1-1-2012

System support for robust data collection in wireless sensing systems

Guoxing Zhan
Wayne State University,

SYSTEM SUPPORT FOR ROBUST DATA COLLECTION IN WIRELESS SENSING SYSTEMS

by

GUOXING ZHAN

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2012

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor Date

_____

_____

_____

ACKNOWLEDGMENTS

I would like to extend my gratitude to all who have helped me in my past five years of study. First, I would like to express my respect and great gratitude to my advisor, Dr. Weisong Shi. Dr. Shi and his MIST lab have been working in the field of computer system research for more than ten years, and have established versatile research fruits and international fame. He motivated me to study the sensor networks; he inspired me with fresh ideas and encouraged me to conquer challenging problems. Being honest, humble and endless seeking for the truth, he has impacted me a lot on my determination to resolve research issues. Also, I would like to thank Dr. Nathan Fisher, Dr. Hongwei Zhang, and Dr. Feng Lin for their grace to serve as my committee members and their valuable suggestions on my dissertation. Dr. Fisher, Dr. Zhang, and Dr. Lin have conducted fruitful research on real-time systems, wireless networks, and control systems, respectively. Their expertise and suggestions are undoubtedly invaluable. Additionally, I am grateful to Dr. Julia Deng for her help with the secure routing research and to all my former and current labmates for their friendly support. Finally, I owe a great deal to my beloved wife, Yan Huang, without the support of whom this work would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background: Wireless Sensing Systems

The advance of technology in electronics has enabled us to remotely collect physically measured information with electric sensors and transform the collected data into knowledge with software. The network facilities, including ethernet, WiFi, and cellular networks, enable remote access to the data sampled by a sensor. A number of potential applications can perform such remote data collection for various purposes, including environmental monitoring, industrial sensing, battlefield awareness, surveillance, forest fire monitoring, wild exploration, remote diagnosis, and so on. As an example, a group of sensors may be deployed in a forest to sense the local humidity or detect fire in the surroundings. The data from the sensors can be wirelessly sent to a nearby station and even further delivered to a remote monitoring center through computer networks. As another example, equipped with a camera sensor and a radio transceiver, a robot may enter a hazardous area to conduct searching tasks. An additional example is in the health care field. A patient may wear a blood pressure sensor and that sensor detects his blood pressure and transmits that value to his smartphone via Bluetooth. The smartphone further delivers the blood pressure reading to a remote hospital through WiFi for diagnosis. An electric sensor is usually pre-programmed to detect ambient conditions. Sensors permeate our life in various forms, e.g., cameras, GPS, thermometers, and blood pressure sensors. According to its making and use, the type of information detected by the sensor varies, including but not limited to environmental attributes, states of other equipment or itself, and conditions of human body. A sensor may sense ambient humidity, record images and noises of its surroundings, determine its own location, detect the voltage of an active

circuit, record the heart rate of a patient, and so on. A sensor may be installed indoors, outdoors, or attached to another object including devices and human. For example, certain large buildings have a few temperature sensors installed inside; a number of earthquake sensors may be distributed in the wild; a moving robotic vehicle can integrate a GPS receiver, a radio transceiver and a camera sensor as part of its circuit board; a smartphone typically comes with a set of sensors (e.g., an accelerometer); a patient may wear a pulse sensor around his hand. It is common that a sensor is equipped with wireless capability through either its integrated radio module or its connection to another wireless device. For example, a small-sized TelosB sensor mote [31] has an internal IEEE 802.15.4 compliant radio and can transmit wirelessly at rate of 250 kbps. As another example, a Bluetooth-enabled heart rate sensor may detect a user's heart rate and sends the data to the user's smartphone via Bluetooth; then the smartphone may further deliver the data through WiFi or its cellular network. The wireless connection of the sensors has brought great convenience and flexibility in the sensor deployment. As a trend, software is playing an important role in remotely collecting data with sensors. The hardware facilities involved in the data collection mainly include sensors, networking devices, storage devices, and computing devices. Normally, these hardware devices execute software instructions and cooperate with one another to automatically perform the data collection. Most of these devices can either be re-programmed by the owners or act according to commands received from other re-programmable devices. In a sense, the hardware is "smart". Often, these devices behave like a modern computer. They may have microcontroller/CPU, memory, storage disks, or network chips. The sensors may contain a software program that commands the sensor on how to detect data and how to send out data. For example, a TelosB sensor mote [31] has a TI MSP430 microcontroller with 10KB memory; it can be programmed via its USB interface. The data transmission to the destination is also normally controlled by software running on the networking devices. The data collected are usually stored at a center computer server and can be accessed by software applications. The server may perform an overall control of the data collection via customized software

programs. Formally, we define a wireless sensing system as a software system that instructs a set of pre-programmed sensors to collect subject information, then wirelessly transmits the data to one or more network gateway devices, afterwards delivers the data to one or more destinations via their associated networks, and finally let software applications access the data collected onto the destinations. In certain scenarios, a network gateway may integrate a few sensors into its circuit. In a wireless sensing system, a sensor may be placed indoors, outdoors, or even attached to another device. The sensor may integrate radio capability into its circuit or connect to another device that is wireless-enabled. The gateway devices are not restricted to conventional routers; instead, they can be a smartphone integrating or connected to the sensor, or a networked computer. The gateway devices are free to choose their networks to deliver the data to the final destinations (typically a single destination), possibly through computer networks or cellular networks. For applications, the data collected onto the destinations can be accessed by certain software. Figure 1.1 illustrates the data flow of a wireless sensing system. The data are first obtained by the sensors through a sensing process. The data are then wirelessly transmitted from the sensors to one or more network gateways. The gateways further deliver the data to one or more destinations (typically one common destination). The destinations are normally computer servers. After that, software applications are able to access the data delivered to the destinations.

1.2 Robust Data Collection in Wireless Sensing Systems: Challenges and Solutions

Faults, errors, failures and attacks are not uncommon in a wireless sensing system and they compromise the effective data collection of a wireless sensing system. As an effort in understanding the potential issues in wireless sensing systems, in 2008, the author co-worked with K. Sha, S. Al-Omariz, etc. on studying the data collected from 13 sensors installed around the Great Lakes [133]. These sensors were mainly located at the St. Clair River, the Detroit River and the Lake Winnebago. These sensors detected their battery voltage, the ambient air temperature and water temperature, the ambient precipitation, as well as the ambient water

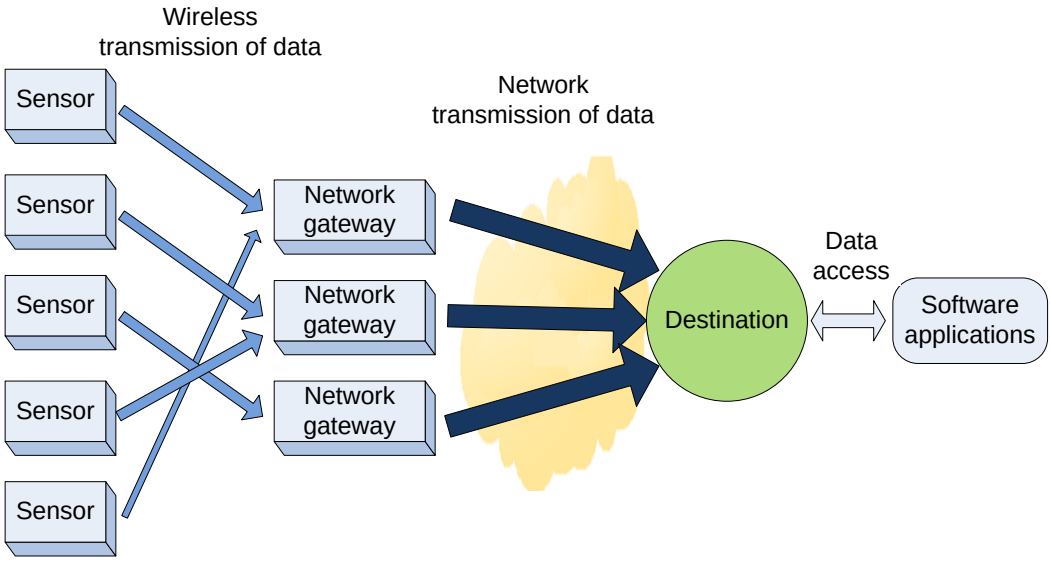


Figure 1.1: The data flow of a wireless sensing system.

level. The data available for our study lasted around one month. The sensors performed the data sampling once every one or two hours. They transmitted the obtained sensing data through radio to a satellite once every several hours; the satellite in turn sent the data also through radio to a central server located in Virginia. Our analysis on the data as well as the associated log discovered certain errors among the collected data and loss of certain data during the whole data collection process [133]. The data from some sensors were occasionally out of their valid range or suspected to be likely wrong according to the our domain expert. Frequent failure occurred at two sensors located at Lake Winnebago: their readings were dramatically different than the readings from other sensors around the same area. It was common to see that the data from certain sensors were missing from several hours to a few days. The analysis on the system log revealed that the data unavailability was caused by several types of errors either during the network transmission or in the sensor hardware. Generally, a robust wireless sensing system needs to address a few important issues in order to accomplish effective data collection. As illustrated in Figure 1.2, the issues may occur at different stages of the data collection. First, the sensors might not produce data correctly when

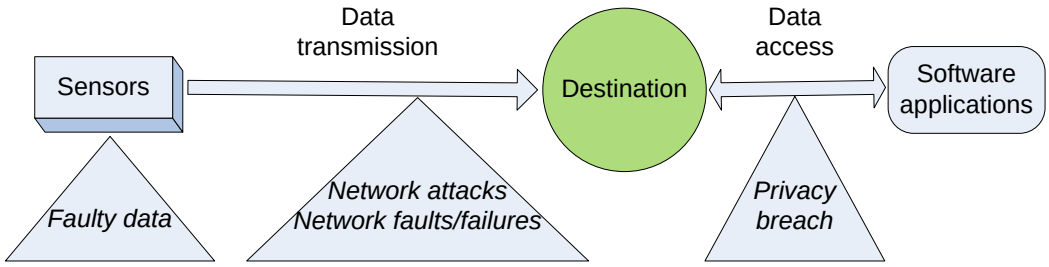


Figure 1.2: The potential issues during data collection of a wireless sensing system.

performing data sampling. The data may become unavailable or erroneous. The cause can be attributed to faults or failures from the sensor hardware and the software running on the sensors. Second, during the transmission of the data, faults, failures, and attacks might occur. A robust wireless sensing system should adopt resilient network protocols that can survive under faults, failures and potential network attacks. Last, when the data collected onto the destinations are being accessed by software applications, there can be privacy concern. The collected data may contain certain private information such as personal data. In such case, an unauthorized software application should not be allowed to access the private data directly. While the previous research has developed certain techniques to address some of these issues in wireless sensing systems, there are a few urgent problems left unsolved. In the following, we will discuss the state-of-the-art on these issues and point out the urgent issues that this dissertation will address.

1.2.1 Data Faults on the Sensors

The sensors used in a wireless sensing system are usually provided by the hardware manufacturers as commodity products. The manufacturers normally design the circuit and preprogram the hardware in a way that enables the sensors to meet with certain accuracy requirement on the detected attributes and certain fault tolerance requirement. In some sense, the hardware manufacturers are responsible for making the sensing functionality highly accurate.

However, as noted in our study of a real-world sensing system [133], in reality, it is not uncommon for the sensors to generate erroneous data. And what's worse, a malicious sensor may intentionally report wrong data to compromise the applications, especially in military applications. Thus, there is a need to detect such erroneous data produced by the sensors. There has been a trend to employ trust management for wireless sensing systems [7] to detect the sensor malfunction. With trust management, a sensor is assigned a trust value to reflect its trustworthiness according to its past performance. The trust management is proved to be effective in improving security [10, 95], supporting decision-making [70, 145], promoting node collaboration [55] and resource sharing [89]. However, to detect erroneous data produced by a sensor, there are a few important issues with directly applying the existing research outcomes. First, most trust research focuses on link-level one-hop communication behaviors, and data integrity is overlooked. Since data collection is the main task of wireless sensing systems, the importance of data integrity should never be underestimated. Second, overcomplicated models often render reputation system hard to apply to deployed wireless sensing systems. Those models may cause too much overhead. Finally, the fair treatment of new transactions and past behaviors, as adopted in the existing work, suffers various attacks. To conquer the challenge in detecting erroneous data produced by the sensors, we proposed a resilient data trustworthiness model, *SensorTrust* (see Chapter 3). While *SensorTrust* is originally designed for a typical type of wireless sensing systems, the idea can be extended to more generic settings. SensorTrust integrates past history and recent risk to accurately identify the current trust level. It employs Gaussian model to rate data integrity in a fine-grained style and a flexible update protocol to adapt to varied context. With acceptable overhead, SensorTrust proves resilient against varied data faults and attacks targeting the data. While the responsibility of ensuring highly accurate sensors is normally left to the sensor manufacturers, there is a particular type of sensing function that draws our attention: localization. So far, due to the algorithm complexity involved in localization, the localization algorithms are left to the system designers in many cases, especially when the GPS positioning is unavailable or

unsatisfactory. The location data are important when we consider wireless sensing systems with mobile sensors. An example application is robotic exploration, where a robot equipped with a few sensors (e.g., cameras, GPS) and a radio transceiver moves through an area to explore the land features. In this application, it is desirable to obtain the real-time location of the robot when it performs the autonomous sensing tasks. Particularly, we studied the localization of small-sized ground robotic vehicles, which have great potential to be deployed in situations that are either uncomfortable for humans or simply too tedious. Although there exist various localization schemes for ground robotic vehicles, they often suffer various issues. These techniques normally utilize GPS, inertial sensors, radio signals, or visual processing. GPS often becomes inoperable in certain environments such as indoors or in wild forests. As an alternative, a localization system may use various waves including electromagnetic waves of various frequency [9, 60, 117, 140, 161]. However, the radio-based positioning requires a set of external devices to generate or receive radio signal and maintaining such a positioning system can be costly and difficult in terms of additional hardware [25, 93, 119], intensive tuning [99], and environmental management. The vision techniques for mobile robot navigation [36] generally heavily rely on sophisticated techniques on the recognition of an object or shape from images and often have restricted spatial and visional requirements. They are relatively costly and difficult to implement or maintain. Additionally, inertial sensors are often used in positioning or navigation systems to detect movement [54, 76, 80, 92, 132]. Though the operation of inertial sensors is independent of external features in the environment and they do not need an external reference, empirically, the inertial sensors are likely to cause cumulative error. To obtain accurate and highly available location data, we proposed LOBOT (see Chapter 4), a low-cost, self-contained localization system for the small-sized ground robotic vehicle. LOBOT provides accurate real-time, three-dimensional positions in both indoor and outdoor environments. Unlike other localization schemes, LOBOT does not require external reference facilities, expensive hardware, careful tuning or strict calibration, and is capable of operating under various indoor and outdoor environments. LOBOT identifies the

local relative movement through a set of integrated inexpensive sensors and well corrects the localization drift by infrequent GPS-augmentation. Our empirical experiments in various temporal and spatial scales show that LOBOT keeps the positioning error well under an accepted threshold.

1.2.2 Network Issues During the Data Transmission

Faults, failures, and attacks during the network transmission may prevent the data from being delivered to the destinations. A major weak link of the chain of data collection lies in the wireless transmission from a sensor to a network gateway. Generally, wireless networks are prone to failures and attacks, which is normally addressed by robust network protocols. As far as wireless sensing systems are concerned, their particular characteristics aggravate the problems. In many applications, especially in a wild environment, the sensors used are battery-powered embedded sensors with limited processing capabilities, such as TelosB motes [31]. With a limited radio communication range, many sensors together with their equipped radio transceivers, comprise a multi-hop wireless ad-hoc network. To send the data to a network gateway, each sensor usually wirelessly sends data out to one of its neighboring sensors, which in turn forwards the data to another sensor. The data reach the network gateway via a multi-hop path through multiple sensors. Such wireless networks consisting of sensors are defined as wireless sensor networks (WSNs) [120, 170] and the network gateway is usually also called a base station. Compared to conventional wireless networks, WSNs consisting of resource-constrained sensors more easily suffer failures and malicious attacks. Most existing network protocols for these networks either assume the honesty of the nodes and focus on how to increase the throughput of the network under faults and failures [1], or attempt to exclude unauthorized participation by encrypting data and authenticating packets [73, 91, 116, 146]. While some of these protocols function well under faults and certain attacks, they can be easily defeated by a malicious attacker who fakes one or more identities by replaying routing information. A malicious node can still participate in the network using another valid node's

identity by replaying their routing information. The multi-hop routing in WSNs offers little protection against the identity deception through replaying routing information. To secure multi-hop routing against adversaries exploiting the replay of routing information, we designed and implemented TARF (see Chapter 5), a robust trust-aware routing framework for dynamic multi-hop wireless networks consisting of sensor nodes. Without tight time synchronization or known geographic information, TARF provides trustworthy and energy-efficient route. Most importantly, TARF proves effective against those harmful attacks developed out of identity deception.

1.2.3 Privacy Protection on the Collected Data

In certain applications of wireless sensing systems, the collected data may contain private information that is not expected to be directly exposed. An example of these applications is the self-monitoring and self-management of patient health [83, 97]. In such a wireless sensing system, users utilize off-the-shelf wireless biomedical sensors to detect their biophysical data such as heart rate and the data are sent out to a remote station through their smartphones [122]. The remote stations may deliver feedback accordingly back to the users. While such applications can lower the medical cost and facilitate remote diagnoses [49], they encounter the obstacle of privacy concern [30, 64, 129, 155]. The existing wireless sensing systems in such areas tend to either not consider privacy protection at all [18, 113] or limit the collected data to its internal use only so as to reduce privacy risks [96, 107]. The restriction of the internal use of data prevents third-party applications from exploring the data and becomes an obstacle to data sharing. The existing privacy research mainly concerns itself about the mechanisms to identify and prevent privacy issues [28, 46, 62] and often does not support arbitrary third-party applications. To conquer the challenge in privacy protection, we proposed Woodward(see Chapter 6), a privacy-preserving wireless sensing system, focusing on health care applications. Woodward protects the user privacy while allowing for the data sharing with arbitrary third-party applications. It adopts an innovative anonymization process that

supports high-precision query and impedes privacy attacks by the overwhelming cost. We implemented Woodward with a health care application and quantitatively evaluated both the query precision and privacy protection.

1.3 Objectives

This dissertation aims to provide system support for robust data collection in wireless sensing systems through addressing a few urgent design issues in existing systems. A wireless sensing system may suffer issues arising at the sensors, during the data transmission, and during the data access by applications. While wireless sensing systems may resemble conventional networked systems in many ways, their unique characteristics determine that certain conventional solutions for networked systems may not work well. Considering a wireless sensing system may be dramatically different than another (depending on their specific applications), the solutions to each of these issues may vary according to the system structures. With certain typical system structures in mind, we have developed approaches to resolve those few urgent problems in the design of wireless sensing systems. However, we would like to emphasize that similar ideas to our approaches can be employed to address the issues in more generic settings. Specially, this dissertation will accomplish the following objectives:

1. With hierarchical wireless sensor networks as an example, develop a resilient trust model to evaluate the trustworthiness of the collected data.

2. Develop a low-cost, self-contained localization system for the small-sized ground robotic vehicle to obtain accurate location data.

3. Design and implement a robust trust-aware routing framework to secure multi-hop routing through a set of sensors in wireless sensing systems.

4. With health care as an exemplary application, develop a privacy-preserving wireless sensing system.

1.4 Contributions

The main contributions of this dissertation are:

1. We developed a resilient trust model, *SensorTrust*, to effectively detect faulty data in wireless sensing systems due to either sensor malfunctioning or malicious attempts to report false data. *SensorTrust* evaluates the trustworthiness of the collected data in wireless sensing systems. While this model is mainly proposed for a certain common architecture of wireless sensing systems, this approach can be generalized to detect data trustworthiness in a more generic setting. *SensorTrust* enables us to accurately identify the current trust level of the data produced by a sensor. *SensorTrust* allows us to consider both past history and recent risk when assessing the data trustworthiness. It also adapts to dynamic environment.

2. We developed a low-cost, self-contained, accurate localization system (LOBOT) for small-sized ground robotic vehicles. This localization system enhances the wireless sensing systems containing mobile sensors by providing more accurate and highly available location data, with only limited overhead in economic cost and management. The hardware devices LOBOT uses are easily-available at low cost. LOBOT is self-contained in that it virtually requires no external devices or external facility management and that it needs no prior information. LOBOT does not require external reference facilities, expensive hardware, careful tuning or strict calibration. Additionally, LOBOT applies to both indoor and outdoor environments and realizes satisfactory performance. Further, LOBOT maintains low cumulative error.

3. We designed and implemented TARF, a robust trust-aware routing framework, to secure multi-hop routing through a set of sensors in wireless sensing systems. Though it is mainly motivated by harmful attackers exploiting the replay of routing information, TARF can also be used to protect the routing layer from other attacks. TARF requires

neither tight time synchronization nor known geographic information. Its resilience and scalability were proved through both extensive simulation and empirical evaluation with large-scale WSNs. We implemented a ready-to-use TinyOS module of TARF with low overhead; this TARF module can be integrated into existing routing protocols with moderate efforts.

4. We developed Woodward, a privacy-preserving wireless sensing system. Though it focuses on health care applications, the design principle in privacy protection can be extended to other wireless sensing systems with privacy concern. Woodward protects the user privacy while allowing arbitrary third-party applications to extract knowledge from the collected data. The anonymization process adopted by Woodward causes overwhelming cost to privacy attackers; it also allows arbitrary third-party applications to perform various query with small under-threshold error.

1.5 Outline

The rest of this dissertation is organized as follows: Chapter 2 reviews the related work; Chapter 3 describes the data trustworthiness model for wireless sensing systems - *SensorTrust*; Chapter 4 describes lobot, a low-cost, self-contained localization system for small-sized ground robotic vehicles; Chapter 5 presents TARF, a trust-aware routing framework to secure multi-hop routing through a set of sensors in wireless sensing systems; Chapter 6 presents Woodward, a privacy-preserving wireless sensing system; Chapter 7 concludes this dissertation and describes future directions.

CHAPTER 2

RELATED WORK

This chapter describes the existing work related to the problems that this dissertation resolves.

2.1 Data Trustworthiness Modeling for WSNs

Trust has been studied in varied contexts for long. It started as an important topic in social science. The effects of trust in commerce was analyzed to help build e-commerce systems, such as eBay [125]. Game theory and reinforcement learning are also used to model the reputation of sellers in [142]. Additionally, trust management is applied to online knowledge sharing [37], peer-to-peer systems and ad-hoc networks [14, 27, 34, 45, 70].

A few general models have been proposed for data trust management of WSNs. Ganeriwal, Balzano and Srivastava proposed a reputation-based framework for high integrity WSNs named RFSN [44]. In the RFSN framework, Bayesian formulation is employed to update reputation metrics with new transaction, density-based outlier detection discovers data outliers, and an aging mechanism is used against the sleeper attack. Other trust model study includes an agent-based trust model by Chen *et al.* [23], ATRM by Boukerche and Li [11], and study on the effects of rating algorithms by Liang and Shi [90]. As far as the hierarchical WSNs are concerned, our *SensorTrust* is a more suitable model. One essential hypothesis in RFSN is that reputation satisfies a Beta distribution. However, to our best knowledge, that hypothesis has not been widely justified yet. In contrast, the Gaussian distribution of data adopted in *SensorTrust* is a well-known fact, and our mathematical analysis shows that the update protocol effectively incorporates long-term reputation and recent risk. Compared to

the binary rating with outlier detection in many existing work, *SensorTrust* employs a Gaussian distribution-based fine-grained method to determine the rating and trust level. Also, in contrast to the aging algorithm used in RFSN, our model utilizes two parameters to gain more flexibility for different contexts.

2.2 Localization Schemes

The radio-based localization schemes roughly fall into two categories: the range-based solutions and the range-free solutions. Most of these schemes usually require a set of surrounding anchor nodes with known position information. Maintaining a proper set of surrounding anchor nodes is vital to such localization solutions. The range-based schemes discover the position by first estimating distances or angles among certain nodes and then applying triangulation or multilateration to compute the location. They utilize various range measurements including Received Signal Strength [4, 29, 61, 130, 134, 143, 158, 159, 169], Time of Arrival [81], Time Difference of Arrival [25, 119, 131] and Angle of Arrival [6, 21, 110, 127]. The range-free localization schemes exploit the proximity information to estimate the location of the target [3, 82, 134, 173], The typical examples include Centroid [15], APIT [56], APS [110], Concave [38] and Self [16].

As another type of popular solutions, Inertial sensors are used in positioning or navigation systems to detect movement [54, 76, 80, 92, 132]. The accelerometer is often perceived as an inexpensive solution for localization. Jackson [66] proposed an accelerometer-based solution for tracking test vehicles on a known stretch of bridge; however, this solution is subject to cumulative error [66]. Hsu proposed an accelerometer based approach for indoor tracking [65], where only theoretical simulation is used for the evaluation. Youssef [160] proposed a hybrid GPS-accelerometer-compass scheme that depends mainly on the low-energy accelerometer and compass sensors. The scheme uses GPS infrequently for synchronization. However, Youssef's work still uses the well-known double integration of acceleration to calculate travel distance. In that work, the cumulative error is not thoroughly analyzed [160].

Other localization techniques include probabilistic approaches and distributed localization. For example, the extended Kalman filter [32] has been extensively used for information fusion in robot navigation problems. Fox [41] proposed an active Markov localization for mobile Robots. Thrun [141] proposed a family of probabilistic localization algorithms known as Monte Carlo Localization for mobile robots. There are also a few distributed localization schemes in wireless sensor networks and wireless ad-hoc networks [8, 24, 87, 138]. As an example, Xiao, Chen, and Zhou [154] proposed a distributed localization system using a moving beacon.

2.3 Trust Management in Secure Routing

Trust and reputation management has been employed in ad hoc networks to secure routing protocols [13, 50, 94, 100, 104, 118, 128, 153, 157]. However, those proposed systems for generic ad hoc networks target relatively powerful hardware platforms such as laptops and smartphones; they can not be applied to WSNs comprising resource-constrained sensor nodes.

Regarding WSNs, secure routing solutions based on trust and reputation management rarely address the "identity theft" exploiting the replay of routing information. A location-based trust-aware routing solution for large WSNs - ATSR - is proposed in [163]. ATSR incorporates a distributed trust model utilizing both direct and indirect trust, geographical information as well as authentication to protect the WSNs from packet misforwarding, packet manipulation and acknowledgements spoofing. Another trust-aware routing protocol for WSNs is TARP (Trust-Aware Routing Protocol) [126]. TARP exploits nodes' past routing behavior and link quality to determine efficient paths. We note that neither ATSR nor TARP offers protection against the identity deception through replaying routing information. It is generally hard to protect WSNs from *wormhole* attacks, *sinkhole* attacks and *Sybil* attacks based on such identity deception. The countermeasures often requires either tight time synchronization or known geographic information [74]. Additionally, Cao, Hu, Chen, Xu, and Zhou proposed

a feedback-based secure routing protocol (FBSR) for WSNs [19], with which sensor nodes incorporate feedback messages included in the MAC layer acknowledgement to avoid network congestion. The feedback message is authenticated with a Keyed One Way Hash Chain (Keyed-OWHC) to prevent feedback fabrication. FBSR also uses a statistics-based detection on a base station to discover potentially compromised nodes. Though the authors claimed that FBSR is resilient against *wormhole* and *Sybil* attacks, such resilience is never evaluated or examined. Additionally, the Keyed-OWHC-based authentication for the feedback message in each MAC layer acknowledgement causes a major overhead in a multi-hop WSN.

2.4 Privacy Protection in Participatory Sensing Systems

The privacy leakage has arisen as one major concern involved in participatory sensing [30, 64, 129, 155]. The privacy attack comes in various forms. Besides the direct data theft, an attacker my attempt to identify a user or his activity either explicitly or implicitly by the user's usage of the computing hardware or software, such as IP/MAC addresses, usage pattern and device fingerprinting [42, 51, 77, 112]. The attacker may also attempt to analyze the data pattern [2, 71, 79], infer the user context [102, 108]

The existing research has explored the privacy protection with diverse approaches. Generally, these approaches fall into one of the following categories [79]: regulatory rules, privacy policies, anonymity, and obfuscation. The regulatory rules and privacy policies rely on administrative regulation and trust relationships. The anonymity-based approaches use pseudonym and group users to generate ambiguity [40, 75, 135]. Many such approaches are based on the concept of k-anonymity or its variants [137, 150], where privacy is obtained when it is unable to distinguish one entity from k-1 other entities. Typical examples occur in location-based services [47, 52, 101, 172]. Some of these approaches are known as ID rotating [86] and mix networksshmatikov:06,benjamin:06. The obfuscation-based approaches protect privacy by reducing the data quality [12, 114, 121], initially introduced for location-based services [39]. This category of approaches are also referred to as "cloaking" in a few

research projects [26, 53, 62]. To quantify the privacy, the researcher have created different metrics. The k-anonymity-based approaches use the size of ambiguity set (k) as the level of privacy [137]. The obfuscation-based approaches may define privacy as the expected magnitude of the noise added onto the data or the duration to be able to track the user [39, 62].

Most of the existing participatory sensing systems [33, 96, 107, 174] use the data to serve only internal applications and thus do not concern themselves with privacy protection. A recent project, *AnonySense* [28], built a participatory sensing system to allow any third-party application to collect data from mobile users. *AnonySense* protects the user privacy by a mix network. The mix network allows users to send messages anonymously and mixes enough messages before reporting to applications. It mainly intends to unlink multiple data records from the same user. However, unlike our Woodward system, *AnonySense* does not the privacy attack based on prior knowledge of a certain user's record.

CHAPTER 3

DATA TRUSTWORTHINESS MODELING

We developed a resilient trust model, *SensorTrust*, to effectively detect faulty data in wireless sensing systems due to either sensor malfunctioning or malicious attempts to report false data. *SensorTrust* evaluates the trustworthiness of the collected data in wireless sensing systems. While this model is mainly proposed for a certain common architecture of wireless sensing systems (hierarchical WSNs), this approach can be generalized to detect data trustworthiness in a more generic setting. *SensorTrust* enables us to accurately identify the current trust level of the data produced by a sensor. *SensorTrust* allows us to consider both past history and recent risk when assessing the data trustworthiness. It also adapts to dynamic environment.

3.1 Introduction

As an important type of wireless sensing systems, wireless sensor networks (WSNs) [120, 170] are ideal candidates for applications to report detected events of interest, such as military surveillance and forest fire monitoring. A WSN comprises battery-powered senor nodes with extremely limited processing capabilities. With a narrow radio communication range, a sensor node wirelessly sends messages to a base station (network gateway) via a multi-hop path.

Wireless sensor networks (WSNs) [120,170] are prone to data faults. A sensor node could report inaccurate data. With the existence of a malicious attacker, the sensor node may even report forged data. Thus, it is important to evaluate the data integrity in WSNs. One type of important approaches for resolving data integrity issues in networked systems is trust management [7]. With trust management, each sensor node in the WSN is assigned a trust value

to reflect its trustworthiness according to its past performance. Trust management of nodes is effective in improving security [10, 95], supporting decision-making [70, 145], promoting node collaboration [55] and resource sharing [89]. However, there are a few important issues with existing work. First, most trust research focuses on link-level communication behaviors, and data integrity is overlooked. Since data collection is the main task of WSNs, the importance of data integrity should never be underestimated. Second, overcomplicated models often render reputation system hard to apply to deployed WSNs. Those models may cause much overhead. Finally, fair treatment of new transactions and past behaviors suffers various attacks.

In this chapter, we proposed a resilient trust model, *SensorTrust*, to evaluate data trustworthiness in hierarchical WSNs. In this model, the aggregator maintains trust estimations for children nodes in terms of data trustworthiness. Unlike previous efforts, our current design of *SensorTrust* mainly focuses on data integrity. *SensorTrust* integrates past history and recent risk in a real-time way that accurately identifies the current trust level. Our model employs Gaussian model [69] to rate data integrity in a fine-grained style, and a flexible update protocol to adapt to varied context. With acceptable overhead, the *SensorTrust* model is evaluated with the real world sensor data from Intel Berkeley Lab and Motelab at Harvard University, and compared with other approaches. The results indicate great advantage of *SensorTrust* to handle faults and attacks.

The rest of this chapter is organized as follows: the detailed mechanism of the *SensorTrust* model is depicted in Section 3.2; the evaluation of *SensorTrust* is given in Section 3.3 separately; Section 3.4 summarizes this chapter.

3.2 SensorTrust Model

We proposed *SensorTrust* to evaluate the data trustworthiness of sensor nodes in hierarchical WSNs. The hierarchical structure has been widely accepted in designing WSNs because it

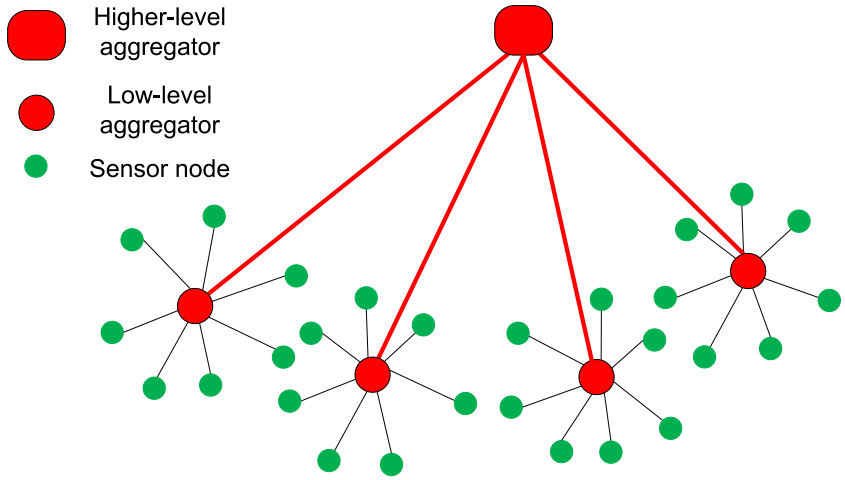


Figure 3.1: A typical hierarchical wireless sensor network.

optimizes network performance [59]. In a hierarchical WSN, each node relays data to its associated lower-level aggregator, and those aggregators forward received data to their higher-level parent aggregators, and the forwarding continues until the top layer of the hierarchy, at which point the data will be sent to the base station. Figure 3.1 shows part of such a hierarchy. Such a hierarchical structure is easy to implement, and it is known that the hierarchical structure enables more efficient use of scarce resources, such as energy, frequency spectrum and bandwidth [20, 58, 136, 144]. Our goal is to establish a trust environment against faults and attacks targeting the data. Since it is usually beneficial to select reliable nodes as aggregators, we assume the aggregators are trustable, and let the aggregators evaluate the trustworthiness of their children nodes as in Section 3.2.1, 3.2.2 and 3.2.3.

3.2.1 Methodology

With *SensorTrust*, the aggregator maintains trust estimations for children nodes, and we integrate its *long-term reputation* and *short-term risk*, and take into consideration both link-level *communication robustness* and *data integrity* (Figure 3.2(a)), with a focus on *data integrity*.

In this chapter, regarding communication, we will only consider one-hop link-level communication rather than multi-hop end-to-end communication. Long-term reputation, also called conventional reputation, refers to its average performance level in its whole past history, and short-term risk identifies to which degree its future behavior is associated with its recent performance. Neither long-term reputation nor short-term risk alone could fully reflect current trustworthiness. On the one hand, a single fault could occasionally happen to even a trustworthy sensor node, but that doesn't necessarily mean the node is unreliable. That suggests the one-sidedness of short-term risk. On the other hand, long-term reputation treats the node's behavior in each transaction equally. But in the real world, a node with good average performance level might begin to behave negatively during recent transactions. That could suggest that the sensor starts to malfunction. The well-known sleeper attack [44] is such a scenario. Therefore, recent performance needs to be viewed differently. Since a node can behave maliciously regarding either (link-level) wireless communication or data management, trustworthiness is evaluated from two aspects: communication robustness and data integrity (Figure 3.2(a)). Most trust research focuses on communication behaviors, and data integrity is overlooked. Since data collection is the main task of WSNs, the importance of data integrity should never be underestimated. Because different applications have their own specific requirements regarding communication trustworthiness and data trustworthiness, we explore communication trustworthiness and data trustworthiness separately.

Our *SensorTrust* model uses a *SensorTrust* value, which is a decimal number in $[0, 1]$, to represent trustworthiness level. It is denoted as T in this chapter. The higher some node's *SensorTrust* value is, the more trustworthy that node is. Specifically, *the SensorTrust value in terms of communication robustness is the estimated probability of a positive communication transaction; the SensorTrust value in terms of data integrity is the estimated probability of integrity of data*. At the beginning (before any transaction happens), the aggregator simply gives its children nodes a *SensorTrust* value of 0, since no evidence of trustworthiness is

available. Each time a sensor node interacts with its associated aggregator (or required by the aggregator to do so), the aggregator evaluates the node's behavior by giving a *rating number* in $[0, 1]$ for this transaction in terms of communication robustness and data integrity respectively. This rating number reflects the aggregator's opinion of the current transaction: the higher the rating numbering is, the more positive the aggregator views the sensor node to be. The rating number together with its latest *SensorTrust* value will be used by the aggregator to update the node's *SensorTrust* value (Figure 3.2(b)).

3.2.2 Transaction Rating

Our *SensorTrust* model rates each transaction by assigning a rating number in $[0, 1]$. We deal with wireless communication *SensorTrust* and data *SensorTrust* separately. In the following, always use R to denote the rating number for a transaction, and T to denote the *SensorTrust* value.

First, though our focus is on data *SensorTrust*, we still give a simple discussion about the communication *SensorTrust*. To rate the transaction between the sensor node and the aggregator regarding communication, we distinguish between positive communication and negative communication. Though there exist other communication behaviors that cannot be simply categorized as positive or negative, we believe it is possible to produce a more concrete rating algorithm for communication *SensorTrust*. If the sensor node is responding as well as expected by the aggregator, i.e. responding in time with the right format, relaying what is supposed to relay, and so on, then this transaction is regarded as positive communication. Otherwise, it's viewed as negative communication. Notice that such rating has nothing to do with data quality. Even if the sensor node is sending faked data, it can be viewed as positive communication, as long as the sensor node is behaving well in terms of transmission itself. To define the rating number, assign rating number $R = 1$ to positive communication, and $R = 0$ to negative communication.

Now, let's focus our attention on the data *SensorTrust*. We want to rate the data integrity for a sensor node, and unlike various approaches based on outlier detection, the rating is based on the widely applied Gaussian model [69], with a fine-grained style. The advantage of using Gaussian model lies in the accuracy of data integrity estimation. As an supplementary measure, first, the obvious abnormal data over the threshold set up according to domain experts will be detected, and rated with 0. Before further defining the rating number for a transaction, we make the following assumption: in the same cluster, the distribution of the data collected by sensor nodes can be depicted by the well-known Gaussian model, i.e., the data are Gaussian distributed. Many measurements of physical phenomena can be approximated, to varying degrees, by the Gaussian distribution. Here the use of the Gaussian model can be justified by the fact that sensor nodes within the same cluster often get closely related measurements due to short distance between them. When testing the Intel Lab data and Motelab data (see Section 3.3), we found that, generally speaking, the data distribution is still well described by Gaussian model, though occasionally irregular tails cause its distribution to slightly deviate from Gaussian model. We demonstrate that deviation with an example. Figure 3.3 displays the normal probability plot of 14 temperature values collected from different motes at Motelab. With normal probability plot, the data is plotted against Gaussian distribution in such a way that the cross makers should form an approximate straight line. Less departures from this straight line indicate less deviation from normality. As we notice that the markers with data values within [64, 88] are very close to the line, and the remaining markers, the so-called tails, depart from the line. The occurrence of such tails is very common in data distribution from a wide range of context where Gaussian model applied still achieves satisfactory results. Additionally, according to the transaction rating described later in this section, data outliers which also brings irregular tails, degrade the *SensorTrust* of the reporting node.

Though the distribution of data for multiple physical attributes can be well modeled by the multi-dimensional Gaussian model, to explain the algorithm in an easier way, we assume

the data are from a single physical attribute. A natural extension can be developed for multi-dimensional attributes. Denote X as the random variable of measurements by sensor nodes, μ as the average value of sampled data received by the aggregator, and σ as the standard deviation of sampled data. Then we have approximately

$$X \sim N(\mu, \sigma), \text{ and } \frac{X-\mu}{\sigma} \sim N(0,1)$$

Let P be the probability density function for the standard Gaussian distribution $N(0,1)$. Then $P(\frac{|d-\mu|}{\sigma})$ depicts the likelihood for X to take the value around d. Since the probability density function P reaches its maximum value at 0 (in other words, when $d = \mu$), for normalization purpose, we define the rating number

$$R = \frac{P(\frac{|d-\mu|}{\sigma})}{P(0)} = e^{-\frac{(d-\mu)^2}{2\sigma^2}}$$

Because the probability density function depicts the likelihood of the occurring of data values, such a rating scheme gives a fine-grained estimation of data accuracy. Table 3.1 lists the rating numbers corresponding to the deviation of the data value from the average value μ. As in Table 3.1, the deviation of 2σ from the average μ leads to a low rating number of 0.135. That is consistent with the following fact: suppose a random value has the distribution $N(\mu, \sigma)$, then it falls out of 2σ from μ with a probability 4.6%. Generally speaking, rare events deserve low trust level, and thus low rating numbers.

Table 3.1: Rating numbers.

$\lvert d-\mu \rvert$	0	0.5σ	σ	1.5σ	2σ	3σ
R	1	0.882	0.607	0.325	0.135	0.011

Though valid data range is usually known by domain experts in the field, outlier detection based on valid range fails to identify those unusually high or unusually low data which are

still within valid range. The proposed rating scheme identifies those data with low trust level by assigning low rating numbers.

