(0, 0.01]$. We vary the threshold probability of one tree, $P_{threshold}$, from 0.05 to 0.3.

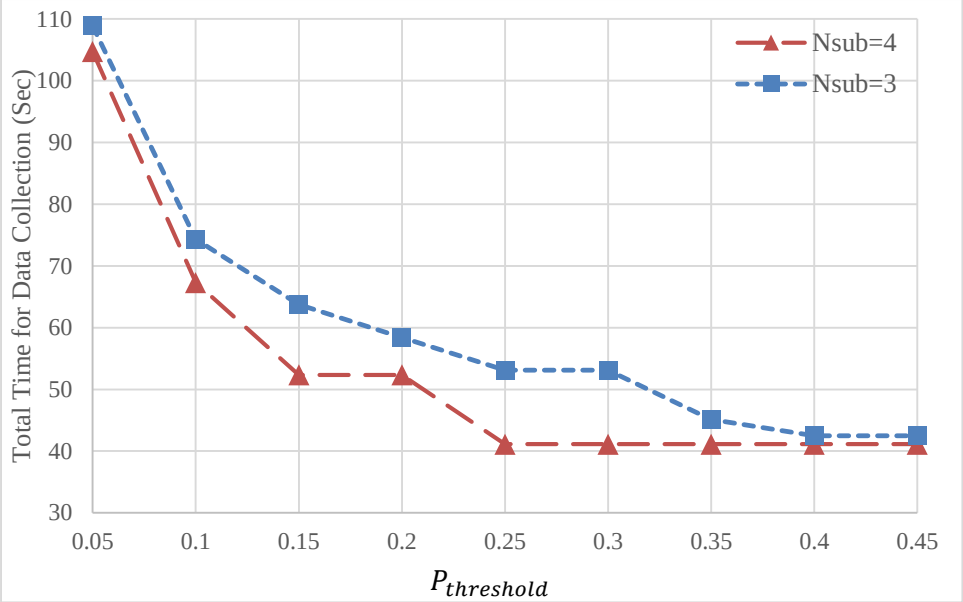


Figure 5.2: Data collection time vs $P_{threshold}$

Figure 5.2 shows our simulation result, and this result reflects the trade-off between security and efficiency of data collection in our problem formation. The reason is that the higher the value of $P_{threshold}$, the higher the number of MD nodes allowed in one single tree (from constraint (4.3)), and each tree tends to be higher. On the other hand, the smaller the value of $P_{threshold}$, the smaller the number of MDs allowed in a tree and each tree in the final deployment tends to be shallow. As the total number of MD nodes in the entire background topology is fixed, in the first case, when the $P_{threshold}$ is large, the number of trees is small, but each tree has higher height. In the second case, when the $P_{threshold}$ is small, the total number of trees is very high, and each tree has low height.

And we know, in a convergecast tree with certain value of N_{sub}, the total number of nodes in one single tree grows exponentially when the height increases. Thus when N_{sub} is fixed, in the first case (when $P_{threshold}$ is large), the summation of heights for all trees is small, and has a shorter total data collection time, while in the second case (when $P_{threshold}$ is small), the summation of heights for all tree is large compared to the first case, and has a higher total data collection time.

In addition, in our simulation the SMTDC yields the same result when $P_{threshold}$ is larger than 0.25 because $P_{threshold}$ influences the number of MD nodes allowed in one tree. When $P_{threshold}$ is larger than 0.25, the number of MD nodes allowed in one tree is more than 57, at which point constraint (4.3) is no longer a determining factor for our SMTDC tree formation.

In contrast, when $P_{threshold}$ is smaller than 0.1, the number of MD nodes allowed in one tree is smaller than 18, at which point every tree is forced to be shallow because of constraint (4.3), thus leading to more trees and greater total height in the WSN, as mentioned previously.

