ABSTRACT

Title of dissertation: SECURITY, TRUST AND COOPERATION IN WIRELESS SENSOR NETWORKS

Shanshan Zheng, Doctor of Philosophy, 2011

Dissertation directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

Wireless sensor networks are a promising technology for many real-world applications such as critical infrastructure monitoring, scientific data gathering, smart buildings, etc.. However, given the typically unattended and potentially unsecured operation environment, there has been an increased number of security threats to sensor networks. In addition, sensor networks have very constrained resources, such as limited energy, memory, computational power, and communication bandwidth. These unique challenges call for new security mechanisms and algorithms. In this dissertation, we propose novel algorithms and models to address some important and challenging security problems in wireless sensor networks.

The first part of the dissertation focuses on data trust in sensor networks. Since sensor networks are mainly deployed to monitor events and report data, the quality of received data must be ensured in order to make meaningful inferences from sensor data. We first study a false data injection attack in the distributed state estimation problem and propose a distributed Bayesian detection algorithm, which could maintain correct estimation results when less than one half of the sensors are

compromised. To deal with the situation where more than one half of the sensors may be compromised, we introduce a special class of sensor nodes called *trusted cores*. We then design a secure distributed trust aggregation algorithm that can utilize the trusted cores to improve network robustness. We show that as long as there exist some paths that can connect each regular node to one of these trusted cores, the network can not be subverted by attackers.

The second part of the dissertation focuses on sensor network monitoring and anomaly detection. A sensor network may suffer from system failures due to loss of links and nodes, or malicious intrusions. Therefore, it is critical to continuously monitor the overall state of the network and locate performance anomalies. The network monitoring and probe selection problem is formulated as a budgeted coverage problem and a Markov decision process. Efficient probing strategies are designed to achieve a flexible tradeoff between inference accuracy and probing overhead. Based on the probing results on traffic measurements, anomaly detection can be conducted. To capture the highly dynamic network traffic, we develop a detection scheme based on multi-scale analysis of the traffic using wavelet transforms and hidden Markov models. The performance of the probing strategy and of the detection scheme are extensively evaluated in malicious scenarios using the NS-2 network simulator.

Lastly, to better understand the role of trust in sensor networks, a game theoretic model is formulated to mathematically analyze the relation between trust and cooperation. Given the trust relations, the interactions among nodes are modeled as a network game on a trust-weighted graph. We then propose an efficient heuristic method that explores network heterogeneity to improve Nash equilibrium efficiency.

Security, Trust and Cooperation in Wireless Sensor Networks

by

Shanshan Zheng

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor Gang Qu
Professor Sennur Ulukus
Professor Michel Cukier
Professor Subramanian Raghavan

Dedication

To my parents Zhiqing and Chunxiang, to my sister Wanqiu, and to Wenjun, for their invaluable love and support.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever. First and foremost I'd like to thank my advisor, Dr. John S. Baras for giving me an invaluable opportunity to work on challenging and extremely interesting projects. His deep insight on amazingly broad area of research, limitless energy and enthusiasm on challenging problems has been the most important support for my work. Dr. Tao Jiang was also a great source of help and encouragement, to whom I am extremely grateful for the guidance and knowledge that she so generously shared. I would also like to thank Dr. Gang Qu, Dr. Sennur Ulukus, Dr. Michel Cukier and Dr. Subramanian Raghavan, for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

Sincerely thanks to my colleagues in the HyNet center and SEIL lab, who have enriched my graduate life in many ways and with whom I always had inspiring and fruitful discussions. Also I would like to thank Kim Edwards for her great administrative support.