3.2.3 SensorTrust Value Update

Given the rating number R for the current transaction and the latest *SensorTrust* value T_{old}, we want to find out the new *SensorTrust* value T_{new}. T_{new} is expected to incorporate both the sensor node's long-term reputation and short-term risk. An intuitive way is to use a weighted average of R and T_{old} as the value of T_{new}. That is what is essentially adopted in the aging mechanism of RFSN [44]. However, that method used against sleeper attacks still suffers periodic attacks. Suppose we update the *SensorTrust* value by $T_{new} = (1 - w) \times T_{old} + w \times R$, $w \in (0, 1)$. We denote such an approach by RFSN-w. Regarding highly sensitive applications, to resist sleeper attacks during which a node behaves well for a while before behaving maliciously, w is expected to be relatively big. The reason is, we wish to degrade such a node's *SensorTrust* value as much as possible after behaving negatively, so that it cannot recover its *SensorTrust* value easily. When it behaves maliciously, it gets a small-valued R, and a big w reaches a small T_{new} according to $T_{new} = (1 - w) \times T_{old} + w \times R$. Unfortunately, a big weight w for the current rating number R leads to the following result: a malicious node can recover its *SensorTrust* value fast by behaving positively for a relatively short time. That is because its past negative behavior can be offset relatively easily due to a big weight for the current rating number. To solve this problem, we update the *SensorTrust* value using two different weights, a relatively big $w_{degrade} \in (0, 1)$ and a relatively small $w_{upgrade} \in (0, 1)$ as follows:

$$T_{new} = \begin{cases} (1 - w_{degrade}) \times T_{old} + w_{degrade} \times R, \text{if } R \leq T_{old} \\ (1 - w_{upgrade}) \times T_{old} + w_{upgrade} \times R, \text{if } R > T_{old} \end{cases}$$

In order to resist the periodic attack, we can choose appropriate $w_{degrade}$ and $w_{upgrade}$ in such a way that the evolution of *SensorTrust* values will be subject to this rule: the *SensorTrust*

value is gained slowly, but can be ruined easily by intense negative behaviors. Whenever the current rating number R is smaller than its latest *SensorTrust* T_{old}, a relatively big $w_{degrade}$ causes *SensorTrust* values to decrease fast to some level between R and T_{old}, even close to R if $w_{degrade}$ is big enough. Whenever the current rating number R is bigger than its latest *SensorTrust* value T_{old}, a relatively small $w_{upgrade}$ prevents the *SensorTrust* value from increasing fast. Such mechanism is effective against periodic attacks.

The two parameters $w_{degrade}$ and $w_{upgrade}$ allow flexible application requirements. $w_{degrade}$ and $w_{upgrade}$ represent the extent to which upgraded and degraded performance are rewarded and penalized, respectively. If any fault and compromise is very likely to be associated with a high risk, $w_{degrade}$ should be assigned a relatively high value to penalize fault and compromise relatively heavily; if a few positive transactions cannot constitute evidence of trustworthiness which requires many more positive transactions, then $w_{upgrade}$ should be assigned a relatively low value. To help users choose suitable $w_{degrade}$ and $w_{upgrade}$, we first observe the following fact: suppose a node's current *SensorTrust* value is 0, after m perfect transactions with rating number 1, its *SensorTrust* value will become $1-(1-w_{upgrade})^m$; suppose a node's current *SensorTrust* value is 1, after n malicious transactions with rating number 0, its *SensorTrust* value will become $(1-w_{degrade})^n$. One possible solution is to let the user empirically decide how many contiguous perfect transactions are expected to happen before upgrading one node's *SensorTrust* value from 0 to 0.8, and how many contiguous malicious transactions are expected to happen before degrading one node's *SensorTrust* value from 1 to 0.2. Answers from user can be used to decide the value of $w_{degrade}$ and $w_{upgrade}$. Further, the value of $w_{degrade}$ can be made context-aware: when negative behaviors happens frequently, raise $w_{degrade}$ as more serious penalty. Further study of parameter selection is left to Section 3.2.4.

3.2.4 Mathematical Study of SensorTrust Evolution

Here we study the evolution of the *SensorTrust* value with a periodic behavior. The study shows, for a periodic behavior, to ensure the *SensorTrust* value gradually approaches the

conventional reputation, it is required that

$$\frac{w_{upgrade}}{1+w_{upgrade}} \leq w_{degrade} \leq min\{1, \frac{w_{upgrade}}{1-w_{upgrade}}\}$$

Denote an arbitrary consecutive rating number sequence as $R_0, R_1, R_2, ..., R_i, R_{i+1}, ...$, with corresponding trustworthiness value $T_0, T_1, T_2, ..., T_i, T_{i+1}, ...$ With the periodic behavior, the rating number sequence should display a similar periodic pattern. To simplify the problem, we consider the following scenario: $R_{i+k\times(n+1)} = 1, \forall 1 \leq i \leq n, \forall k \geq 0$, and $R_{j\times(n+1)} = 0, \forall j \geq 1$. This is actually the rating sequence for a periodic communication behavior in which a successful delivery is rated with a rating number 1, and an unsuccessful delivery is rated with a rating number 0. The corresponding conventional reputation is easily seen to be the successful delivery ratio $\frac{n}{n+1}$. The following two claims are considered equivalent: (1) for such a periodic behavior, the *SensorTrust* value gradually approaches the conventional reputation $\frac{n}{n+1}$; (2)

$$\forall n \geq 1, \quad \lim_{k->\infty} T_{k\times(n+1)} \leq \frac{n}{n+1}$$

and

$$\lim_{k->\infty} T_{k\times(n+1)-1} \geq \frac{n}{n+1}$$

Thus, the actual problem here is to decide when the claim (2) becomes true. We will briefly go over the major steps.

After some computation,

$$\lim_{k->\infty} T_{k\times(n+1)}$$
$$= \frac{(1-w_{degrade})\times[1-(1-w_{upgrade})^n]}{1-(1-w_{upgrade})^n\times(1-w_{degrade})}$$

Thus,

$$\lim_{k->\infty} T_{k\times(n+1)} \leq \frac{n}{n+1}$$

after transformation, is equivalent to

$$w_{degrade} \geq 1 - \frac{n}{n+1-(1-w_{upgrade})^n}$$

Fortunately, with a bit calculus, we can prove that

$$\frac{n}{n+1-(1-w_{upgrade})^n}$$

is an increasing function in terms of n. Again, that leads to the following fact:

$$\forall n \geq 1, w_{degrade} \geq 1 - \frac{n}{n+1-(1-w_{upgrade})^n}$$

is equivalent to

$$w_{degrade} \geq \frac{w_{upgrade}}{1+w_{upgrade}}$$

Therefore,

$$\forall n \geq 1, \lim_{k->\infty} T_{k\times(n+1)} \leq \frac{n}{n+1}$$

if and only if

$$w_{degrade} \geq \frac{w_{upgrade}}{1+w_{upgrade}}$$

Applying a similar procedure, we can prove that

$$\forall n \geq 1, \lim_{k->\infty} T_{k\times(n+1)-1} \geq \frac{n}{n+1}$$

is equivalent to

$$w_{degrade} \leq \frac{w_{upgrade}}{1-w_{upgrade}}$$

Therefore, with a periodic behavior, if the *SensorTrust* value is required to approach the conventional reputation, then our model requires that

$$\frac{w_{upgrade}}{1+w_{upgrade}} \leq w_{degrade} \leq \frac{w_{upgrade}}{1-w_{upgrade}}$$

Now, we will estimate the amplitude of trustworthiness variation.

$$\begin{aligned} & \lim_{k->\infty}(T_{k\times(n+1)-1} - T_{k\times(n+1)}) \\ = & \lim_{k->\infty} \frac{T_{k\times(n+1)}}{1-w_{big}} - \lim_{k->\infty} T_{k\times(n+1)} \\ = & \frac{w_{big}\times[1-(1-w_{small})^n]}{1-(1-w_{small})^n\times(1-w_{big})} \\ \leq & w_{big} \end{aligned}$$

$\lim_{k->\infty}(T_{k\times(n+1)-1} - T_{k\times(n+1)})$ is an increasing function in terms of w_{big}, w_{small} respectively. Small values of w_{big}, w_{small} lead to small fluctuation of trustworthiness around the average rating number. On the other hand, real applications may require tuning w_{small} to fit the favored speed to gain a good reputation, and tuning w_{big} to fit the favored magnitude of penalty for downgrading performance.

3.3 Evaluation of SensorTrust

In this section, we evaluate *SensorTrust* with data collected from the Intel Berkeley Research lab [98] and Motelab [147] at Harvard University. To implement *SensorTrust* onto the aggregator in a WSN, the aggregator maintains a data table to store the data received from its

children sensor nodes during the most recent transmission period, and another *SensorTrust* table to store current *SensorTrust* values for children nodes. The data table is updated whenever the aggregator receives data, and the *SensorTrust* table is updated once during each transmission period. With the transaction rating results, the update protocol only involves very simple computations. The main cost comes from the storage cost to maintain the two tables, and the implementation of the rating algorithm. Suppose the cluster of the aggregator has a size m. Then both the storage cost and the computation cost are $\Theta(m)$. With polynomial approximation to the exponential function in the rating algorithm, the total overhead is still acceptable.

In the Intel lab, humidity, temperature, light and voltage data were collected from 54 sensors deployed around 35 days, sampled once every 31 seconds. Regarding Motelab data, our uploaded program collected similar data from 14 sensors for 16 hours, sampled once every 10 seconds.

3.3.1 Evaluation with Intel Lab Data

We analyze the efficacy of *SensorTrust* in identifying the trustworthiness of sensor nodes at Intel lab data. Experiments are conducted to compute *SensorTrust* values of different nodes with varying $w_{upgrade}$ and $w_{degrade}$. The results show that the *SensorTrust* value integrates both long-term reputation and short-term risk of wireless communication and data integrity, and accurately captures the change of trustworthiness. Also, $w_{upgrade}$ and $w_{degrade}$ are flexible enough to satisfy specific application requirements.

As a first example, Figure 3.4(a), with a zoom-in view at Figure 3.4(b), depicts the communication *SensorTrust* and the conventional reputation of mote 1 during $[5.55 \times 10^4 \times 31, 5.825 \times 10^4 \times 31]$(seconds). As mentioned in Section 3.2.3, we denote by RFSN-w the approach which updates the *SensorTrust* value by $T_{new} = (1 - w) \times T_{old} + w \times R$, $w \in (0, 1)$. We compute the communication *SensorTrust* with three sets of parameters: (1) $w_{upgrade} = 0.03, w_{degrade} = 0.05$; (2) RFSN-0.03: $w_{upgrade} = w_{degrade} = 0.03$; (3)

RFSN-0.05: $w_{upgrade} = w_{degrade} = 0.05$. Basically, the *SensorTrust* value is more sensitive to changing behavior than the conventional reputation. The conventional reputation at time t is the number of successful deliveries from the beginning time 0 till t divided by the number of all attempted deliveries till t. The conventional reputation in the Figure 3.4 looks almost constant (not actually constant), due to the fact that a relative short-time change doesn't much impact the conventional reputation computed since time 0. Actually, the sensor communication frequently failed for a few hours after the 5.605×10^4-th second. Though conventional reputation doesn't even seem to fall down, the *SensorTrust* values with those three sets of parameters go down towards 0 fast since the 5.605×10^4-th second. When the connection is well maintained for a length of time, the *SensorTrust* values rise, even above its conventional reputation. Comparing the three *SensorTrust* series with different parameters: RFSN-0.03 and RFSN-0.05 underestimate potential risk before the 5.605×10^4-th second, due to the fact that the equality of $w_{upgrade}$ and $w_{degrade}$ causes relatively fast recovery of *SensorTrust* values from communication failures; In contrast, *SensorTrust* values with $w_{upgrade} = 0.03, w_{degrade} = 0.05$ tend to better reflect the potential risk while maintaining a relatively accurate estimate of the sensor's trust level based on its past performance. As soon as the failures since the 5.605×10^4-th second are identified, the *SensorTrust* value is penalized relatively heavily with $w_{degrade} = 0.05$. Though RFSN-0.05 causes the same extent of penalty, the setting of $w_{upgrade} = 0.05$ causes overestimate of the *SensorTrust* values when the communication is better maintained before the 5.605×10^4-th second.

Now we observe how *SensorTrust* values for physical data evolve through another example. Take $w_{upgrade} = 0.0149$ and $w_{degrade} = \frac{w_{upgrade}}{1-w_{upgrade}^2}$ in the following. All the temperature data series from 54 motes is plotted in Figure 3.5(a), with the highlighted series from mote 4. Mote 4's temperature *SensorTrust* is plotted in Figure 3.5(b). It can be easily seen that when mote 4 temperature data get closer to the average temperature, its *SensorTrust* value increases; when mote 4 temperature data start to deviate more from the average temperature, its *SensorTrust* value decreases.

3.3.2 Attack Analysis with Motelab Data

Next we analyze the efficacy of *SensorTrust* against faulty data and malicious data manipulation. We generate a few common types of faults and attacks against the Motelab data. The results indicate the resilience of *SensorTrust* is very good.

The first type is a combination of sleeper attack and stuck-at fault [44]. The sleeper attack is a scenario wherein a node starts misbehaving after creating a good reputation through positive behaviors for a certain amount of time. And the stuck-at fault represents a sensor getting stuck at a wrong data value (sticky value) and remaining there permanently or intermittently. As a combination, we keep the original accurate data of a mote during a certain amount of time, and use sticky value to replace the data thereafter. The experiments show that *SensorTrust* value typically plummets down from a high level towards 0 once the stuck-at fault occurs. As an example, we use a sticky temperature value 0 to replace the temperature data after 1000 samples at mote 63. In Figure 3.6(a), the *SensorTrust* values in terms of temperature data for all motes are plotted against the time, with the thick red line for mote 63, and blue for other motes. The mote 63 creates a good trust level for itself during the first 1000 samples. However, that reputation is quickly destroyed after a few reports of sticky value.

In another type of attack, we generate random data at randomly selected sensor nodes. Due to the random nature of wild environment, it is not easy to tell random data within a reasonable range from normal data. The experiments show that the mote generating random data have much lower *SensorTrust* values than other motes. As an example, we generate random temperature data from 0 to 50 degree Celsius at mote 63 As seen in Figure 3.6(b), mote 63 (thick red line) maintains *SensorTrust* values no more than 0.3, while other nodes (blue) gain *SensorTrust* values of at least 0.8 after the reputation accumulation stage.

The third type of attack is periodic attack. In a periodic attack [168], every time the attacker successfully achieved a cover reputation T1, he will launch attacks until his trust value drops to T2. Then he will provide some good services again to re-build his reputation. It can continue doing so without being detected. The experiments show that with a relatively

big $w_{degrade}$ and a relatively small $w_{upgrade}$, nodes compromised by the periodic attack tend to have much lower *SensorTrust* values than others. In the following example of the periodic attack, mote 63 reports reports sticky value 0 once after correctly reporting data every 10 times. With $w_{upgrade} = 0.02$ and $w_{degrade} = 0.05$, the *SensorTrust* values of mote 63 are much lower than those of other motes (see Figure 3.6(c)).

Finally, we generate anomaly data at randomly selected motes. Results show that motes reporting anomaly data have very low *SensorTrust* values. As an example, we modify the temperature data at mote 63 to be 7 degrees higher than the original data. As seen in Figure 3.6(d), the *SensorTrust* values of mote 63 (thick red line) are always lower than 0.08, while other motes (blue) gradually gain *SensorTrust* values of above 0.5 or even much higher.

3.3.3 Investigation of Multiple Malicious Nodes

During the attack analysis experiments with the Motelab data as described in Section 3.3.2, we note that the number of malicious nodes impacts the performance of *SensorTrust*. Naturally, the more malicious nodes present in a WSN, the more noise it causes to a data trustworthiness system. When the number of the malicious nodes increases, it poses greater difficulty to identify those malicious nodes. Another factor impacting *SensorTrust* is the type of the attack. With a different type of attack, the performance of *SensorTrust* usually shows a slight difference; with more specific domain knowledge about the sensing data, the attackers can often launch more powerful attacks. While it is not likely to exhaust all the types of attacks, we present our investigation of the resilience of *SensorTrust* against multiple malicious nodes exploiting certain known data range and random data generation. Our investigation indicates that with most nodes being honest, *SensorTrust* generally performs well in the presence of multiple malicious nodes. Its performance starts to show reasonable degradation when a large portion of the nodes become malicious attackers.

Specifically, in the investigation experiments, we provide a set of Gaussian-distributed data randomly generated as the original authentic data set. Totally there are 100 nodes, each

of which reports data once in a time unit, from time 1 through time 1000. At any moment from time 1 to time 1000, the original data reported by all the nodes are Gaussian-distributed, with a mean value of 50 and a standard variance of 5. The Gaussian-distributed data are generated randomly through the Matlab software. Now, as an attack, the first few nodes start maliciously reporting data from a uniform distribution on the data interval [30, 70]. To identify the malicious nodes, we will compute the *SensorTrust* values: any node with an average *SensorTrust* value of lower than 0.5 on the time interval [900,1000] will be deemed as a malicious one. Figure 3.7 illustrates the evolution of *SensorTrust* for both honest nodes (upper thin blue lines) and malicious nodes (lower thick red lines). Figure 3.7(a), (b), (c) and (d) indicate that *SensorTrust* can distinguish most malicious nodes from honest nodes with the existence of multiple malicious nodes. Note that the *SensorTrust* values of the malicious nodes slightly increase with an increasing number of malicious nodes. The *SensorTrust* values of the honest nodes fall into the interval [0.6, 0.8]. With 5, 10 or 20 malicious nodes, the *SensorTrust* values of the malicious nodes are usually under 0.5, as shown in Figure 3.7(a), (b) and (c). However, with 30 malicious nodes, the *SensorTrust* values of certain malicious nodes are often around 0.5, as shown in Figure 3.7(d). That implies the greater difficulty in identifying the malicious nodes since their *SensorTrust* values seem to build up a more positive profile.

Experiments with from 1 such malicious node through 60 malicious nodes result in a 0 false positive rate and an reasonably decreasing recall rate. The 0 false positive rate indicates that *SensorTrust* generally do not misjudge an honest node in being a malicious one. The recall rate reflects the percentage of identified malicious nodes among all the malicious nodes. The recall rate is perfectly 100% until the number of malicious nodes goes beyond 42. The recall rate drops to around 60% with 54 malicious nodes, and 30% with 60 malicious nodes.

Overall, our experiments show that *SensorTrust* generally performs well in identifying the trustworthiness of a node in the presence of multiple malicious nodes. Thus, with low

SensorTrust values, the malicious nodes are distinguished from the honest nodes. When a significant portion of the network consists of malicious nodes, the performance of *SensorTrust* starts to degrade.

3.4 Summary

In this chapter, we proposed a resilient trust model, *SensorTrust*, to evaluate data trustworthiness in hierarchical WSNs. In this model, the aggregator maintains trust estimations for children nodes. Unlike previous efforts, our current design of *SensorTrust* mainly focuses on data integrity, though communication robustness can also be incorporated. With this model, past history and recent risk are synthesized in a real-time way that accurately identifies the current trust level. Our model employs the Gaussian model to rate data integrity in a fine-grained style, and a flexible update protocol to adapt to different applications. With acceptable overhead, the *SensorTrust* model is evaluated with the real world sensor data from Intel Berkeley Lab and Motelab at Harvard University, and compared with other approaches. The results indicate great advantage of *SensorTrust* to handle varied faults and attacks.

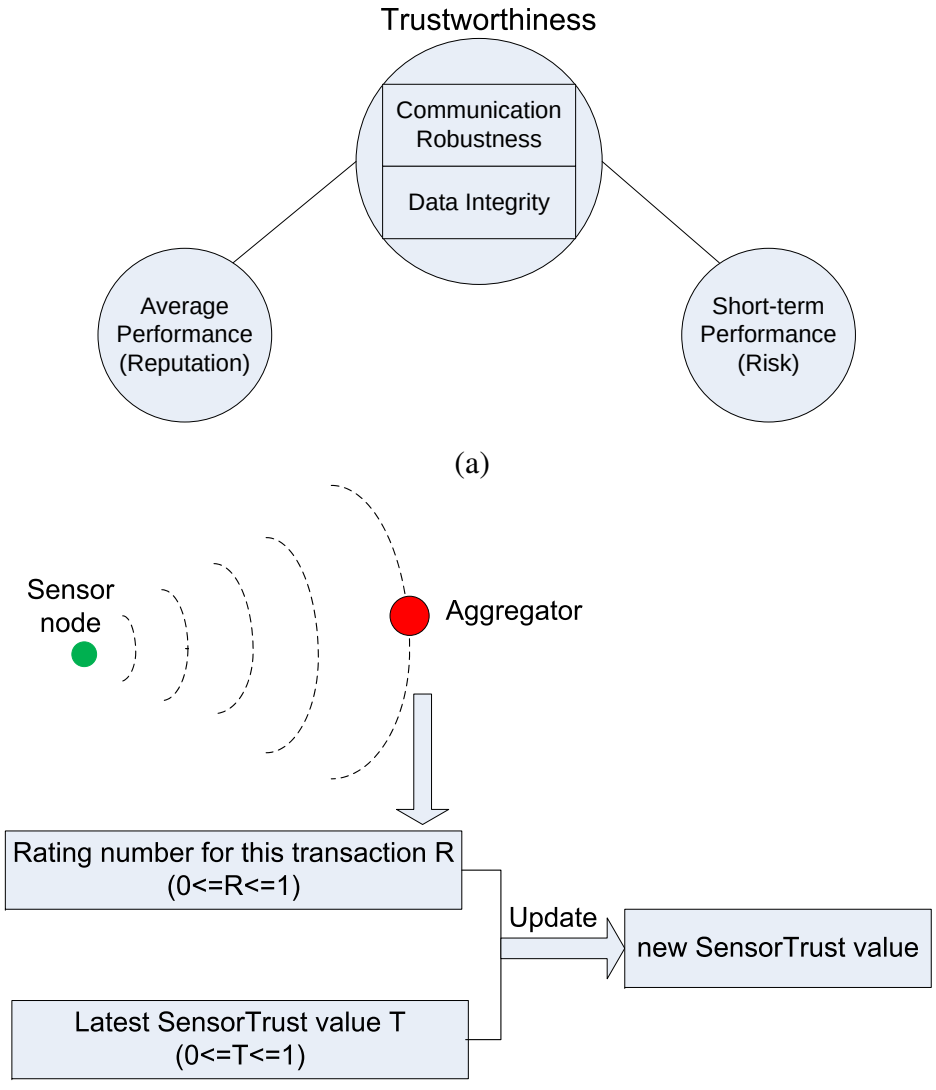


Figure 3.2: (a) The *SensorTrust* model, (b) the update protocol of *SensorTrust*.

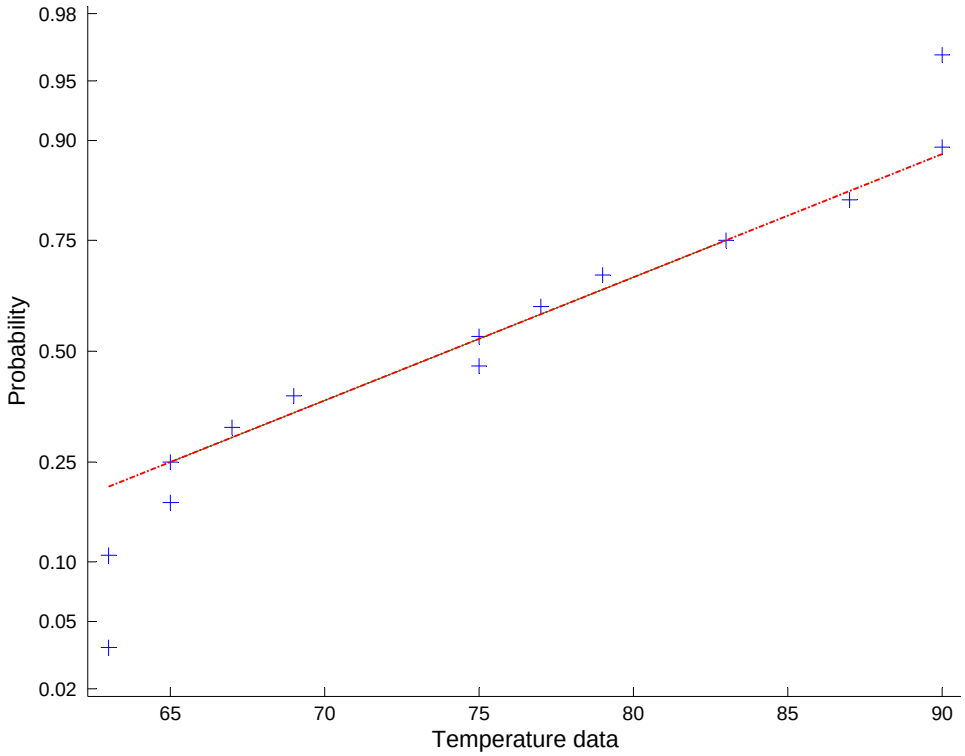
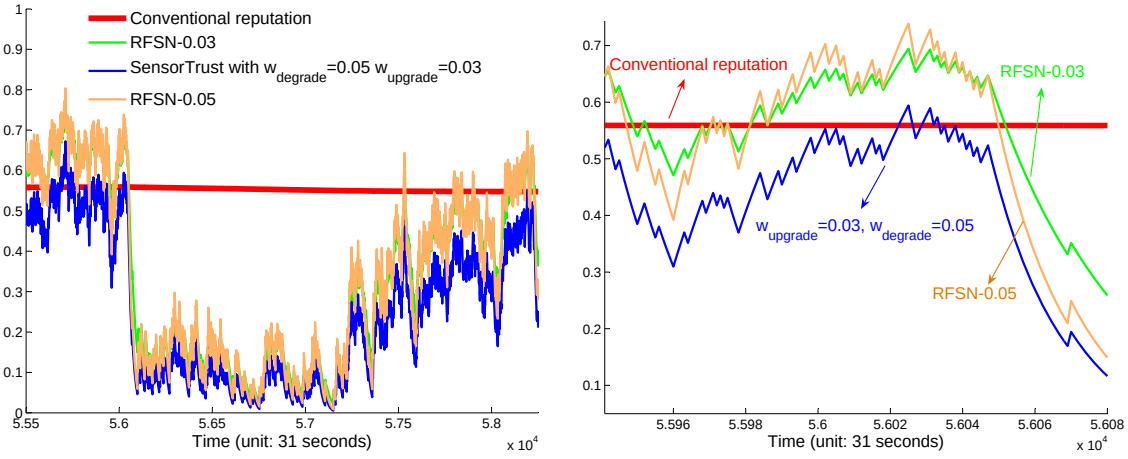


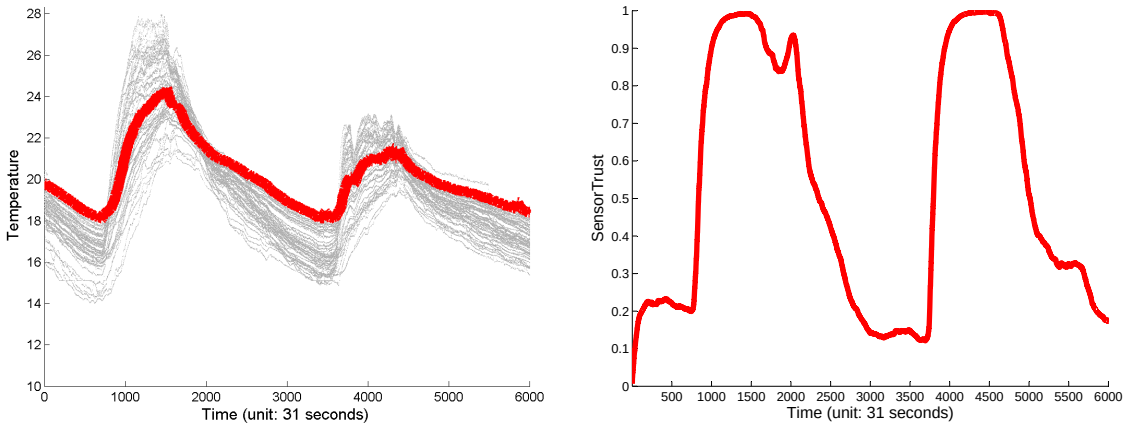
Figure 3.3: Normality testing with normal probability plot.



(a) Reputation Versus Trust

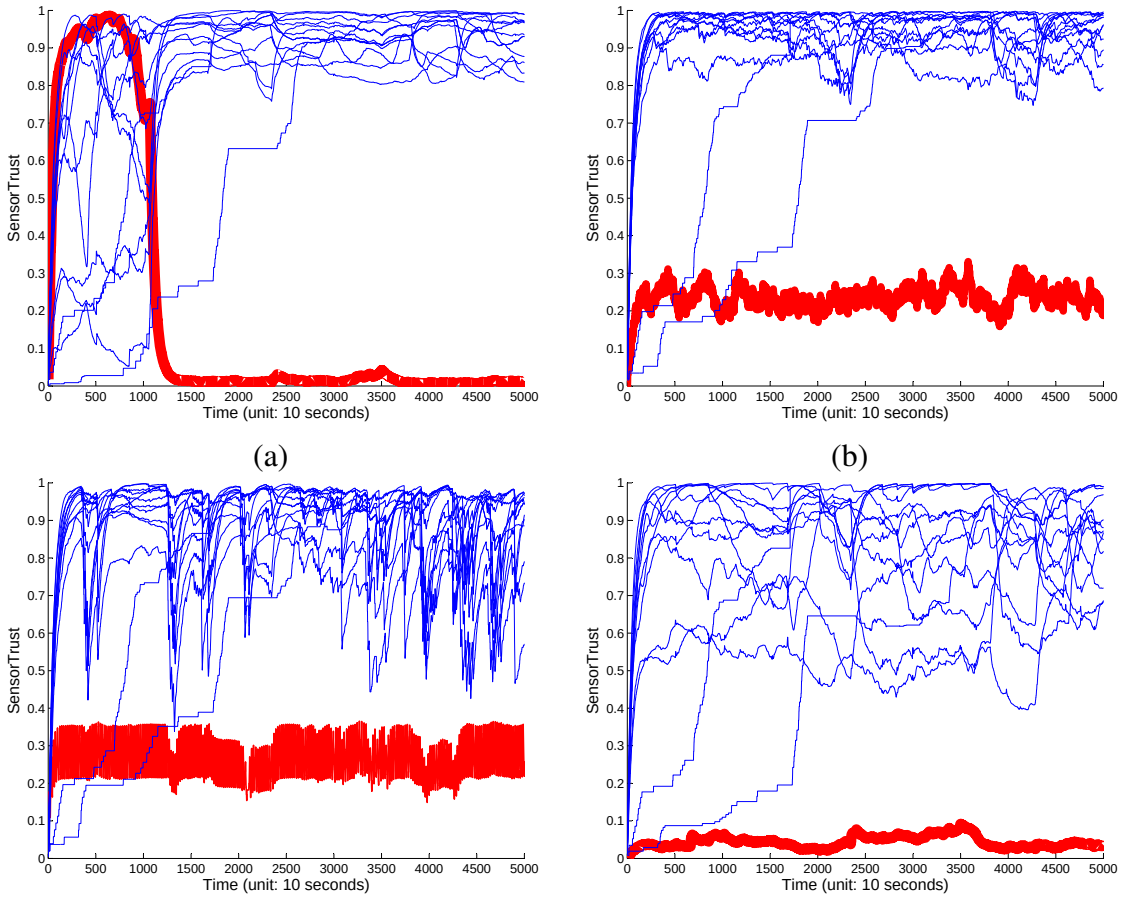
(b) Zoom-in view of (a)

Figure 3.4: Communication between mote 1 and the aggregator (the conventional reputation looks almost constant, though not actually constant, due to the fact that a relative short-time change doesn't much impact the conventional reputation computed since time 0.)



(a) (b)

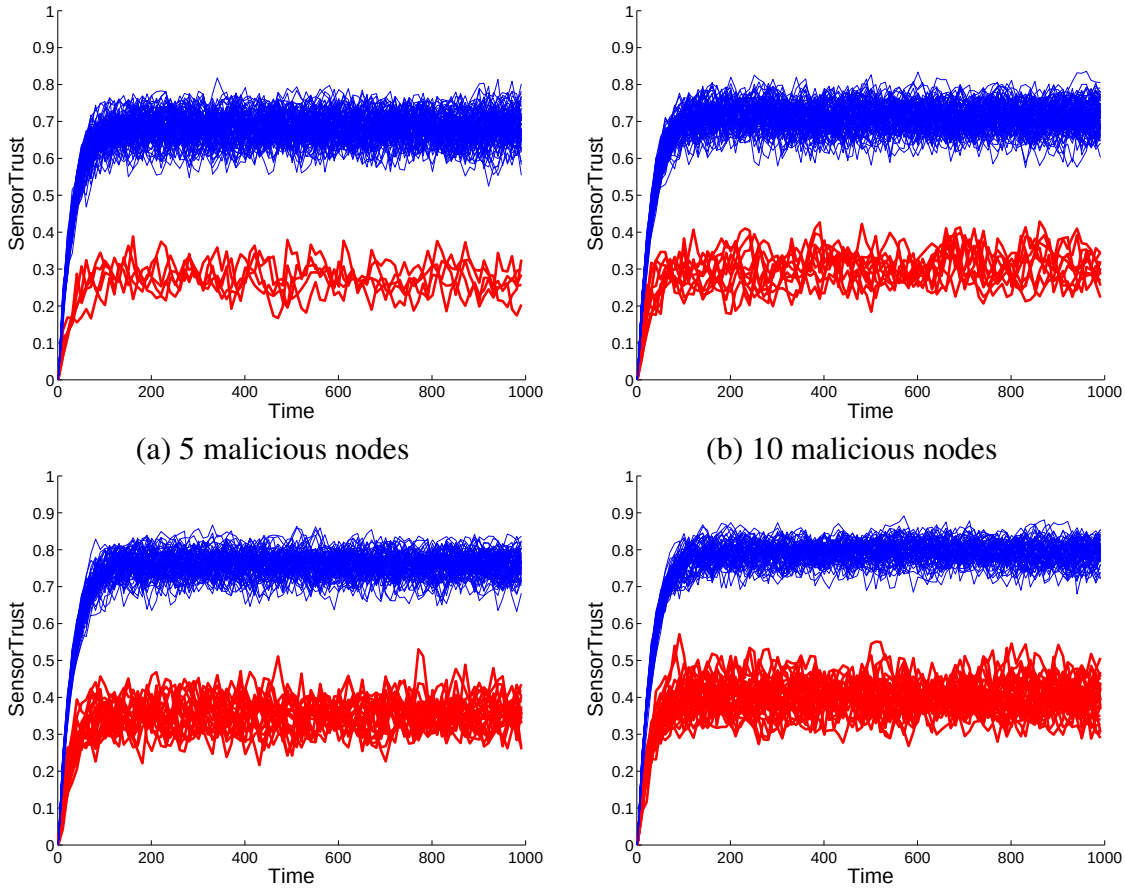
Figure 3.5: (a) temperature data series from all motes (gray) with mote 4 data highlighted (thick red line), (b) *SensorTrust* value in terms of temperature data for mote 4.



(a) (b)

(c) (d)

Figure 3.6: *SensorTrust* values for all motes (blue) with a faulty or attacked mote highlighted (thick red line). Types of faults and attacks: (a) sleeper attack and stuck-at fault, (b) random data generation, (c) periodic attack, (d)data abnormality.



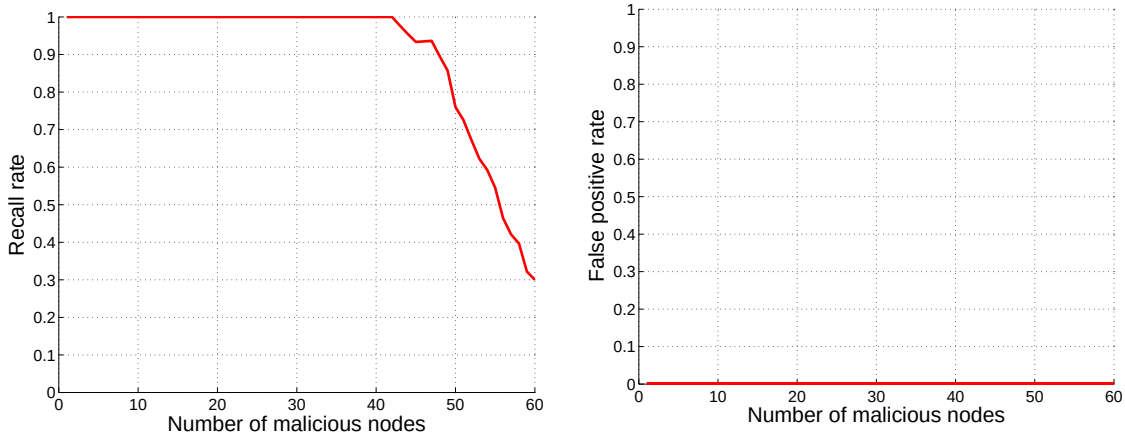
(a) 5 malicious nodes

(b) 10 malicious nodes

(c) 20 malicious nodes

(d) 30 malicious nodes

Figure 3.7: *SensorTrust* values for 100 nodes with the first few nodes being malicious. The upper thin blue lines are for honest nodes while lower thick red lines are for malicious ones. The number of malicious nodes is: (a) 5, (b) 10, (c) 20, (d)30.



(a) Recall rate.

(b) False positive rate.

Figure 3.8: Identity malicious nodes with a *SensorTrust* threshold of 0.5: (a) recall rate; (b) false positive rate (perfectly 0).

CHAPTER 4

LOCALIZATION OF SMALL-SIZED GROUND ROBOTIC VEHICLES

We developed a low-cost, self-contained, accurate localization system (LOBOT) for small-sized ground robotic vehicles. This localization system enhances the wireless sensing systems containing mobile sensors by providing more accurate and highly available location data, with only limited overhead in economic cost and management. The hardware devices LOBOT uses are easily-available at low cost. LOBOT is self-contained in that it virtually requires no external devices or external facility management and that it needs no prior information. LOBOT does not require external reference facilities, expensive hardware, careful tuning or strict calibration. Additionally, LOBOT applies to both indoor and outdoor environments and realizes satisfactory performance. Further, LOBOT maintains low cumulative error.

4.1 Introduction

Small-sized ground robotic vehicles have great potential to be deployed in situations that are either uncomfortable for humans or simply too tedious. For example, a robot may become part of industrial operations, or become part of a senior citizen's life, or become a tour guide for an exhibition center. The robot is kept as small as possible to allow access through narrow passageways such as a tunnel. To fulfill these missions, the robotic vehicle often has to obtain its accurate localization in real time. Considering the difficulty or impossibility in frequent calibration or the management of external facilities, it is desirable to have a self-contained positioning system for the robot: ideally, the localization system should be completely integrated onto the robot instead of requiring external facilities to obtain the position; the system

should work indoors and outdoors without any human involvement such as manual calibration or management. Meanwhile, the cost is expected to be as low as possible.

There exist various localization schemes for ground robotic vehicles. These techniques normally utilize GPS, inertial sensors, radio signals, or visual processing. GPS often becomes inoperable in certain environments such as indoors or in wild forests. Additionally, the GPS operations consume power quickly. As an alternative, a localization system may use various waves including electromagnetic waves of various frequency (e.g., common WiFi radio, Ultra-wideband [161], RFID radio [140], Infrared [117]), laser beam [60], and ultrasound [9]. The radio-based positioning is among the most popular techniques. This technology requires a set of external devices to generate or receive radio signal; as the reference nodes, these external devices should have known positions. The accuracy of the radio-based positioning strongly depends on the proper calibration of the reference devices and the target node [148, 149] as well as a friendly radio environment. Maintaining such a positioning system can be costly and difficult in terms of additional hardware [25, 93, 119], intensive tuning [99], and environmental management. It is also vulnerable to interference from other signals, thus affecting the accuracy of positioning.

Another category of solutions is vision techniques for mobile robot navigation [36]. Generally, these techniques heavily rely on sophisticated techniques on the recognition of an object or shape from images and often have restricted spatial and visional requirements. The performance usually strongly depends on the environment in which the robot operates and the localization suffers frequent failure. Additionally, they may require a known map of the environment. Overall, the vision-based positioning is relatively costly and difficult to implement or maintain.

Additionally, inertial sensors are often used in positioning or navigation systems to detect movement [54, 76, 80, 92, 132]. Different than the radio-based and the vision-based techniques, the operation of inertial sensors is independent of external features in the environment

and they do not need an external reference. The inertial sensors mainly comprise accelerometers and gyroscopes (gyros). An accelerometer measures specific force and a gyroscope measures angular rate. Many inertial systems often require extremely accurate inertial sensors to maintain accuracy, which often causes high cost and calibration difficulty. Being widely-available and inexpensive, the accelerometer is often perceived as a solution for localization. The accelerometer-based positioning schemes generally use the following formula to derive distance from a given acceleration a: $s(t) = \int \int a(t)dtdt$. In spite of being theoretically well founded, empirically, the double integral is likely to cause cumulative error. The methods proposed to correct this error often have not been thoroughly evaluated yet.

To resolve the aforementioned issues, we proposed LOBOT, a low-cost, self-contained localization system for the small-sized ground robotic vehicle. LOBOT identifies the real-time localization through a set of self-integrated inexpensive sensors including an accelerometer, a magnetic field sensor, several motor rotation sensors, and infrequent GPS-augmentation. It detects local relative position with a combination of the accelerometer, the magnetic field sensor and the motor rotation sensors. LOBOT infrequently invokes the GPS-augmentation to assist in identifying global location and correcting drifting errors. LOBOT can be applied to both indoor and outdoor environments. These extra sensing devices including the GPS receiver are integrated onto the ground robotic vehicle and only induce a limited cost to the vehicle. LOBOT does not require any external facilities or prior information and it virtually needs no effort of external maintenance. LOBOT is free of many common requirements or issues raised in other localization schemes such as radio-based schemes and vision-technique-based schemes, such as expensive hardware, external reference facilities, careful tuning or strict calibration, and prior map information. It also has significant improvement in location precision over the purely-accelerometer-based approach. We developed a prototype of the LOBOT system and conducted various field evaluation. The empirical results indicate the satisfactory performance of LOBOT.

The rest of this chapter is organized as follows: the detailed mechanism of LOBOT is described in Section 4.2; the implementation and empirical evaluation of LOBOT are given in Section 4.3; the summary of this chapter is presented in Section 4.4.