5.1.3 The influence of load balancing factor

We simulate the relationship between load balancing factor and the data collection time. We still use the topology with 100~150 MD nodes. We assume the threshold probability of one tree, $P_{threshold}$, to be fixed to a certain value, 0.15, and we vary the average number of child MD nodes in the tree, N_{sub}, from 1.125 to 6. We run experiments on three groups of data; for each group, p_i satisfies a normal distribution, with average value equal to 0.005, 0.01, 0.015, but falls within the range of $(0, 0.02]$. And we use the function $T_{level}(N_{sub})$ of equation (3.7) to simulate the time spent on each level of the tree. We have to mention that the total number of MD nodes is not exactly the same after the tree SMTDC formation process (but within the range of 100~150 MDs), especially for N_{sub} larger than 5. The reason is that it requires minor adjustments for some relay MDs to satisfy N_{sub} constraint (4.6) and (4.7). But this adjustment does not influence the predicting data collection time in SMTDC tree formation.

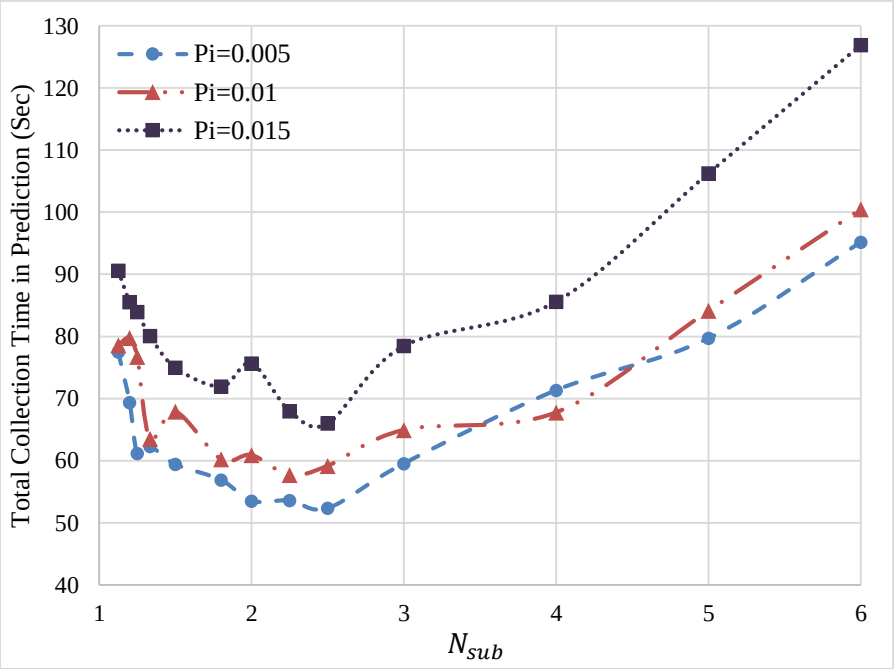


Figure 5.3: Total collection time (Sec) vs. N_{sub} for different p_i

In Figure 5.3, we show the prediction time for data collection versus load balancing parameter N_{sub} in a 100+ MDs topology. This observation reflects the trade-off between load balancing and efficiency. More specifically, if load balancing parameter N_{sub} is too small, e.g. N_{sub} less than two, the summation of heights/levels for the entire topology tends to be large. Even if the time spent in each level is relatively small (equation (3.7)), the multiplication result of the total levels and the time spent in each level (formula (4.1)) is large, hence it has a longer data collection time. On the other hand, if balancing parameter N_{sub} is too large, e.g. N_{sub} larger than 5, even though the summation of heights for the entire topology tends to be small, according to our prediction model (equation (3.7)) the time spent on each level becomes longer (a single half-duplex device has to communicate with its child nodes sequentially, without parallel communication), and the prediction time (the multiplication result) according to our objection formula (4.1) increases again. The simulation result in Figure 5.3 matches our observations previously that, with total number of n hardware-limited half-duplex MD nodes, the data collection time of a line topology, in which N_{sub} is one, and a star topology, in which N_{sub} is $n-1$, is much longer than those of other tree topologies. The reason is that there is no parallel communication for relay MD nodes in line topology and star topology.