This dissertation is prepared through collaborative participation in the Multi-Scale Systems Center (MuSyC), supported by the Defense Advanced Research Projects Agency (DARPA) under award number 013641-001, through the FRCP of SRC and DARPA. It is also supported by US Air Force Office of Scientific Research MURI award FA9550-09-1-0538, and Army Research Office MURI award W911-NF-0710287.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Sensor networks

Recent advances of MEMS technology have made the manufacturing of small sensors technically and economically feasible. Sensor networks provide the ability to enable remote interactions with the physical world at unprecedented physical scales, with a broad set of sensors and at low cost. These tiny sensor nodes consist of sensing, data processing, and communication components, usually named as "Motes". A typical example of a sensor node can be the Iris Mote, which has a 16MHz processor with 8KB RAM, 128 KB program flash and 512KB measurement flash memories. There are also a little more advanced types of sensor nodes such as the Intel Mote 2 (Imote2), which has a 32-bit XScale processor, running at adjustable frequency from 13 to 416 MHz, with 256KB SRAM, 32 MB flash memory and 32 MB SDRAM. Both of these Motes have an integrated 802.15.4 radio and an external 2.4 GHz antenna. They can be equipped with multiple onboard sensors, such as accelerometers, magnetometers, passive infrared sensors, acoustic and video sensors, etc.. Fig. 1.1 shows the Imote2 and the associated sensor board.

A wireless sensor network consists of a set of spatially scattered Motes that can measure various properties of the environment, implement local and distributed inferences, and make responses to events or queries. Sensor networks are usually

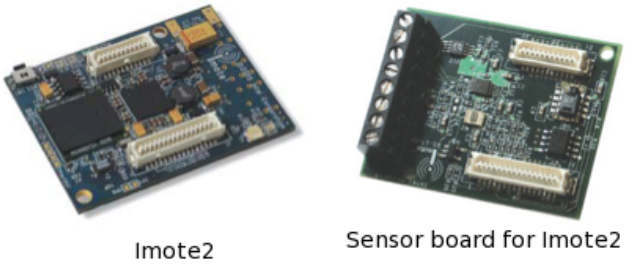


Figure 1.1: Imote2 and the sensor board

ad-hoc and infrastructure free. Each node communicates with others by forming a multi-hop radio path and maintain connectivity in a decentralized manner. The connectivity among nodes can also be dynamic as nodes can join or leave the network at any time. Typical sensor network applications include critical infrastructure monitoring, emergency disaster relief, scientific data gathering, smart buildings, personal medical systems, context-aware services, etc.. However, a great deal remains to be done in order for this vision to be realized. Especially, given the typically unattended and potentially unsecured operation environment, there has been an increased number of security threats to sensor networks.

Although wireless sensor networks come from wireless ad hoc networks, there are important distinctions between them, which can greatly affect the system designs including the security design. Compared to ad hoc networks, sensor networks have limited energy, computational capability, memory capacity, and communication bandwidth. Design of sensor network algorithms must take into account these resource constraints. For example, in a sensor network, the nodes usually have to locally process the information they collect, then only the fused data is sent back to the network operator. Efficient collaborative signal processing algorithms would

be needed to enable such local fusion of data. To save sensor energy for longer life time, techniques for operating in low energy state are required, such as placing the nodes in some type of sleep mode. The emergence of new technologies for designing energy harvesting devices is also promising for more energy efficient use of sensor networks. Also, sensor networks are usually deployed with a large number of nodes. For example, in surveillance applications, while the sensitivity of individual sensors may be highly variable, with high node density they will be able to reliably detect intruders. Therefore, scalability is another important issue that needs to be considered for sensor network designs.

1.2 Main contributions and thesis organization

Given the unique characteristics of sensor networks, their security problems are more complicated and challenging. In this dissertation we focus on some important security issues in sensor networks and propose novel algorithms and models. The main contributions of this dissertation are:

1. Data trust in sensor networks:

    - Sensor networks are mainly deployed to detect events and report data. To make meaningful inferences from sensor data, the quality of the received data must be ensured. One of the important sources of false sensor data is due to malicious attacks where the attackers compromise sensors and inject false measurements. We study an example of false data injection attack in the distributed state estimation problem and show that

the estimation results can vary arbitrarily even if only one sensor is compromised. We then propose a distributed Bayesian detection algorithm that can maintain correct estimation results as long as less than one half of the sensors are compromised, without causing extra communication costs.

- Trust has been widely used for many applications, including e-commerce, peer to peer networks, web-based services and distributed computing. Trust-based methods can provide a complementary security mechanism for sensor networks. We present a special construction of trust system in sensor networks, namely, the *trusted cores*, which are a special class of nodes that have much higher levels of security than other sensor nodes. We design a secure distributed trust aggregation algorithm that can utilize the trusted cores to enhance data integrity in sensor networks. We show that as long as there exist some paths that can connect each regular node to one of these trusted cores, the network can not be subverted by the attackers even if more than half of the sensors are compromised.