4.2 The Design of LOBOT

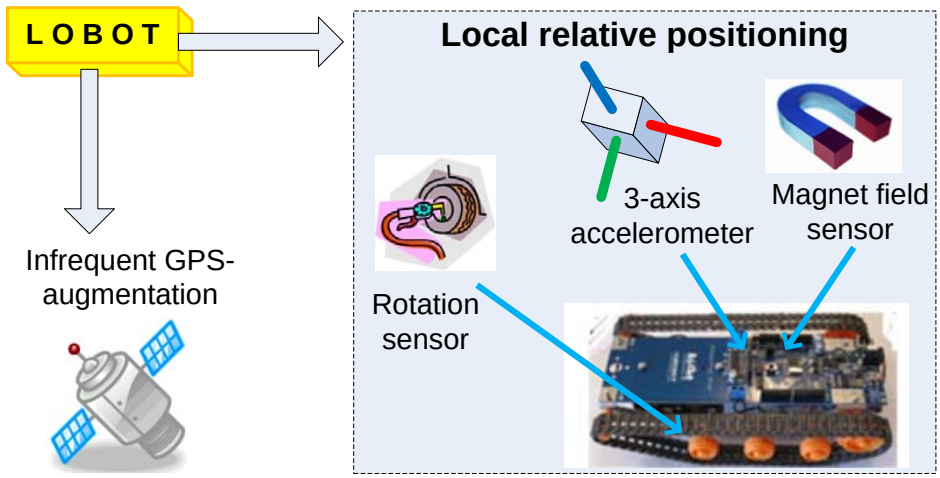


Figure 4.1: The design of LOBOT.

LOBOT localizes a robotic vehicle with a hybrid approach consisting of infrequent absolute positioning through a GPS receiver and local relative positioning based on a 3D accelerometer, a magnetic field sensor and several motor rotation sensors (Figure 4.1). All these sensors are installed on the robotic vehicle. The motor rotation sensors are to detect the rotational movement of the motors and thus infer the travel distance of the robot. An embedded microcontroller inside the robot vehicle takes central control of these sensors and is also responsible for computing the current absolute position. LOBOT infrequently uses GPS to obtain an absolute position and utilizes the accelerometer, the magnetic field sensor and the motor rotation sensors to measure local relative movement since the last known absolution position through GPS. With the GPS data, correction is performed to reduce the cumulative error from the local relative positioning component. The infrequent use of GPS reduces the dependence on the environmental impact, e.g., a small area without GPS signal. As a matter

of fact, even if GPS is available, LOBOT may still only uses the local relative component over a short time period instead of GPS because GPS is known to have error of up to 20m while the local relative component has much lower error over a short time elapse. Additionally, the infrequent use of GPS saves electric power.

The local relative positioning component measures the instantaneous three-dimensional moving direction through both the accelerometer and the magnetic field sensor. It also measures the momentary travel distance for every small amount of time elapse through the rotation sensors attached to the vehicle motors. With the moving direction data together with the momentary travel distance, we can obtain an estimate of the movement vector. This seemingly straightforward strategy, however, has encountered a few major technical issues that arise in practical applications. One lies in the distinction between the world reference system and the on-board relative reference system. Another factor that impacts the localization practice is the way the robotic vehicle operates the motors to move. A further complication comes from the cumulative error.

The overall procedure for LOBOT to decide the position is illustrated by Figure 4.2. Roughly, the local relative positioning infers the momentary moving orientation (Subsection 4.2.2) and estimates the momentary travel distance (Subsection 4.2.3), with the aid of the accelerometer, the magnetic sensor, and the rotation sensors. The local relative positioning accumulates these momentary estimates to compute the position of the vehicle at any time. Over certain time elapse, the infrequent GPS-augmentation is conducted and is used to perform drift correction (Subsection 4.2.4) so as to obtain better position estimate.

LOBOT is a low-cost, self-contained system. All the necessary hardware devices needed to perform the positioning are a GPS receiver, a 3D accelerometer, a magnetic field sensor, and several motor rotation sensors. LOBOT only needs the commodity versions of these devices that come with moderate precision and low prices. For ease of development, our prototype uses a GPS receiver, a 3D accelerometer, a magnetic field sensor from an unlocked HTC Legend smartphone that is sold at no more than $300 at the time of this writing. The

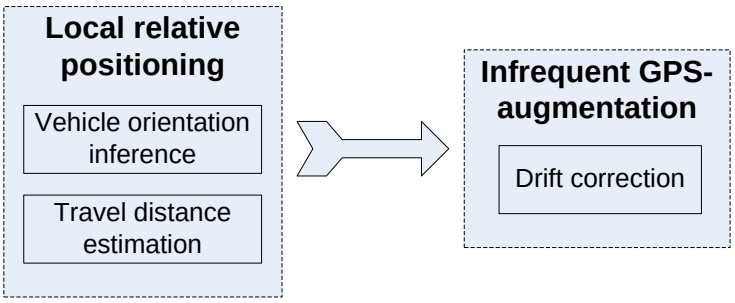


Figure 4.2: The overall procedure of LOBOT.

motor rotation sensors used in this prototype is obtained from a brand of hobby servo motor that sells at $20. Given a complete circuit design, the actual cost of manufacturing a microcontroller chip integrating all these raw sensors (including the GPS receiver) can very likely be brought down to well under $100 per set. Additionally, all these sensing devices including the GPS receiver can be well powered by the battery of the HTC legend smartphone. Compared with the intense power needed to drive a robotic vehicle, these sensing devices induce only limited overhead in the power consumption. Thus, LOBOT is a low-cost system. The self-containedness of LOBOT is reflected in two aspects: virtually no requirement of external devices or external facility management; no prior information needed. All the necessary devices are attached to the body of the robotic vehicle that we need to localize. Except for GPS, LOBOT does not require any external devices (e.g., a reference anchor point). The GPS satellite network is maintained by official organizations and thus the use of a GPS receiver virtually needs no effort to maintain external facilities. Unlike many positioning schemes based on vision recognition techniques, LOBOT does not require prior information of the environment either.

4.2.1 Reference Frames

To determine the current moving orientation, we will first need to make a choice on the reference frame. The direction is expressed in a coordinate system relative to the reference

frame chosen. Here we will first briefly cover the definition of the reference frames and their meanings. We adopt a right-handed orthogonal reference frame, *LOBOTFrame*$\{X_L, Y_L, Z_L\}$ as follows: the Y axis is parallel to the magnetic field of the earth and points towards the magnetic north pole; the Z axis points towards the sky and is parallel to the gravitational force; the X axis is defined as the outer vector product of a unit vector of Y and that of Z so that $\{X_L, Y_L, Z_L\}$ defines a right-handed orthogonal reference frame. For the purpose of measuring relative movement, the choice of the origin does not affect our result and thus we omit the origin when describing the reference frames. Additionally, we assume that in an area being explored by the robot the directions of both the gravitational force and the earth's magnetic field are constant. As a matter of fact, the gravitational direction rarely changes in a city-magnitude area. The change of the earth's magnetic field direction in such an area is usually also negligible without the existence of another strong magnetic field. If the strength of another magnetic field is so strong that it causes a noticeable difference on the readings of the magnetic sensor, LOBOT will switch to the pure GPS-based mode if the GPS service is available. Thus, we have a well-defined reference frame *LOBOTFrame* for measuring the relative movement of the vehicle. Roughly, the X axis is tangential to the ground at the robot's current location and points east; the Y axis is tangential to the ground and points north (it is slightly different than the magnetic north); the Z axis roughly points towards the sky and is perpendicular to the ground.

Before introducing how to determine the robot's moving orientation, we first show three other closely related right-handed orthogonal reference frames. Unlike *LOBOTFrame*, these frames change as the robot moves. The first one is the reference frame relative to the rigid body of the robot, which we name *VehicleBodyFrame*. *VehicleBodyFrame* is not a static frame when the vehicle moves. Specifically, *VehicleBodyFrame* is a right-handed orthogonal reference frame $\{X_V, Y_V, Z_V\}$, described as follows: the Y axis is parallel to the lines connecting the centers of a motor and another motor right behind it, and points to the front; the Z axis points towards the sky and is perpendicular to the surface containing all the centers of

the motors; the X axis is defined as the outer vector product of a unit vector of the Y axis and that of the Z axis so that $\{X_V, Y_V, Z_V\}$ defines a right-handed orthogonal reference frame (the X axis points to the right side of the vehicle).

Another relative reference frame, denoted as *AccelerometerBodyFrame*, is also a right-handed orthogonal reference frame $\{X_A, Y_A, Z_A\}$ on which the accelerometer reading is based. Usually the 3D reading from an accelerometer indicates how the measured acceleration is decomposed into these three axis directions. This reference frame is relative to the circuit board of the accelerometer and is defined by the manufacturer. Two of the axes are often parallel to the circuit board. Similarly, the last reference frame which we name as *MagneticSensorBodyFrame*, is another right-handed orthogonal relative reference frame $\{X_M, Y_M, Z_M\}$ on which the magnetic sensor reading is based. Note that *VehicleBodyFrame*, *AccelerometerBodyFrame* and *MagneticSensorBodyFrame* may all change when the vehicle moves; however, a fixed installation ensures inherent unchanged relations between *VehicleBodyFrame* and the two latter frames and such relations can be decided during installation.

More Intuitive Description of Reference Frames

We now illustrate the several reference frames used through figures and more intuitive description. *LOBOTFrame* is used as the reference frame when deciding the momentary moving orientation of a ground robotic vehicle. Figure 4.3(a) illustrates the three axes of *LOBOTFrame*: roughly, the X axis roughly points east; the Y axis points towards the magnetic north pole; the Z axis points outwards into the sky. *LOBOTFrame* only depends on current earth orientation and can be regarded as static. *VehicleBodyFrame* is used to reflect the current orientation of the rigid body of the robot. *VehicleBodyFrame* changes as the vehicle moves. Figure 4.3(b) illustrates the three axes of *VehicleBodyFrame*: roughly, the X axis points to the right side of the vehicle; the Z axis points outwards into the sky; the Y axis points towards the front.

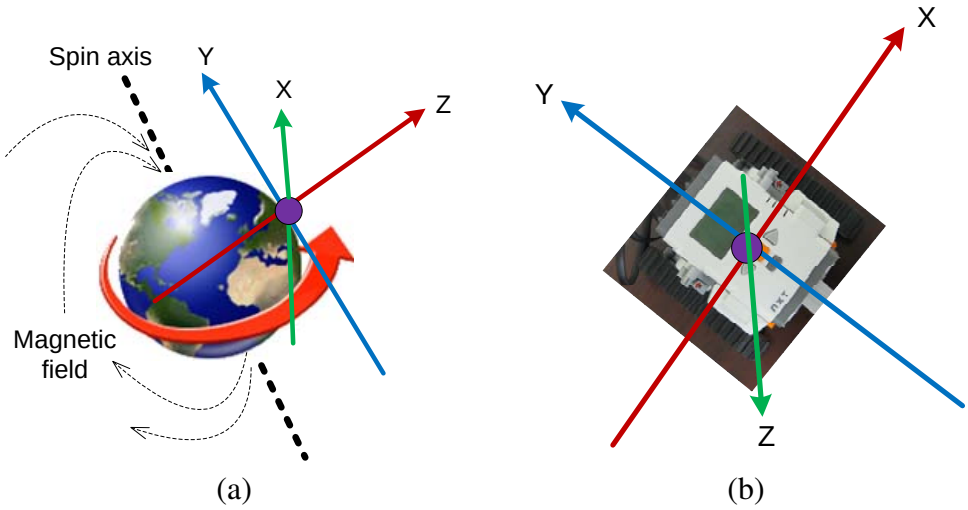


Figure 4.3: The three axes of (a) *LOBOTFrame* and *VehicleBodyFrame*.

AccelerometerBodyFrame and *MagneticSensorBodyFrame* are the references frames with which the three-dimensional sensing readings are interpreted from an accelerometer and a magnetic sensor respectively. These reference frames are defined by the hardware manufacturers and reflect the current orientation of the corresponding sensor boards. When these sensors are attached to a fixed position on a robotic vehicle, *AccelerometerBodyFrame* and *MagneticSensorBodyFrame* change as the vehicle moves. Figure 4.4 illustrates a possible configuration of the three axes of a sensor board: roughly, the X axis points to the right side of the sensor board; the Y axis points towards the front; the Z axis points outwards into the sky;

Table 4.1 summarizes these reference frames and their dependencies.

Table 4.1: Reference frames and their dependencies

Frame	Depends on
LOBOTFrame	Earth
VehicleBodyFrame	Vehicle body orientation
AccelerometerBodyFrame	Accelerometer sensor board orientation
MagneticSensorBodyFrame	Magnetic sensor board orientation

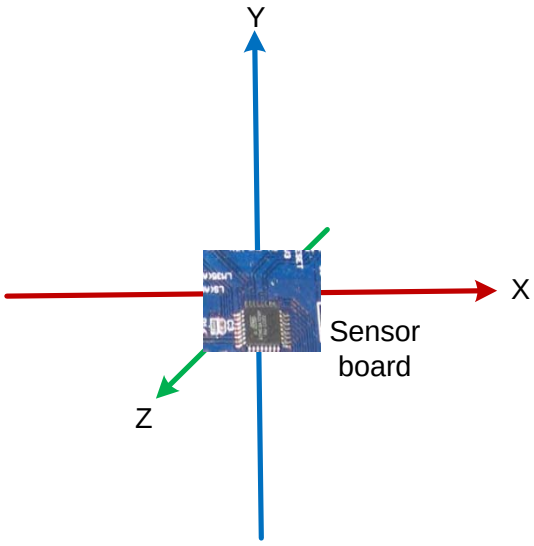


Figure 4.4: The three axes of of a sensor board.

Finally, we briefly describe the relationship between each of these reference frames. The relationship between *VehicleBodyFrame* ($\{\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V\}$) and *LOBOTFrame* ($\{\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L\}$) basically reflects the current orientation of the vehicle. Specially, we need to express the three vectors $\{\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V\}$ in terms of the other three vectors $\{\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L\}$. Then we will know which direction the vehicle moves towards (on earth), including the slope of the current ground surface. The relationship between *AccelerometerBodyFrame* and *VehicleBodyFrame* reflects the mounting direction of the accelerometer on the vehicle. There is a similar relationship between *MagneticSensorBodyFrame* and *VehicleBodyFrame*.

4.2.2 Inferring Orientation of Robotic Vehicle

Now we describe how LOBOT infers the current instantaneous moving direction of the robotic vehicle relative to *LOBOTFrame*, which is a static frame (relative to the earth). Denote the unit vectors along the axes of each reference frame (normalized basis vector) as in Table 4.2. To infer the orientation of the vehicle, it is enough to express $\{\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V\}$ in

Table 4.2: Reference frames and their normalized basis vectors

Frame	Normalized basis vectors
LOBOTFrame	$\{\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L\}$
VehicleBodyFrame	$\{\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V\}$
AccelerometerBodyFrame	$\{\widehat{X}_A, \widehat{Y}_A, \widehat{Z}_A\}$
MagneticSensorBodyFrame	$\{\widehat{X}_M, \widehat{Y}_M, \widehat{Z}_M\}$

terms of $\{\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L\}$. Given the gravitational acceleration vector g, then

$$\widehat{Z}_L = -\frac{g}{\|g\|} \tag{4.1}$$

Let the normalized accelerometer reading be (a_1, a_2, a_3) relative to *AccelerometerBodyFrame*. Then

$$\widehat{Z}_L = -\frac{g}{\|g\|} = a_1 \cdot \widehat{X}_A + a_2 \cdot \widehat{Y}_A + a_3 \cdot \widehat{Z}_A \tag{4.2}$$

Similarly, given the normalized reading (m_1, m_2, m_3) from the magnetic sensor, we have

$$\widehat{Y}_L = m_1 \cdot \widehat{X}_M + m_2 \cdot \widehat{Y}_M + m_3 \cdot \widehat{Z}_M \tag{4.3}$$

Let T_{AV} be the transformation matrix between *AccelerometerBodyFrame* and *VehicleBodyFrame*, T_{MV} be the transformation matrix between *MagneticSensorBodyFrame* and *VehicleBodyFrame*, so that

$$(\widehat{X}_A, \widehat{Y}_A, \widehat{Z}_A) = (\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V) \cdot T_{AV} \tag{4.4}$$

$$(\widehat{X}_M, \widehat{Y}_M, \widehat{Z}_M) = (\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V) \cdot T_{MV} \tag{4.5}$$

Thus, we have the following equations:

$$\begin{aligned} \widehat{Z}_L &= (a_1, a_2, a_3) \cdot (\widehat{X}_A, \widehat{Y}_A, \widehat{Z}_A)' && (4.6) \\ &= (a_1, a_2, a_3) \cdot T'_{AV} \cdot (\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V)' && (4.7) \\ \widehat{Y}_L &= (m_1, m_2, m_3) \cdot (\widehat{X}_M, \widehat{Y}_M, \widehat{Z}_M)' && (4.8) \\ &= (m_1, m_2, m_3) \cdot T'_{MV} \cdot (\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V)' && (4.9) \end{aligned}$$

Now, we are able to construct a special orthogonal matrix as the transformation matrix T_{LV} between *LOBOTFrame* and *VehicleBodyFrame* as follows: the second column vector of T_{LV} is:

$$((m_1, m_2, m_3) \cdot T'_{MV})' = T_{MV} \cdot (m_1, m_2, m_3)' \tag{4.10}$$

The third column vector is:

$$((a_1, a_2, a_3) \cdot T'_{AV})' = T_{AV} \cdot (a_1, a_2, a_3)' \tag{4.11}$$

The first column vector will be the outer product of the second column vector and the third column vector. T_{LV} is determined in this way because the unique transformation matrix between $\{\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L\}$ and $\{\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V\}$ must be an orthogonal matrix with a determinant 1. Consequently, we have constructed the transformation matrix T_{LV} between *LOBOTFrame* and *VehicleBodyFrame* from T_{AV}, T_{MV}, the accelerometer readings and the magnetic sensor readings, such that

$$(\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L) = (\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V) \cdot T_{LV} \tag{4.12}$$

All the above computation involves only a limited number of basic arithmetic operations. Considering that an orthogonal matrix has its inverse being its transpose, we have

$$(\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V) = (\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L) \cdot T_{LV}^{-1} \quad (4.13)$$

$$= (\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L) \cdot T_{LV}^{'} \quad (4.14)$$

Therefore, we have achieved expressing $\{\widehat{X}_V, \widehat{Y}_V, \widehat{Z}_V\}$ in terms of $\{\widehat{X}_L, \widehat{Y}_L, \widehat{Z}_L\}$ through limited algebraic arithmetic operations and thus determined the orientation of the vehicle. The question whether the robotic vehicle is moving forward or backward can be decided from the readings (positive or negative) of the rotation sensors.

Note that the above derivation assumes that the readings of the accelerometer reflect the gravitational force. When the robotic vehicle is moving, the accelerometer measurement often involves the movement acceleration. However, the movement acceleration for such a robotic vehicle is usually a very small fraction of the gravitational acceleration. As verified in our experiments, the effect of movement acceleration is negligible; even if it might show a considerable value during speeding up and braking, the time elapse in which it occurs is so short that it almost has no observable effect to localization.

4.2.3 Travel Distance

After inferring the instantaneous orientation of the robotic vehicle, we also need to know the momentary travel distance so as to compute the momentary relative movement. The rotation sensor attached to a motor continually measures the rotating angle. Let r be the rotation sensor reading in degrees, d be the wheel's diameter, then the travel distance of the wheel's movement is $\frac{r \cdot \pi \cdot d}{360}$. In the case of slippage and obstacle, a few recent research projects have been developed to handle such issues using methods such as sensing modalities and obstacle avoidance [123].

Another important issue we need to address relates to the way the robotic vehicle operates its

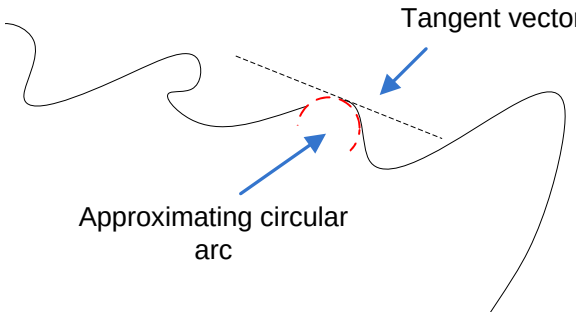


Figure 4.5: Approximate curved path locally by circular arcs.

motors. It is common that a robotic vehicle may make turns or follow a curved path through adjusting its two sides of motors at different speeds and even in reverse direction. Now, the question is how to calculate the moving distance given two different rotation sensor readings, one on each side. First, we observe that any small segment of movement, in a short enough time, can be perceived as part of a circular movement around a certain origin. This observation can be made even when the two sides of wheels move in reverse direction. As an extreme scenario, when the vehicle makes a turn by reversing the two sides of motors at exactly the same magnitude of speed, the approximating arc has a radius of zero. In mathematical terms, a local curve, if short enough, can be approximated by a small arc with the same curvature and tangential at the intersection, as illustrated in Figure 4.5. The curvature reflects how fast the curve turns at a point and depends on both the first derivative and second derivative of the curve. Approximating a curve locally with such an approximating arc produces a negligible cumulative difference when computing distance; that is because the approximating arc locally has almost the same first and second derivatives.

We claim that the travel distance of the robotic vehicle can be approximated by the average of the two side motor's travel distance. A motor may rotate either forward or backward; it rotates forward (backward) in an attempt to move the vehicle forward (backward). Correspondingly, in addition to the absolute distance measured, each reading of rotation sensor is assigned a sign: positive for forward rotation and negative for backward rotation. When

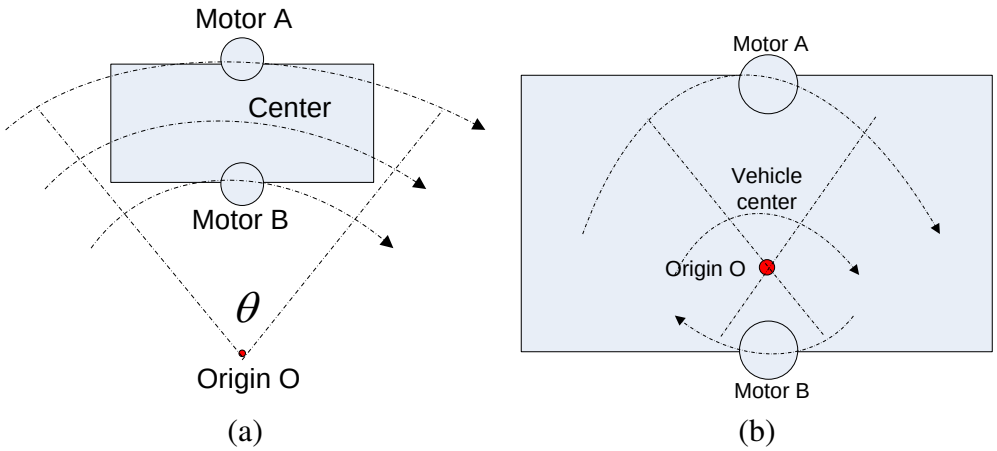


Figure 4.6: Travel distance with different-pace motors: (a)same direction; (b)reverse direction.

the two sides' motors are moving in reverse direction, a positive distance is recorded as one side's reading and a negative distance for the other side. The robotic vehicle's direction is determined by the resulting average's sign. First, we discuss the case when the two motors are moving in the same direction but at different pace. As illustrated in Figure 4.6(a), the center of the vehicle moves in a arc equally between Motor A's trace arc and Motor B's trace arc. It is straightforward that the center's arc length is the average of Motor A's arc length and Motor B's. Thus, we just theoretically proved the claim in the case that Motor A and B move in the same direction but at different pace. Next, we discuss the case that Motor A and B move in reverse direction. In this case, as shown in Figure 4.6(b), the origin O around which the whole vehicle almost circularly moves is between the two motors. It is closer to the one with the smaller absolute pace. A bit straightforward geometry shows that the center's travel distance is the average of Motor A's and B's, with Motor A and B having different signs. The sign of the average determines the moving direction of the vehicle center.

4.2.4 Drift Correction

As in many inertial systems, the localization computed through movement direction and travel distance tends to show drifting effect after a while. Figure 4.7 compares the the trace

retrieved in one of our outdoor experiments through our local relative positioning and through GPS. We observe that positioning purely through local relative positioning gradually drifts from the correct position and finally accumulates large error. Thus, LOBOT needs to apply

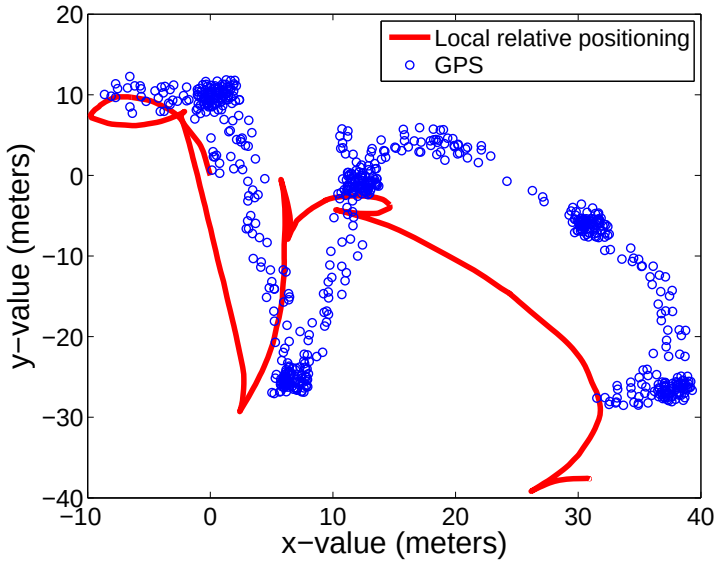


Figure 4.7: Drift in local relative positioning.

drift correction to the localized results by utilizing the absolute position obtained from GPS.

LOBOT requests GPS sampling in an adaptive way that incorporates both location accuracy and energy use. The more frequent GPS sampling likely results in better correction of positioning; but more frequent GPS sampling also means significantly higher cost of power consumption [111, 175]. Roughly, LOBOT adjust its GPS sampling frequency according to the magnitude of the cumulative error of the local relative positioning. When the cumulative error of the local relative positioning between the current GPS sampling and its preceding GPS sampling increases, LOBOT increases its GPS sampling frequency accordingly; otherwise, LOBOT reduces its GPS sampling frequency. Specifically, let $CErrThd$ be the tolerant threshold of the cumulative error of the local relative positioning between two consecutive GPS samplings; P be the time elapse between the two most recent GPS samplings; $CErr$ be the cumulative error of the local relative positioning between these two most recent GPS

samplings. Then the time elapse from the most recent GPS sampling to the next GPS sampling will be $P \cdot CErrThd/CErr$. In practice, to increase stability, LOBOT adopts a GPS sampling gap period slightly lower than $P \cdot CErrThd/CErr$. When the GPS signal is not available, LOBOT periodically wakes up the GPS receiver to check its availability and then puts it to sleep.

LOBOT assumes identical distribution of cumulative error among all time periods of equal length. Let the probability sample space be the set X of all possible localization-related events, $err(X,t)$ be the random error of local relative positioning at time t, and $corr(X,t)$ be the correction at time t. $corr(X,t)$ is the difference between the position obtained through relative positioning and the ground truth. err and $corr$ are both stochastic processes. Let the time start at 0 (last successful GPS request), end at T(the current GPS reading time); assume LOBOT performs local relative positioning at time $1, 2, 3..., T-1, T$. Here we analyze the correction with these simplified assumptions in mind; in fact, our reasoning works with a more general situation with the same logic. Then $corr(X,0) = 0$. We have

$$corr(X,t) = \sum_{i=0}^{t} err(X,i), 0 < t < T \tag{4.15}$$

According to the maximal-likelihood estimation, an optimal estimate of $corr(X,t)$ is its mean value

$$E(corr(X,t)) = \sum_{i=0}^{t} E(err(X,i)) \tag{4.16}$$

$$= t \cdot E(err(X,1)) \tag{4.17}$$

We also have

$$E(err(X,1)) = E(corr(X,T))/T \tag{4.18}$$

Therefore, combining the above two equations, we have

$$E(corr(X,t)) = t \cdot E(err(X,1)) \quad (4.19)$$

$$= t \cdot E(corr(X,T))/T \quad (4.20)$$

Again, based on the principle of maximal-likelihood estimation, the mean value $E(corr(X,T))$ has its estimated value being the difference between the current GPS-supplied reading and the last position obtained through relative positioning. Additionally, an optimal estimate of the random correction $corr(X,t)$ at time t is $t \cdot E(corr(X,T))/T$. Therefore, to correct the drift at time t, we only need to estimate $E(corr(X,T))$ and then add $t \cdot E(corr(X,T))/T$ to the original position estimate. $E(corr(X,T))$ is estimated to be the difference between the current GPS-supplied reading and the last position obtained through local relative positioning.

Finally, it is possible that LOBOT is inactivate first and then becomes active when there is no GPS signal. In this situation, LOBOT is only able to compute its relative movement until it receives a GPS signal in the future. Once a GPS sampling is available, it starts to trace back and restore all the absolute location before that point. If no GPS signal is available, LOBOT will interpolate one of it absolute position linearly with respect to time and derive the rest using its recorded relative movement.

4.3 Implementation And Empirical Evaluation

To implement LOBOT, we used a low-cost LEGO MINDSTORM NXT 2.0 vehicle robot [85] and a moderately priced HTC Legend smart phone [84] as shown in Figure 4.8. The HTC Legend phone is mounted onto the robot, merely to supply a set of sensors: an accelerometer, a magnetic sensor and a GPS. In our experiments, the HTC phone is lifted higher to avoid the magnetic interference from both the robot and the ground. Powered by six AA batteries, this LEGO NXT robot moves on its two servo motors (one on the left and the other on right). The

Figure 4.8: The LEGO NXT robot and the HTC Legend phone.

two servo motors can rotate at their own user-specified speeds, either in the same direction or reverse, providing flexible movement. Their rotating speeds can be changed by user programs at any moment. The LEGO NXT has a set of built-in rotation sensors to continually measure the rotating distance of each motor. The HTC Legend phone has an accelerometer (G-sensor), a magnetic sensor (digital compass) and an internal GPS. Our programs control the motor's movement, collect the data from rotation sensors, the accelerometer, the magnetic sensor as well as GPS.

We performed repeated experiments indoors and outdoors on the main campus of Wayne State University, scaling from 1m x 1m (meter) areas up to areas of 50m x 50m. The LEGO robot randomly moves from its minimal speed (the speed of a snail) to its full speed (several inches per second) and may change its speed and direction every few seconds. It may also operate its two motors at different pace or reversely to follow curved path and make turns. These experiments computed the location data on all three axes: x (East), y (North) and z (upward). Each experiment lasts from 1 minute to 20 minutes. The programmed robot randomly decided its next movement after every certain amount of time from 5 seconds to 1 minute.

The two approaches, LOBOT and the purely accelerometer-based approach, were both executed simultaneously during each experiment. The GPS raw data were collected during outdoor experiments when applicable. To get the ground truth, we performed manual

recording of positions in most cases and camera-assisted positioning in small areas. Our experiments indicate that the purely accelerometer-based approach cannot achieve satisfactory results within the context of localizing a ground robotic vehicle like the LEGO robot we used. In contrast, LOBOT, with a low-cost setting, realizes relatively accurate positioning either indoors or outdoors. Although the pure local relative positioning component of LOBOT shows the cumulative drifting effect, LOBOT well compensates the drift through the infrequent GPS-augmentation.

4.3.1 Groud Truth Retrieval

To get the ground truth, we performed both manual measurement and a camera-assisted positioning approach. We performed manual recording of positions for both indoor and outdoor experiments of varied scales. For each experiment, from 5 to 20 positions were manually recorded and the recordings were temporally evenly distributed.

As for the experiments within small areas, to facilitate the information retrieval, we performed a series of camera-assisted positioning to replace the manual measurement as the ground truth. Such experiments are restricted to those occurring indoors and within a 1m x 1m square coverage. The restriction is out of two considerations: first, it is difficult to deploy the camera-assisted positioning outdoors; second, for experiments in a relatively large area, with our approach, the images produced cannot well distinguish a robot from other spots on the images without further sophisticated (often slow) object recognition techniques. As verified against the manual recordings, the camera-assisted positioning has an accuracy of within 6cm.

For the camera to detect the location, as in Figure 4.9, we place the robot vehicle on a small flat area, over which a camera is installed. The camera faces strictly perpendicularly towards the ground. A small piece of red tag is attached to the center of the robot on the top, which distinguishes that spot from every other spot. Taking advantage of that fact, our program continuously collects the latest frame data from the camera at a speed of no less

than one frame per second and analyzes the frame to retrieve the position of the robot in the image. It successfully finds the central red spot with at least 95% of the frames. The occasional failure is due to glaring image shots and illumination change. To map a spot found in the image back to the original physical position, we only need to scale the image linearly to certain point when its new scale exactly matches the physical area the camera actually covers.

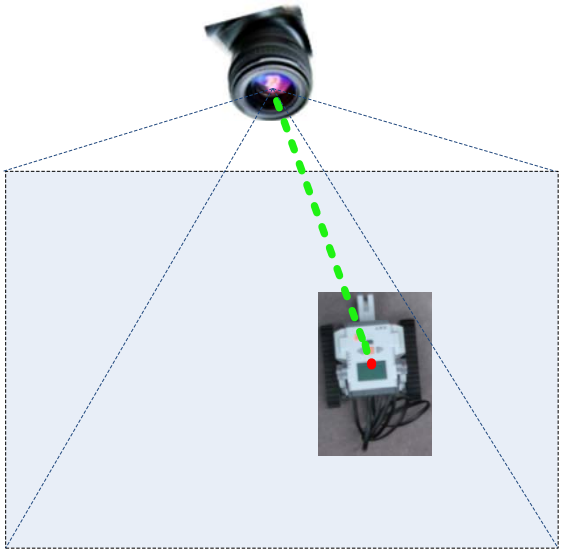


Figure 4.9: Camera-based positioning.

4.3.2 Inaccuracy of Sensing Data

Before dipping into the detailed performance analysis, we would like to observe the inaccuracy of the received sensing data. The sensing data usually display certain deviation from the true sensing value due to various issues from the hardware or the software. When such inaccuracy starts to accumulate, the resulting location might noticeably deviate from the ground truth. A successful localization system should at least be able to reduce the cumulative errors. It is noteworthy that the various positioning techniques often differ not by their theoretical soundness, but by their capability to resist data inaccuracy. The purely accelerometer-based

positioning approach has its strong theoretical foundation from the Newton's Second Law of Motion; however, the position resolved from the acceleration data might quickly deviate from the ground truth. Admittedly, our LOBOT system is also impacted by the cumulative error from the rotation sensor, the accelerometer, the magnetic sensor and the GPS. Fortunately, in the first place, LOBOT tends to have much lower cumulative error than the accelerometer-based approach; further, after performing the GPS-augmentation, the remnant of the cumulative error is well under an acceptable range, considering the low cost of LOBOT.

Our collected data suggest that all the sensors except GPS are able to capture the small movement changes. We retrieved a series of raw data from the accelerometer, the magnetic field sensor, the motor rotation sensor, and the GPS receiver. The data indicate that these sensors are sensitive to even small movement changes. As examples, we show certain data in Figure 4.10, Figure 4.11 and Figure 4.12. Figure 4.10 plots the eastern component of the detected instantaneous orientation based on the magnetic sensor and the accelerometer. When the robotic vehicle changes movement direction, the computed orientation responds by fast adjustment. Figure 4.11 shows the y-component of the instantaneous acceleration detected by the accelerometer when the robotic vehicle randomly adjusts its speed. The acceleration data quickly captures such speed changes. Figure 4.12 describes the motor rotational distance over each few milliseconds detected by the rotation sensor when the vehicle randomly switch between random movement and standing. Though the rotation sensor may produce errors, the figure roughly matches the actual movement pattern.

While these sensors are capable of capturing instantaneous movement, the accuracy of the positioning results are strongly impacted by the specific localization approaches being used. The sensing error varies, depending on the sensors. Generally, the magnetic sensor, motor rotation sensor, and the accelerometer tend to show small instantaneous sensing error; the GPS receiver may produce a relative large error in location (Figure 4.13). The very small instantaneous inaccuracy of the acceleration data could lead to large positioning errors if the acceleration is used as the exclusive raw data for positioning. That is due to the major

quadratic effect in computing the travel distance from the acceleration: $S = vt + \frac{1}{2}at^2$, a being the acceleration. Even with a perfect instantaneous acceleration, the inaccuracy resulting from applying that value as estimation for a whole small time interval could be detrimental.

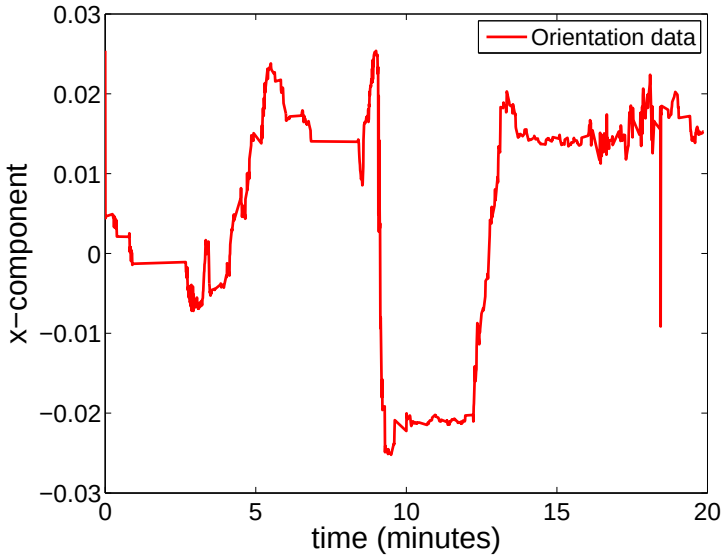


Figure 4.10: Sensing data: orientation.

While the schemes aforementioned may suffer from the quadratic effect, LOBOT involves only linear computation among the raw data. It tends to accumulate errors much slower than the accelerometer-based approach. Figure 4.14 compares the resulting location along the x-axis in one of our experiments among the three approaches: LOBOT, the purely accelerometer-based approach, and the manual measurement. As reflected by the manual measurement, the vehicle moves along x-axis back and forth in varied speeds and with constant standing. The location result from LOBOT matches the manual measurement with very small cumulative errors. However, the purely accelerometer-based approach wrongly suggests that the vehicle is almost standing all over the time. In spite of the fact that the acceleromter can capture sensitive movement, the quadratic effect in approximating errors and sensing errors has developed into a serious location deviation. In another experiment as described by Figure 4.15, the vehicle stays almost static along the z-axis as suggested by

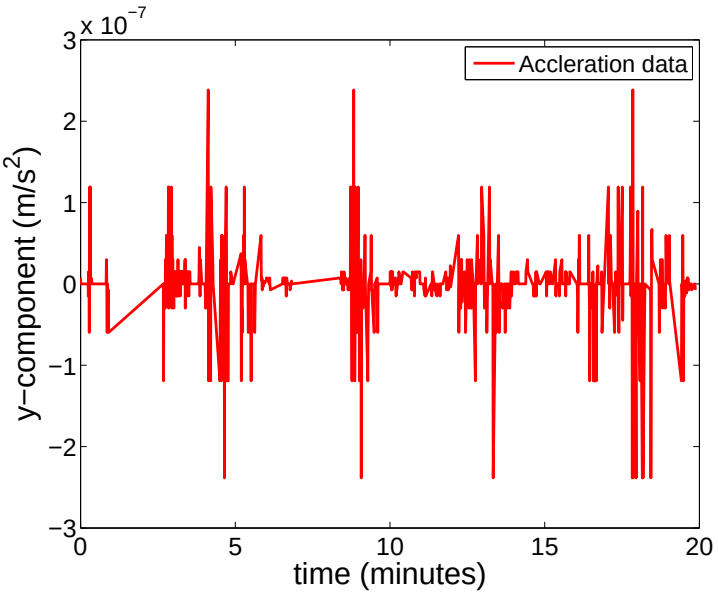


Figure 4.11: Sensing data: acceleration.

the manual movement. Similarly, LOBOT successfully detects the standing of the vehicle; however, the double integration of the erroneous accelerometer data falsely informs the large movement over time.

Although a single GPS reading can have error of up to three meters in our experiments, unlike the relative position based on accumulation, the GPS positioning does not accumulate errors: a previous inaccuracy GPS reading would not affect the current GPS reading. Finally, when the GPS-augmentation is applied to the drifting outcome of the local relative positioning component, the resulting location solution is satisfactory.

4.3.3 Evaluation of Local Relative Positioning

LOBOT strongly relies on the low cumulative errors of its local relative positioning component. A major portion of the experiments were performed to evaluate the local relative positioning. Both the manual measurement and the camera-assisted positioning were used to gain the ground truth. Though most results are from experiments on relatively flat planes (2D experiments), we also carried out 3D experiments of localizing the robot on surfaces with a

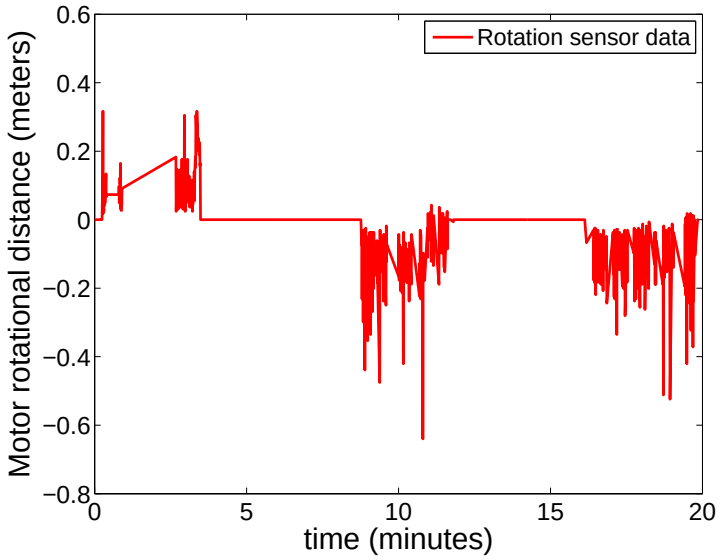


Figure 4.12: Sensing data: motor rotation.

slope. LOBOT does not favor one dimension over another. As a matter of fact, any two dimensions from a 3D experiment can be viewed as a 2D experiment. For that reason, the major analysis is on the 2D experiments while the 3D experiments exhibit similar characteristics.

Two-Dimensional Experiments

We present the 2D trace of the robot as well as the time series of the movement on each single dimension. The results show the relatively low cumulative errors of LOBOT and the large deviation of the purely accelerometer-based approach.