We have to mention that when p_i is smaller than 0.005, equally, the number of MDs allowed in one tree is more than 44.5, which means that, for a 100 MD topology, constraint (4.3) is no longer a determining factor for our SMTDC tree formation algorithm. But when p_i is larger than 0.015, the number of MDs allowed in one tree is smaller than 15; at that time, constraint (4.3) becomes a determining factor. And each tree allows fewer MDs than the case when p_i is 0.005 or 0.01, thus our objective function (4.1) becomes larger.

5.1.4 Simulation of iterative SMTDC tree formation algorithm

In this part we test our heuristic iterative SMTDC tree formation algorithm (Algorithm 2, mentioned in section 4.1.3). Rather than specifying a certain value of N_{sub} and bringing the predefined N_{sub} into Algorithm 1, we initialize $N_{sub}^{(0)}$ to three different values and run Algorithm 2 under the same map of 100+ MDs, and we choose p_i to satisfy a roughly normal distribution, with average value equal to 0.01, and p_i within the range of $(0, 0.02]$. In addition, t_e, t_d, t_t, and α all choose the same settings as the second group of experiments in the previous section.

Figures 5.4 and 5.5 show the result of our simulation. In these figures, the x-axis represents the round of iterations. The y-axis in Figure 5.4 represents the value of $N_{sub}^{(i)}$ in round i, and the y-axis in Figure 5.5 represents the value of $T_{total}^{(i)}$ in round i. The large marker dots in both figures represent the round after which Algorithm 2 converges to a certain value and terminates.