2. Sensor network monitoring and anomaly detection:

   - Network monitoring is important for ensuring correct sensor network functioning. The low bandwidth and energy constraints are the main challenges for sensor network monitoring. We design a trust-assisted probing strategy to monitor a hierarchical sensor network, where network heterogeneity is exploited for better bandwidth and energy efficiency. The

basic idea is to collect path measurements through a two-phase probing. The first phase is used to identify the suspicious areas in the network by sending a small subset of probes. It is formulated as a budgeted maximum coverage problem. We propose an approximation algorithm to efficiently solve this problem using linear programming duality. The second phase aims at locating individual links that are responsible for the observed path anomalies with minimum communication cost. We formulate the second phase as a Markov decision problem and solve it using an efficient heuristic method. We provide thorough validations through simulations under different network settings and performance comparisons with existing algorithms.

- Based on traffic measurements collected by probing, anomaly detection algorithms can be applied to decide whether the observations are normal or not. However, recent studies have shown that node mobility along with spatial correlation of the monitored phenomena in sensor networks can lead to observation data that have long range dependency, which could significantly increase the false alarm rates for traditional anomaly detection methods [73]. We develop an anomaly detection scheme based on multi-scale analysis of the long range dependent traffic to address this challenge. In the proposed detection scheme, discrete wavelet transforms are used to approximately de-correlate the traffic data and capture data characteristics at different time scales. The remaining dependencies are

captured by a multi-level hidden Markov model in the wavelet domain. To estimate the model parameters, we propose an online discounting Expectation Maximization (EM) algorithm, which also tracks variations of the estimated models over time. Network anomalies are detected as abrupt changes in the tracked model variations. We evaluate our detection scheme thoroughly using numerical long range dependent time series and also malicious scenarios simulated using the NS-2 network simulator.

3. Game theoretic analysis of security and trust:

   - We present a game theoretic analysis of the efficiency of establishing trust for improving node cooperation. The trust relations among nodes are modeled as a trust-weighted network, and we study a graphical game in this network where the nodes' payoffs are affected by their trust relations. We characterize the Nash equilibrium and the social optimum point of this game and show that the game efficiency has a close relationship to Bonacich centralities of nodes in the trust-weighted network. We also provide a method for improving game efficiency by allocating heterogeneous resources to different nodes according to their Bonacich centralities. We provide both experimental and theoretical analysis on the resulting improvement of the game efficiency.

The rest of this dissertation is organized as follows. Chapter 2 reviews security issues in sensor networks, discusses security objectives, threat models and the associated countermeasures. Chapter 3 and Chapter 4 focus on data trust in

sensor network. Chapter 3 investigates the faulty data problem in sensor networks and presents a distributed Bayesian detection algorithm. Chapter 4 discusses the concept of trust in sensor networks and presents the trust aggregation algorithm which uses trusted cores to enhance data integrity. Chapter 5 and Chapter 6 focus on sensor network monitoring and anomaly detection. Chapter 5 presents our trust assisted probing strategy and anomaly localization algorithm. Chapter 6 presents our anomaly detection algorithm for long range dependent traffic. Lastly, Chapter 7 formulates a game theoretic model to study security and trust in sensor networks.

Chapter 2

Security in Wireless Sensor Networks

In this chapter, we review security issues in sensor networks. Security objectives, threat models and the associated countermeasures for sensor networks are discussed.

2.1 Security objectives

The security objectives of wireless sensor networks are summarized as follows.

- **Data Integrity:** This requirement measures the trustworthiness of the information provided by the sensor network. The quality of the received information allows the sensor network to correctly perform its function, e.g., formulate local or distributed inferences, make responses to events or queries, etc. A violation of data integrity results in deception; authorized nodes receive false information but believe the faulty information to be true.