According to our 10 experiments with each running 20 minutes in 12m x 12m areas, the trace resulting from LOBOT has an accuracy of within 2.5 meters compared to manual recordings. One such experiment is shown in Figure 4.16. In Figure 4.16, the (x, y) coordinates by LOBOT are relatively close to the manual recordings. In contrast, the accelerometer-based approach tends to suggest almost "no-movement" on the plane and dramatic movement on the third dimension (the altitude). As in Figure 4.16, the results from the accelerometer-based approach falsely "suggest" that the robot moves within a small circle with 1m radius.

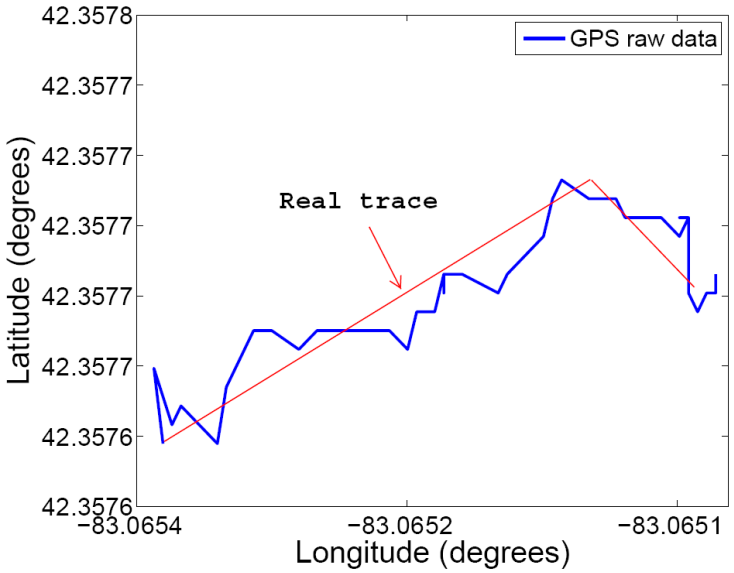


Figure 4.13: Sensing data: GPS.

Since the movement is on flat plane surfaces, LOBOT naturally verifies the limited movement on the third dimension. The altitude from LOBOT is within a range from -0.5m to 0.5m through 20 minutes. One such example is presented in Figure 4.17. In contrast, the accelerometer-based approach often falsely reports a dramatic movement on the third dimension. Again as in Figure 4.17, according to that approach, the robot is driving down a steep slope though it never leaves the flat plane ground. As for such results, it is reasonable to suspect that the acceleration data on the third dimension might have a constant large negative deviation from its true zero value and that the deviation could have resulted from an inaccurate gravitational constant or simply the sensing errors. However, the acceleration data on the third dimension seems to suggest only very small constant deviation of the acceleration data might exist. The corresponding data for the same previous experiment is extracted and shown in Figure 4.18. The figure indicates that the acceleration data oscillates around zero. To explain the dramatic error on the z-value of the accelerometer-based approach, we note that this approach involves a quadratic expression of the time and thus the time elapse accumulates such errors very fast.

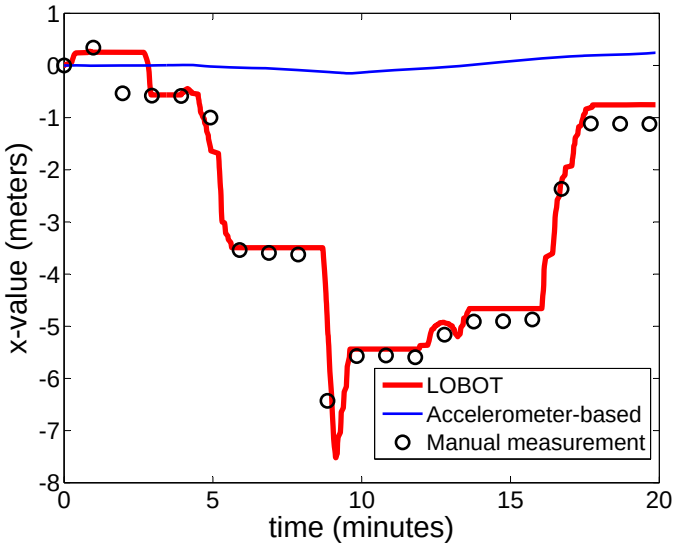


Figure 4.14: Time series of x-value

In addition to the trace, the time series of the components of the movement vector on each dimension also confirms the satisfactory performance of LOBOT's local relative positioning. With the same experiment in Figure 4.16, the time series of the x is almost perfectly close to the ground truth. The time series of y values is plotted in Figure 4.19. The y values of LOBOT exhibit a deviation of up to 1.75m over 20 minutes. On the other hand, as for the accelerometer-based approach, the figure displays almost static y values.

Finally, getting the ground truth through the camera-assisted positioning allows better examination of LOBOT. As found in our experiments, the error of LOBOT generally accumulates slowly; however, occasionally a relative noticeable transient error occurs due to accidents such as slippage. Despite the cumulative errors, the trace LOBOT retrieves generally follows the overall movement trend. As in one experiment (Figure 4.20), the robot moved for one minute, over which the local relative positioning performs almost perfectly except when a slippage occurred around the position (-0.07, 0.33). After the slippage, the trace curve still has a very similar shape as the camera-retrieved ground truth, however, with

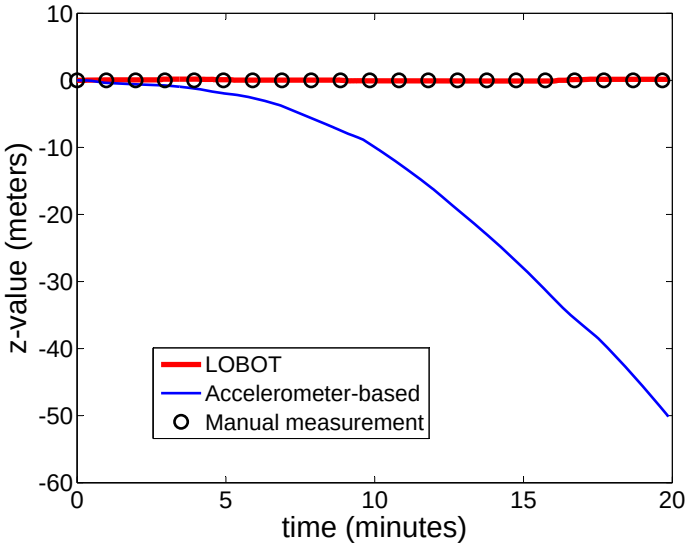


Figure 4.15: Comparisons: z-value

a shifting effect. When such a noticeable error happens, after the GPS-augmentation, the results can often be adjusted to be relatively close to the ground truth.

Three-Dimensional Experiments

Compared to the 2D experiments, our 3D experiments show similar performance of LOBOT's local relative positioning. Admittedly, the deviation from the third dimension adds to the overall positioning error. However, as for LOBOT, the addition of errors is often comparable to the errors from the two other dimensions. In its design, LOBOT does not treat the third dimension different than the other two. In one experiment, the robot climbs up a east-bound slope of 16.7 degrees. Figure 4.21 shows the (x,z) value pair of LOBOT against the manual measurement. The x value is the distance projected onto the east and the z value is the height from the base. The slope computed is almost the same as the one from manual measurement.

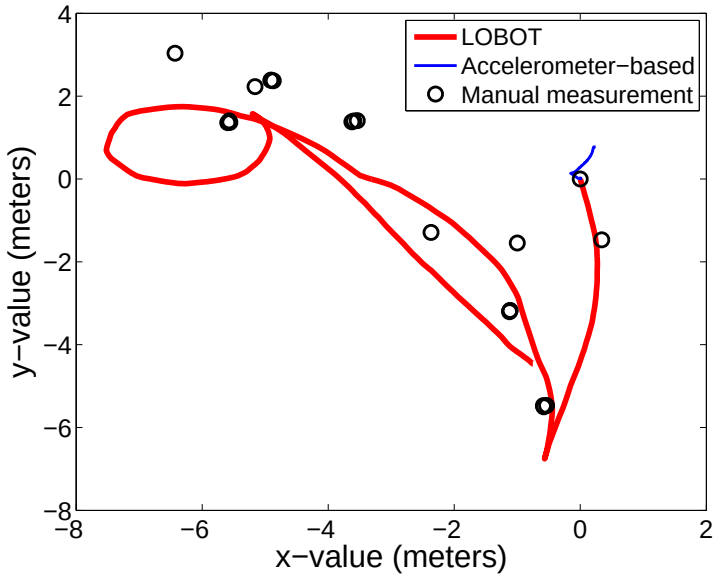


Figure 4.16: Trace comparison of a 2D experiment.

4.3.4 Evaluation of LOBOT with GPS-Augmentation

We performed a few outdoor experiments in GPS-available areas of up to 50m x 50m. To obtain the ground truth, the GPS on the HTC Legend phone is turned on and computes positions at least once every three seconds. Since the GPS's (longitude, latitude) data can be locally viewed as Cartesian coordinates, we mapped the GPS data onto a meter-based distance coordinate through linear regression. The trace produced by LOBOT is compared against the continuous GPS timestamped trace. The empirical analysis shows that the LOBOT's local relative positioning produces an inaccuracy of up to 18m; with one-time GPS-augmentation, the error is well under 8m. Without the GPS-augmentation, the trace retrieved still has a similar shape to the ground truth but with a drift. The result of one experiment is illustrated in Figure 4.22. In Figure 4.22, the thicker red line is the trace produced by the LOBOT without the GPS-augmentation, the small circles are the GPS trace, and the thinner green line is the trace by LOBOT with the correction from the last GPS-detected position. The one-time adjustment from the GPS data largely corrects the drift. With the same experiment, we performed a two-time adjustment: first correction based on the GPS data collected in the middle

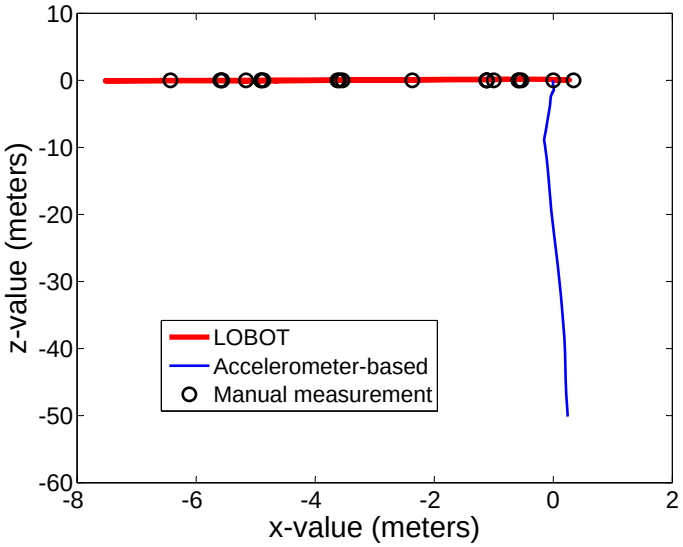


Figure 4.17: (x,z) trace comparison.

of the experiment time; the other correction based on the last GPS data. Interestingly, the two-time adjustment does not seem to suggest much improvement over the one-time adjustment, as shown in Figure 4.23. The main reason is, the GPS measurement itself is known to have inherent inaccuracy.

4.3.5 Impact of Time Interval Selections

So far, we have assumed that the accelerometer, the magnetic sensor, and the rotation sensors collect data periodically with a default time interval of 0.085s. Our empirical experiments indicate that any time interval under 1s would have produced a very close trace with only slight distortion. When the time interval increases to 2s or greater, the distortion becomes noticeable in certain scenarios that the robot changes its movement pattern at a fast rate. Figure 4.24 shows such distortion of LOBOT's local relative positioning in one of our experiments. In Figure 4.24, with an interval of 1.085s, the trace has a slight drift of around 0.2m. When the time interval increases to 2.085s and 3.085s, the distortion becomes apparent.

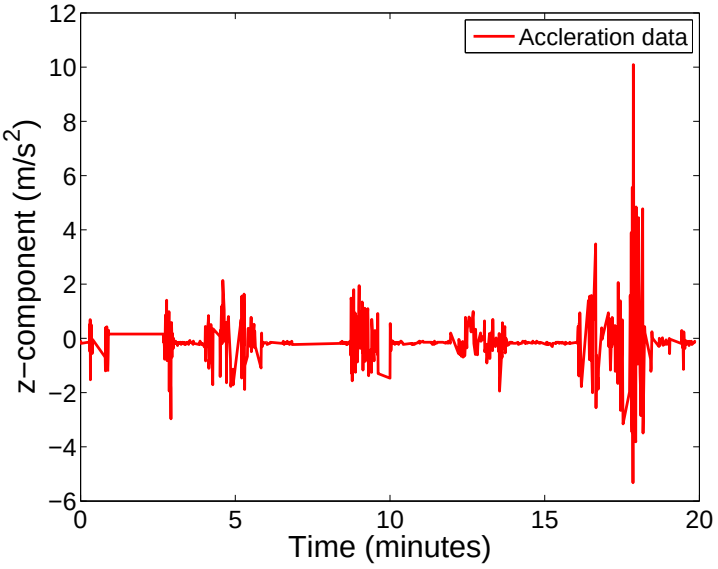


Figure 4.18: Sensing data: acceleration

4.4 Summary

We proposed LOBOT, a low-cost, self-contained, accurate localization system for small-sized ground robotic vehicles. LOBOT localizes a robotic vehicle with a hybrid approach consisting of infrequent absolute positioning through a GPS receiver and local relative positioning based on a 3D accelerometer, a magnetic field sensor and several motor rotation sensors. LOBOT fuses the information from an accelerometer, a magnetic sensor and motor rotation sensors to infer the movement of the robot through a short time period; then the inferred movement is corrected with infrequent GPS-augmentation. The hardware devices LOBOT uses are easily-available at low cost. LOBOT is self-contained in that it virtually requires no external devices or external facility management and that it needs no prior information. Unlike other localization schemes such as radio-based solutions, LOBOT does not require external reference facilities, expensive hardware, careful tuning or strict calibration. Additionally, LOBOT applies to both indoor and outdoor environments and realizes satisfactory performance. We developed a prototype of LOBOT and conducted extensive field experiments. The empirical experiments of various temporal and spatial scales with LOBOT verified its

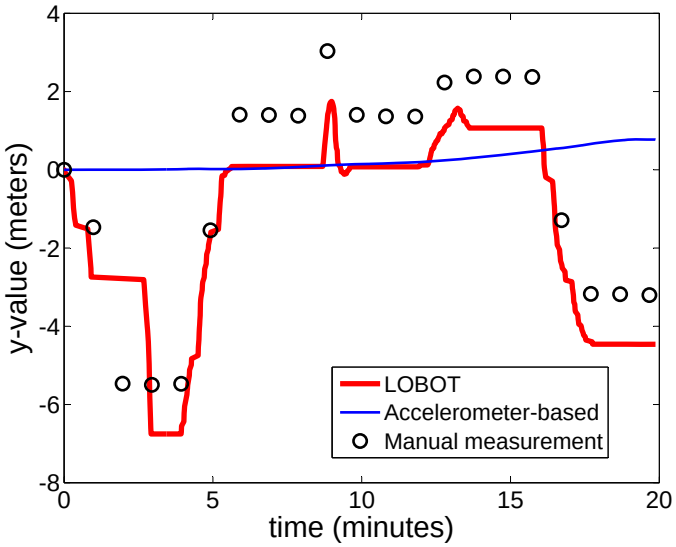


Figure 4.19: Time series of y-value

accuracy. In contrast to the accelerometer-based approach, LOBOT succeeds in maintaining low cumulative error. The GPS-augmentation greatly enhances LOBOT's resilience.

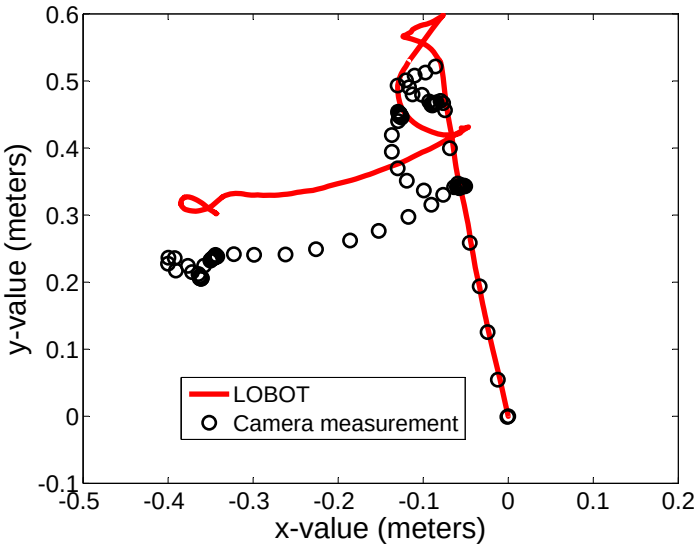


Figure 4.20: LOBOT trace with cumulative errors.

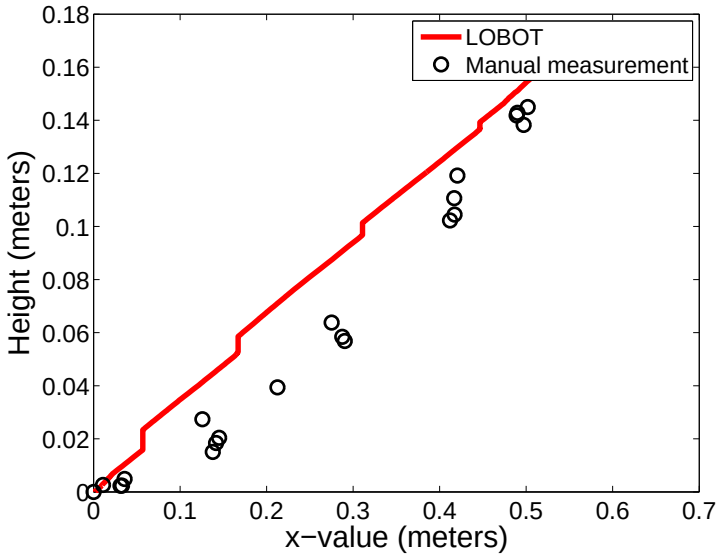


Figure 4.21: Three-dimensional experiment.

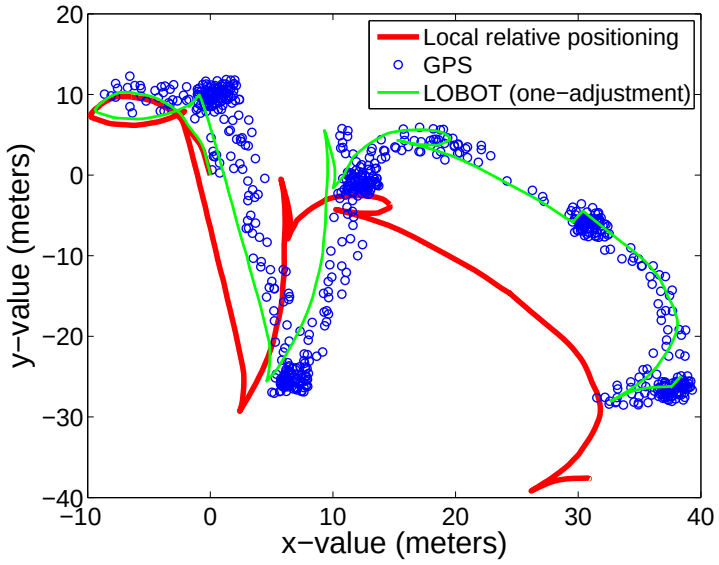


Figure 4.22: Outdoor experiments with one-time GPS-augmentation.

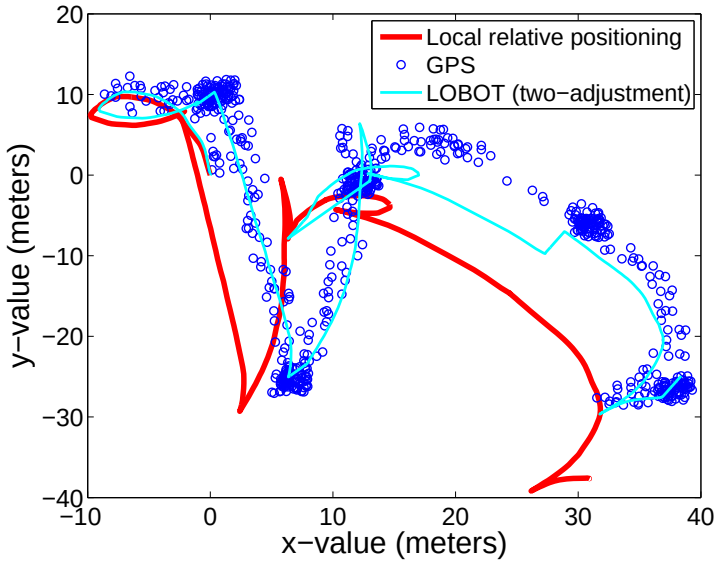


Figure 4.23: Outdoor experiments with two-time GPS-augmentation.

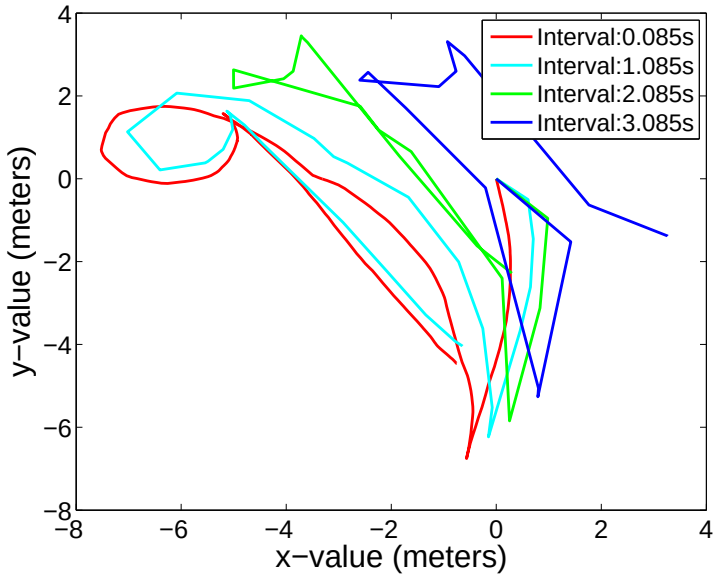


Figure 4.24: Time interval choices.

CHAPTER 5

TRUST-AWARE ROUTING FRAMEWORK TO SECURE MULTIHOP ROUTING

We designed and implemented TARF, a robust trust-aware routing framework, to secure multi-hop routing through a set of sensors (WSNs) in wireless sensing systems. Though it is motivated by harmful attackers exploiting the replay of routing information, TARF can also be used to protect the routing layer from other attacks. TARF requires neither tight time synchronization nor known geographic information. Its resilience and scalability were proved through both extensive simulation and empirical evaluation with large-scale WSNs. We implemented a ready-to-use TinyOS module of TARF with low overhead; this TARF module can be integrated into existing routing protocols with moderate efforts.

5.1 Introduction

As an important type of wireless sensing systems, wireless sensor networks (WSNs) [120, 170] are ideal candidates for applications to report detected events of interest, such as military surveillance and forest fire monitoring. A WSN comprises battery-powered senor nodes with extremely limited processing capabilities. With a narrow radio communication range, a sensor node wirelessly sends messages to a base station (network gateway) via a multi-hop path. However, the multi-hop routing of WSNs often becomes the target of malicious attacks. An attacker may tamper nodes physically, create traffic collision with seemingly valid transmission, drop or misdirect messages in routes, or jam the communication channel by creating radio interference [152]. This chapter focuses on the kind of attacks in which adversaries misdirect network traffic by identity deception through replaying routing information. Based

on identity deception, the adversary is capable of launching harmful and hard-to-detect attacks against routing, such as *selective forwarding*, *wormhole* attacks, *sinkhole* attacks and *Sybil* attacks [74].

As a harmful and easy-to-implement type of attack, a malicious node simply replays all the outgoing routing packets from a valid node to forge the latter node's identity; the malicious node then uses this forged identity to participate in the network routing, thus disrupting the network traffic. Those routing packets, including their original headers, are replayed without any modification. Even if this malicious node cannot directly overhear the valid node's wireless transmission, it can collude with other malicious nodes to receive those routing packets and replay them somewhere far away from the original valid node, which is known as a *wormhole* attack [67]. Since a node in a WSN usually relies solely on the packets received to know about the sender's identity, replaying routing packets allows the malicious node to forge the identity of this valid node. After "stealing" that valid identity, this malicious node is able to misdirect the network traffic. For instance, it may drop packets received, forward packets to another node not supposed to be in the routing path, or even form a transmission loop through which packets are passed among a few malicious nodes infinitely. It is often difficult to know whether a node forwards received packets correctly even with overhearing techniques [74]. *Sinkhole* attacks are another kind of attacks that can be launched after stealing a valid identity. In a *sinkhole* attack, a malicious node may claim itself to be a base station through replaying all the packets from a real base station [78]. Such a fake base station could lure more than half the traffic, creating a "black hole". This same technique can be employed to conduct another strong form of attack - *Sybil* attack [109]: through replaying the routing information of multiple legitimate nodes, an attacker may present multiple identities to the network. A valid node, if compromised, can also launch all these attacks.

The harm of such malicious attacks based on the technique of replaying routing information is further aggravated by the introduction of mobility into WSNs and the hostile network

condition. Though mobility is introduced into WSNs for efficient data collection and various applications [5, 43, 57, 88, 156, 162, 167], it greatly increases the chance of interaction between the honest nodes and the attackers. Additionally, a poor network connection causes much difficulty in distinguishing between an attacker and a honest node with transient failure. Without proper protection, WSNs with existing routing protocols can be completely devastated under certain circumstances. In an emergent sensing application through WSNs, saving the network from being devastated becomes crucial to the success of the application.

Unfortunately, most existing routing protocols for WSNs either assume the honesty of nodes and focus on energy efficiency [1], or attempt to exclude unauthorized participation by encrypting data and authenticating packets. Examples of these encryption and authentication schemes for WSNs include TinySec [73], Spins [116], TinyPK [146], and TinyECC [91]. Admittedly, it is important to consider efficient energy use for battery-powered sensor nodes and the robustness of routing under topological changes as well as common faults in a wild environment. However, it is also critical to incorporate security as one of the most important goals; meanwhile, even with perfect encryption and authentication, by replaying routing information, a malicious node can still participate in the network using another valid node's identity.

In addition to the cryptographic methods, trust and reputation management has been employed in generic ad hoc networks and WSNs to secure routing protocols. Basically, a system of trust and reputation management assigns each node a trust value according to its past performance in routing. Then such trust values are used to help decide a secure and efficient route. However, the proposed trust and reputation management systems for generic ad hoc networks target only relatively powerful hardware platforms such as laptops and smartphones [13, 50, 100, 104, 118, 128, 153, 157]. Those systems cannot be applied to WSNs due to the excessive overhead for resource-constrained sensor nodes powered by batteries. As far as WSNs are concerned, secure routing solutions based on trust and reputation management rarely address the identity deception through replaying routing information [126, 163]. The

countermeasures proposed so far strongly depends on either tight time synchronization or known geographic information while their effectiveness against attacks exploiting the replay of routing information has not been examined yet [74].

At this point, to protect WSNs from the harmful attacks exploiting the replay of routing information, we have designed and implemented a robust trust-aware routing framework, TARF, to secure routing solutions in wireless sensor networks. Based on the unique characteristics of resource-constrained WSNs, the design of TARF centers on *trustworthiness* and *energy efficiency*. Though TARF can be developed into a complete and independent routing protocol, the purpose is to allow existing routing protocols to incorporate our implementation of TARF with moderat effort and thus producing a secure and efficient fully-functional protocol. Unlike other security measures, TARF requires neither tight time synchronization nor known geographic information. Most importantly, TARF proves resilient under various attacks exploiting the replay of routing information, which is not achieved by previous security protocols. Even under strong attacks such as *sinkhole* attacks, *wormhole* attacks as well as *Sybil* attacks, and hostile mobile network condition, TARF demonstrates steady improvement in network performance. The effectiveness of TARF is verified through extensive evaluation with simulation and empirical experiments on large-scale WSNs. Finally, we have implemented a ready-to-use TARF module with low overhead, which as demonstrated can be integrated into existing routing protocols with ease; the demonstration of a proof-of-concept mobile target detection program indicates the potential of TARF in WSN applications.

For the rest of this chapter, we start by stating the assumptions and goals of this chapter and the notations used in Section 5.2. Then we elaborate the design of TARF in Section 5.3, including the routing procedure as well as the *EnergyWatcher* and *TrustManager* components. In Section 5.4, we present the simulation results of TARF against various attacks through replaying routing information in static, mobile and RF-shielding conditions. Section 5.5 further presents the implementation of TARF, empirical evaluation at a large sensor

network and a resilient proof-of-concept mobile target detection application based on TARF. Finally, Section 5.6 summarizes this chapter.

5.2 Assumptions and Goals

We target secure routing for data collection tasks, which are one of the most fundamental functions of WSNs. In a data collection task, a sensor node sends sampled data to a remote base station with the aid of intermediate nodes, as shown in Figure 5.1(a). Though there could be more than one base station, our routing approach is not affected by the number of base stations; to simplify our discussion, we will assume that there is only one base station. It is possible for an adversary to replay all the packets from a base station, possibly through a *wormhole*, and thus to forge the identity of the base station. If necessary, the adversary can spoof the acknowledgement packet of the base station, too. Such identity deception can result in the following situation: a large amount of packets are attracted to this fake base station and are never delivered to the real base station (see Figure 5.1(b)). Essentially, an adversary can forge the identity of any legal node through replaying that node's outgoing routing packets. With a forged identity, an attacker may launch a series of other attacks, including packet dropping, network looping and *Sybil* attacks. Additionally, this chapter does not address denial-of-service (DoS) [152] attacks, where an attacker intends to damage the network by exhausting its resource. For instance, we do not address DoS attacks such as congesting the network by replaying numerous packets or physically jamming the network.

Further, we assume no data aggregation is involved. Nonetheless, our approach can still be applied to cluster-based WSNs, where data are aggregated by clusters before being relayed. In a cluster-based WSN, cluster headers themselves form a sub-network; after certain data reach a cluster header, the aggregated data will be routed to a base station only through such a sub-network consisting of cluster headers. Our framework can then be applied to this sub-network to achieve secure routing for cluster-based WSNs.

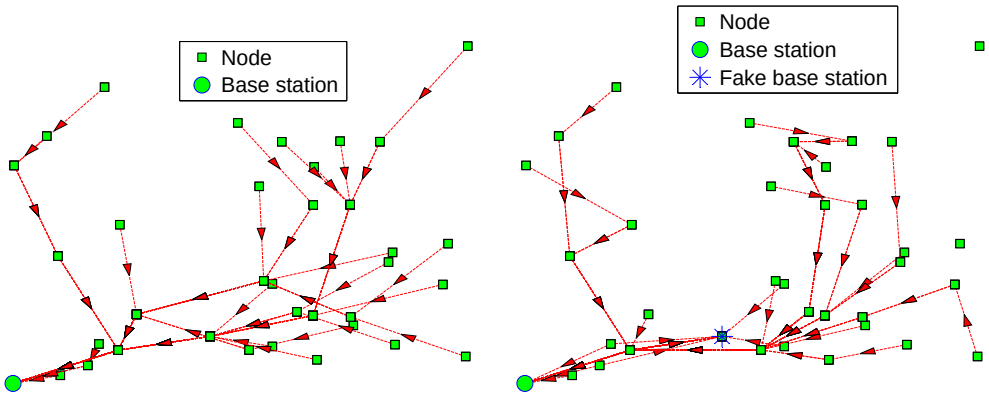


Figure 5.1: Multi-hop routing: (a) normal scenarios; (b) a fake base station attracts traffic.

Additionally, we make certain assumptions regarding the format of packets in TARF. We assume all data packets and routing packets, including their packet headers, are authenticated; a packet can be forwarded only after its authenticity is verified. Whether data encryption is implemented can be decided by the application. We note that a regular sensor node (not a base station) may not afford a strong authentication mechanism that costs too much computation overhead, and that an adversary may physically compromise that node and hack the authentication scheme. Thus, we only require moderate authentication on a sensor node to add difficulty to the attackers. However, we do require a strong authentication on a base station node with a high processing capability; such a requirement is adopted to guarantee that an adversary is not able to manipulate or forge a broadcast message from the base station at will. That requirement is crucial to TARF; it is also key to any successful secure routing protocol. This strong authentication requirement can be achieved by existing broadcast authentication schemes [22, 116, 124].

Every data packet is assumed to have at least the following fields: the sender id, the sender sequence number, the next-hop node id (the receiver in this one-hop transmission), the source id (the node that initiates the data), and the source's sequence number. We insist that the source node's information should be included for the following reasons. First, that

allows the base station to identify which data packets are initiated but undelivered; Second, a WSN cannot afford the overhead to transmit all the one-hop information to the base station. Regarding routing packets, they should have at least the following fields: the source id, the source's sequence number, and the next-hop id. In addition, we assume that after receiving a data packet, a node will send out an acknowledgement packet which may not be authenticated. While strong acknowledgement authentication for each hop may enhance security, it leads to major computation overhead and network delay considering the multi-hop routing pattern. Further, any sensor node but a base station could be physically captured, compromised and hacked to reveal its detailed authentication mechanism. Thus security through acknowledgement authentication cannot be guaranteed. Acknowledgement spoofing may be exploited by an attacker, admittedly, but TARF is to direct a node a to circumvent an attacker spoofing acknowledgement based on the trust management.

Next, we present the goals of TARF.

High Throughput *Throughput* is defined as the ratio of the number of all data packets delivered to the base station to the number of all sampled data packets. In our simulation, *throughput* at a moment is computed over the period from the beginning time (0) until that particular moment. Note that single-hop re-transmission may happen, and that duplicate packets are considered as one packet as far as *throughput* is concerned. *Throughput* reflects how efficiently the network is collecting and delivering data. Here we regard high *throughput* as one of our most important goals.

Energy Efficiency Efficient energy use is significant for battery-powered sensor nodes, and data transmission accounts for a major portion of energy consumption. We evaluate energy efficiency by the average energy cost to successfully deliver a unit-sized data packet from a source node to the base station. Note that link-level re-transmission should be given enough attention when considering energy cost since each re-transmission causes a noticeable increase in energy consumption. If every node in a WSN consumes approximately the same energy to transmit a unit-sized data packet, we can use another metric *hop-per-delivery* to

evaluate energy efficiency. Under that assumption, the energy consumption depends on the number of hops, i.e. the number of one-hop transmissions occurring. To evaluate how efficiently energy is used, we can measure the average hops per delivery, i.e., the number of all hops divided by the number of all delivered data packets, abbreviated as *hop-per-delivery*.

Scalability & Adaptability TARF should work well with WSNs of large magnitude under highly dynamic contexts. We will examine its scalability through empirical experiments on Motelab [105], a large-scale WSN testbed (see Section 5.5.3); the adaptability of TARF will be evaluated through simulation under mobile and hash network conditions (see Section 5.4).

Here we do not include other aspects such as latency, load balance, or fairness. Low latency, balanced network load, and good fairness requirements can be enforced in specific routing protocols incorporating TARF.

5.3 Design of TARF

TARF secures the multi-hop routing in WSNs against intruders exploiting the replay of routing information by evaluating the trustworthiness of neighboring nodes. It identifies such intruders that misdirect noticeable network traffic by their low trustworthiness and routes data through paths circumventing those intruders to achieve satisfactory *throughput*. TARF is also energy-efficient, highly scalable, and well adaptable. Before introducing the detailed design, we first introduce several necessary notions here.

Neighbor For a node N, a neighbor (neighboring node) of N is a node that is reachable from N with one-hop wireless transmission.

Trust level For a node N, the trust level of a neighbor is a decimal number in $[0, 1]$, representing N's opinion of that neighbor's level of trustworthiness. Specifically, the trust level of the neighbor is N's estimation of the probability that this neighbor correctly delivers data received to the base station. That trust level is denoted as T in this chapter.

Energy cost For a node N, the energy cost of a neighbor is the average energy cost to successfully deliver a unit-sized data packet with this neighbor as its next-hop node, from N to the base station. That energy cost is denoted as E in this chapter.

5.3.1 Overview

TARF integrates trustworthiness and energy efficiency in making routing decisions. For a node N to route a data packet to the base station, N only needs to decide to which neighboring node it should forward the data packet. That chosen neighbor is N's next-hop node. Once the data packet is forwarded to that next-hop node, the remaining task to deliver the data to the base station is fully delegated to it, and N is totally unaware of what routing decision its next-hop node makes. To choose its next-hop node, N considers both the trustworthiness and the energy efficiency of its neighbors. For that, N maintains a neighborhood table with trust level values and energy cost values for certain known neighbors. It is sometimes necessary to delete some neighbors' entries to keep the table size acceptable. The technique of maintaining a neighborhood table of a moderate size is demonstrated by Woo, Tong and Culler [151]; TARF may employ the same technique.

In TARF, in addition to data packet transmission, there are two types of routing information that need to be exchanged: broadcast messages from the base station about undelivered data packets and energy cost report messages from each node. Neither message needs acknowledgement. A broadcast message from the base station is broadcast to the whole network; each node receiving a fresh broadcast message from the base station will broadcast it to all its neighbors once. The freshness of a broadcast message is checked through its field of source sequence number. The other type of exchanged routing information is the energy cost report message from each node, which is broadcast to only its neighbors once. Additionally, any node receiving such an energy cost report message will not forward it.

For each node N in a WSN, to maintain such a neighborhood table with trust level values and energy cost values for certain known neighbors, two components, *EnergyWatcher*

and *TrustManager*, run on the node (Figure 5.2). *EnergyWatcher* is responsible for recording the energy cost for each known neighbor, based on N's observation of one-hop transmission to reach its neighbors and the energy cost report from those neighbors. A compromised node may falsely report an extremely low energy cost to lure its neighbors into selecting this compromised node as their next-hop node; however, these TARF-enabled neighbors eventually abandon that compromised next-hop node based on its low trustworthiness as tracked by *TrustManager*. *TrustManager* is responsible for tracking trust level values of neighbors based on network loop discovery and broadcast messages from the base station about undelivered data packets. Once N is able to decide its next-hop neighbor according to its neighborhood table, it sends out its energy report message: it broadcasts to all its neighbors its energy cost to deliver a packet from the node to the base station. The energy cost is computed as in Section 5.3.3 by *EnergyWatcher*. Such an energy cost report also serves as the input of its receivers' *EnergyWatcher*.

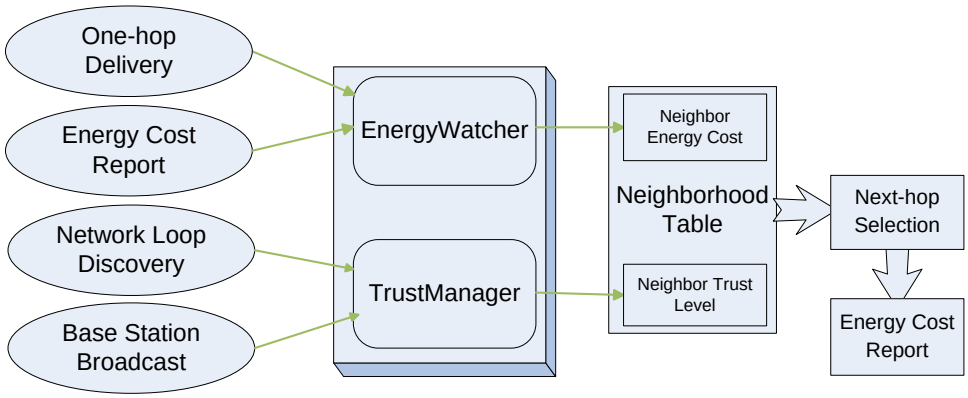


Figure 5.2: Each node selects a next-hop node based on its neighborhood table, and broadcast its energy cost within its neighborhood. To maintain this neighborhood table, *EnergyWatcher* and *TrustManager* on the node keep track of related events (on the left) to record the energy cost and the trust level values of its neighbors.

5.3.2 Routing Procedure

TARF, as with many other routing protocols, runs as a periodic service. The length of that period determines how frequently routing information is exchanged and updated. At the beginning of each period, the base station broadcasts the information about undelivered data packets during the past few periods to the whole network once, which triggers the exchange of routing information in this new period. Whenever a node receives such a broadcast message from the base station, it knows that the most recent period has ended and a new period has just started. In this way, no tight time synchronization is required for a node to keep track of the beginning or ending of a period. During each period, the *EnergyWatcher* on a node monitors energy consumption of one-hop transmission to its neighbors and processes energy cost reports from those neighbors to maintain energy cost entries in its neighborhood table; its *TrustManager* also keeps track of network loops and processes broadcast messages from the base station about undelivered data to maintain trust level entries in its neighborhood table.

To maintain the stability of its routing path, a node may retain the same next-hop node until the next fresh broadcast message from the base station occurs. Meanwhile, to reduce traffic, its energy cost report could be configured to not occur again until the next fresh broadcast from the base station. If a node does not change its next-hop node selection until the next broadcast from the base station, that guarantees all paths to be loop-free, as can be deducted from the procedure of next-hop node selection. However, as noted in our experiments, that would lead to slow improvement in routing paths. Therefore, we allow a node to change its next-hop selection in a period when its current next-hop node performs the task of receiving and delivering data poorly.

Next, we introduce the structure and exchange of routing information as well as how nodes make routing decisions in TARF.

Structure and Exchange of Routing Information

A broadcast message from the base station fits into a fixed number of packets; in our implementation, it fits into one packet. Such a message consists of a few pairs of <the node id of a source node, an undelivered sequence interval [a, b] with a significant length>. To reduce overhead, only a few such pairs are selected to be broadcast. The undelivered sequence interval [a, b] is explained as follows: the base station searches the source sequence numbers received in the past few periods, identifies which source sequence numbers for the source node with this id are missing, and chooses certain significant interval [a, b] of missing source sequence numbers as an undelivered sequence interval. For example, the base station may have all the source sequence numbers for the source node 2 as {109, 110, 111, 150, 151} in the past two periods. Then [112, 149] is an undelivered sequence interval. Since the base station is usually connected to a powerful platform such as a desktop, a program can be developed on that powerful platform to assist in recording all the source sequence numbers and finding undelivered sequence intervals. The reason for searching over more than one period is to identify as many undelivered data packets as possible. To illustrate that, consider this example: suppose the source sequence numbers of delivered data packets from node 2 are {1, 2, 3} for the 1st period and {200, 201, 203} for the 2nd period; then simply searching over a single period would not discover the undelivered packets unless every node is required to send a fixed number of data packets over each period.