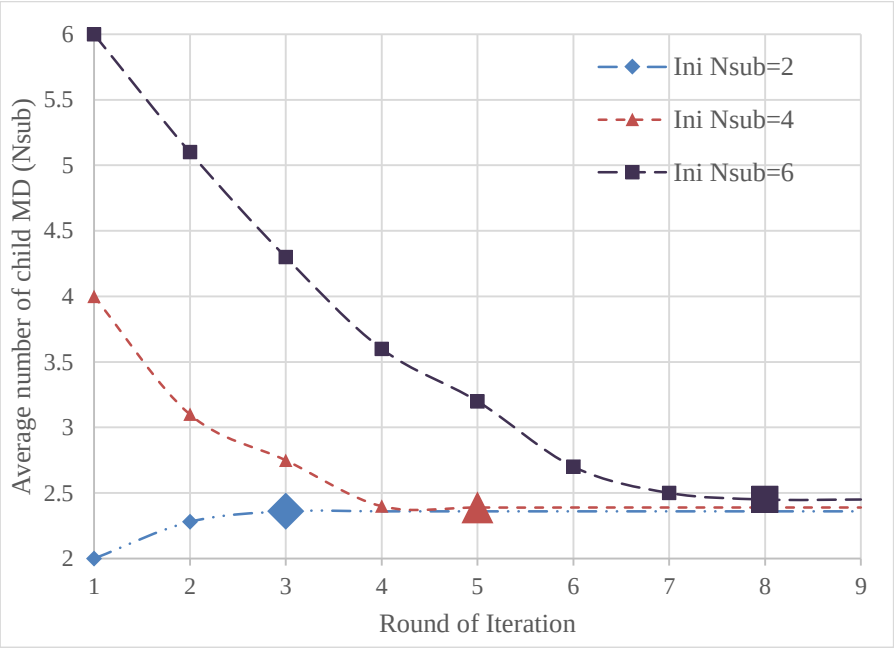


Figure 5.4: $N_{sub}^{(i)}$ vs. round of iteration

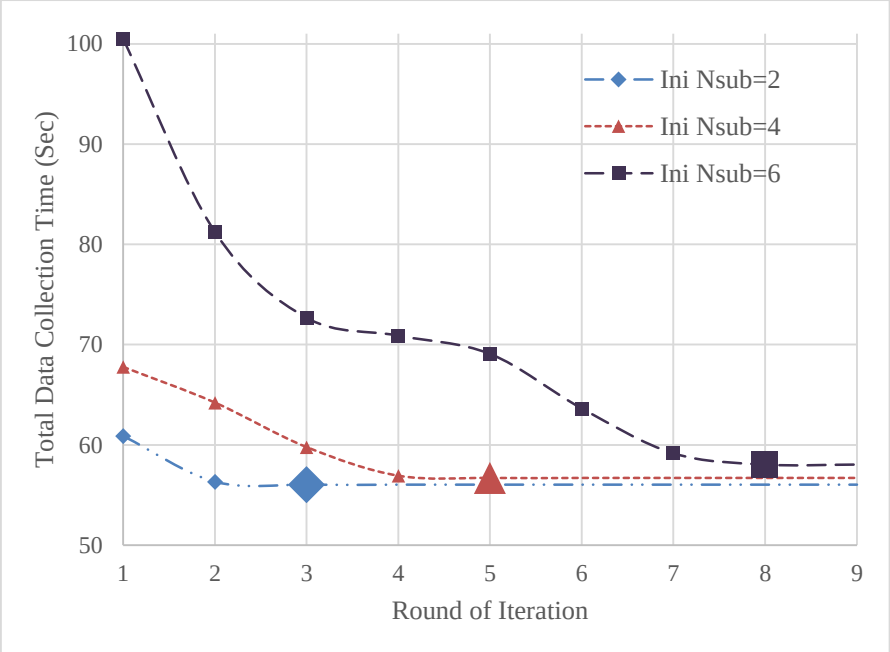


Figure 5.5: Total data collection time vs. round of iteration

We can see from both groups of experiments that, when the round of iteration increases, the $N_{sub}^{(i)}$ and the total time converge to a certain value, which means after a certain number of rounds of iteration, the tree topologies do not change too much between two iterations (the termination points are the large markers in the two figures). In Figure 5.5, we can also see that when the rounds of iteration increase, the total data collection time decreases, because in each round of iteration, we use Algorithm 1 to generate a better tree, and we change the load balancing constraint (equations (4.6) and (4.7)) before we run the next round of iteration.

5.2 Performance of SMTDC scheduling

In this section, we analyze the performance of the SMTDC scheduling assignment method discussed in section 4.2. Specifically, after we generate a tree topology using the SMTDC tree formation method in section 4.1, we apply our scheduling arrangement approach to observe the improvement of data collection time compared to the expectation model in section 4.1. We emphasize again that the total data collection time in section 4.1 is calculated by using the objective function (4.1) in the SMTDC tree formation part. The main purpose of section 4.1 is to build optimized tree topologies with the potential for fast data collection by using the SMTDC scheduling approach in section 4.2. So, in this section, the total data collection time is recalculated by our SMTDC scheduling approach, which tends to be smaller and more precise than the prediction time in section 4.1.

We also compare SMTDC scheduling to random scheduling method, and all simulations of scheduling arrangement methods under OMNET++. We use a background topology with about 100 MD nodes. The $P_{threshold}$ is fixed to a certain value, 0.15. And p_i satisfies a roughly normal distribution, with the average value equal to 0.01, and p_i within the range of $(0, 0.02]$. We vary the average number of child MD nodes in the tree, N_{sub}, from 1.125 to 6 to generate different tree topologies (using Algorithm 1). For each specific tree topology with certain N_{sub}, we can get three data collection times; one expectation time is calculated by objective function (4.1) and another is generated by running the random transmission scheduling model, where every relay MD in the tree topology sends a top-down message by random sequence order to its child MDs, and every relay MD receives a bottom-up message with FIFO sequence. The third data collection time is generated by our SMTDC scheduling approach. These three groups of data are presented in Figure 5.6. Our SMTDC scheduling approach shows a great

improvement over the SMTDC time expectation model and performs better than the random scheduling approach. We can see from Figure 5.6 that SMTDC scheduling has a great improvement especially for large N_{sub}; e.g., when N_{sub} larger than 5, SMTDC scheduling is 45% faster than SMTDC prediction.