Accordingly, each node in the network stores a table of <the node id of a source node, a forwarded sequence interval [a, b] with a significant length> in the past few periods. The data packets with the source node and the sequence numbers falling in this forwarded sequence interval [a, b] have already been forwarded by this node. When the node receives a broadcast message with undelivered sequence intervals, its *TrustManager* will be able to identify which data packets forwarded by this node are not delivered to the base station. Considering the overhead to store such a table, old entries will be deleted once the table is full.

Once a fresh broadcast message from the base station is received, a node immediately invalidates all the existing energy cost entries: it is ready to receive a new energy report from its neighbors and choose its new next-hop node afterwards. Also, it is going to select a node either after a timeout is reached or after it has received an energy cost report from some highly trusted candidates with acceptable energy cost. A node immediately broadcasts its energy cost to its neighbors only after it has selected a new next-hop node. That energy cost is computed by its *EnergyWatcher* (see Section 5.3.3). A natural question is which node starts reporting its energy cost first. For that, note that when the base station is sending a broadcast message, a side effect is that its neighbors receiving that message will also regard this as an energy report: the base station needs 0 amount of energy to reach itself. As long as the original base station is faithful, it will be viewed as a trustworthy candidate by *TrustManager* on the neighbors of the base station. Therefore, those neighbors will be the first nodes to decide their next-hop node, which is the base station; they will start reporting their energy cost once that decision is made.

Route Selection

Now, we introduce how TARF decides routes in a WSN. Each node N relies on its neighborhood table to select an optimal route, considering both energy consumption and reliability. TARF makes good efforts in excluding those nodes that misdirect traffic by exploiting the replay of routing information.

For a node N to select a route for delivering data to the base station, N will select an optimal next-hop node from its neighbors based on trust level and energy cost and forward the data to the chosen next-hop node immediately. The neighbors with trust levels below a certain threshold will be excluded from being considered as candidates. Among the remaining known neighbors, N will select its next-hop node through evaluating each neighbor b based on a trade-off between T_{Nb} and $\frac{E_{Nb}}{T_{Nb}}$, with E_{Nb} and T_{Nb} being b's energy cost and trust level value in the neighborhood table respectively (see Section 5.3.3, 5.3.4). Basically, E_{Nb} reflects the

energy cost of delivering a packet to the base station from N assuming that all the nodes in the route are honest; $\frac{1}{T_{Nb}}$ approximately reflects the number of the needed attempts to send a packet from N to the base station via multiple hops before such an attempt succeeds, considering the trust level of b. Thus, $\frac{E_{Nb}}{T_{Nb}}$ combines the trustworthiness and energy cost. However, the metric $\frac{E_{Nb}}{T_{Nb}}$ suffers from the fact that an adversary may falsely reports extremely low energy cost to attract traffic and thus resulting in a low value of $\frac{E_{Nb}}{T_{Nb}}$ even with a low T_{Nb}. Therefore, TARF prefers nodes with significantly higher trust values; this preference of trustworthiness effectively protects the network from an adversary who forges the identity of an attractive node such as a base station. For deciding the next-hop node, a specific trade-off between T_{Nb} and $\frac{E_{Nb}}{T_{Nb}}$ is demonstrated in Figure 5.16 (see Section 5.5.2).

The remaining delivery task is fully delegated to that selected next-hop neighbor, and N is totally unaware of what routing decision its chosen neighbor is going to make. Next, the chosen node will repeat what N has done, i.e., delegating the left routing task to its own chosen next-hop neighbor. In this way, instead of finding out a complete path to the base station, each node is only responsible for choosing its next-hop node, thus saving considerable cost in computation and routing information exchange. As an example shown in Figure 5.3, node a is trying to forward a packet to the base station. After comparing both the trust level and energy cost among its neighbors 1, 2 and b, a decides that b is the most promising next-hop node for data delivery and forwards the data packet to b immediately. b is free to make its own decision for routing the packet to the base station. b decides that its neighbor c is a better candidate than its neighbor 3. After that, the task is delegated to c, and c continues to delegate the job to d. Finally, d delivers the packet to the base station. Observe that in an ideal misbehavior-free environment, all nodes are absolutely faithful, and each node will choose a neighbor through which the routing path is optimized in terms of energy; thus, an energy-driven route is achieved.

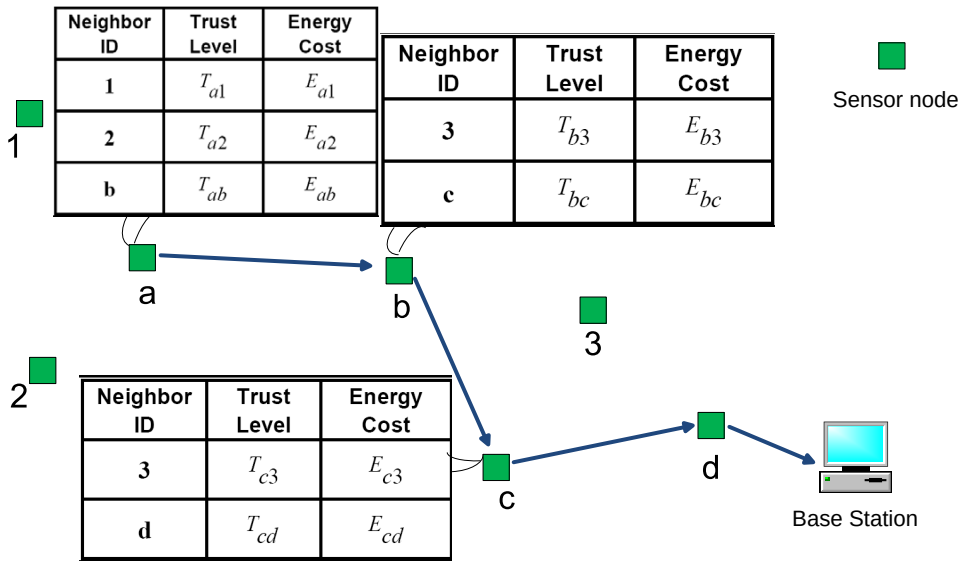


Figure 5.3: Routing illustration.

5.3.3 *EnergyWatcher*

Here we describe how a node N's *EnergyWatcher* computes the energy cost E_{Nb} for its neighbor b in N's neighborhood table and how N decides its own energy cost E_N. Before going further, we will clarify some notations. E_{Nb} mentioned is the average energy cost of successfully delivering a unit-sized data packet from N to the base station, with b as N's next-hop node being responsible for the remaining route. Here, one-hop re-transmission may occur until the acknowledgement is received or the number of re-transmissions reaches a certain threshold. The cost caused by one-hop re-transmissions should be included when computing E_{Nb}. Suppose N decides that A should be its next-hop node after comparing energy cost and trust level. Then N's energy cost is $E_N = E_{NA}$. Denote $E_{N\to b}$ as the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. Note that the re-transmission cost needs to be considered. With the above notations, it is straightforward to establish the following relation:

$$E_{Nb} = E_{N\to b} + E_b$$

Since each known neighbor b of N is supposed to broadcast its own energy cost E_b to N, to compute E_{Nb}, N still needs to know the value $E_{N\to b}$, i.e., the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. For that, assuming that the endings (being acknowledged or not) of one-hop transmissions from N to b are independent with the same probability p_{succ} of being acknowledged, we first compute the average number of one-hop sendings needed before the acknowledgement is received as follows:

$$\sum_{i=1}^{\infty} i \cdot p_{succ} \cdot (1 - p_{succ})^{i-1} = \frac{1}{p_{succ}}$$

Denote E_{unit} as the energy cost for node N to send a unit-sized data packet once regardless of whether it is received or not. Then we have

$$E_{Nb} = \frac{E_{unit}}{p_{succ}} + E_b$$

The remaining job for computing E_{Nb} is to get the probability p_{succ} that a one-hop transmission is acknowledged. Considering the variable wireless connection among wireless sensor nodes, we do not use the simplistic averaging method to compute p_{succ}. Instead, after each transmission from N to b, N's *EnergyWatcher* will update p_{succ} based on whether that transmission is acknowledged or not with a weighted averaging technique. We use a binary variable Ack to record the result of current transmission: 1 if an acknowledgement is received; otherwise, 0. Given Ack and the last probability value of an acknowledged transmission p_{old_succ}, an intuitive way is to use a simply weighted average of Ack and p_{old_succ} as the value of p_{new_succ}. That is what is essentially adopted in the aging mechanism [44]. However, that method used against sleeper attacks still suffers periodic attacks [164]. To solve this problem, we update the p_{succ} value using two different weights as in our previous work [164], a

relatively big $w_{degrade} \in (0, 1)$ and a relatively small $w_{upgrade} \in (0, 1)$ as follows:

$$p_{new\_succ} = \begin{cases} (1 - w_{degrade}) \times p_{old\_succ} + w_{degrade} \times Ack, \text{if } Ack = 0 \\ (1 - w_{upgrade}) \times p_{old\_succ} + w_{upgrade} \times Ack, \text{if } Ack = 1 \end{cases}$$

The two parameters $w_{degrade}$ and $w_{upgrade}$ allow flexible application requirements. $w_{degrade}$ and $w_{upgrade}$ represent the extent to which upgraded and degraded performance are rewarded and penalized, respectively. If any fault and compromise is very likely to be associated with a high risk, $w_{degrade}$ should be assigned a relatively high value to penalize fault and compromise relatively heavily; if a few positive transactions can't constitute evidence of good connectivity which requires many more positive transactions, then $w_{upgrade}$ should be assigned a relatively low value.

5.3.4 TrustManager

A node N's *TrustManager* decides the trust level of each neighbor based on the following events: discovery of network loops, and broadcast from the base station about undelivered data packets. For each neighbor b of N, T_{Nb} denotes the trust level of b in N's neighborhood table. At the beginning, each neighbor is given a neutral trust level 0.5. After any of those events occurs, the relevant neighbors' trust levels are updated.

Note that many existing routing protocols have their own mechanisms to detect routing loops and to react accordingly [48, 115, 151]. In that case, when integrating TARF into those protocols with anti-loop mechanisms, *TrustManager* may solely depend on the broadcast from the base station to decide the trust level; we adopted such a policy when implementing TARF later (see Section 5.5). If anti-loop mechanisms are both enforced in the TARF component and the routing protocol that integrates TARF, then the resulting hybrid protocol may overly react towards the discovery of loops. Though sophisticated loop-discovery methods exist in the currently developed protocols, they often rely on the comparison of specific routing cost to reject routes likely leading to loops [48]. To minimize the effort to integrate TARF

and the existing protocol and to reduce the overhead, when an existing routing protocol does not provide any anti-loop mechanism, we adopt the following mechanism to detect routing loops. To detect loops, the *TrustManager* on N reuses the table of <the node id of a source node, a forwarded sequence interval [a, b] with a significant length> (see Section 5.3.2) in the past few periods. If N finds that a received data packet is already in that record table, not only will the packet be discarded, but the TrustManager on N also degrades its next-hop node's trust level. If that next-hop node is b, then T_{old_Nb} is the latest trust level value of b. We use a binary variable $Loop$ to record the result of loop discovery: 0 if a loop is received; 1 otherwise. As in the update of energy cost, the new trust level of b is

$$T_{new\_Nb} = \begin{cases} (1 - w_{degrade}) \times T_{old\_Nb} + w_{degrade} \times Loop, \text{if } Loop = 0 \\ (1 - w_{upgrade}) \times T_{old\_Nb} + w_{upgrade} \times Loop, \text{if } Loop = 1 \end{cases}$$

Once a loop has been detected by N for a few times so that the trust level of the next-hop node is too low, N will change its next-hop selection; thus, that loop is broken. Though N cannot tell which node should be held responsible for the occurrence of a loop, degrading its next-hop node's trust level gradually leads to the breaking of the loop.

On the other hand, to detect the traffic misdirection by nodes exploiting the replay of routing information, *TrustManager* on N compares N's stored table of <node id of a source node, forwarded sequence interval [a, b] with a significant length> recorded in the past few periods with the broadcast messages from the base station about undelivered data. It computes the ratio of the number of successfully delivered packets which are forwarded by this node to the number of those forwarded data packets, denoted as $DeliveryRatio$. Then N's

TrustManager updates its next-hop node b's trust level as follows:

$$T_{new\_Nb} = \begin{cases} (1 - w_{degrade}) \times T_{old\_Nb} + w_{degrade} \times DeliveryRatio, \\ \text{if } DeliveryRatio < T_{old\_Nb}. \\ (1 - w_{upgrade}) \times T_{old\_Nb} + w_{upgrade} \times DeliveryRatio, \\ \text{if } DeliveryRatio >= T_{old\_Nb}. \end{cases}$$

Effectiveness of TrustManager against Various Attacks

TrustManager effectively identities the low trustworthiness of various attacks. Once the low trust levels of an adversary is recognized by *TrustManager*, the route selection procedure, according to its preference of trustworthy nodes, enables a valid node to avoid choosing an adversary as its next-hop node (see Section 5.3.2). The various attacks developed out of identity deception through replaying routing information, including *wormhole* attacks, *sink-hole* attacks, *Sybil* attacks and other misforwarding behaviors, all aim to cheat a valid node into choosing a neighboring attacker as its next-hop node. Though the valid node may be lured into the trap for a while since the attacker usually appears to be attractive, from the base broadcast messages, eventually the valid node realizes the data packets forwarded to its next-hop node is rarely delivered to the base station. Thus the next-hop node is marked as having a low trust level by *TrustManager*. A *Sybil* attack, due to its presence with multiple fake identities, could take longer for TrustManager to recognize than other attacks.

As an example, suppose an adversary M forges the identity of the base station by replaying all the routing packets from the base station. At first, it is able to deceive its neighbors into believing that M is a base station; as a result, M may attract a large amount of data packets, which never reach the base station. However, after the base station broadcasts the information about those undelivered packets, M's neighbors will downgrade M's trust level values in their neighborhood table. Note that M is only capable of replaying but is not capable of manipulating or generating authenticated broadcast messages, and that M usually cannot prevent other nodes from receiving a broadcast message from the base station. As

time elapses, M's neighbors will start realizing that M is not trustworthy and will look for other next-hop candidates that are more reliable. Similarly, if M forges the identity of another valid appealing node, M's neighbors will gradually realize that M is not reliable.

Additionally, once a valid node identifies a trustworthy honest neighbor as its next-hop node, it tends to keep that next-hop selection without considering other seemingly attractive nodes such as a fake base station. That tendency is caused by both the preference to maintain stable routes and the preference to highly trustable nodes.

5.4 Simulation and Evaluation

To further evaluate the efficacy of TARF in terms of energy efficiency and *throughput*, we have developed a reconfigurable emulator of wireless sensor networks on a two-dimensional plane with Matlab [103]. To effectively simulate a WSN, this emulator uses the object-oriented technique to construct two classes of objects: WSNMANAGER and NODE, to represent the whole network and a sensor node. The interaction between nodes are emulated through event passing. The routing function for a node can be rewritten to adopt different routing protocols; different maps can also be ported into this simulator. To simulate the unreliable wireless transmission, the outcome of one-hop packet transmission is decided by the following model: suppose a node A is wirelessly transmitting a packet to node B, the probability for B to successfully receive such a packet is assumed to be

$$1 - (min(dist, MAX\_DIST)/MAX\_DIST)^8,$$

where $dist$ is the distance from A to B, and MAX_DIST is the maximal transmission range. In our experiments, MAX_DIST is defined as 100m; initially, 35 nodes are randomly distributed within a 300*300 rectangular area as in Figure 5.4(a), and a base station is placed at the origin [0, 0]. All the nodes have the same power level and the same maximal transmission range of 100m. For easier reference, we define a virtual time unit as used in our simulation:

each node samples 6 times in every virtual time unit; the timing gap between every two consecutive samplings of the same node is equivalent. Essentially, the virtual time unit can reflect any length of actual time. We simulate the sensor network in 1440 consecutive virtual time units. Unless specified otherwise, the length of a period is 1 virtual time unit.

5.4.1 Three Types of Network Topology

Regarding the network topology, we set up three types of network topologies. The first type is the static-location case under which all nodes stand still and the dynamics of the network come from the unstable radio and the malicious behaviors. The specific placement of nodes are shown in Figure 5.4(a).

The second type is a group-motion-with-noise case based on Reference Point Group Mobility (RPGM) model [63, 171]. Basically, the RPGM Model mimics the behavior of a set of nodes moving in one or more groups: each group moves as a whole according to the trajectory of its logic center in order to perform certain group tasks. Additionally, a random motion vector is applied to each node. It models various scenarios such as battlefield situations and recovery scenarios [171]. In the experiments, we use a customized GPGM model as follows: all nodes fall into two groups (G1 and G2) with nearly equal sizes as indicated in Figure 5.4(a); each group will move around its virtual centroid, with the virtual trajectories of the two centroids as illustrated in Figure 5.4(b). Specifically, at any moment t (using the virtual time unit), the virtual position of G1's centroid is {20*[6*t-sin(6*t)], 20*[6*t+sin(6*t)]}, and that of G2's is {20*[6*t+sin(6*t)], 20*[6*t-sin(6*t)]}. The two trajectories coil around each other, creating abundant opportunities for nodes to interact with one another. We adopt such coiled trajectories to expose nodes to attackers as well as constantly changing network topology, so that we may realize how resilient TARF could be under a hostile environment. To well mimic the reality, we added a random Gaussian noise with a standard deviation of 20 meters to the trajectories of all nodes. The noise injected further increases the dynamics of

the network topology. Figure 5.4(c) displays the location of all nodes after 0.5 virtual time unit.

The last type of dynamic network incorporated in the experiments is the addition of scattered RF-shielded areas to the aforementioned group-motion-with-noise case. In an RF-shielded area, any outgoing and incoming radio signal is completely blocked though a node falling into such an area can still move out of it. In our experiments, specifically, thinking of the ground as a grid divided into cells of 100m by 100m, an RF-shielded square of 20m by 20m is placed in each such cell. The distribution of the scattered RF-shielded areas in a square of 600m by 600m is illustrated in Figure 5.4(d).

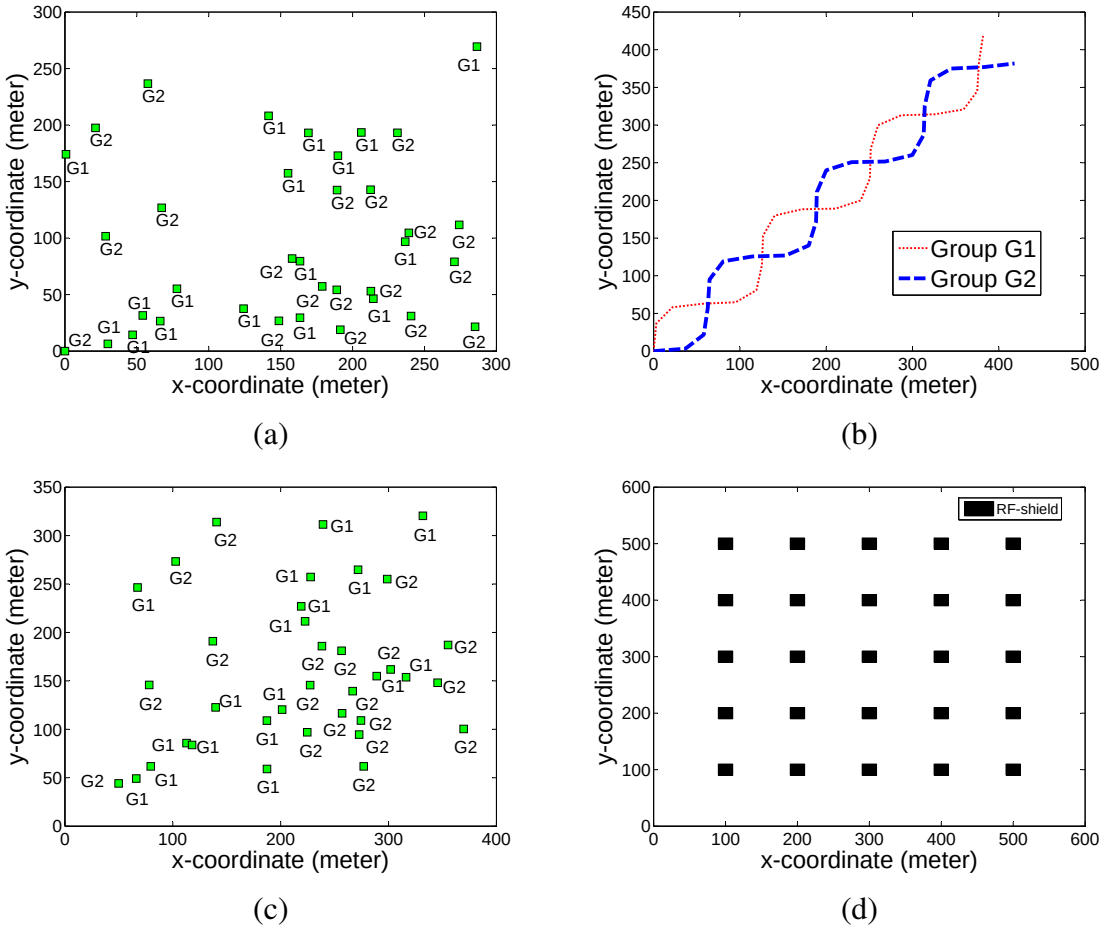


Figure 5.4: (a) initial location of all nodes in two groups-G1 and G2; (b) virtual trajectories of G1 and G2; (c) location of nodes after 0.5 virtual time unit; (d) RF-shielded areas.

5.4.2 Simulation Results

The performance of TARF is compared to that of a link connectivity-based routing protocol adapted from what is proposed by Alec Woo, Terence Tong and David Culler [151]. That link connectivity-based routing protocol is designed for low-power wireless sensor networks under dynamic network topology; experiments indicated its better performance in terms of energy efficiency and *throughput*, compared with other protocols based on Shortest-Path, Minimum Transmission, Broadcast, and Destination Sequenced Distance Vector (DSDV) [68, 115, 151]. For later convenience, we will simply refer to link connectivity-based routing protocol as *Link-connectivity*. Similarly to TARF, with the *Link-connectivity* protocol, each node makes its routing decision in a distributed manner; the next-hop node is selected among its neighborhood table according to an link estimator based on exponentially weighted moving average (EWMA). In our simulation with the *Link-connectivity* protocol, the next-hop selection is also run periodically. As noted in our experiments, the *Link-connectivity* protocol demonstrates strong adaptability into dynamic network conditions. However, as different than TARF, the *Link-connectivity* protocol assumes that all nodes are honest.

As we will see from the experimental results, in the presence of misbehaviors, the *throughput* in TARF is often much higher than that in *Link-connectivity*; the *hop-per-delivery* in the *Link-connectivity* protocol is generally at least comparable to that in TARF. For the TARF protocol in the simulation, unless mentioned otherwise, *EnergyWatcher* uses the parameters $w_{upgrade} = 0.1$, $w_{degrade} = 0.2$; *TrustManager* uses the parameters $w_{upgrade} = 0.1$, $w_{degrade} = 0.3$. For both TARF and *Link-connectivity*, if not specified otherwise, the period length is set to 1.

First, we conduct experiments to study the performance of TARF and *Link-connectivity* under a misbehavior-free environment; the results show that TARF and *Link-connectivity* have comparable performance when there is no adversary. Second, we evaluate TARF under three common types of attacks: (1) a certain node forges the identity of the based station

by replaying broadcast messages, also known as the *sinkhole* attack; (2) a set of nodes colludes to form a forwarding loop; and (3) a set of nodes drops received data packets. These experiments were conducted in the static case, the group-motion-with-noise case, and the addition of RF-shielded areas to the group-motion-with-noise case separately. Generally, under these common attacks, TARF produces a substantial improvement over *Link-connectivity* in terms of data collection and energy efficiency. Further, we evaluate TARF under more severe attacks: multiple moving fake bases and multiple *Sybil* attackers. As before, the experiments are conducted under all the three types of network topology. Under these two types of most severe attacks which almost devastates the *Link-connectivity* protocol, TARF succeeds in achieving a steady improvement over the *Link-connectivity* protocol. Finally, we discuss the choice of the period length and the trust updating scheme. Our experiments reveal that a shorter period or a faster trust updating scheme may not necessarily benefit TARF.

Comparable Performance in a Misbehavior-Free Environment

Under a misbehavior-free environment, the two protocols have comparable performance in packet delivery and energy efficiency. Under a misbehavior-free environment, according to the TARF protocol, a node may still perceive its neighbors as having different trust levels, due to the fact that the node cannot well distinguish between malicious behavior and failed delivery due to environmental effects. However, such misperception of trust, which reflects the instability of radio transmission, has a limited impact towards the performance of TARF. The comparability is verified by a few experiments under the three types of network topologies separately: static location, group-motion-with-noise, and group-motion-with-noise across RF-shielded areas. Figure 5.5 demonstrates the results from an experiment. Under each topology, as time elapses, the *throughput* of the TARF protocol gradually approaches the *throughput* of the *Link-connectivity* protocol. With the static location, after 100 periods, both TARF and *Link-connectivity* can achieve a *throughput* of at least 95% (see Figure 5.5(a)). Note that during first few periods, the performance of both protocols usually fluctuate much.

In the simulation, to focus on the long term evolution of the protocols' performance, we may omit the first few periods when presenting the graphic results. Our later empirical experiments (see Section 5.5.3, 5.5.4) indicates the significant improvement of TARF over *Link-connectivity* in fighting against attacks even during the early stage. As displayed in Figure 5.5(c), with a group motion pattern with noise, the *throughput* of both protocols goes below 32%. The group-motion-with-noise setting results in a a large portion of packets being lost due to the fast-changing network connection. Though TARF has a *throughput* slightly lower, it gradually catches up. Note that the *throughput* is calculated over the period from the beginning to the current moment. TARF does not negatively impact the overall *throughput*. Similarly, in the case of group-motion-with-noise across the RF-shielded areas (see Figure 5.5(e)), the *throughput* from both protocols is greatly impacted by the dynamic network; it further goes down to no more than 24%. In this case, TARF also gradually approaches *Link-connectivity* in *throughput*. Concerning the energy usage, TARF has a *hop-per-delivery* that is at least not higher than that of the *Link-connectivity* protocol. Some of our simulation results even show a lower *hop-per-delivery* for TARF (see Figure 5.5(b)(d)(e)), i.e., better energy-efficiency. That is because TARF selects routes that either have less hops or need less retransmission in these experiments.

Resilience under Common Attacks

Now, we evaluate the resilience of TARF under three common types of attacks: (a) a certain node forges the identity of the based station by replaying broadcast messages; (b) a set of nodes colludes to form a forwarding loop; and (c) a set of nodes drops received data packets. Specifically, in our simulation, these three attacks are: (a) a compromised node at the "heart" of the network becomes a fake base station through replaying the routing information from the base station (see Figure 5.6(a)); (b) 5 nodes close to the base station collude to form a network loop (see Figure 5.6(b)); (c) 6 nodes drop any data packet received (see Figure 5.6(c)). The simulation results show that, in the case of a static location, TARF maintains a high

throughput and a low *hop-per-delivery* under these attacks, which is generally a tremendous improvement over the *Link-connectivity* protocol. In the case of group-motion-with-noise or crossing RF-shielded areas, compared to the almost devastating impact of these attacks to the *Link-connectivity* protocol, TARF shows a steady improvement in packet delivery and energy-efficiency. Considering the precious value of each data packet under an emergent sensing mission in a hostile environment, such an improvement can be vital.

Under attack scenario (a) (see Figure 5.6(a)), the fake base station attempts to attract a significant portion of the network traffic by cheating nearby nodes to believe in its false identity. With the *Link-connectivity* protocol, a node cannot distinguish between such a fake base station and the real base station, which results in at least half network traffic being directed to the "blackhole". In addition to the low *throughput*, the *Link-connectivity* protocol also shows relatively low energy efficiency (see Figure 5.7), i.e., a relatively higher *hop-per-delivery*, since too many "hops" end up with sending the data finally to the fake base station. However, with TARF, a significant amount of data packets are delivered with routes circumventing the fake base. The improvement of TARF over *Link-connectivity* in *throughput* is very noticeable in the static location case: the *throughput* doubles (see Figure 5.7(a)). In the case of group motion and crossing the RF-shielded areas, though the fake base station together with the hostile mobile network condition strongly limit the *throughput* (see Figure 5.5(c)(e)), compared with the *Link-connectivity* protocol, TARF still succeeds in saving a considerable amount of data packets from being misdirected (see Figure 5.7(c)(e)). Such a considerable amount of data saved protects the WSN from being devastated; it can be of significant value in a critical mission based on the data collection task from a WSN. Concerning energy efficiency, TARF produces a significantly lower *hop-per-delivery* than *Link-connectivity* does (see Figure 5.7(b)(d)(f)). The main reason is that TARF not only selects trustworthy route paths but also considers energy consumption when making a route decision. Compared with the *hop-per-delivery* in a misbehavior-free environment (see Figure 5.5(b)(d)(f)), with the aid of TARF, the existence of the fake base station only results in a *hop-per-delivery* that is no

more than 30% higher. Considering that *hop-per-delivery* takes into account those hops ending up with directing the packets to the fake base in vain, the energy cost of each successful route identified by TARF is comparable to that in a misbehavior-free environment.

Under attack scenario (b) (see Figure 5.6(b)), the network loop comprised of 5 compromised nodes close to the base station attempts to cheat nearby node into forwarding packets into this infinite loop. The compromised nodes inside the loop appear to behave normally to the outside nodes: they receive packets from the outside and issue acknowledgements; they forward the received packets immediately to the next node after the reception (but to its "conspirators"). Luckily enough, in our experiment of the *Link-connectivity* protocol in the static location case (see Figure 5.8(a)), a few nearby nodes are able to select route paths avoiding the loop, thus escaping from such a trap. Note that the preference of route stability in the *Link-connectivity* protocol helps retain those lucky choices. The lucky choices are the result of the random instability of the radio communication. However, such luck does not happen in the case of group-motion-with-noise and crossing the RF-shielded areas (see Figure 5.8(c)(e)): the *Link-connectivity* protocol suffers the devastation by that loop. The motion of the network greatly increases the chance of the interaction between the loop and its nearby regular nodes, so that a great majority of data packets are intercepted by this loop. Similarly to the scenario of attack (a), TARF achieves a high *throughput* in the static location case (see Figure 5.8(a)), and displays a steady improvement in saving the network from being devastated in the case of group-motion-with-noise and crossing RF-shielded areas (see Figure 5.8(c)(e)). Despite the 5 nodes close to the base station in the loop being compromised, TARF manages to find paths with acceptable energy efficiency in the static case (see Figure 5.8(b)). The high *hop-per-delivery* value (thus low energy efficiency) in the case of group motion and crossing RF-shielded areas (see Figure 5.8(d)(f)) are mainly caused by the low *throughput*.

Under attack scenario (c) (see Figure 5.6(c)), 6 nodes a bit far away from the base station drop any data packet received. Like the nodes in a loop, these 6 nodes appears to be normal nodes in receiving packets and acknowledgement, but do not forward any packet. The *Link-connectivity* protocol in this simulation does not provide mechanism to check whether a packet sent is forwarded by the receiver. Observing the geographic location of these 6 nodes in the network, they may not pose as much threat to the network as the aforementioned network loop does. Since these 6 nodes are not much attractive, the *Link-connectivity* protocol still produces a *throughput* of at least 90%, a bit lower than TARF (see Figure 5.9(a)). Unlike the static case, in the case of group-motion-with-noise and crossing RF-shielded areas (see Figure 5.9(c)(e)), TARF shows relatively great improvement over the *Link-connectivity* protocol. That is because the movement of nodes creates a better condition for these 6 nodes to jeopardize the network; TARF helps the network in recognizing these compromised nodes. Another impact of the geographic location of these 6 nodes is towards the *hop-per-delivery*. As shown in Figure 5.9(b)(d)(e), the *hop-per-delivery* of the *Link-connectivity* protocol is just a bit higher than TARF. The reason for such seemingly "efficient" energy usage for the *Link-connectivity* protocol is that the *Link-connectivity* protocol causes the delivery of a large amount of packets to be interrupted at these 6 compromised node, thus resulting in seemingly "short" routing paths.

Resilience of TARF against Multiple Moving Fake Bases

Now we test the resilience of TARF against a strong form of attack where multiple fake base stations move fast in the network. Specifically, in each of our experiments, 1 to 5 nodes are compromised and become fake bases, each moving along its closed loop-shaped path with small random turbulence (see Figure 5.10(a)). During each virtual unit of time, each fake base finishes a "round trip" along its path. In the case the whole network moves in groups, each fake base also moves along with its original group in addition to its loop-shaped motion. Note that these fake bases move in a small "neighborhood" instead of moving across the

whole network. Such local motion enables the fake bases to be able to acknowledge many data packets sent to it. Otherwise, if they move across too wide an area, then the incapability to receive and acknowledge packets directed to it would make their neighbors recognize them as nodes with poor radio connection; in that case, the *Link-connectivity* protocol to a certain degree could circumvent these fake bases. In our experiments with these locally moving fake bases, we note that though the local movement of these fake bases poses great threat against the network, depending on the specific location and the movement pattern, it may not necessarily cause a worse *throughput* than the static fake bases. We conduct experiments with 1 fake base, 3 fake bases and 5 fake bases separately. The numbering of the fake bases is displayed in Figure 5.10(a). Each set of fake bases in our experiments consist of the first few fake bases.

Overall, in these attacks, TARF shows a steady improvement over the *Link-connectivity* in *throughput*, as shown in Figure 5.11. Generally speaking, the less attackers there are, the more potential of improvement TARF has. However, our experiments show certain exception. For example, in one experiment with 5 moving fake bases and a network with static location, TARF achieves a slightly higher *throughput* than with 3 moving fake bases (see Figure 5.11(b)). The performance of TARF is related to the network topology. In the static case, TARF produces a at least 60% *throughput*, even with these 5 moving fake bases. In the case of group-motion-with-noise and crossing the RF-shielded areas, though the space of improvement is limited due to the hostile network connection, TARF still demonstrates a steady improvement over the *Link-connectivity* protocol (see Figure 5.11(c)(d)(e)(f)).

Resilience of TARF against Multiple Sybil Attackers

Now we conduct experiments to test TARF against multiple *Sybil* attackers. In a *Sybil* attack, an adversary presents multiple identities to harm the network. In our experiments, we set up 1 to 5 *Sybil* attackers; each set of attacker comprise the first few numbered attackers (see Figure 5.10(b)). Each attacker uses a single identity in each 3 periods, and then switch to

another identity in the next 3 periods; it forges the identities of all valid nodes over time. The reason for a *Sybil* attacker to keep an identity for 3 periods is to accumulate considerable "reputation" for that forged identity. *Sybil* attackers are usually hard to detect. TARF does not attempt to "physically" identify a *Sybil* attacker; instead, TARF enables a node to send data to a promising next-hop identity that has more likelihood to deliver data. Under these Sybil attacks, the *throughput* of TARF is compared with that of *Link-connectivity*. The graphic results of our experiments are illustrated in Figure 5.12. Similar to the scenario of multiple moving fake base stations, TARF shows a significant improvement over the *Link-connectivity* protocol in *throughput* in the case of a physically static network (see Figure 5.12(a)(b)). In the case of group motion and crossing the RF-shielded areas (see Figure 5.12(c)(d)(e)(f)), TARF achieves a stably increasing *throughput* over time.

Discussion of TARF: Period Length and Update Speed

Here we discuss the selection of the period length and update speed for TARF. Intuitively, it appears that a short period and faster update speed of trust might help TARF in identifying malicious attackers more quickly. However, the downside of adopting a short period or fast update is that doing so may cause a node to misjudge an honest neighbor as an attacker. Due to the instability and the randomness of the radio communication, carefulness should be taken when updating the trust level of a neighbor, especially in a mobile environment. Our experiments indicate that simply shortening the period length or expediting trust update for TARF does not necessarily produce positive improvement. In certain cases, that even impairs the performance.

In addition to TARF, we also conduct experiments using the following variant of TARF - TARF_QUICKDEGRADE: when updating trust with $DeliveryRatio$, *TrustManager* (see Section 5.3.4) adopts a varying value of $w_{degrade}$ that increases linearly as $DeliveryRatio$ decreases. Essentially, when the routing via a neighboring node is performing worse, TARF_QUICKDEGRADE

degrades the trust level of that neighbor in a faster manner than TARF does. In several experiments, we compare the *throughput* metric of the following protocols: TARF with period being 1 virtual time unit, TARF with period being 0.5 virtual time unit, TARF_QUICKDEGRADE with period being 1, and TARF_QUICKDEGRADE with period being 0.5. Basically, we rerun certain experiments aforementioned now with different protocols. Certain graphic results are presented in Figure 5.13. Interestingly, TARF_QUICKDEGRADE(period=1) and TARF_QUICKDEGRADE(period=0.5) show lower *throughput* than TARF(period=1) during several experiments in the following scenarios: one fake base in a physically static network (Figure 5.13(a)), one fake base in the case of group-motion-with-noise (Figure 5.13(b)), a network loop of 5 nodes in the case of group-motion-with-noise (Figure 5.13(c)), and one fake base in the case of crossing the RF-shielded areas (Figure 5.13(d)). This fact indicates that fast degradation of trust may not always improve the performance and that the parameters involved in trust update should be carefully selected. The indication is also supported by a few other experiments. Regarding the period length, though TARF(period=0.5) seems to show a higher *throughput* than TARF(period=1) in Figure 5.13(a)(c), TARF(period=1) gradually develops a slightly better performance than TARF(period=0.5) in Figure 5.13(b)(d). To explain, though a shorter period seems to provide a better "real-time" trust estimation of a route, the distributed routing decision by the individual nodes spoils that benefit: any premature estimation of the quality of an individual link in a short period may compromise any path going through that link. Thus, shorter periods does not necessarily bring a higher throughput.

5.5 Implementation and Empirical Evaluation

In order to evaluate TARF in a real-world setting, we implemented the *TrustManager* component on TinyOS 2.x, which can be integrated into the existing routing protocols for WSNs with moderate effort. Originally, we had implemented TARF as a self-contained routing protocol [165] on TinyOS 1.x before this second implementation. However, we decided to

re-design the implementation considering the following factors. First, the first implementation only supports TinyOS 1.x, which was replaced by TinyOS 2.x; the porting procedure from TinyOS 1.x to TinyOS 2.x tends to frustrate the developers. Second, rather than developing a self-contained routing protocol, the second implementation only provides a *TrustManager* component that can be easily incorporated into the existing protocols for routing decisions. The detection of routing loops and the corresponding reaction are excluded from the implementation of *TrustManager* since many existing protocols, such as Collection Tree Protocol [48] and the link connectivity-based protocol [151], already provide that feature. As we worked on the first implementation, we noted that the existing protocols provide many nice features, such as the analysis of link quality, the loop detection and the routing decision mainly considering the communication cost. Instead of providing those features, our implementation focuses on the trust evaluation based on the base broadcast of undelivered information, and such trust information can be easily reused by other protocols. Finally, instead of using TinySec [73] exclusively for encryption and authentication as in the first implementation on TinyOS 1.x, this re-implementation let the developers decide which encryption or authentication techniques to employ; the encryption and authentication techniques of TARF may be different than that of the existing protocol.

5.5.1 TrustManager Implementation Details

The *TrustManager* component in TARF is wrapped into an independent TinyOS configuration named `TrustManagerC`. `TrustManagerC` uses a dedicated logic channel for communication and runs as a periodic service with a configurable period, thus not interfering with the application code. Though it is possible to implement TARF with a period always synchronized with the routing protocol's period, that would cause much intrusion into the source code of the routing protocol. The current `TrustManagerC` uses a period of 30 seconds; for specific applications, by modifying a certain header file, the period length may be

re-configured to reflect the sensing frequency, the energy efficiency and trustworthiness requirement. `TrustManagerC` provides two interfaces (see Figure 5.14), `TrustControl` and `Record`, which are implemented in other modules. The `TrustControl` interface provides the commands to enable and disable the trust evaluation, while the `Record` interface provides the commands for a root, i.e., a base station, to add delivered message record, for a non-root node to add forwarded message record, and for a node to retrieve the trust level of any neighboring node. The implementation on a root node differs from that on a non-root node: a root node stores the information of messages received (delivered) during the current period into a record table and broadcast delivery failure record; a non-root node stores the information of forwarded messages during the current period also in a record table and compute the trust of its neighbors based on that and the broadcast information. Additionally, to avoid the problem of possible "gaps" between two continuous periods, for each origin involved, the broadcast message from a root also includes the corresponding minimal and maximal sequence number received during the current period. Noting that much implementation overhead for a root can always be transferred to a more powerful device connected to the root, it is reasonable to assume that the root would have great capability of processing and storage.

For a root node, the record table keeps the delivered message intervals for up to 100 source nodes, with up to 20 non-overlapped significant delivered intervals for each individual origin. Once the table already contains 100 source nodes' record, it will not enter any record from another new source node. The table size is decided to be limited so that a root can run TARF on its own without the aid from a powerful computer connected to it. That consideration brings convenience for experiments on a remote WSN testbed. It is also viable to remove that limit and transfer the overhead to the powerful computer connected to the root. A root broadcasts two types of delivery failure record: at most three packets of significant undelivered intervals for individual origins and at most two packets of the id's of the origins without any record in the current period. For each origin, at most three significant undelivered intervals are broadcast. For a non-root node, considering the processing and memory usage overhead,

the record table keeps the forwarded message intervals for up to 20 source nodes, with up to 5 non-overlapped intervals for each individual origin. Our later experiments verify that such size limit of the table on a non-root node produces a resilient TARF with moderate overhead. The record table on a node keeps adding entries for new origins until it is full.

Whenever a non-root node receives a fresh trust broadcast message that is not in its cache, it executes the following action and then posts a task to re-broadcast the message to its neighbors: for each node involved in both its record table and the broadcast message, it computes the number of the undelivered messages and that of the forwarded messages; concerning the number of the forwarded messages, a forwarded interval in the record table is counted only if it overlaps with an undelivered interval; regarding the number of the undelivered messages, we only count the overlapping part of the undelivered intervals in the broadcast message and the forwarded intervals. The counting is implemented in this way because the limited memory leads to the incompleteness of all types of records. These two numbers are then used to calculate the current delivery ratio, which is later used to update the trust level of the corresponding neighbor. After three seconds from the reception of a fresh broadcast message, the node will assume that all broadcast messages have been received and processed. Then it starts to compute the trust level for each known neighbor. To protect the radio stack buffer, a message queue is implemented to store the broadcast messages waiting to be sent; whenever a `sendDone` event is fired, the message queue, if not empty, dequeues a message and post a task to send it. With our current implementation, a valid trust value is an integer between 0 and 100, and any node is assigned an initial trust value of 50. The weigh parameters are: $w_{upgrade} = 0.1$, $w_{degrade} = 0.3$. The trust table of a non-root node node keeps the trust level for up to 10 neighbors. Considering that an attacker may present multiple fake id's, two techniques may be employed. One countermeasure is to increase the size of the trust table to a reasonable magnitude. The other technique, as currently implemented, evicts entries with a trust level close to the initial trust of any node. Such eviction policy is to ensure that the trust

table remembers those neighbors with high trust and low trust; any other neighbor not in this table is deemed to have the initial trust value of 50.

5.5.2 Incorporation of TARF into Existing Protocols

To demonstrate how this TARF implementation can be integrated into the exiting protocols with moderate effort, we incorporated TARF into a collection tree routing protocol (CTP) [48]. The CTP protocol is efficient, robust, and reliable in a network with highly dynamic link topology. It quantifies link quality estimation in order to choose a next-hop node. The software platform is TinyOS 2.x. Figure 5.15 demonstrates the procedure to perform the integration. First, as in every TinyOS program, the `TrustControl` interface and the `Record` interface are wired to the `TrustManagerC` component properly. Then, call the `TrustControl.start` command when a `booted` event is fired to enable the trust evaluation; call the `Record.addForwarded` command for a non-root node to add forwarded record once a data packet has been forwarded; call the `Record.addDelivered` command for a root to add delivered record once a data packet has been received by the root. Finally, inside the CTP's task to update the routing path, call the `Record.getTrust` command to retrieve the trust level of each next-hop candidate; an algorithm taking trust into routing consideration is executed to decide the new next-hop neighbor (see Figure 5.16).

Similar to the original CTP's implementation, the implementation of this new protocol decides the next-hop neighbor for a node with two steps: Step 1 traverses the neighborhood table for an optimal candidate for the next hop; Step 2 decides whether to switch from the current next-hop node to the optimal candidate found. For Step 1, as in the CTP implementation, a node would not consider those links congested, likely to cause a loop, or having a poor quality lower than a certain threshold. This new implementation prefers those candidates with higher trust levels; in certain circumstances, regardless of the link quality, the rules deems a neighbor with a much higher trust level to be a better candidate (see Figure 5.16). The preference of highly trustable candidates is based on the following consideration: on the one hand,

it creates the least chance for an adversary to misguide other nodes into a wrong routing path by forging the identity of an attractive node such as a root; on the other hand, forwarding data packets to a candidate with a low trust level would result in many unsuccessful link-level transmission attempts, thus leading to much re-transmission and a potential waste of energy. When the network *throughput* becomes low and a node has a list of low-trust neighbors, the node will exclusively use the trust as the criterion to evaluate those neighbors for routing decisions. As show in Figure 5.16, it uses trust/cost as a criteria only when the candidate has a trust level above certain threshold. The reason is, the sole trust/cost criteria could be exploited by an adversary replaying the routing information from a base station and thus pretending to be an extremely attractive node. As for Step 2, compared to the CTP implementation, we add two more circumstances when a node decides to switch to the optimal candidate found at Step 1: that candidate has a higher trust level, or the current next-hop neighbor has a too low trust level.

This new implementation integrating TARF requires moderate program storage and memory usage. We implemented a typical TinyOS data collection application, MultihopOscilloscope [106], based on this new protocol. The MultihopOscilloscope application, with certain modified sensing parameters for our later evaluation purpose, periodically makes sensing samples and sends out the sensed data to a root via multiple routing hops. Originally, MultihopOscilloscope uses CTP as its routing protocol. Now, we list the ROM size and RAM size requirement of both implementation of MultihopOscilloscope on non-root Telosb [139] motes in Table 5.1. The enabling of TARF in MultihopOscilloscope increases the size of ROM by around 1.3KB and the size of memory by around 1.2KB.

Table 5.1: Size comparison of MultihopOscilloscope implementation

Protocol	ROM (bytes)	RAM (bytes)
CTP	31164	3579
TARF-enabled CTP	34290	4767

5.5.3 Empirical Evaluation on Motelab

To evaluate how effective TARF is against deception through replaying routing information in the real world, we tested the performance of TARF on Motelab [105] at Harvard University. As a public test bed of wireless sensor networks, at the time of our experiments, totally 184 TMote Sky sensor motes were deployed in the Electrical Engineering and Computer Science building: approximately 97 nodes functioned properly while the rest were either removed or disabled. Any of these motes has a 8MHz TI MSP430 processor, 10KB of RAM, 1Mbit of Flash memory, and a 2.4GHz Chipcon CC2420 radio with an indoor range of approximately 100 meters. These motes are distributed over many rooms at three floors, with two to four motes in most rooms. In addition to the wireless connectivity among the nodes on each same floor, there also exists certain wireless connection between nodes from different floors.

We developed a simple data collection application in TinyOS 2.x that sends a data packet containing an increasing sequence number every five seconds; the data packet is supposed to be delivered to a base station node (root). This application was executed on 91 functioning non-root nodes on Motelab. This program does not include any sensing functionality: whenever the five-second periodic timer fires, it posts a task to send out a data packet. The absence of sensing from the program offers the benefit that the experiments would not be impacted by the processing time for sensing. Also, later experiments show that the setting of sending every five seconds does not cause congestion to the network. For comparison, we used CTP and the TARF-enabled CTP implementation as the routing protocols for the data collection program separately. The TARF-enabled CTP has a TARF period of 30 seconds. The CTP and the TARF-enabled CTP implemented were wired into the data collection application separately, and the resulting programs were uploaded onto Motelab. Additionally, we conducted an attack with five fake base stations that formed a *wormhole*. Whenever the base station sent out any packet, three fake base stations which overheard that packet replayed the complete packet without changing any content including the node id. Other fake base stations overhearing that replayed packet would also replay the same packet. Through replaying the packets

from the base station, these fake base stations essentially forged the id of the real base station: a node would recognize those fake base stations as the real base station. Such a *wormhole* effectively creates fake base stations even in a remote location. And such a replaying technique does not require any knowledge on what encryption or authentication techniques have been possibly adopted by the base station. Note that there is a distinction between such malicious replay and the forwarding when a well-behaved node receives a broadcast from the base station. When a well-behaved node forwards a broadcast packet from the base station, it will include its own id in the packet so that its receivers will not recognize the forwarder as a base station. We conducted the first experiment by uploading the program with the CTP protocol onto 91 motes (not including those 5 selected motes as fake bases in later experiments), and no attack was involved here. Then, in another experiment, in addition to programming those 91 motes with CTP, we also programmed the five fake base stations so that they stole the id the base station through replaying. In the last experiment, we programmed those 91 motes with the TARF-enabled CTP, and programmed the five fake base stations as in the second experiment. The root is programmed with CTP in the first and second experiments, and with the TARF-enabled CTP in the last experiment.

Each of our programs run for 30 minutes. As illustrated in Figure 5.17(a), the existence of the five *wormhole* attackers greatly degraded the performance of CTP: the number of the delivered data packets in the case of CTP with the five-node *wormhole* is no more than 14% that in the case of CTP without adversaries. The TARF-enabled CTP succeeded in bringing an immense improvement over CTP in the presence of the five-node *wormhole*, almost doubling the *throughput*. That improvement did not show any sign of slowing down as time elapsed. The number of nodes from each floor that delivered at least one data packet in each six-minute sub-period is plotted in Figure 5.17(a), Figure 5.17(b) and Figure 5.17(c) separately. On each floor, without any adversary, at least 24 CTP nodes were able to find a successful route in each six minute. However, with the five fake base stations in the *wormhole*, the number of CTP nodes that could find a successful route goes down to 9 for the first floor; it decreases

to no more than 4 for the second floor; as the worst impact, none of the nodes on the third floor ever found a successful route. A further look at the data showed that all the nine nodes from the first floor with successful delivery record were all close to the real base station. On the other hand, the CTP nodes relatively far away from the base station, such as those on the second and the third floor, had little luck in making good routing decisions. When TARF was enabled on each node, the nodes close to the *wormhole* became aware of the fake base stations; most nodes made correct routing decisions circumventing the attackers. That improvement can be verified by the fact that the number of the TARF-enabled nodes with successful delivery record under the threat of the *wormhole* is close to that of CTP nodes with no attackers, as shown in Figure 5.17(a), Figure 5.17(b) and Figure 5.17(c).

5.5.4 Application: Mobile Target Detection in the Presence of an Anti-Detection Mechanism

To demonstrate how TARF can be applied in networked sensing systems, we developed a proof-of-concept resilient application of target detection. This detection application relies on a deployed wireless sensor network to detect a target that could move, and to deliver the detection events to a base station via multiple hops with the TARF-enabled CTP protocol. The detection report collected by the base station is sent to a server for a visualized report. To simplify the task of detecting the target, in our experiment, the target used is a TelosB mote that sends out an AM (Active Message) packet every three seconds of a particular type. A node in this detection application receiving such a type of packet from the target issues a detection report, which will be sent to the base station with the aforementioned TARF-enabled CTP protocol.

The experiment is set up within a clear floor space of 90 by 40 inches with 15 TelosB motes (see Figure 5.18(a)). To make the multi-hop delivery necessary, the transmission power of all the Telosb motes except two fake base stations in the network is controlled through both software reduction and attenuator devices, so that the controlled transmission range is within

30 inches. To add mobility to the target, this target mote is mounted on a LEGO MINDSTORM NXT 2.0 vehicle robot [85]. The target uses an anti-detection mechanism utilizing a fake base station close to the real base station and another remote base station close to the target and mounted on another LEGO vehicle robot. The two fake base stations, operated with a maximal transmission power and a corresponding transmission range of at least 100 feet, collude to form a *wormhole*: the fake base station close to the base station replays all the packets from the base station immediately with a powerful radio; the remote fake base station, after receiving those packets, immediately replays it again with a powerful radio. This anti-detection mechanism tricks some network nodes into sending their event reports into these fake base stations instead of the real base station. Though the fake base station close to the real base station is capable of cheating the whole network alone by itself with its powerful radio for a certain amount of time, it can be easily recognized by remote nodes as a poor next-hop candidate soon by most routing protocols based on link quality: that fake base station does not acknowledge the packets "sent" to it via a single hop from remote nodes with a weak radio since it cannot really receive them. Thus, the anti-detection mechanism needs to create such a *wormhole* to replay the packets from the base station remotely.

The target node 14 and the fake base station 13 close to it move across the network along two parallel tracks of 22 inches back and forth (see Figure 5.18(b)); they travel on each forward or backward path of 22 inches in around 10 minutes. The experiment lasts 30 minutes. For comparison, three nodes 9, 10 and 11 programmed with the CTP protocol are paired with another three nodes 6, 7 and 8 programmed with the TARF-enabled CTP (see Figure 5.18(b)); each pair of nodes are physically placed close enough. All the other nodes, except for the fake base stations and the target node, are programmed with the TARF-enabled CTP. To fairly compare the performance between CTP and the TARF-enabled CTP, we now focus on the delivered detection reports originating from these three pairs of nodes: pair (9, 6), (10, 7) and (11, 8). For the timestamp of any detection report from these six nodes, we plot a corresponding symbol: a purple circle for the nodes with the TARF-enabled CTP; a

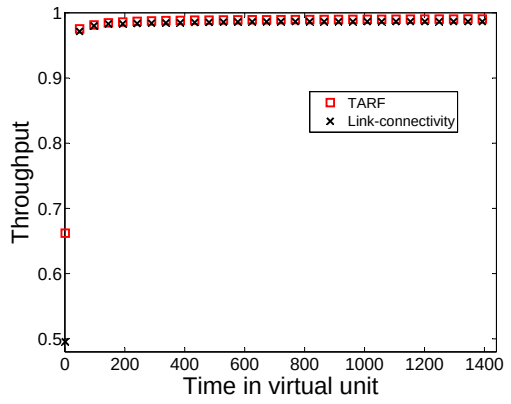
black cross for the CTP nodes. The resulting detection report is visualized in Figure 5.19(a). Roughly, the TARF nodes report the existence of the target seven times as often as the CTP nodes do. More specifically, as shown in Figure 5.19(b), in the pair (9, 6), no report from CTP node 9 is delivered while 46 reports from TARF node 6 is delivered; in the pair (10, 7), no report from CTP node 10 is delivered while 80 reports from TARF node 7 is delivered; in the pair (11, 8), 40 reports from CTP node 11 is delivered while 167 reports from TARF node 8 is delivered. Taking into account the spatial proximity between each pair of nodes, the TARF-enabled CTP achieves an enormous improvement in target detection over the original CTP.

The demonstration of our TARF-based target detection application implies the significance of adopting a secure routing protocol in certain critical applications. The experimental results indicate that TARF greatly enhances the security of applications involving multi-hop data delivery.

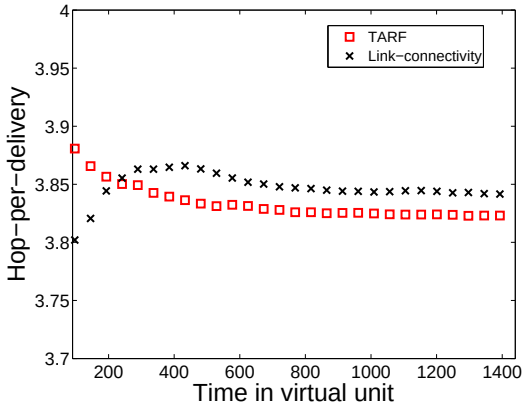
5.6 Summary

We have designed and implemented TARF, a robust trust-aware routing framework for WSNs, to secure multi-hop routing in dynamic WSNs against harmful attackers exploiting the replay of routing information. TARF focuses on trustworthiness and energy efficiency, which are vital to the survival of a WSN in a hostile environment. With the idea of trust management, TARF enables a node to keep track of the trustworthiness of its neighbors and thus to select a reliable route. Our main contributions are listed as follows. (1) Unlike previous efforts at secure routing for WSNs, TARF effectively protects WSNs from severe attacks through replaying routing information; it requires neither tight time synchronization nor known geographic information. (2) The resilience and scalability of TARF is proved through both extensive simulation and empirical evaluation with large-scale WSNs; the evaluation involves both static and mobile settings, hostile network conditions, as well as strong attacks such as

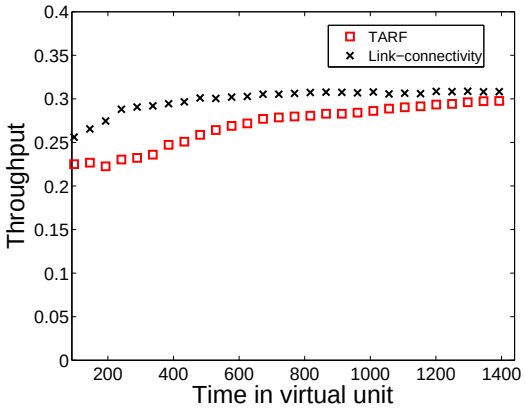
wormhole attacks and *Sybil* attacks. (3) We have implemented a ready-to-use TinyOS module of TARF with low overhead; as demonstrated in the chapter, this TARF module can be integrated into existing routing protocols with the moderate effort, thus producing secure and efficient fully-functional protocols. (4) Finally, we demonstrate a proof-of-concept mobile target detection application that is built on top of TARF and is resilient in the presence of an anti-detection mechanism; that indicates the potential of TARF in WSN applications. We believe that the idea of TARF can also be applied to general ad hoc networks and peer-to-peer networks to fight against similar attacks.



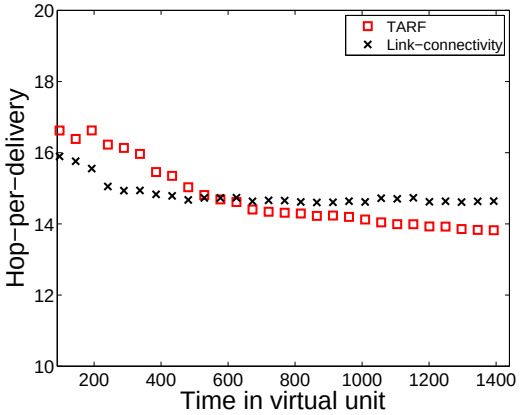
(a) Static location
No misbehavior



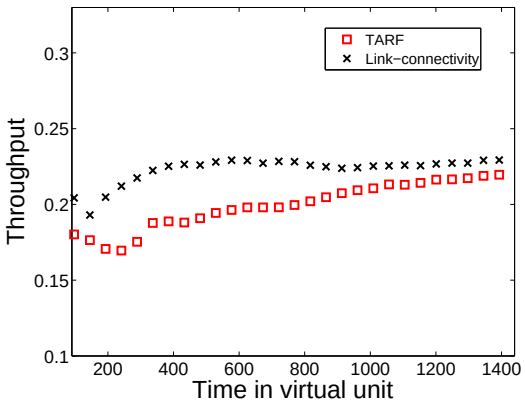
(b) Static location
No misbehavior



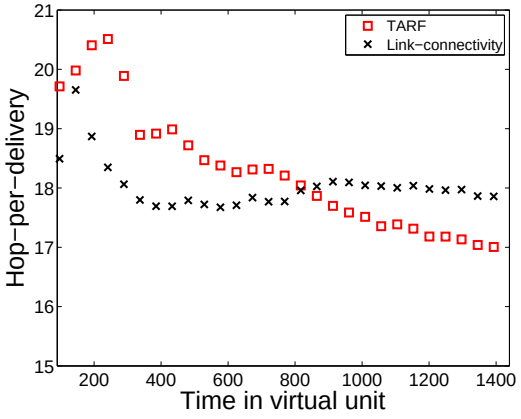
(c) Group motion with noise
No misbehavior



(d) Group motion with noise
No misbehavior

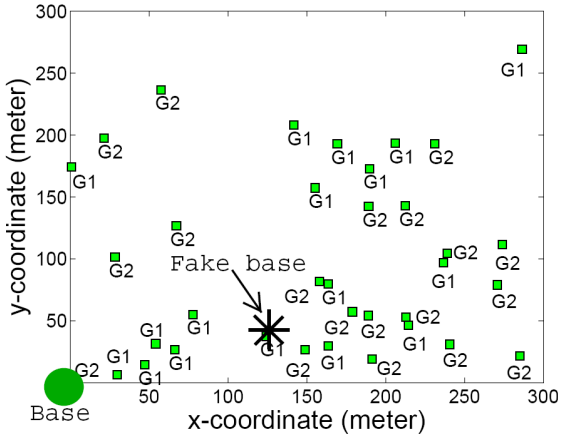


(e) Crossing RF-shield
No misbehavior

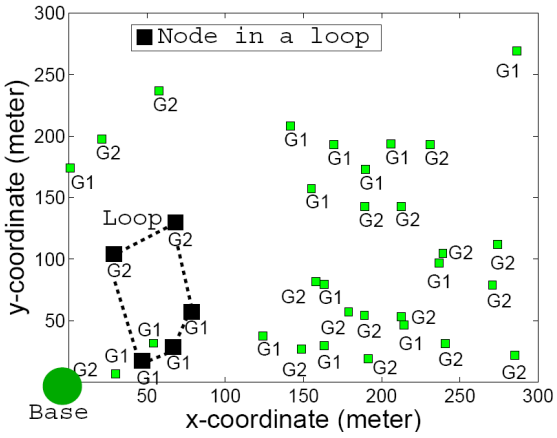


(f) Crossing RF-shield
No misbehavior

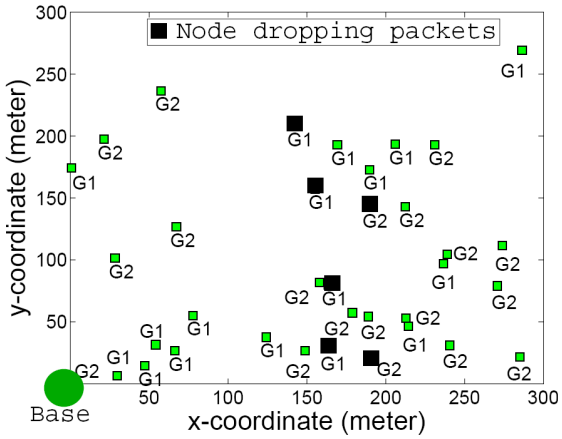
Figure 5.5: Under a misbehavior-free environment, TARF and *Link-connectivity* are comparable in *throughput* and *hop-per-delivery*.



(a) One fake base

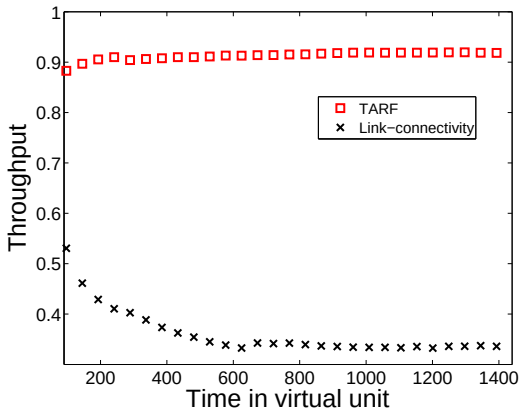


(b) A network loop

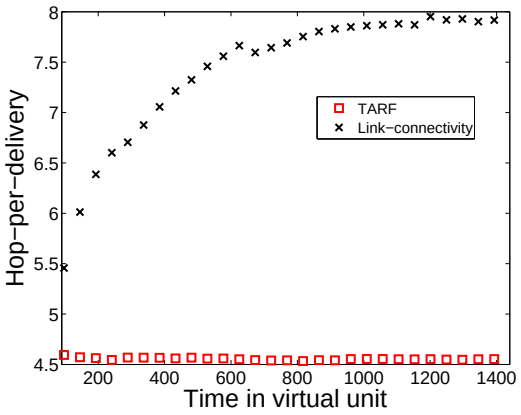


(c) 6 nodes dropping packets

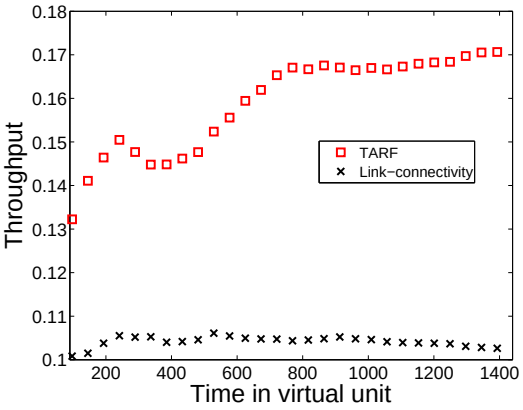
Figure 5.6: Common attacks (small green squares are regular nodes): (a) a fake base station (black star); (b) a network loop consisting of 5 nodes (big black squares); (c) 6 nodes dropping packets (big black squares).



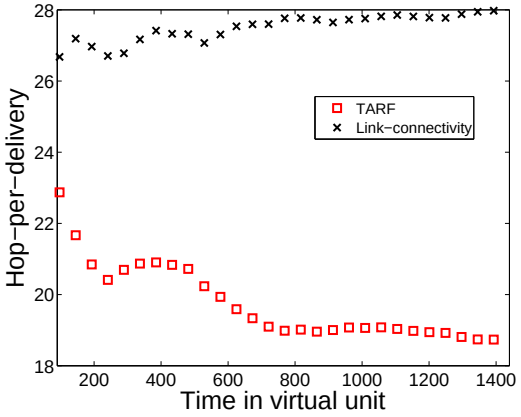
(a) Static location
One fake base



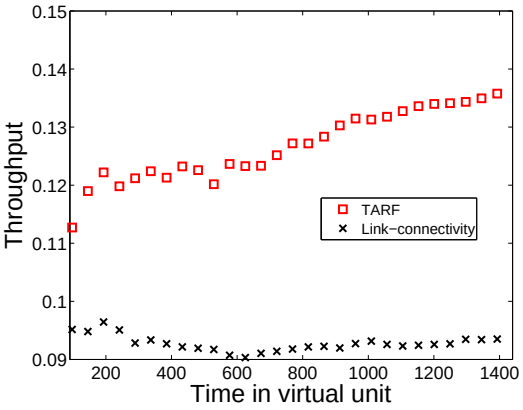
(b)Static location
One fake base



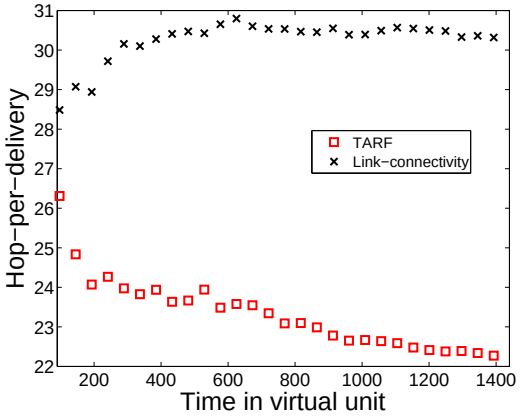
(c) Group motion
One fake base



(d) Group motion
One fake base

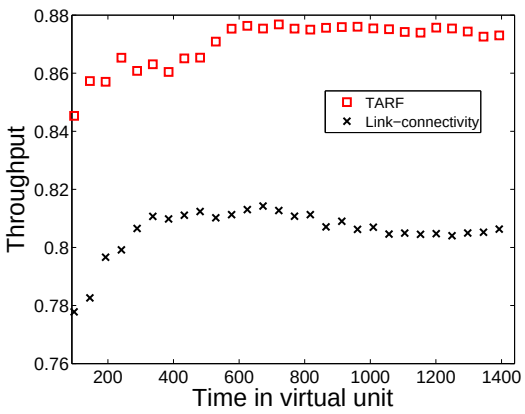


(e) Crossing RF-shield
One fake base

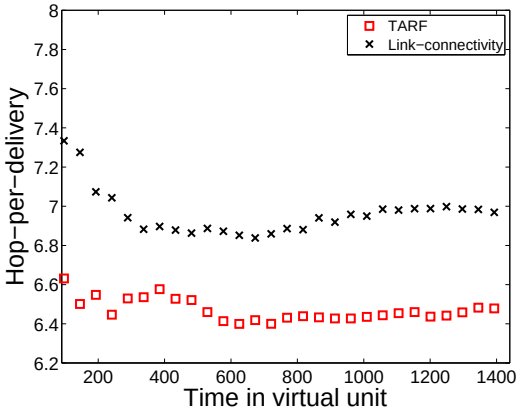


(f) Crossing RF-shield
One fake base

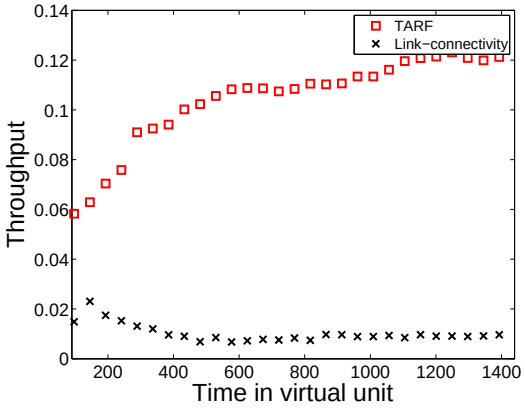
Figure 5.7: With one fake base, TARF shows higher *throughput* and lower *hop-per-delivery* than the *Link-connectivity* protocol.



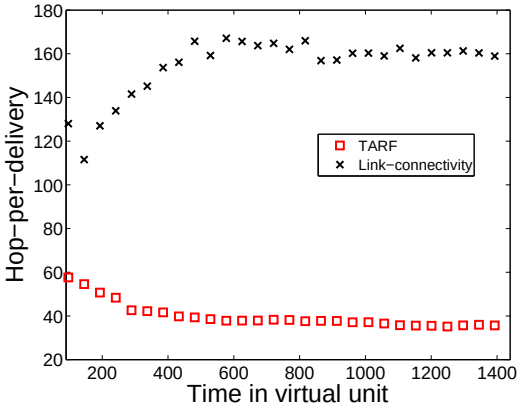
(a) Static location
A network loop of 5 nodes



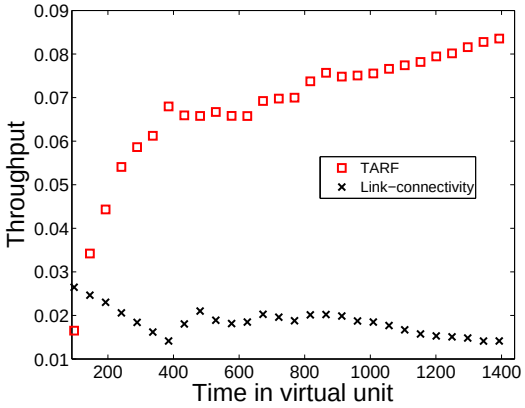
(b)Static location
A network loop of 5 nodes



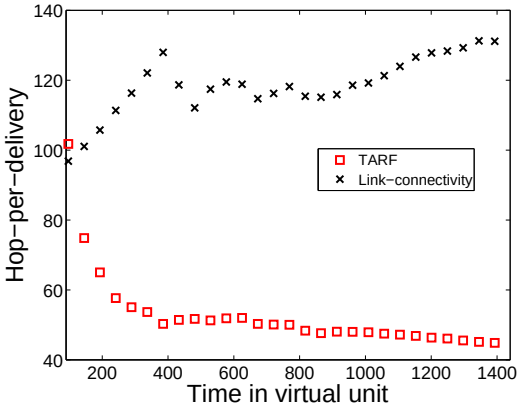
(c) Group motion with noise
A network loop of 5 nodes



(d) Group motion with noise
A network loop of 5 nodes

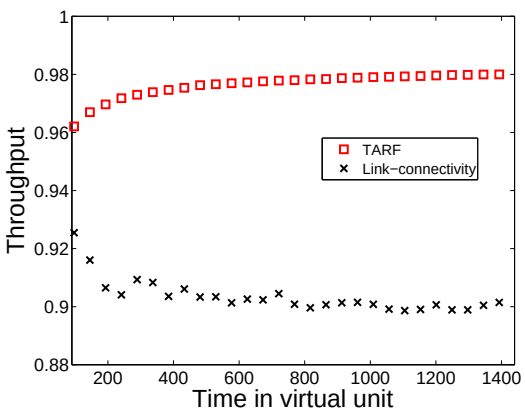


(e) Crossing RF-shield
A network loop of 5 nodes

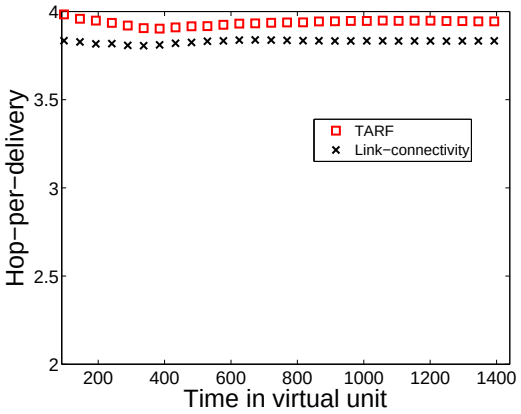


(f) Crossing RF-shield
A network loop of 5 nodes

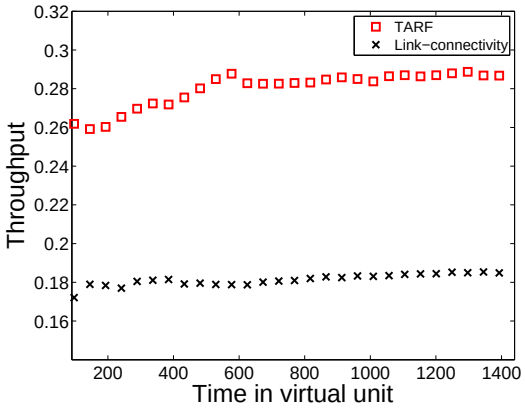
Figure 5.8: With a network loop, TARF shows higher *throughput* and lower *hop-per-delivery* than the *Link-connectivity* protocol.



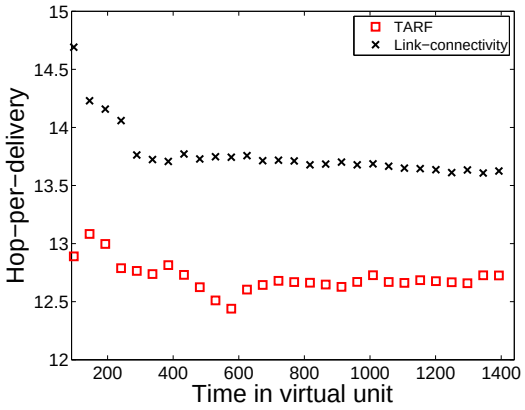
(a) Static location
6 nodes dropping



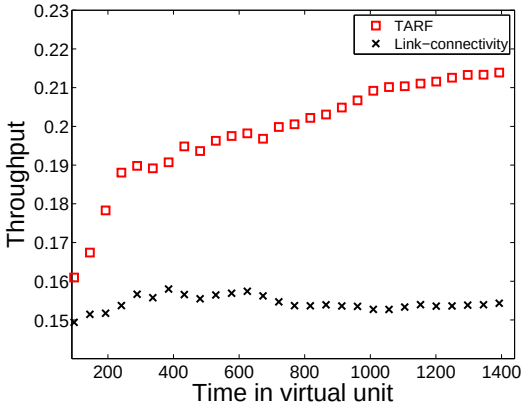
(b)Static location
6 nodes dropping



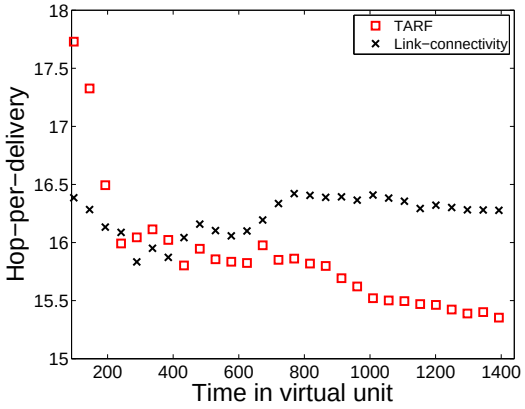
(c) Group motion with noise
6 nodes dropping



(d) Group motion with noise
6 nodes dropping

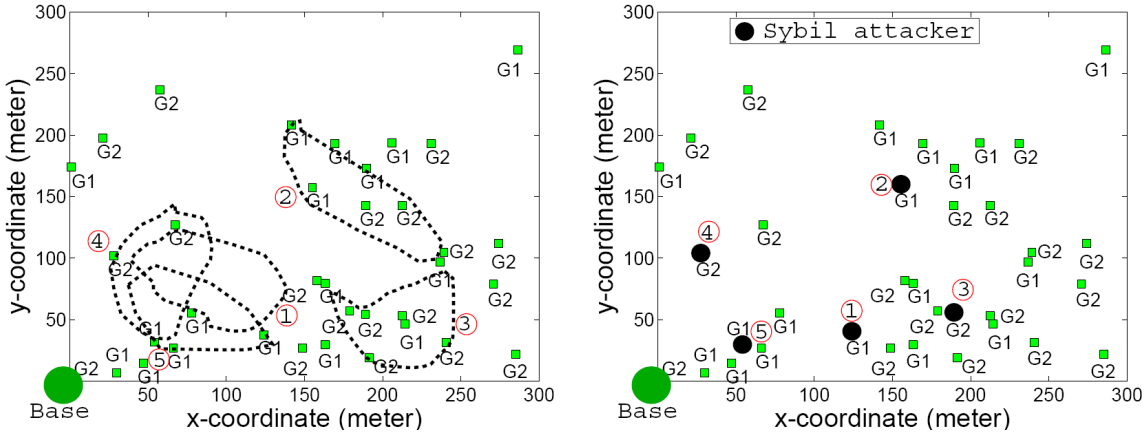


(e) Crossing RF-shield
6 nodes dropping



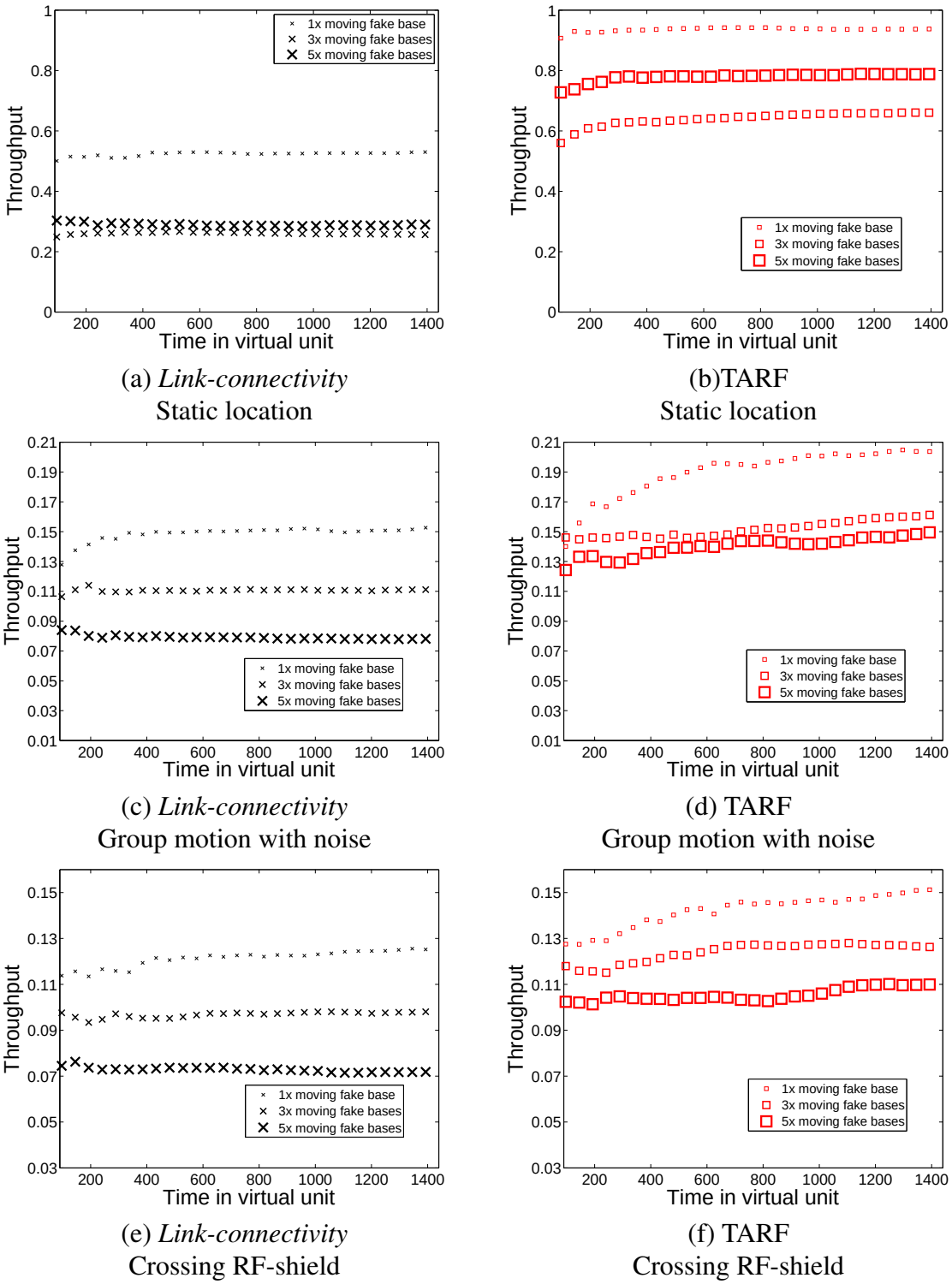
(f) Crossing RF-shield
6 nodes dropping

Figure 5.9: With certain nodes dropping packets, TARF shows higher *throughput* and lower *hop-per-delivery* than the *Link-connectivity* protocol.



(a) Motion trace of 5 fake bases (b) 5 *Sybil* nodes

Figure 5.10: Severe attacks: multiple moving fake base stations and *Sybil* attackers.



(a) *Link-connectivity*
Static location

(b)TARF
Static location

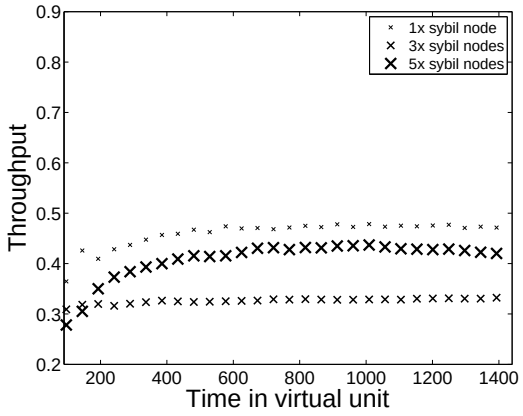
(c) *Link-connectivity*
Group motion with noise

(d) TARF
Group motion with noise

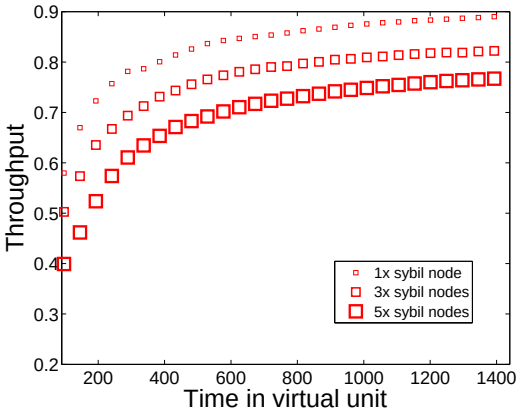
(e) *Link-connectivity*
Crossing RF-shield

(f) TARF
Crossing RF-shield

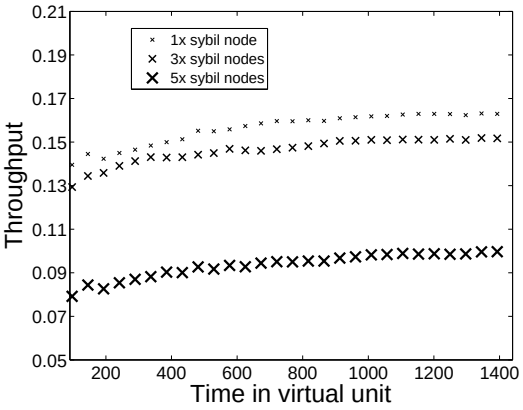
Figure 5.11: With multiple moving fake bases, TARF displays a steady improvement over the *Link-connectivity* protocol in *throughput*.