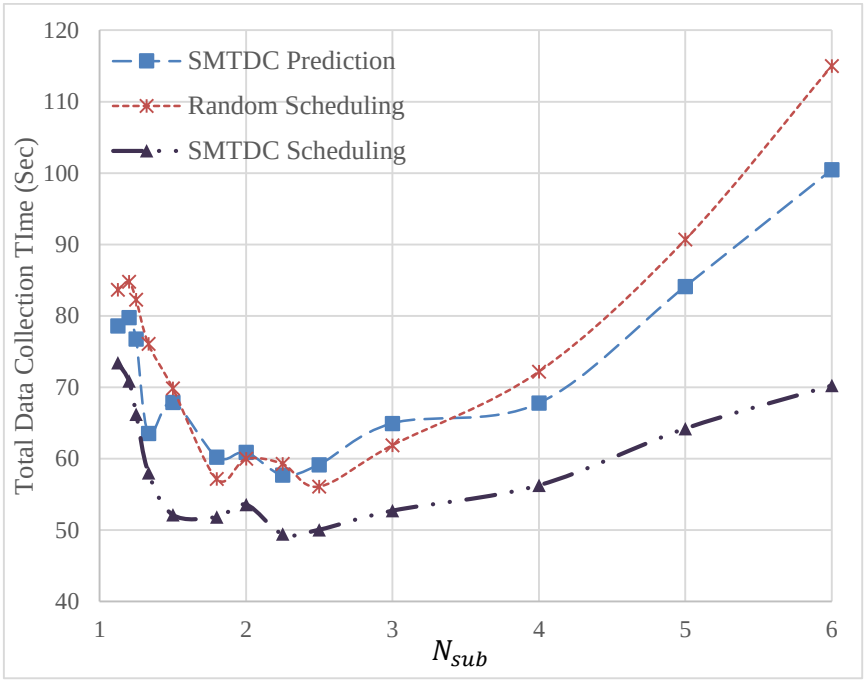


Figure 5.6: Comparison of random scheduling, SMTDC prediction model, and SMTDC scheduling

Chapter 6 CONCLUSION

In this thesis, we studied the secure minimum data collection time in large scale, stationary WSNs where nodes are hardware-limited devices and they communicate in a half-duplex model. We proposed a load balanced, secure, and efficient data collection protocol, called SMTDC, with path-constrained mobile data collectors. The SMTDC can be adapted and cooperate with any universal security communication protocol where message passing can be separated into three serial stages: encryption, transmission, and decryption. SMTDC has two phases, the first of which is to build optimized convergecast tree topologies in the WSN which have relatively minimal height, and are also well balanced. A heuristic approximation algorithm by means of a linear relaxation is presented to build optimized trees, and the algorithm calculates one expectation time with the predefined average number child nodes, N_{sub}. When N_{sub} is uncertain, we use an iterative method to get close-to-optimal tree topologies. To further improve the data collection time from our expectation, in the second phase of SMTDC, we develop a scheduling arrangement algorithm, with a heuristic idea that lets the time spent on each subtree overlap with other subtrees as much as possible. Through simulations conducted using real topologies, we demonstrated that SMTDC performs well, as planned. To the best of our knowledge, the SMTDC is the first protocol used for minimizing the data collection time in a WSN that considers and combines the security and load balancing issues together.

APPENDIX A SPECIFICATIONS OF RASPBERRY PI

Table A.1. Raspberry Pi specifications

Price	$ 35
SoC	Broadcom BCM 2835
CPU	700 MHz ARM1176JZFS core with floating point
GPU	Broadcom VideoCore IV @250MHz
Memory (SDRAM)	512MB
Onboard Storage	SD/MMC/SDIO card slot
Onboard Network	10/100 Ethernet
Size	85.60mm x 85.60mm x 56mm (or roughly 3.37″ 2.21″ 0.83″)
Weight	45 g (1.6 oz)
Operating systems	Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Raspbian OS, RISC OS, Slackware Linux.