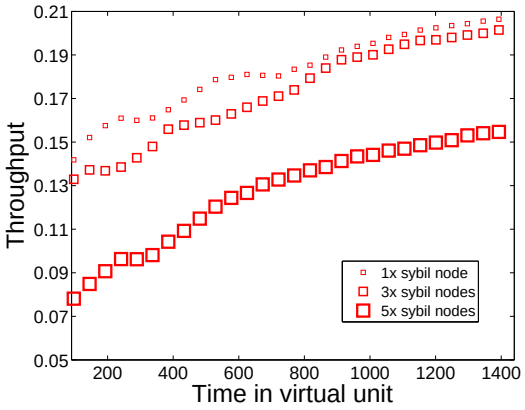
(a) *Link-connectivity*
Static location



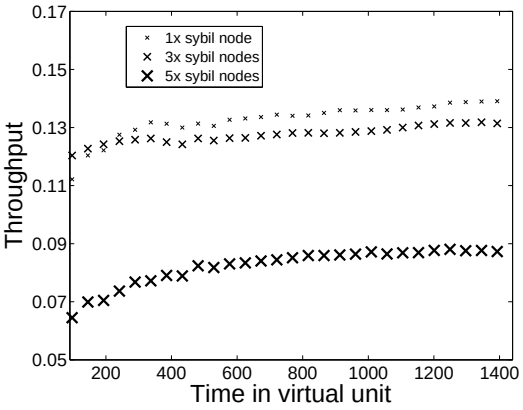
(b)TARF
Static location



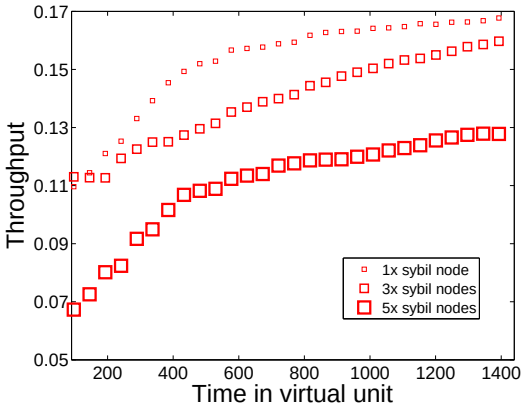
(c) *Link-connectivity*
Group motion with noise



(d) TARF
Group motion with noise

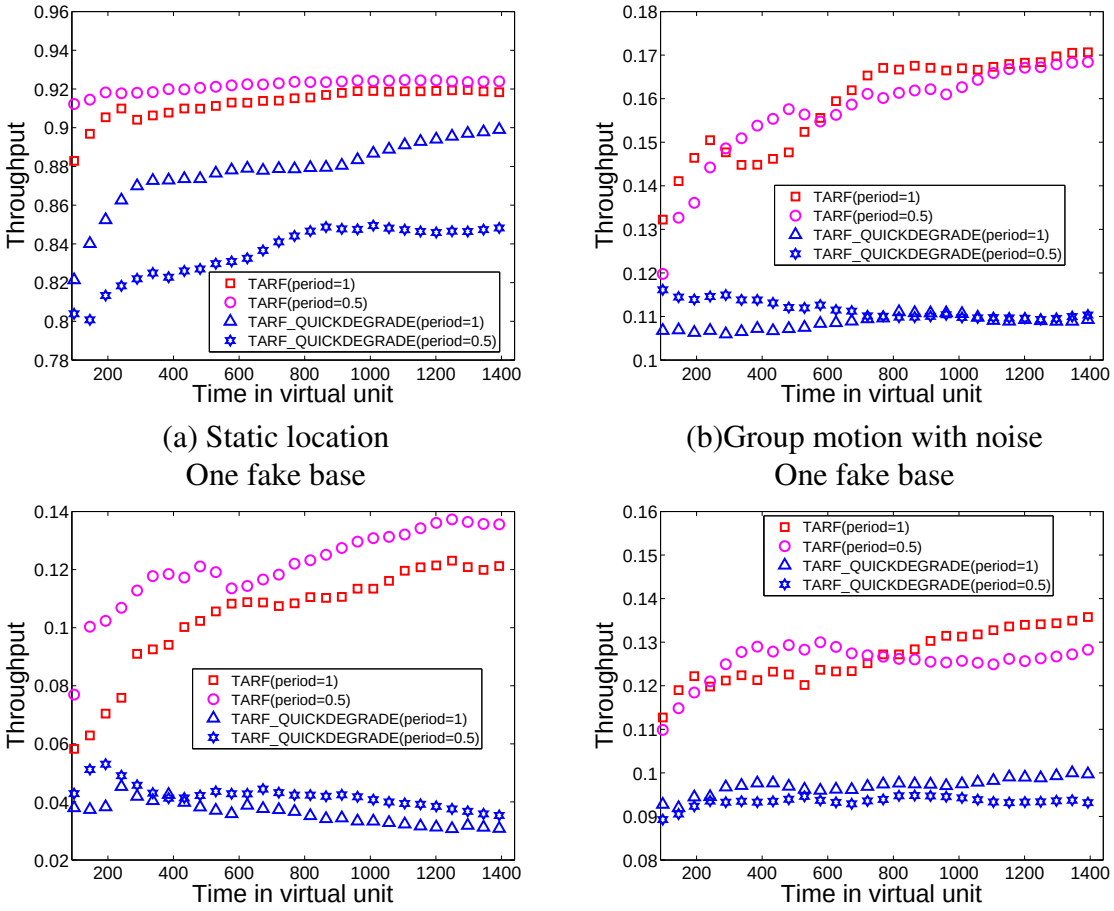


(e) *Link-connectivity*
Crossing RF-shield



(f) TARF
Crossing RF-shield

Figure 5.12: With multiple *Sybil* nodes, TARF demonstrates steady improvement over the *Link-connectivity* in *throughput*.



(a) Static location
One fake base

(b)Group motion with noise
One fake base

(c) Group motion with noise
A network loop of 5 nodes

(d) Crossing RF-shield
One fake base

Figure 5.13: Quicker update or shorter period for TARF does not necessarily improve *throughput*.

```
configuration TrustManagerC {
  provides {
    interface TrustControl;
    interface Record;
  }
  implementation {
    .....
  }
}
```

```
interface TrustControl
{
  //enable trust evaluation
  command error_t start();
  //disable trust evaluation
  command error_t stop();
}
```

```
interface Record
{
  //for a root to add delivered record <source node id, source sequence number>
  command void addDelivered(am_addr_t src, uint8_t seq);
  //for a non-root node to add forwarded record <source id, source sequence, next-hop id>
  command void addForwarded(am_addr_t src, uint8_t seq, am_addr_t next);
  //return the trust level of a node
  command uint16_t getTrust(am_addr_t id);
}
```

Figure 5.14: *TrustManager* component.

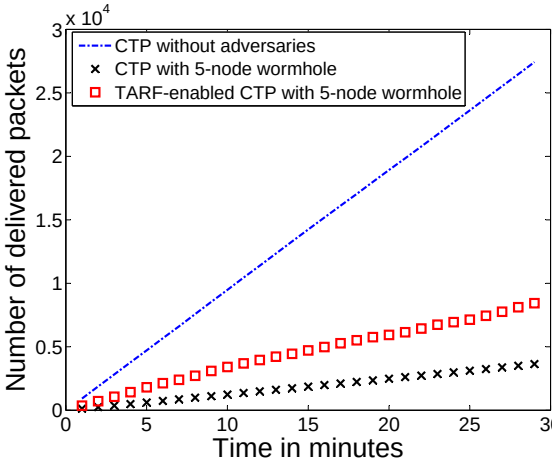
```
event void Boot.booted() {
   ……..
   //enable trust evaluation
   call TrustControl.start();
}
event void SubSend.sendDone(message_t* msg, error_t error) {
   …….....
   //a non-root node records forwarded messages for trust evaluation
   call Record.addForwarded(call CollectionPacket.getOrigin(msg), call
CollectionPacket.getSequenceNumber(msg), call  AMPacket.destination(msg));
   …….....
}
event message_t* SubReceive.receive(message_t* msg, void* payload, uint8_t len) {
   …….....
   //a root records delivered messages for trust evaluation
   call Record.addDelivered(call CollectionPacket.getOrigin(msg), call
CollectionPacket.getSequenceNumber(msg) );
   …….....
}
task void updateRouteTask() {
   …….....
   //retrieve the trust level of each next-hop candidate
   trust=call Record.getTrust(entry->neighbor);
   // integrate trust and the existing protocol's cost metric to decide the optimal next-hop
   …….....
}
```

Figure 5.15: Integration of CTP and TARF (red bigger font).

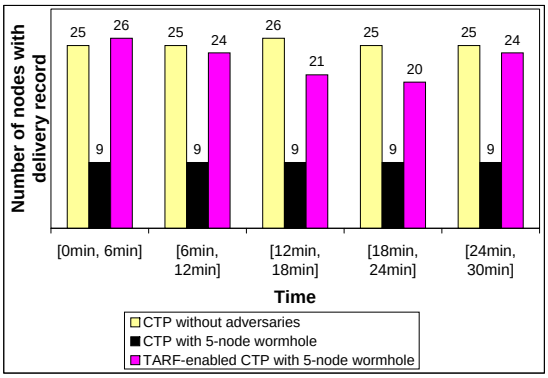
```
//Step 1. traverse the neighborhood table for an optimal candidate for the next hop
optimal_candidate = NULL
//the cost of routing via the optimal candidate provided by the existing protocol, initially infinity
optimal_cost = MAX_COST
//the trust level of the optimal candidate, initially 0
optimal_trust = MIN_TRUST
for each candidate in the neighborhood table
    if link is congested, or may cause a loop, or does not pass quality threshold
        continue
    better =  false
    if candidate.trust >= optimal_trust && candidate.cost < optimal_cost
        better = true
    //prefer trustworthy candidates
    if candidate.trust >= TRUST_THRESHOLD && optimal_trust < TRUST_THRESHOLD
        better = true
    if candidate.trust >= ESSENTIAL_DIFFERENCE_THRESHOLD + optimal_trust
        better = true
    //effective when all nodes have low trust due to network change or poor connectivity
    if candidate.trust >= 3 * optimal_trust / 2
        better = true
    //add restriction of trust level requirement
    if candidate.trust >= TRUST_THRESHOLD && candidate.trust / candidate.cost >
optimal_trust / optimal_cost
        better = true
    if better == true
        optimal_candidate = candidate
        optimal_cost = candidate.cost
        optimal_trust = candidate.trust

//Step 2. decide whether to switch from the current next-hop node to the optimal candidate found:
if optimal_trust >= currentNextHop.trust                                    \
 || currentNextHop.trust <= TRUST_THRESHOLD                                 \
 || current link is congested and switching is not likely to cause loops    \
 || optimal_cost + NEXTHOP_SWITCH_THRESHOLD < currentNextHop.cost   \
    currentNextHop = optimal_candidate
```

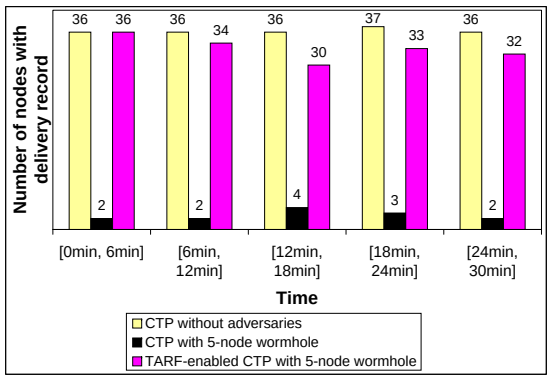
Figure 5.16: Routing decision incorporating trust management.



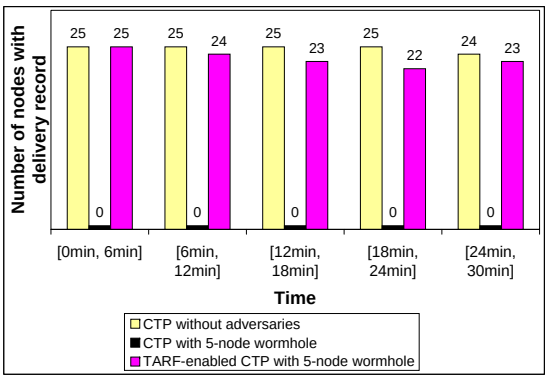
(a) All three floors



(b) First floor

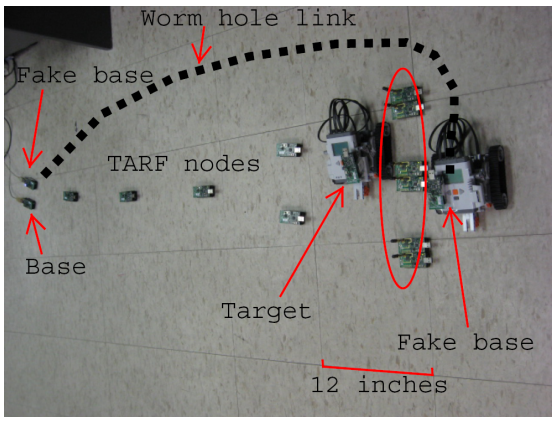


(c) Second floor

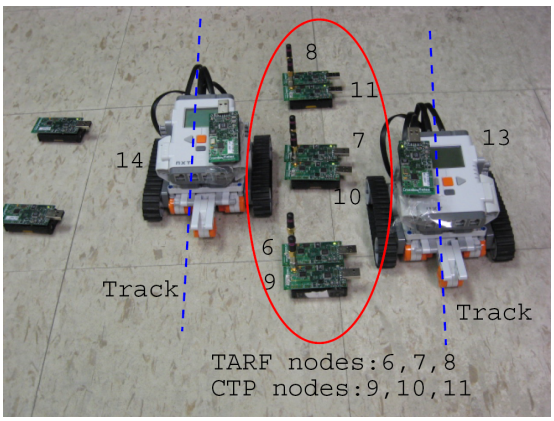


(d) Third floor

Figure 5.17: Empirical comparison of CTP and TARF-enabled CTP on Motelab: (a) number of all delivered data packets since the beginning; number of nodes on (b) the first floor, (c) the second floor and (d) the third floor that delivered at least one data packet in sub-periods.

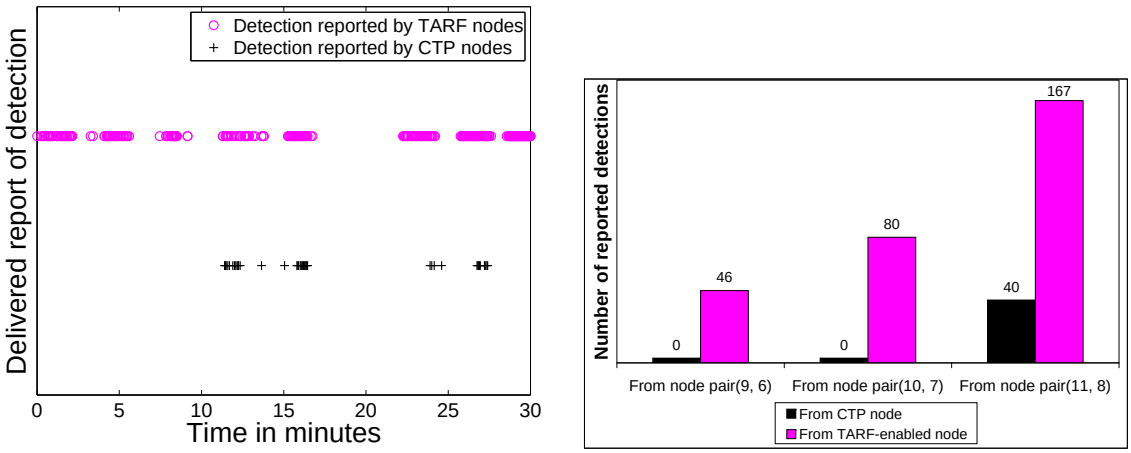


(a) A snapshot of the network.



(b) A closer look.

Figure 5.18: Deployment of a TARF-enabled wireless sensor network to detect a moving target under the umbrella of two fake base stations in a *wormhole*.



(a) Detection report. (b) Number of reported detections.

Figure 5.19: Comparison of CTP and the TARF-enabled CTP in detecting the moving target.

CHAPTER 6

PRIVACY-PRESERVING PARTICIPATORY SENSING SYSTEM

We developed Woodward, a privacy-preserving wireless sensing system. Though it focuses on health care applications, the design principle in privacy protection can be extended to other wireless sensing systems with privacy concern. Woodward protects the user privacy while allowing arbitrary third-party applications to extract knowledge from the collected data. The anonymization process adopted by Woodward causes overwhelming cost to privacy attackers; it also allows arbitrary third-party applications to perform various query with small under-threshold error.

6.1 Introduction

An increasing number of network-enabled computing devices permeate our daily lives. Some typical network-enabled consumer devices include smartphones, PDAs, and in-vehicle infotainment systems. In addition to their network capabilities such as WiFi, GPRS and Bluetooth, these devices are often either equipped with internal sensors such as GPS, accelerometers or able to connect to various external sensors including biomedical sensors. This trend has laid the foundation for participatory sensing, in which daily network-enabled devices, such as smart phones, are used to "form interactive, participatory sensor networks that enable public and professional users to gather, analyze and share local knowledge" [17, 35]. An important category of participatory sensing applications is towards the self-monitoring and self-management of patient health [83, 97]. With the off-the-shelf wireless biomedical sensors, a participatory sensing system will be able to collect biophysical data such as heart rate from the patient and deliver feedback accordingly back to the patient [122]. The data

collection and the feedback delivery are both performed by software through the computer networks. Such applications can lower the medical cost and facilitate remote diagnoses [49].

While participatory sensing can bring great benefit in areas such as health care, there is a rise of concern over privacy leakage [30,64,129,155]. When a user participates in a participatory sensing task, the sensing application could leak his personal information to an adversary. That would greatly discourage the user's involvement. Unfortunately, much existing work on participatory sensing focuses on how to build the software infrastructure to enable applications [18, 113] and generally does not take privacy into consideration. Meanwhile, certain participatory sensing systems [96, 107] tend to limit the use of collected data to internally developed applications only so as to reduce the risk of privacy leakage. The restriction of the internal use of data prevents third-party applications from exploring the data and becomes an obstacle to data sharing. Additionally, the existing privacy research mainly concerns itself about the mechanisms to identify and prevent privacy issues [28, 46, 62] and often does not support arbitrary third-party applications.

To conquer the challenge, with health care as the main focus area, we proposed Woodward, a privacy-preserving participatory sensing system. Woodward protects the user privacy and facilitates the data sharing with the third-party applications. It adopts an innovative anonymization process that allows high-precision query and impedes privacy attacks by overwhelming cost. We implemented Woodward with a health care application and quantitatively evaluated the query precision and privacy protection.

The rest of this chapter is organized as follows: we give an overview of the Woodward system in Section 6.2; the design of the Woodward server is described in Section 6.3; the implementation of Woodward is given in Section 6.4; the quantitative empirical evaluation is in Section 6.5; the summary of this chapter is presented in Section 6.6.

6.2 System Overview

Woodward is a privacy-preserving system to facilitate participatory sensing on network-enabled computing devices. This system allows arbitrary third-party applications to perform various query. We use self-monitoring and self-management of patient health as the main applications to demonstrate the system. As shown in Figure 6.1, in this Woodward system, a user utilizes his network-enabled device (e.g., smartphone) and a few sensors including biomedical sensors to collect healthcare-related data and sends them to a central server - the Woodward server. The sensors are either integrated into the user device or connected wirelessly (e.g., via Bluetooth). The Woodward server stores the data, validates the data, anonymizes the data for privacy protection, and interacts with the user and arbitrary third-party applications. The third-party applications can only access the anonymized data on the Woodward server and can submit health status feedback for a record accessed to the Woodward server. The Woodward server then delivers the feedback to the designated user. For other types of applications rather than health care, the information flow is still the same; only the sensors and the applications are replaced accordingly. Thus, the Woodward system consists of three components (Figure 6.1): the users submitting the data with network-enabled devices and sensors; arbitrary third-party applications; and the core component - the Woodward server, which is trusted by the users. The third-party applications do not retrieve the data from the user directly; instead, all the data are sent to the Woodward server and the applications are only allowed to access the anonymized data from the Woodward server. This requirement is crucial for privacy protection and data reuse. If a third-party application directly accesses a user's original data, the user privacy is hardly guaranteed. Additionally, the third-party application does not directly deliver its generated feedback to a user because the application should not know the user's contact information due to the privacy requirement. Instead, the application submits the feedback for an anonymized record to the Woodward server first; the Woodward server then internally maps that anonymous subject of that feedback onto its true identity and delivers the feedback to the user.

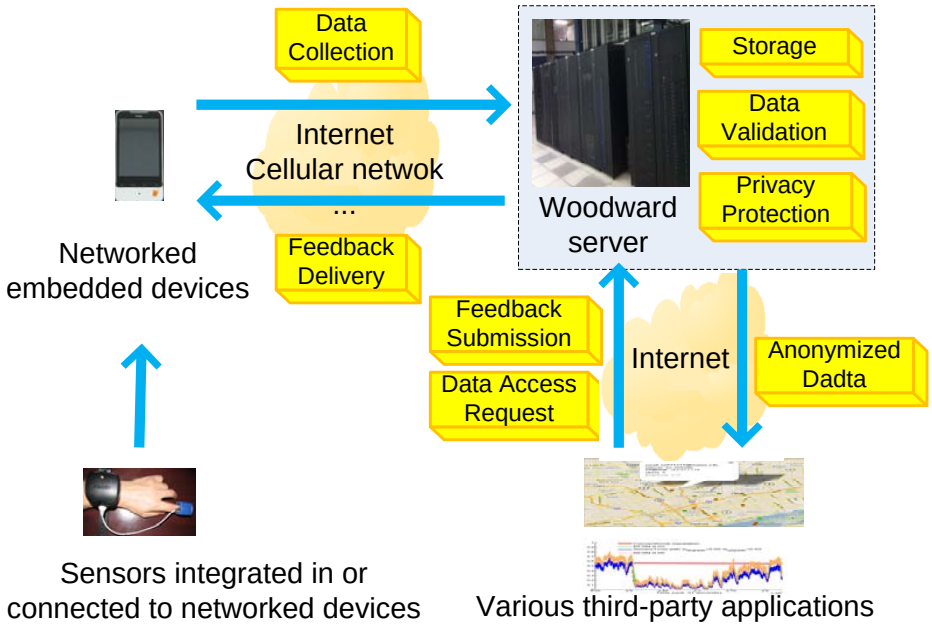


Figure 6.1: The design of Woodward.

The application should be aware that the data have gone through the anonymization process that adds noise to the original data. The anonymization process guarantees that any statistical query, including percentile query of any single value, will be highly precise. A statistical query concerns the statistical features that are based on the probability distribution of the data. Additionally, and importantly, for common values that occur frequently, the noise is small; for values that occur rarely, the noise can be large. Note that on the one hand, rare values, if exposed with only small noise, have a good chance of being traced by attackers with prior knowledge. On the other hand, with small noise added, it is safe to expose common values. The density of a neighborhood of a value can be decided either from a published result from the system or simply from performing a query; the range of the possible noise can also be determined similarly. Generally, most values fall into moderately densely populated areas and the anonymized data closely resemble the original data.

6.3 The Design of the Woodward Server

The Woodward server is designed to store received data, perform data validation, provide flexible application query interface and user feedback, and protect the user privacy. The design also aims to provide high-precision query answers and cause only small performance overhead. Before considering any other issues, we first want to present our approach to protect the user privacy as it closely relates to the rest of the system.

6.3.1 Privacy Protection at the Woodward Server

To protect the user privacy, we first describe the privacy threat model. The Woodward server does not present the true identity or contact information of a user to an application and an attacker will not be able to take advantage of any identity or contact information. However, based on certain prior knowledge about a particular user, the attacker may attempt to identify a certain anonymized record owned by that user. For example, the attacker might happen to know that a user named Alice has an unusually high heart rate of 190 bpm. Then the attacker might search through all the anonymized data exposed by the Woodward server. If the data anonymization is not performed properly and only one record has an unusually high heart rate (though at a different value), the the attacker can decide that that record belongs to Alice. Thus, this attacker just identified an anonymized record about Alice. If that same record also contains other information (e.g., age) the attacker is interested in, the attacker could access that information and start harmful activities against Alice. Therefore, under this privacy threat model, with prior knowledge of an attribute (or multiple attributes) about a user, an attacker attempts to link at least one anonymized record to that user and exploit that record for other private information of the user. For simplicity, the current design of the anonymization process assumes that an attacker only has the prior knowledge of a single attribute of a user. But the approach can be generalized to handle attacks based on prior knowledge of multiple attributes.

The Woodward server performs the anonymization process on the received data; and the application query is executed against the anonymized data. The anonymization uses different schemes according to the types of the data. Our major interest here is the numeric biophysical data of the user (e.g., heart rate). We will first describe the anonymization for other types of data. For identity information such as the name and email address, the server maintains a secret one-to-one mapping that maps each identity into a unique meaningless symbol (e.g., a byte string). The mapping is maintained in such a way that it is impossible for a third-party to reverse the map to find out the original identity corresponding to an anonymized symbol. The reason that an identity symbol is still needed is that in a conventional entity-relationship database query often needs to know if two records are associated with the same user or not. For discrete attributes with only a very small number of possible values, the value is generally directly exposed to the applications unless the user indicates that the data should not be exposed; in the latter case, an "unknown" value will replace the original value for the application query. For an attribute that can only have a very small number of possible values, there can be a large number of users having each same value; thus, that information is usually not sensitive and Woodward usually directly exposes a discrete attribute with a very small set of values unless specified otherwise by the users. For text or binary data, the data are either completely hidden from the query or exposed to the application query, as specified by the user. Regarding location data, the Woodward server anonymizes the exact location to a city-magnitude area. Though it is possible to exploit the existing approaches for location anonymization [172], for our purpose with health care information, we are satisfied with this simple scheme.

For the numeric biophysical data of the user, we need to take extra care to perform anonymization. These data are greatly valued by many applications. But exposing them directly (even without any explicit identity information) can be exploited by attackers with certain prior knowledge, as described in our privacy threat model. The Woodward server

adds noise to the biophysical data and presents the anonymized data to third-party applications. The anonymization process has a few requirements. First of all, the anonymization should keep a transformed value as close to its original value as possible while protecting the privacy; and the overall statistical features of the original and the anonymized data should be very close to each other. The condition helps maintain the precision of a general statistical value of the form $\int_S g(x) \cdot f(x)dx$, where x is the random variable of our concern (e.g., heart rate), $f(x)$ is the probability density function of x, $g(x)$ is the random numeric function the application is interested in, and S is the measurable set on which the integration is applied. One indication of this condition is that the probability density function of the anonymized data will be very close to that of the original data. This condition also indicates that the change of $g(x)$ is limited after anonymization. Theoretically, the well-known Schwarz's Inequality [69] helps establish a loose upper bound on the deviation due to anonymization:

$$|\int_S \psi_1(x) \cdot \psi_2(x)dx| <= \sqrt{\int_S \psi_1(x)^2 dx} \cdot \sqrt{\int_S \psi_2(x)^2 dx}$$

To see this, let $\widetilde{f}(x)$ be the probability density function for the anonymized data. Then the deviation of the statistic is

$$\begin{aligned} & |\int_S g(x) \cdot \widetilde{f}(x)dx - \int_S g(x) \cdot f(x)dx| \\ = \; & |\int_S g(x) \cdot (\widetilde{f}(x) - f(x))dx| \\ <= \; & \sqrt{\int_S g(x)^2 dx} \cdot \sqrt{\int_S (\widetilde{f}(x) - f(x))^2 dx} \end{aligned}$$

Second, note that the value with a dense neighborhood will only need low random noise for the anonymization purpose. Within a densely populated area, a small noise can easily confuse the attacker with many records of similar values. Last, for a value within a sparse neighborhood, a random noise of moderate size should be considered. To frustrate the attacker, such

a random noise must fulfill the requirement that there should be at least a moderate number of other data values that can possibly be anonymized to be the same anonymized value.

Figure 6.2 and Figure 6.3 visualize the idea. 10000 random values (original data) are generated between 50 and 220, based on the normal distribution. The anonymized process is applied to get the anonymized data. Figure 6.2 shows the histograms of the original data and the anonymized data. As indicated by Figure 6.2, the original and the anonymized data show

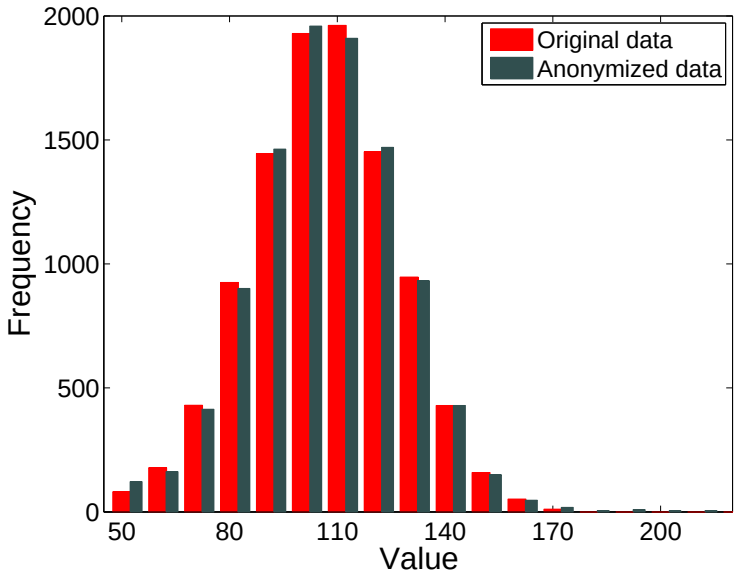


Figure 6.2: Histograms of original and anonymized data with normal distribution.

similar frequency distribution. Figure 6.3 compares the original data and the anonymized data more closely. For each pair of (original value, anonymized value), a point is plotted. For data falling into the intense interval [80, 140] (Figure 6.3), the anonymized values show very limited deviation from their original values. By contrast, for the sparse data out of that interval, the difference between the original and the anonymized values can be as large as 60 (Figure 6.3).

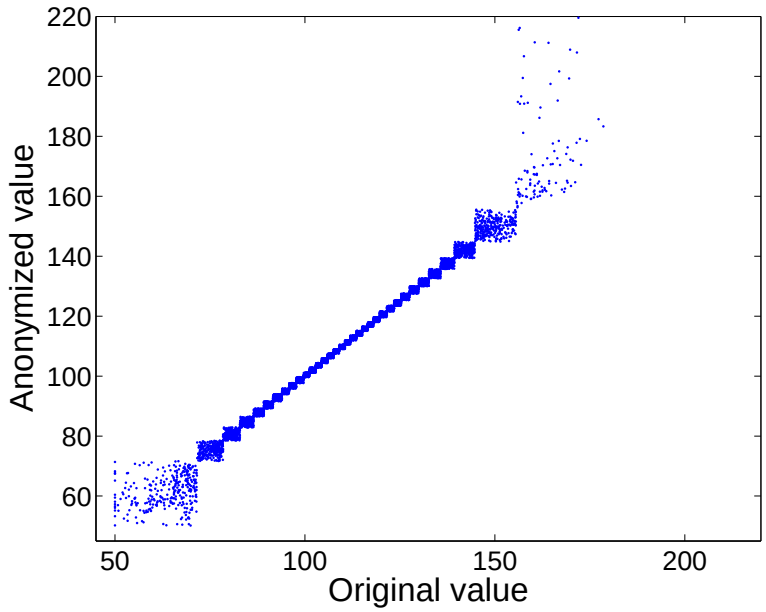


Figure 6.3: Comparison of anonymized data and original data with normal distribution.

6.3.2 Anonymization on Numeric Biophysical Data

Now, we describe our anonymization algorithm for the numeric biophysical data. Assume the size of the database has been large enough; otherwise, a query against a database with a few records will not be permitted due to privacy concern. The algorithm first divides the range of the numeric data into a series of contiguous neighborhood in the form of open or half-open intervals $(-\infty, I_0), [I_0, I_1), [I_1, I_2), [I_2, I_3), ..., [I_n, +\infty)$. Each interval contains a similar number of data values occurrence, with a value of multiple occurrence counted multiple times; the exceptions happen around those values that occur more than a few times. we denote that common relative frequency of almost all the neighborhoods (intervals) as *RFNBH*, i.e., the ratio of the data occurrence in that neighborhood to the total data occurrence. Thus, a neighborhood of a small size is denser than a neihgborhood of a greater size. The larger a neighborhood is, the sparser it is. After the neighborhood division, the anonymization process is applied to each existing value and incoming value according to the neighborhood they fall in. Each anonymized value should still fall in the same neighborhood as its original

neighborhood; it indicates that the difference of cumulative relative frequency between the original and the anonymized data should not exceed *RFNBH* - the common relative frequency of a neighborhood. In other words, the difference between the statistical distribution of the original and the anonymized data is dominated by *RFNBH*. The magnitude of the random noise applied is decided in a way that is on average proportional to the neighborhood size. In other words, the sparser the neighborhood is, the higher noise is likely to be applied to the data values there. Importantly, the randomness of the noise applied discourages an attacker by presenting a whole neighborhood of candidate values to a malicious query targeting a particular user. There is a trade-off on the value of *RFNBH*: the smaller *RFNBH* is, the better query precision the anonymization maintains and the more privacy risk there is. A specific algorithm is given in Algorithm 1, with subprocedures in Function 2 and Function 3. Table 6.1 summarizes the notations and samples parameters used.

Table 6.1: Major notations and parameters used with sample values in parentheses.

Symbol	Meaning
X	An attribute.
x	An existing or incoming data value of attribute X.
$\widetilde{x}$	An anonymized value of x.
LBOUND(50)	the lower bound of the attribute value.
UBOUND(220)	the upper bound of the attribute value.
NbhList	List of neighborhoods covering the range of attribute X.
TOTAL	Total frequency of data occurrences.
RFNBH(0.03)	The common relative frequency of the neighborhoods
NZRT(0.5)	Ratio of noise magnitude over neighborhood size.
NZCF(0.7)	The confidence that noise falls into a major interval.
NZTH(10)	Maximal noise threshold.

6.3.3 Other Design Aspects

The Woodward server stores the anonymized data and allows an arbitrary third-party application to request database query on the anonymized database. The way how an application can

Algorithm 1 For a numeric-valued attribute *X* in a database, anonymize its values.

procedure ANONYMIZE(attribute *X*)
  NbhList=NEIGHBORHOODDIVISION(*X*);
  for each existing/incoming value x of attribute *X* **do**
    identify its neighborhood $[I_{left}, I_{right})$ in *NbhList*;
    store $\widetilde{x}$=ANONYMIZE(x, $[I_{left}, I_{right})$);
  end for
end procedure

Function 2 Divide the attribute range into a series of contiguous neighborhood and return the neighborhood list.

function NEIGHBORHOODDIVISION(attribute *X*)
  NbhList={} ▷ List of neighborhoods
  $I_{right} = I_{left} =$ *LBOUND*;
  while $I_{left} <$ *UBOUND* **do**
    NumOfData $= 0$;
    while *NumOfData*$<$ *RFNBH* $*$ *TOTAL* AND $I_{right} <$ *UBOUND* **do**
      NumOfData += occurrence frequency of I_{right};
      $I_{right} = min\{$*UBOUND*$, x | x > I_{right}\}$;
    end while
    if $I_{right} <$ *UBOUND* **then**
      Add $[I_{left}, I_{right})$ onto *NbhList*;
    else
      Add $[I_{left}, I_{right}]$ onto *NbhList*;
    end if
    $I_{left} = I_{right}$;
  end while
  return *NbhList*;
end function

Function 3 Return the anonymized value of x within its neighborhood $[I_{left}, I_{right})$.

function ANONYMIZE(Data x, $[I_{left}, I_{right})$)
  ALeft $= max(x -$ *NZTH*$, x -$ *NZRT* $* (x - I_{left}))$;
  ARight $= min(x +$ *NZTH*$, x +$ *NZRT* $* (I_{right} - x))$;
  XFR = relative occurrence frequency of x;
  ALeft $= x - (x -$ *ALeft*$)/(100 *$ *XFR* $+ 1)$;
  ARight $= x + ($*ARight* $- x)/(100 *$ *XFR* $+ 1)$;
  randomly generate $\widetilde{x}$ with probabilistic distribution: *NZCF* in $[$*ALeft*, *ARight*$)$; 1-*NZCF* in $[I_{left},$ *ALeft*$)$ and $[$*ARight*$, I_{right})$;
  return $\widetilde{x}$;
end function

send its query request mimics the way a user accesses a regular remote database: besides typical database privilege authorization, no other restriction applies. This gives the application the maximal freedom.

Regarding the data storage, the Woodward server stores the original and the anonymized data onto separate storage units. For any new data, an online anonymization is performed; the original and the anonymized data are then stored separately. For large data, we may use RAID or even a cluster database with a shared-nothing structure.

For data submitted from an anonymous user, a validation process is performed to protect the database from pollution by erroneous data. To detect an abnormal value, the value is checked against the statistical distribution of the existing data. Further, we can take advantage of the source reputation for data validation. The specific data validation approach can be found in our previous work [166].

6.4 Implementation

We implemented a complete Woodward system, with health care as the application area. It includes a user component, a server component and an application component.

We developed a user client program on a HTC Legend Android phone for data collection and feedback retrieval. As illustrated by Figure 6.4, the Android client program reads the heart rate wirelessly from a Nonin's Bluetooth-enabled sensor Avant 4100 worn around the user's wrist. In addition to the heart rate, the Android client program also collects location data with the internal GPS and get user input for a questionnaire about the user information such as age and email. All the collected data are sent to the Woodward server via WiFi. The same Android client program also displays the feedback generated by applications once it is available. The client exchanges messages with the Woodward server using XML. For the sake of security, all the network communication is protected by Secure Socket Layer (SSL).

The Woodward server we developed stores data sent from the user with MySQL. An original copy and an anonymized copy are stored in separate databases. The anonymization

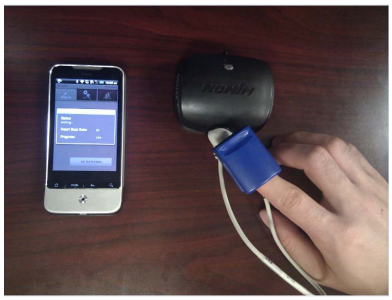


Figure 6.4: The client program on an Android phone.

on a numeric attribute is performed according to Algorithm 1. When the server starts, it first prepares for the anonymization by performing the neighborhood division. Then, whenever a new records arrives, the server performs the online anonymization based on the outcome of the neighborhood division, with a small overhead. The server maintains a secret one-to-one map between all true user names and their anonymized names. The anonymized names are generated from a secure random string generator that guarantees universal uniqueness. The server allows any third-party application to perform read-only access to the anonymized copy with SQL. It also accepts the feedback an application generates towards a user and delivers the feedback to the user. The third-party application is not allowed to directly access the original copy. For the feedback, the application specifies the anonymized name of the user and the server maps that to the true user.

We created a sample third-party application that informs certain users of potential risk of cardiovascular disease according to their heart rate readings. Specifically, whenever a user is found to have a high heart rate of at least 97% percentile among the group of user similar to his age [72], the application submits such feedback to the Woodward server targeting that user: "Your heart rate appears to be considerably higher than your peers. That reveals a certain risk of cardiovascular diseases. If you are interested in more details or need subscription service, please contact our eHealth group at xxx-xxx-xxxx."

6.5 Evaluation of Privacy Protection and Query Accuracy

For the anonymization process applied to a numeric biophysical attribute, we evaluated the effectiveness of privacy protection and query accuracy. We generated a series of random heart rate readings between 50bpm and 220bpm and applied the anonymization process according to Algorithm 1 and the parameters from Table 6.1. The original random data were generated based on one of the following seven statistical distributions: uniform distribution, binomial distribution, normal distribution, Poisson distribution, chi-Squared distribution, Weibull distribution, and exponential distribution. For each distribution, 10,000 random values were generated as a complete set. Table 6.2 summarizes the statistical characteristics of the original random data, including the value range, the average, and the standard deviation. Table 6.3 summarizes the same statistics for the corresponding anonymized data. Regarding the average and the standard deviation, there are very limited differences between the original and the anonymized data. Additionally, except for the uniform distribution, the anonymization tends to enlarge the range of the data by various sizes. To explain that, note the data are sparsely distributed at either the left or the right end of the original range, except for the uniform distribution. According to the anonymization process, the sparse areas tend to get larger noises. The larger noises at either end of the original range result in the enlarged range of the anonymized data. That effect is visualized in Figure 6.3: the further a point is away from the diagonal line in the figure, the larger deviation there is.

Table 6.2: Original random data.

Random data	Range	Average	Std Dev
Uniform distribution	50.02—220.00	135.26	49.12
Binomial distribution	82.00—136.00	110.10	7.39
Normal distribution	50.00—178.56	109.91	20.02
Poisson distribution	85.00—139.00	110.01	7.73
Chi-Squared distribution	51.26—210.28	85.04	18.89
Weibull distribution	55.52—154.27	104.44	15.06
Exponential distribution	50.00—180.93	62.10	12.00

Table 6.3: Anonymized data.

Anonymized data	Range	Average	Std Dev
Uniform distribution	50.00—219.95	135.30	49.15
Binomial distribution	50.08—219.12	110.29	9.89
Normal distribution	50.15—219.56	109.96	20.58
Poisson distribution	50.18—215.59	109.93	9.36
Chi-Squared distribution	50.05—219.82	85.19	19.71
Weibull distribution	50.10—219.98	104.45	16.16
Exponential distribution	50.00—213.00	62.35	13.36

Despite the differences in the range, overall, the empirical distributions of the original data and the anonymized data show very limited differences. We have seen the small differences of the frequency histograms for the normal distribution in Figure 6.2. Additionally, their empirical cumulative distribution displays an almost perfect match.

The noise (i.e., the difference between an original value and its anonymized value) can vary, from a small scale to a very large scale. But on average, the noise tends to be small. Table 6.4 summarizes the magnitude of the noise. Though the noise can range from 0 to 108, the average noise is no more than 1.4, with its standard deviation less than 5.