APPENDIX B DEDUCTION OF SECURITY CONSTRAINT

Assume that every sensor in a WSN, $MD_i (i \in \mathcal{M})$, encounters a security problem, such as key leakage or eavesdropping, with probability p_i (not identical). The probability that the tree k encounters a security problem because of an individual MD is $P_{attacked}(T_k)$. We say the tree satisfies the security constraint if $P_{attacked}(T_k)$ is no larger than some predefined threshold $P_{threshold}$, so we have:

$$P_{attacked}(T_k) = 1 - \prod_{i:MD_i \in T_k} (1 - p_i) = 1 - (1 - p_i)^{|T_k|} \leq P_{threshold}$$

$$1 - P_{threshold} \leq \prod_{i:MD_i \in T_k} (1 - p_i)$$

$$\log(1 - P_{threshold}) \leq \log(\prod_{i:MD_i \in T_k} (1 - p_i)) = \sum_{i:MD_i \in T_k} \log(1 - p_i)$$

$$-\log(1 - P_{threshold}) \geq -\sum_{i:MD_i \in T_k} \log(1 - p_i)$$

$$\log\left(\frac{1}{1 - P_{threshold}}\right) \geq \sum_{i:MD_i \in T_k} \log(\frac{1}{1 - p_i})$$

……

$$\sum_{i \in \mathcal{M}} (\log(\frac{1}{1 - p_i}) \sum_{i:MD_i \in T_k} X_{i,j,h}^k) \leq \log\left(\frac{1}{1 - P_{threshold}}\right), \quad \forall k \in \mathcal{R}$$

$$\sum_{i \in \mathcal{M}} \sum_{j \in \mathbb{S}(i)} \sum_{h=1}^{H} X_{i,j,h}^k \bullet \log\left(\frac{1}{1 - p_i}\right) \leq log\left(\frac{1}{1 - P_{threshold}}\right), \quad \forall k \in \mathcal{R}$$

REFERENCES

[1] Pathan, Al-Sakib Khan, ed. *Security of Self-Organizing Networks: MANET, WSN, WMN, VANET*. CRC Press, 2010.

[2] Chen, Tzung-Shi, Hua-Wen Tsai, and Chih-Ping Chu. "Adjustable convergecast tree protocol for wireless sensor networks." *Computer Communications* 33.5 (2010): 559-570.

[3] Gao, Shuai, Hongke Zhang, and Sajal K. Das. "Efficient data collection in wireless sensor networks with path-constrained mobile sinks." *Mobile Computing, IEEE Transactions on* 10.4 (2011): 592-608.

[4] Zhang, Wensheng, and Guohong Cao. "Optimizing tree reconfiguration for mobile target tracking in sensor networks." *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. Vol. 4. IEEE, 2004.

[5] Incel, Özlem Durmaz, et al. "Fast data collection in tree-based wireless sensor networks." *Mobile Computing, IEEE Transactions on* 11.1 (2012): 86-99.

[6] Jin, Haiming, et al. "Secure data collection in constrained tree-based Smart Grid environments." *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*. IEEE, 2014.

[7] Tabassum, Rehana, et al. "Scapach: Scalable password-changing protocol for smart grid device authentication." *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*. IEEE, 2013.

[8] Gong, Li, et al. "Protecting poorly chosen secrets from guessing attacks."*Selected Areas in Communications, IEEE Journal on* 11.5 (1993): 648-656.

[9] Moeller, Scott, et al. "Routing without routes: the backpressure collection protocol." *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010.

[10] Raymond, David R., and Scott F. Midkiff. "Denial-of-service in wireless sensor networks: Attacks and defenses." *Pervasive Computing, IEEE* 7.1 (2008): 74-81.

[11] Khan, Shafiullah, Kok-Keong Loo, Tahir Naeem, and Mohammad Abrar Khan, "Denial of service attacks and challenges in broadband wireless network," *International Journal of Computer Science and Network Security*, 8.7 (2008):1-6.

[12] Wang, Bao-Tung, and Henning Schulzrinne. "An IP traceback mechanism for reflective DoS attacks." *Electrical and Computer Engineering, 2004. Canadian Conference on*. Vol. 2. IEEE, 2004.

[13] Newsome, James, et al. "The sybil attack in sensor networks: analysis & defenses." *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*. ACM, 2004.

[14] Zhang, Qinghua, et al. "Defending against sybil attacks in sensor networks."*Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. IEEE, 2005.

[15] Khalil, Issa, Saurabh Bagchi, and Ness B. Shroff. "LITEWORP: a lightweight countermeasure for the wormhole attack in multihop wireless networks."*Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 2005.