Table 6.4: Noise magnitude

Noise magnitude	Range	Average	Std Dev
Uniform distribution	0.00—5.54	1.35	1.10
Binomial distribution	0.00—90.66	0.83	4.75
Normal distribution	0.00—59.63	0.95	2.33
Poisson distribution	0.00—79.57	0.71	3.50
Chi-Squared distribution	0.00—66.17	0.90	2.77
Weibull distribution	0.00—78.49	0.88	3.18
Exponential distribution	0.00—108.13	0.63	3.56

The noise is closely related to the size of the division interval that the original value falls in. Roughly, the noise tends to be small for short intervals and greater for longer intervals. Table 6.5 summarizes the length of the division intervals. The length can vary from 0.3 to 116, corresponding to the various densities. The longer the division interval is, the sparser

the neighborhood is. The noise, on average, is roughly proportional to the length the division interval. Table 6.6 summarizes the ratio of noise magnitude to division interval length. Though that ratio can vary from 0 to 1, generally, its average is from 0.18 to 0.27.

Table 6.5: Length of division intervals used for anonymization

Interval length	Range	Average	Std Dev
Uniform distribution	1.66—5.84	5.00	0.68
Binomial distribution	1.00—93.00	7.08	20.54
Normal distribution	1.43—64.42	5.00	11.15
Poisson distribution	1.00—88.00	7.39	19.99
Chi-Squared distribution	1.11—76.56	5.00	13.10
Weibull distribution	1.09—83.28	5.00	14.47
Exponential distribution	0.34—115.97	5.00	19.80

Table 6.6: Ratio of noise magnitude to division interval length

Noise/division interval	Range	Average	Std Dev
Uniform distribution	0.0000125—0.99	0.27	0.22
Binomial distribution	0.0000038—1.00	0.19	0.26
Normal distribution	0.0000141—0.99	0.26	0.21
Poisson distribution	0.0000036—1.00	0.18	0.25
Chi-Squared distribution	0.0000666—1.00	0.26	0.21
Weibull distribution	0.0000171—0.99	0.27	0.21
Exponential distribution	0.0000066—0.99	0.26	0.21

The percentile query shows that it is moderately accurate to use the anonymized data for estimating the percentile of an original value. Roughly, the percentile difference should have 100**RFNBH* as its upper threshold, where *RFNBH* is the common relative frequency of the neighborhoods. That corresponds to how the anonymization process divides the intervals and adds noise. In the meantime, for discrete-valued numeric data (i.e., integers), the percentile difference may exceed 100**RFNBH* because of the biased noise introduced by the discreteness. If it is allowed to use non-discrete anonymized values, we may well control the percentile difference under that upper threshold with the following anonaymization process:

first apply tiny random noise to the original data, then apply the original anonymization process to the data which has absorbed the tiny noise. The reason of applying tiny noise first is to break the clustering of discrete values and facilitate the splitting of the domain into division intervals (discrete values tend to cluster onto a few values). Table 6.7 lists the percentile rank difference between the original data and the anonymized data for each distribution. Except for the two discrete distribution (binomial distribution and Poisson distribution), the data of all other distribution has a percentile rank difference between -3 and 3, which matches 100**RFNBH* (*RFNBH*=0.03). Additionally, the latter has a 0 difference on average.

Table 6.7: Percentile query accuracy

Percentile rank difference	Range	Average	Std Dev
Uniform distribution	-3—3	0.00	1.06
Binomial distribution	-6—4	-1.90	1.49
Normal distribution	-3—3	-0.00	1.05
Poisson distribution	-7—4	-1.83	1.53
Chi-Squared distribution	-3—3	-0.00	1.06
Weibull distribution	-3—3	-0.00	1.07
Exponential distribution	-3—3	-0.00	1.06

Finally, the experiments show that our anonymization process highly protects the user privacy and discourages an attacker by the anonymized data. To quantify the efforts that the attacker needs to maliciously identify a user, for each numeric record, we define the attack cost as the number of records that falls between the original value and the anonymized value. Intuitively, the attack cost reflects the minimum number of records to check starting with the original value and before coming across the anonymized value. The attacker with certain prior knowledge on a certain user would have to examine through at least all those records before finding out the corresponding anonymized record. This attack cost is the minimum cost that impedes the privacy attack and thus a very conservative estimation. The actual cost can be much higher since an attacker can never be sure of the exact number of records falling between the original value and the anonymized value. The higher the attack cost is, the better

our anonymization process protects the privacy. The attack cost for each original value can vary and is independent of the division interval it falls into. The attack cost roughly reflects the frequency of data falling into the corresponding division interval. Figure 6.5 illustrates the attack cost for each value from a normal-distributed data set with 10,000 records. Visually, the attack cost is independent of where the value lies. A value around the left end (50) can have as a high attack cost as a value in the middle. Table 6.8 summarizes the statistics of the attack cost for each distribution. Not only does the attack cost vary a lot, it also has a high value (86–197) on average. Figure 6.6 illustrates the empirical cumulative distribution of the attack cost for the normal distribution data. The figure reveals: with a likelihood of 60%, the attack cost is at least 50; with a likelihood of 33%, the attack cost is at least 100; with a likelihood of 8%, the attack cost is at least 200.

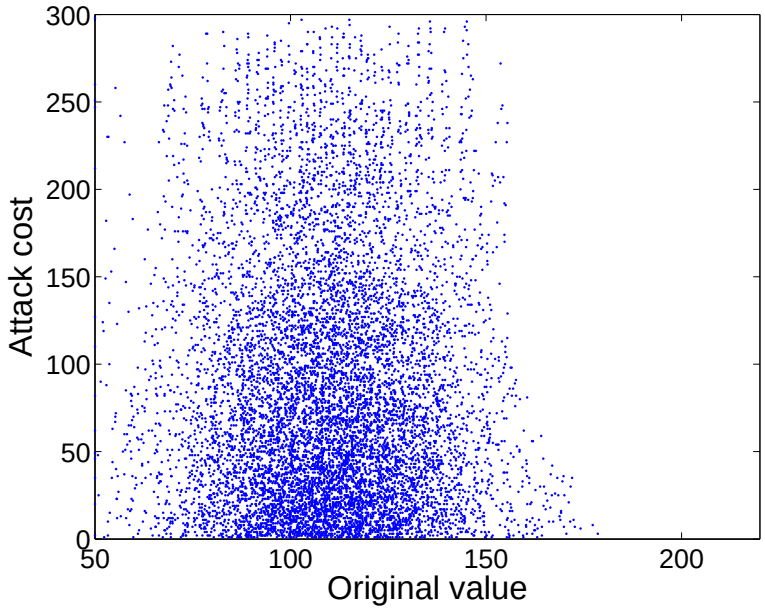


Figure 6.5: Attack cost of identifying record from anonymized normal-distributed data with prior knowledge.

Table 6.8: Attack cost of identifying record from anonymized data with prior knowledge

Attack cost	Range	Average	Std Dev
Uniform distribution	1—300	87.31	70.80
Binomial distribution	1—530	196.69	140.62
Normal distribution	1—298	86.45	70.32
Poisson distribution	1—671	189.58	141.51
Chi-Squared distribution	1—300	87.67	70.22
Weibull distribution	1—299	88.95	71.37
Exponential distribution	1—299	87.26	70.99

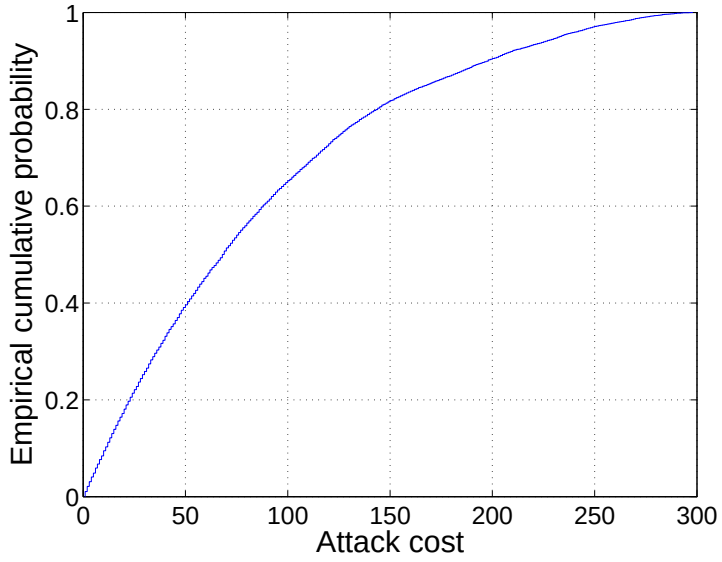


Figure 6.6: Empirical CDF for attack cost of identifying record from normal-distributed anonymized data with prior knowledge.

6.6 Summary

We proposed Woodward, a privacy-preserving participatory sensing system, focusing on health care applications. Unlike the existing participatory sensing systems, Woodward protects the user privacy while supplying the anonymized data to arbitrary third-party applications. The innovative anonymization process adopted by Woodward causes overwhelming cost to privacy attackers; it also allows arbitrary third-party applications to perform various query with small under-threshold error. These features are not achievable by the existing

privacy protection schemes. We implemented Woodward with a health care application and evaluated the query precision and privacy protection quantitatively. In the future, we plan to generalize the anonymization process to multi-dimensional data so as to further impede the privacy attacks exploiting prior knowledge of multiple attributes. Additionally, we will develop versatile applications based on Woodward.

CHAPTER 7

CONCLUSIONS AND FUTURE DIRECTIONS

7.1 Conclusions

This dissertation studied how to provide system support for robust data collection in wireless sensing systems through addressing a few urgent design issues in existing systems. A wireless sensing system may suffer issues arising at the sensors, during the data transmission, and during the data access by applications. While wireless sensing systems may resemble conventional networked systems in many ways, their unique characteristics determine that certain conventional solutions for networked systems may not work well. With certain typical system structures, we have developed approaches to resolve those few urgent problems in the design of wireless sensing systems. Similar ideas to our approaches can be employed to address the issues in more generic settings.

First, we developed a resilient trust model, *SensorTrust*, to effectively detect faulty data in wireless sensing systems due to either sensor malfunctioning or malicious attempts to report false data. *SensorTrust* evaluates the trustworthiness of the collected data in wireless sensing systems. While this model is mainly proposed for a certain common architecture of wireless sensing systems (hierarchical WSNs) , this approach can be generalized to detect data trustworthiness in a more generic setting. In this model, an aggregator maintains trust estimations for its children nodes. With this model, past history and recent risk are synthesized in a real-time way that accurately identifies the current trust level. Our model employs the Gaussian model to rate data integrity in a fine-grained style, and a flexible update protocol to adapt to different applications. With acceptable overhead, the *SensorTrust* model was evaluated with the real world sensor data from Intel Berkeley Lab and Motelab at Harvard University,

and compared with other approaches. The results indicate great advantage of *SensorTrust* to handle varied faults and attacks.

Then, we developed a low-cost, self-contained, accurate localization system (LOBOT) for small-sized ground robotic vehicles. This localization system enhances the wireless sensing systems containing mobile sensors by providing more accurate and highly available location data, with only limited overhead in economic cost and management. LOBOT localizes a robotic vehicle with a hybrid approach consisting of infrequent absolute positioning through a GPS receiver and local relative positioning based on a 3D accelerometer, a magnetic field sensor and several motor rotation sensors. LOBOT fuses the information from an accelerometer, a magnetic sensor and motor rotation sensors to infer the movement of the robot through a short time period; then the inferred movement is corrected with infrequent GPS-augmentation. The hardware devices LOBOT uses are easily-available at low cost. LOBOT is self-contained in that it virtually requires no external devices or external facility management and that it needs no prior information. Unlike other localization schemes such as radio-based solutions, LOBOT does not require external reference facilities, expensive hardware, careful tuning or strict calibration. Additionally, LOBOT applies to both indoor and outdoor environments and realizes satisfactory performance. We developed a prototype of LOBOT and conducted extensive field experiments. The empirical experiments of various temporal and spatial scales with LOBOT verified its accuracy. In contrast to the accelerometer-based approach, LOBOT succeeds in maintaining low cumulative error. The GPS-augmentation greatly enhances LOBOT's resilience.

Additionally, we designed and implemented TARF, a robust trust-aware routing framework, to secure multi-hop routing through a set of sensors (WSNs) in wireless sensing systems. Though it is motivated by harmful attackers exploiting the replay of routing information, TARF can also be used to protect the routing layer from other attacks. TARF focuses on trustworthiness and energy efficiency, which are vital to the survival of a WSN in a hostile environment. With the idea of trust management, TARF enables a node to keep track of the

trustworthiness of its neighbors and thus to select a reliable route. Unlike previous efforts at secure routing for WSNs, TARF effectively protects WSNs from severe attacks through replaying routing information; it requires neither tight time synchronization nor known geographic information. The resilience and scalability of TARF were proved through both extensive simulation and empirical evaluation with large-scale WSNs; the evaluation involved both static and mobile settings, hostile network conditions, as well as strong attacks such as *wormhole* attacks and *Sybil* attacks. We implemented a ready-to-use TinyOS module of TARF with low overhead; this TARF module can be integrated into existing routing protocols with moderate effort, thus producing secure and efficient fully-functional protocols. Additionally, we demonstrated a proof-of-concept mobile target detection application that was built on top of TARF and was resilient in the presence of an anti-detection mechanism; that indicates the potential of TARF in WSN applications.

Finally, we developed Woodward, a privacy-preserving wireless sensing system. Though it focuses on health care applications, the design principle in privacy protection can be extended to other wireless sensing systems with privacy concern. Unlike the existing wireless sensing systems, Woodward protects the user privacy while allowing arbitrary third-party applications to extract knowledge from the collected data. The anonymization process adopted by Woodward causes overwhelming cost to privacy attackers; it also allows arbitrary third-party applications to perform various query with small under-threshold error. These features are not achievable by the existing privacy protection schemes. We implemented Woodward with a health care application and evaluated the query precision and privacy protection quantitatively.

7.2 Future Directions

As the future directions, it will be beneficial to study how to extend our approaches to systematically establish robust data collection for wireless sensing systems.

First, the data trustworthiness modeling (*SensorTrust*) in this dissertation is mainly based on the assumption that the data from one sensor should be consistent with the data from certain other sensors. Such consistency requirement can be relaxed to be extended to more generic settings. As one extension, we may consider the consistency among multiple attributes detected by the sensors. It is likely that one attribute is related to another attribute. If inconsistency is detected among multiple attributes, then the trustworthiness can be downgraded. We may also consider the temporal consistency within the data from a single sensor. The temporal evolvement of an attribute reading detected by a sensor often displays certain patterns, depending on the characteristics of the attribute. Drastic changes in the readings over a short time often indicate potential sensor malfunction.

Second, the localization of small-sized ground robotic vehicles (LOBOT) can be improved in a few ways. Certain investigation can be conducted to evaluate how other probabilistic inference algorithms such as Kalman filter can help reduce the errors from drifting. Slippage of the vehicles should also be considered; there are a few existing projects that help resolve the slippage issue. Additionally, the impact of the magnetic interference on the accuracy of localization can be studied. Further, empirical analysis can be conducted on the energy overhead of the localization approaches utilizing various sensors.

Third, the trust-aware routing framework we proposed for wireless sensor networks (TARF) can be applied to other attacks targeting the routing layer. The underlying mechanism of our proposed approach does not assume that only the attacks exploiting the replay of the routing information exist. It can also defend against other attacks towards the routing layer preventing packet delivery. Empirical experiments will be needed to verify the performance. Meanwhile, we did not consider the MAC layer attacks. Existing approaches normally address those attacks through a combination of defense on both the software and hardware sides; pure software approaches may not work well.

Last, the anonymization we proposed for the Woodward system can be further improved in a few ways. The existing research in protecting privacy of location information can be

incorporated into Woodward. It is also possible to generalize the anonymization process to multi-dimensional data so as to further impede the privacy attacks exploiting prior knowledge of multiple attributes. Additionally, we would like to encourage the development of versatile applications based on Woodward.

BIBLIOGRAPHY

[1] J. Al-Karaki and A. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications*, 11(6):6–28, Dec. 2004.

[2] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7:275–286, October 2003.

[3] Aline Baggio and Koen Langendoen. Monte carlo localization for mobile wireless sensor networks. *Ad Hoc Netw.*, 6:718–733, July 2008.

[4] P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775 – 784 vol.2, 2000.

[5] L. Bai, F. Ferrese, K. Ploskina, and S. Biswas. Performance analysis of mobile agent-based wireless sensor network. In *Proceedings of the 8th International Conference on Reliability, Maintainability and Safety (ICRMS 2009)*, pages 16 –19, 20-24 2009.

[6] P. Biswas, H. Aghajan, and Y. Ye. Integration of angle of arrival information for multi-modal sensor network localization using semidefinite programming. In *In Proceedings of 39th Asilomar Conference on Signals, Systems and Computers*, 2005.

[7] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[8] B. Blywis, M. Gu andnes, F. Juraschek, and S. Gliech. A localization framework for wireless mesh networks - anchor-free distributed localization in the des-testbed. In

Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on, pages 1 –10, sept. 2010.

[9] Gaetano Borriello, Alan Liu, Tony Offer, Christopher Palistrant, and Richard Sharp. Walrus: wireless acoustic location with room-level resolution using ultrasound. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, pages 191–203, New York, NY, USA, 2005. ACM.

[10] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. A novel solution for achieving anonymity in wireless ad hoc networks. In *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 30–38, 2004.

[11] A. Boukerche and X. Li. An agent-based trust and reputation management scheme for wireless sensor networks. *Global Telecommunications Conference*, 3, 2005.

[12] A.J. Bernheim Brush, John Krumm, and James Scott. Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 95–104, New York, NY, USA, 2010. ACM.

[13] S. Buchegger and J. Boudec. Performance analysis of the confidant protocol. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc '02)*, pages 226–236, New York, NY, USA, 2002. ACM.

[14] S. Buchegger and J. Boudec. A robust reputation system for mobile ad-hoc networks. *Proceedings of P2PEcon*, 2004.

[15] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5), October 2000.

[16] Nirupama Bulusu, John Heidemann, Deborah Estrin, and Tommy Tran. Self-configuring localization systems: Design and experimental evaluation. *ACM Trans. Embed. Comput. Syst.*, 3:24–60, February 2004.

[17] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava. Participatory sensing. *WSW06 at SenSys 06*, 2006.

[18] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, and R. Peterson. People-centric urban sensing. In *WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet*, 2006.

[19] Z. Cao, J. Hu, Z. Chen, M. Xu, and X. Zhou. Fbsr: feedback-based secure routing protocol for wireless sensor networks. *International Journal of Pervasive Computing and Communications*, 2008.

[20] S. Cha, M. Jo, J. Lee, D. Kim, S. Kang, K. Cho, N. Lee, and Y. Kim. Hierarchical node clustering approach for energy savings in wsns. In *SERA '07: Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications*, pages 253–259, Washington, DC, USA, 2007. IEEE Computer Society.

[21] Ho-lin Chang, Jr-ben Tian, Tsung-Te Lai, Hao-Hua Chu, and Polly Huang. Spinning beacons for precise indoor localization. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 127–140, New York, NY, USA, 2008. ACM.

[22] S. Chang, S. Shieh, W. Lin, and C. Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS '06)*, pages 311–320, New York, NY, USA, 2006. ACM.

[23] H. Chen, H. Wu, X. Zhou, and C. Gao. Agent-based trust model in wireless sensor networks. In *Proceedings of the Eighth ACIS International Conference on Software*

Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pages 119–124, 2007.

[24] Bing Hwa Cheng, Lieven Vandenberghe, and Kung Yao. Distributed algorithm for node localization in wireless ad-hoc networks. *ACM Trans. Sen. Netw.*, 6(1):8:1–8:20, January 2010.

[25] X. Cheng, A. Teler, G. Xue, and D. Chen. Tps: a time-based positioning scheme for outdoor wireless sensor networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, march 2004.

[26] Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, GIS '06, pages 171–178, New York, NY, USA, 2006. ACM.

[27] B Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[28] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, 2008.

[29] Jose A. Costa, Neal Patwari, and Alfred O. Hero, III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sen. Netw.*, 2:39–64, February 2006.

[30] L. Cox, A. Dalton, and V. Marupadi. Smokescreen: flexible privacy controls for presence-sharing. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 233–245, New York, NY, USA, 2007. ACM.

[31] Crossbow technology inc.

[32] James L. Crowley. Mathematical foundations of navigation and perception for an autonomous mobile robot. In *Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics*, pages 9–51, London, UK, 1996. Springer-Verlag.

[33] T. Das, P. Mohan, V. Padmanabhan, R. Ramjee, and A. Sharma. Prism: platform for remote sensing using smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 63–76, New York, NY, USA, 2010. ACM.

[34] C. Davis. A localized trust management scheme for ad hoc networks. *3rd International Conference on Networking (ICN04)*, 2004.

[35] L. Deng and L. P. Cox. Livecompare: grocery bargain hunting through participatory sensing. In *HotMobile '09: Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, pages 1–6, New York, NY, USA, 2009. ACM.

[36] G.N. Desouza and A.C. Kak. Vision for mobile robot navigation: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237 –267, feb 2002.

[37] L. Ding, P. Kolari, S. Ganjugunte, T. Finin, and A. Joshi. Modeling and evaluating trust network inference. In *Seventh International Workshop on Trust in Agent Societies at AAMAS 2004*, 2004.

[38] L. Doherty, K.S.J. pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1655 –1663 vol.3, 2001.

[39] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *In Pervasive*, pages 152–170, 2005.

[40] Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Comput. Surv.*, 42:5:1–5:35, December 2009.

[41] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[42] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, Berkeley, CA, USA, 2006. USENIX Association.

[43] S.R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '03)*, volume 1, pages 377 – 381 Vol.1, 1-5 2003.

[44] S. Ganeriwal, L. Balzano, and M. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Trans. Sen. Netw.*, 2008.

[45] A. Garg, R. Battiti, and R. Cascella. Reputation management: experiments on the robustness of rocq. *Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings*, pages 725–730, April 2005.

[46] B. Gedik and L. Liu. Energy-aware data collection in sensor networks: A localized selective sampling approach. Technical report, Georgia Institute of Technology, 2005.

[47] Aris Gkoulalas-Divanis, Panos Kalnis, and Vassilios S. Verykios. Providing k-anonymity in location based services. *SIGKDD Explor. Newsl.*, 12:3–10, November 2010.

[48] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pages 1–14, New York, NY, USA, 2009. ACM.

[49] I. Gondal, M. Iqbal, M. Woods, and S. Sehgal. Integrated sensing and diagnosis – the next step in real time patient health care. In *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, pages 581 –586, july 2007.

[50] W. Gong, Z. You, D. Chen, X. Zhao, M. Gu, and K. Lam. Trust based routing for misbehavior detection in ad hoc networks. *Journal of Networks*, 5(5), May 2010.

[51] Xun Gong, Negar Kiyavash, and Nikita Borisov. Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 684–686, New York, NY, USA, 2010. ACM.

[52] Zhenqiang Gong, Guang-Zhong Sun, and Xing Xie. Protecting privacy in location-based services using k-anonymity without cloaked region. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, MDM '10, pages 366–371, Washington, DC, USA, 2010. IEEE Computer Society.

[53] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of ACM MobiSys'03*, May 2003.

[54] Santanu Guha, Kurt Plarre, Daniel Lissner, Somnath Mitra, Bhagavathy Krishna, Prabal Dutta, and Santosh Kumar. Autowitness: locating and tracking stolen property while tolerating gps and radio outages. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 29–42, New York, NY, USA, 2010. ACM.

[55] Q. He, D. Wu, and P. Khosla. Sori: A secure and objective reputation-based incentive scheme for ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, pages 825–830, 2004.

[56] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 81–95, New York, NY, USA, 2003. ACM.

[57] J. Hee-Jin, N. Choon-Sung, J. Yi-Seok, and S. Dong-Ryeol. A mobile agent based leach in wireless sensor networks. In *Proceedings of the 10th International Conference on Advanced Communication Technology (ICACT 2008)*, volume 1, pages 75 –78, 17-20 2008.

[58] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, page 8020, Washington, DC, USA, 2000. IEEE Computer Society.

[59] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

[60] J.A. Hesch, F.M. Mirzaei, G.L. Mariottini, and S.I. Roumeliotis. A laser-aided inertial navigation system (l-ins) for human localization in unknown indoor environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5376 –5382, may 2010.

[61] J. Hightower, R. Want, and G. Borriello. Spoton: An indoor 3d location sensing technology based on rf signal strength. Technical report, University of Washington, Department of Computer Science and Engineering, Seattle,WA, February 2000.

[62] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[63] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '99)*, pages 53–60, New York, NY, USA, 1999. ACM.

[64] N. Hopper, E. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 82–91, New York, NY, USA, 2007. ACM.

[65] C. Hsu and C. Yu. An accelerometer based approach for indoor localization. In *Proceedings of the 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, UIC-ATC '09, Washington, DC, USA, 2009. IEEE Computer Society.

[66] J.D. Jackson, D.W. Callahan, and P. F. Wang. Location tracking of test vehicles using accelerometers. In *Proceedings of the 5th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing*. World Scientific and Engineering Academy and Society (WSEAS), 2006.

[67] M. Jain and H. Kandwal. A survey on complex wormhole attack in wireless ad hoc networks. In *Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT '09)*, pages 555 –558, 28-29 2009.

[68] D. B. Johnson et al. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic Publishers, 1996.

[69] O. Kallenberg. Foundations of modern probability. Springer, 2002.

[70] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

[71] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. In *Proceedings of the 2nd ACM international*

workshop on Wireless mobile applications and services on WLAN hotspots, WMASH '04, pages 110–118, New York, NY, USA, 2004. ACM.

[72] William B. Kannel, Craig Kannel, Ralph S. Paffenbarger Jr., and L.Adrienne Cupples. Heart rate and cardiovascular mortality: The framingham study. *American Heart Journal*, 1987.

[73] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proc. of ACM SenSys 2004*, November 2004.

[74] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[75] Douglas J. Kelly, Richard A. Raines, Michael R. Grimaila, Rusty O. Baldwin, and Barry E. Mullins. A survey of state-of-the-art in anonymity metrics. In *Proceedings of the 1st ACM workshop on Network data anonymization*, NDA '08, pages 31–40, New York, NY, USA, 2008. ACM.

[76] J. Kim, J. Lee, G. Jee, and T. Sung. Compensation of gyroscope errors and gps/dr integration. In *Position Location and Navigation Symposium, 1996., IEEE 1996*, pages 464 –470, apr 1996.

[77] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. In *Security and Privacy, 2005 IEEE Symposium on*, pages 211 – 225, may 2005.

[78] I. Krontiris, T. Giannetsos, and T. Dimitriou. Launching a sinkhole attack in wireless sensor networks; the intruder side. In *Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications(WIMOB '08)*, pages 526 –531, 12-14 2008.

[79] John Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, 13:391–399, August 2009.

[80] P. Lamon and R. Siegwart. Inertial and 3d-odometry fusion in rough terrain - towards real 3d navigation. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1716 – 1721, sept.-2 oct. 2004.

[81] S. Lanzisera, D.T. Lin, and K.S.J. Pister. Rf time of flight ranging for wireless sensor network localization. In *Intelligent Solutions in Embedded Systems, 2006 International Workshop on*, pages 1 –12, june 2006.

[82] Loukas Lazos and Radha Poovendran. Serloc: secure range-independent localization for wireless sensor networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, WiSe '04, pages 21–30, New York, NY, USA, 2004. ACM.

[83] JaeHo Lee. Smart health: Concepts and status of ubiquitous health with smartphone. In *ICT Convergence (ICTC), 2011 International Conference on*, pages 388 –389, sept. 2011.

[84] Htc legend. `http://www.htc.com/europe/specification.aspx?p_id=313`, 2010.

[85] Lego mindstorm nxt 2.0. `http://mindstorms.lego.com`, 2009.

[86] Ming Lei, Xiaoyan Hong, and S.V. Vrbsky. Protecting location privacy with dynamic mac address exchanging in wireless networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 49 –53, nov. 2007.

[87] Bin Li, Wenjie Wang, Qinye Yin, Rong Yang, and Yubo Li. Distributed localization in wireless sensor networks using rotary antennas. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 2443 –2447, oct. 2010.

[88] Y. Li, J. Xu, B. Zhao, and G. Yang. A new mobile agent architecture for wireless sensor networks. In *Proceedings of the 3rd IEEE Conference on Industrial Electronics and Applications (ICIEA 2008)*, pages 1562 –1565, 3-5 2008.

[89] Z. Liang and W. Shi. Pet: A personalized trust model with reputation and risk evaluation for p2p resource sharing. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 7*. IEEE Computer Society, 2005.

[90] Z. Liang and W. Shi. Analysis of ratings on trust inference in open environments. *Perform. Eval.*, 65(2):99–128, 2008.

[91] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks (IPSN '08)*, pages 245–256. IEEE Computer Society, 2008.

[92] B. Liu, M. Adams, and J. Ibanez-Guzman. Minima controlled recursive averaging noise reduction for multi-aided inertial navigation of ground vehicles. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3408 – 3414, aug. 2005.

[93] Juan Liu, Ying Zhang, and Feng Zhao. Robust distributed node localization with error management. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 250–261, New York, NY, USA, 2006. ACM.

[94] L. Liu and W. Shi. Trust and reputation management. *Internet Coomputing*, 14(5), sep/oct 2010.

[95] Z. Liu, A. Joy, and R. Thompson. A dynamic trust model for mobile ad hoc networks. In *Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 80–85, 2004.

[96] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *the 8th ACM Conference*

on Embedded Networked Sensor Systems, 2010.

[97] A. Lymberis. Smart wearable systems for personalised health management: current r d and future challenges. In *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, 2003.

[98] S. Madden. Intel lab data, 2004. http://berkeley.intel-research.net/labdata.

[99] Miklós Maróti, Péter Völgyesi, Sebestyén Dóra, Branislav Kusý, András Nádas, Ákos Lédeczi, György Balogh, and Károly Molnár. Radio interferometric geolocation. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 1–12, New York, NY, USA, 2005. ACM.

[100] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom '00)*, pages 255–265. ACM, 2000.

[101] Sergio Mascetti, Claudio Bettini, X. Sean Wang, Dario Freni, and Sushil Jajodia. Providenthider: An algorithm to preserve historical k-anonymity in lbs. In *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, MDM '09, pages 172–181, Washington, DC, USA, 2009. IEEE Computer Society.

[102] Yutaka Matsuo, Naoaki Okazaki, Kiyoshi Izumi, Yoshiyuki Nakamura, Takuichi Nishimura, Kôiti Hasida, and Hideyuki Nakashima. Inferring long-term user properties based on users' location history. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2159–2165, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[103] Matlab. `http://www.mathworks.com`, 2010.

[104] P. Michiardi and R. Molva. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the IFIP TC6/TC11*

Sixth Joint Working Conference on Communications and Multimedia Security, pages 107–121, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.

[105] Motelab. `http://motelab.eecs.harvard.edu`, 2005.

[106] Multihoposcilloscope. `http://www.tinyos.net`, 2006.

[107] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, 2009.

[108] William M. Newman, Margery A. Eldridge, and Michael G. Lamming. Pepys: generating autobiographies by automatic tracking. In *Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work*, pages 175–188, Norwell, MA, USA, 1991. Kluwer Academic Publishers.

[109] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proc. of the 3rd International Conference on Information Processing in Sensor Networks (IPSN'04)*, April 2004.

[110] D. Niculescu and Badri Nath. Ad hoc positioning system (aps) using aoa. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1734 – 1743 vol.3, march-3 april 2003.

[111] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 299–314, New York, NY, USA, 2010. ACM.

[112] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pages 99–110, New York, NY, USA, 2007. ACM.

[113] A. Parker, S. Reddy, T. Schmid, K. Chang, S. Ganeriwal, M. B. Srivastava, M. Hansen, J. Burke, D. Estrin, M. Allman, and V. Paxson. Network system challenges in selective sharing and verification for personal, social, and urban-scale sensing applications. In *the Fifth Workshop on Hot Topics in Networks (HotNets-V)*, 2006.

[114] Sai Teja Peddinti and Nitesh Saxena. On the limitations of query obfuscation techniques for location privacy. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 187–196, New York, NY, USA, 2011. ACM.

[115] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[116] A. Perrig, R. Szewczyk, W. Wen, D. Culler, and J. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks Journal (WINET)*, 8(5):521–534, September 2002.

[117] N. Petrellis, N. Konofaos, and G.Ph. Alexiou. Target localization utilizing the success rate in infrared pattern recognition. *Sensors Journal, IEEE*, 6(5):1355 –1364, oct. 2006.

[118] A. Pirzada and C. McDonald. Trusted greedy perimeter stateless routing. In *Proceedings of the 15th IEEE International Conference on Networks (ICON 2007)*, pages 206 –211, 19-21 2007.

[119] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking(MobiCom'00)*, August 2000.

[120] V. Rajaravivarma, Yi Yang, and Teng Yang. An overview of wireless sensor network and applications. In *System Theory, 2003. Proceedings of the 35th Southeastern Symposium on*, pages 432 – 436, march 2003.

[121] Abhishek RayChaudhuri, Ujwal K. Chinthala, and Amiya Bhattacharya. Obfuscating temporal context of sensor data by coalescing at source. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11:41–42, April 2007.

[122] Ramona Rednic, John Kemp, Elena Gaura, and James Brusey. Networked body sensing: Enabling real-time decisions in health and defence applications. In *Advanced Computer Science and Information System (ICACSIS), 2011 International Conference on*, pages 17 –24, dec. 2011.

[123] G. Reina, L. Ojeda, A. Milella, and J. Borenstein. Wheel slippage and sinkage detection for planetary rovers. *Mechatronics, IEEE/ASME Transactions on*, 11(2):185 –195, april 2006.

[124] K. Ren, W. Lou, K. Zeng, and P. Moran. On broadcast authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 6(11):4136 –4144, november 2007.

[125] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebays reputation system. 2002.

[126] A. Rezgui and M. Eltoweissy. Tarp: A trust-aware routing protocol for sensor-actuator networks. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS 2007)*, 8-11 2007.

[127] P Rong and M.L. Sichitiu. Angle of arrival localization for wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, sept. 2006.

[128] H. Safa, H. Artail, and D. Tabet. A cluster-based trust-aware routing protocol for mobile ad hoc networks. *Wirel. Netw.*, 16(4):969–984, 2010.

[129] T. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: privacy trends in consumer ubiquitous computing. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, 2007.

[130] C. Savarese, J.M. Rabaey, and J. Beutel. Location in distributed ad-hoc wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 4, pages 2037 –2040 vol.4, 2001.

[131] A. Savvides, C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking(MobiCom'01)*, July 2001.

[132] N. Schmitz, J. Koch, M. Proetzsch, and K. Berns. Fault-tolerant 3d localization for outdoor vehicles. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 941 –946, oct. 2006.

[133] K. Sha, G. Zhan, S. Al-Omari, T. Calappi, W. Shi, and C. Miller. Data quality and failures characterization of sensing data in environmental applications. In *CollaborateCom 2008*, 2008.

[134] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing)*, June 2003.

[135] Vitaly Shmatikov and Ming-Hsiu Wang. Measuring relationship anonymity in mix networks. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, WPES '06, pages 59–62, New York, NY, USA, 2006. ACM.

[136] M. Soltan, M. Maleki, and M. Pedram. Lifetime-aware hierarchical wireless sensor network architecture with mobile overlays. *Radio and Wireless Symposium, 2007 IEEE*, pages 325–328, Jan. 2007.

[137] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:571–588, October 2002.

[138] Bin Tang, Xianjin Zhu, A. Subramanian, and Jie Gao. Dal: A distributed localization in sensor networks using local angle measurement. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1 –6, aug. 2009.

[139] Telosb. `http://www.xbow.com`, 2005.

[140] Lasse Thiem, Björn Riemer, Marcus Witzke, and Thomas Luckenbach. Rfid-based localization in heterogeneous mesh networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 415–416, New York, NY, USA, 2008. ACM.

[141] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dallaert. Robust monte carlo localization for mobile robots. *Artif. Intell.*, 128:99–141, May 2001.

[142] T. Tran and Rc Cohen. A reputationcoriented reinforcement learning strategy for agents in electronic marketplaces. Computational Intelligence, 2002.

[143] Alex Varshavsky, Eyal de Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason. Gsm indoor localization. *Pervasive Mob. Comput.*, 3:698–720, December 2007.

[144] M. Vemula, M. Bugallo, and P. Djuric. Target tracking in a two-tiered hierarchical sensor network. *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 4:IV–IV, May 2006.

[145] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, page 150, 2003.

[146] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN '04)*, pages 59–64, New York, NY, USA, 2004. ACM.

[147] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 483–488, April 2005.

[148] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 73 – 80, april 2005.

[149] Kamin Whitehouse, Chris Karlof, and David Culler. A practical evaluation of radio signal strength for ranging-based localization. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11:41–52, January 2007.

[150] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 754–759, New York, NY, USA, 2006. ACM.

[151] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the First ACM SenSys'03*, November 2003.

[152] A. Wood and J. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, Oct 2002.

[153] J. Liu X. Li, M. R. Lyu. Taodv: A trusted aodv routing protocol for mobile ad hoc networks. In *Proceedings of Aerospace Conference*, 2004.

[154] Bin Xiao, Hekang Chen, and Shuigeng Zhou. Distributed localization using a moving beacon in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 19(5):587–600, May 2008.

[155] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, 2009.

[156] W. Xue, J. Aiguo, and W. Sheng. Mobile agent based moving target methods in wireless sensor networks. In *IEEE International Symposium on Communications and Information Technology (ISCIT 2005)*, volume 1, pages 22 – 26, 12-14 2005.

[157] Z. Yan, P. Zhang, and T. Virtanen. Trust evaluation based security solution in ad hoc networks. In *Proceeding of the 7th Nordic Workshop on Secure IT Systems*, 2003.

[158] K. Yedavalli and B. Krishnamachari. Sequence-based localization in wireless sensor networks. *Mobile Computing, IEEE Transactions on*, PP(99):1, 2009.

[159] K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasan. Ecolocation: a sequence based technique for rf localization in wireless sensor networks. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 285 – 292, april 2005.

[160] M. Youssef, M.A. Yosef, and M. El-Derini. Gac: Energy-efficient hybrid gps-accelerometer-compass gsm localization. In *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, dec. 2010.

[161] K. Yu and I. Oppermann. Uwb positioning for wireless embedded networks. In *Radio and Wireless Conference, 2004 IEEE*, pages 459 – 462, sept. 2004.

[162] L. Yuan and X. Wang. Study on data gathering algorithm based on mobile agent and wsn for emergent event monitoring. In *Proceedings of International Symposium on Computer Network and Multimedia Technology (CNMT 2009)*, pages 1 –5, 18-20 2009.

[163] T. Zahariadis, H. Leligou, P. Karkazis, P. Trakadas, I. Papaefstathiou, C. Vangelatos, and L. Besson. Design and implementation of a trust-aware routing protocol for large wsns. *International Journal of Network Security & Its Applications (IJNSA)*, 2(3), July 2010.

[164] G. Zhan, W. Shi, and J. Deng. Poster abstract: Sensortrust - a resilient trust model for wsns. In *Proceedings of the 7th International Conference on Embedded Networked Sensor Systems (SenSys'09)*, 2009.

[165] G. Zhan, W. Shi, and J. Deng. Tarf: A trust-aware routing framework for wireless sensor networks. In *Proceeding of the 7th European Conference on Wireless Sensor Networks (EWSN'10)*, 2010.

[166] Guoxing Zhan, Weisong Shi, and Julia Deng. Sensortrust: A resilient trust model for wireless sensing systems. *Pervasive and Mobile Computing*, 2011.

[167] L. Zhang, Q. Wang, and X. Shu. A mobile-agent-based middleware for wireless sensor networks data fusion. In *Proceedings of Instrumentation and Measurement Technology Conference (I2MTC '09)*, pages 378 –383, 5-7 2009.

[168] Q. Zhang and T. Yu. On the modeling of honest players in reputation systems. *Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on*, pages 249–254, June 2008.

[169] Yu Zhang, Lin Zhang, and Xiuming Shan. Ranking-based statistical localization for wireless sensor networks. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 2561 –2566, 31 2008-april 3 2008.

[170] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann Publishers, 2004.

[171] Q. Zheng, X. Hong, and S. Ray. Recent advances in mobility modeling for mobile ad hoc network research. In *Proceedings of the 42nd Annual Southeast Regional Conference (ACM-SE 42)*, pages 70–75, New York, NY, USA, 2004. ACM.

[172] Ge Zhong and Urs Hengartner. Toward a distributed k-anonymity protocol for location privacy. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, 2008.

[173] Ziguo Zhong and Tian He. Achieving range-free localization beyond connectivity. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 281–294, New York, NY, USA, 2009. ACM.

[174] C. Zhu, K. Li, Q. Lv, L. Shang, and R. Dick. iscope: personalized multi-modality image search for mobile devices. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pages 277–290, New York, NY, USA, 2009. ACM.

[175] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 315–330, New York, NY, USA, 2010. ACM.

ABSTRACT

SYSTEM SUPPORT FOR ROBUST DATA COLLECTION IN WIRELESS SENSING SYSTEMS

by

GUOXING ZHAN

August 2012

Advisor: Dr. Weisong Shi

Major: Computer Science

Degree: Doctor of Philosophy

This dissertation studied how to provide system support for robust data collection in wireless sensing systems through addressing a few urgent design issues in the existing systems. A wireless sensing system may suffer issues arising at the sensors, during the data transmission, and during the data access by applications. Due to the unique characteristics of wireless sensing systems, certain conventional solutions for networked systems may not work well with these issues. We developed approaches to resolve these urgent problems in the design of wireless sensing systems. Specially, we have achieved the following: (1) we developed a resilient trust model to effectively detect faulty data in wireless sensing systems due to either sensor malfunctioning or malicious attempts to report false data; (2) we developed a low-cost, self-contained, accurate localization system for small-sized ground robotic vehicles, which enhances the wireless sensing systems containing mobile sensors by providing more accurate and highly available location data, with only limited overhead in economic cost and management; (3) we designed and implemented a robust trust-aware routing framework to secure multi-hop routing through a set of sensors in wireless sensing systems; (4) we developed a privacy-preserving wireless sensing system, which protects the user privacy while allowing arbitrary third-party applications to extract knowledge from the collected data.

AUTOBIOGRAPHICAL STATEMENT

GUOXING ZHAN

Guoxing Zhan is a Ph.D. candidate in the Department of Computer Science at Wayne State University. He received a M.S. in Mathematics from Chinese Academy of Sciences in 2007 and another M.S in Computer Science from Wayne State University in 2009. Mr. Zhan is interested in research on participatory sensing, wireless sensor network, mobile computing, networking and systems security, trust management, and information processing. Several of his research papers have been presented at international conferences or published by research journals. Additionally, Mr. Zhan has instructed a few computer science labs consisting of interactive short lectures and hands-on experience.