[16] Hu, Yih-Chun, Adrian Perrig, and David B. Johnson. "Packet leashes: a defense against wormhole attacks in wireless networks." *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. Vol. 3. IEEE, 2003.

[17] Tiwari, Mukesh, et al. "Designing intrusion detection to detect black hole and selective forwarding attack in WSN based on local information." *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International Conference on*. IEEE, 2009.

[18] Karakehayov, Zdravko. "Using REWARD to detect team black-hole attacks in wireless sensor networks." *Wksp. Real-World Wireless Sensor Networks* (2005): 20-21.

[19] Singh, Virendra Pal, Sweta Jain, and Jyoti Singhai. "Hello flood attack and its countermeasures in wireless sensor networks." *International Journal of Computer Science* 7.3 (2010): 23.

[20] Xu, Wenyuan, et al. "The feasibility of launching and detecting jamming attacks in wireless networks." *Proceedings of the 6th ACM International Symposium on Mobile ad hoc Networking and Computing*. ACM, 2005.

[21] Hu, Yih-Chun, Adrian Perrig, and David B. Johnson. "Packet leashes: a defense against wormhole attacks in wireless networks." *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. Vol. 3. IEEE, 2003.

[22] Bandara, Dilum and Anura P. Jayasumana, “An enhanced topdown cluster and cluster tree formation algorithm for wireless sensor networks”. *In Proceedings of the Second International Conference on Industrial and Information Systems (ICIIS)*, 37–42, August 2007.

[23] Erciyes, Kayhan, Deniz Ozsoyeller, and Orhan Dagdeviren. "Distributed algorithms to form cluster based spanning trees in wireless sensor networks."*Computational Science–ICCS 2008*. Springer Berlin Heidelberg, 2008. 519-528.

[24] Chen, Ying, Pedro Henrique Gomes, and Bhaskar Krishnamachari. "Multi-channel Data Collection for Throughput Maximization in Wireless Sensor Networks." *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*. IEEE, 2014.

[25] Dai, Hui, and Richard Han. "A node-centric load balancing algorithm for wireless sensor networks." *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*. Vol. 1. IEEE, 2003.

[26] Song, Wen-Zhan, et al. "Time-optimum packet scheduling for many-to-one routing in wireless sensor networks." *The International Journal of Parallel, Emergent and Distributed Systems* 22.5 (2007): 355-370.

[27] Florens, Cédric, Massimo Franceschetti, and Robert J. McEliece. "Lower bounds on data collection time in sensory networks." *Selected Areas in Communications, IEEE Journal on* 22.6 (2004): 1110-1120.

[28] Revah, Yoram, and Michael Segal. "Improved lower bounds for data-gathering time in sensor networks." *Networking and Services, 2007. ICNS. Third International Conference on*. IEEE, 2007.

[29] Pathan, Al-Sakib Khan, Hyung-Woo Lee, and Choong Seon Hong. "Security in wireless sensor networks: issues and challenges." *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*. Vol. 2. IEEE, 2006.

[30] Abbasi, Ameer Ahmed, and Mohamed Younis. "A survey on clustering algorithms for wireless sensor networks." *Computer communications* 30.14 (2007): 2826-2841.

[31] Zungeru, Adamu Murtala, Li-Minn Ang, and Kah Phooi Seng. "Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison." *Journal of Network and Computer Applications* 35.5 (2012): 1508-1536.

[32] Saleem, Muhammad, Gianni A. Di Caro, and Muddassar Farooq. "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions." *Information Sciences* 181.20 (2011): 4597-4624.

[33] Raspberry Pi - Teach, Learn, and Make with Raspberry Pi https://www.raspberrypi.org/

[34] Choi, Hongsik, Ju Wang, and Esther A. Hughes. "Scheduling for information gathering on sensor network." *Wireless Networks* 15.1 (2009): 127-140.

[35] Perrig, Adrian, John Stankovic, and David Wagner. "Security in wireless sensor networks." *Communications of the ACM* 47.6 (2004): 53-57.

[36] "Wash., dc data set," 2014. [Online]. Available: http://data.octo.dc.gov/

[37] "GUROBI Solver," 2014. [Online]. Available: http://www.gurobi.com/