  Data integrity can be violated in several ways. Firstly, sensor nodes may report data that are not representative of their intended environment. For example, sensors may be compromised by the attacker, or the environment around sensors may be intentionally affected by the attacker, e.g., the attacker may put a magnet on top of a magnetometer or change the location of the sensor node. Secondly, the message may be modified by unauthorized users during trans-

mission, i.e., message tampering. Thirdly, as in-network processing of the collected data is common in sensor networks, the attacker may compromise data integrity during data aggregation. For example, assuming that sensor nodes are gathering temperature data and need to report to the base station the average temperature value from their neighbors, the attacker may corrupt this average value. Since sensor networks are mainly deployed to monitor events and report data, data integrity would be an important requirement for sensor network security. In this thesis, Chapter 3 and Chapter 4 focus on data integrity in sensor networks.

- **Availability:** this requirement means that the information collected by sensors is accessible and usable upon demand by authorized users. A violation of network availability results in denial of service: the prevention of authorized access to sensor measurements. Adjusting the traditional security mechanisms to fit within wireless sensor networks could weaken the availability of a sensor network. For example, the additional computation and communication from executing security mechanisms can consume additional energy, thus reduce the lifetime of a sensor network; moreover, if a centralized scheme is used, the single point of failure will also threaten availability of the sensor network.

- **Data Authentication:** This requirement allows the receiver to verify that the data really originates from the claimed sender. In the case of two party communication, data authentication can be achieved by a purely symmetric mechanism; the sender and receiver share a secret key to compute the message

authentication code of all communicated data. Authentication is also necessary for administrative tasks such as network reprogramming or controlling sensor node duty cycle.

- **Data Confidentiality:** This requirement addresses the need to ensure that data collected by the sensor network will not leak outside and used by unauthorized parties. The unauthorized users should also be prevented to learn the information collected by sensors. Privacy is a special case of confidentiality when the data collected is personal. Data confidentiality can be achieved by cryptographic techniques, such as access control and privacy preserving schemes.

2.2 Threat model

The types of threats in sensor networks would be different from the threats against traditional computer networks. In traditional computer networks, such as the Internet, infrastructures such as routers and DNS servers are usually well protected. Important computers can be kept in physically secure areas, and there will be some level of redundancy and diversity that allows the infrastructure to survive attacks. However, wireless sensor networks are usually deployed with no pre-existing infrastructures. A sensor network is a collection of autonomous nodes that communicate with each other by forming a multi-hop radio network and the connectivity may be maintained in a decentralized manner. Furthermore, sensor networks have very limited resources, such as communication bandwidth and energy, and they are

often deployed in an unattended and potentially unsecured environment. All of these characteristics would bring unique types of threats against sensor networks.

2.2.1 Threat taxonomy

A general way to categorize the threats and attacks on sensor networks is as follows.

- **Insider vs. outsider attackers:** An insider attacker has access to the encryption keys or other secrets used by the network. In contrast, an outsider attacker has no access to the sensor network secrets. Therefore, insider attackers can conduct much more complex attacks and lead to much more severe damages.

- **Passive vs. active attackers:** A passive attacker is interested in collecting sensitive data from the network, which compromises the confidentiality requirement. For example, a passive attacker can eavesdrop packets in transmission or analyze traffic patterns to get sensitive information, which may be used later to launch active attacks. Such kind of attacks are much easier in a wireless network than in a traditional wired network due to the ease to access wireless channels. In contrast, the goal of an active attacker is to disrupt the network functionality and degrade network performance. For example, the attacker can inject false data into the network by pretending to be a legitimate node and mislead the network into making wrong decisions.

2.2.2 Attacks on sensor networks

We further categorize the attacks on sensor networks into three classes: attacks on sensor nodes, on network communication stacks and application protocols.

2.2.2.1 Attacks on sensor nodes

- **Physical tampering:** Current sensor hardware does not provide any resistance to physical tampering due to the requirement of low cost. Once the Motes are physically captured, the attackers can easily extract the cryptographic keys from the hardware. Physical attacks on sensor nodes can be classified into two types: invasive and non-invasive.

  In invasive attacks, the attackers have access to the chip level components of the device, and then use reverse engineering and probing techniques to get unlimited access to the information stored on the chip. In non-invasive attacks, the device is neither opened nor physically tampered with. An example of a non-invasive attack could be the side-channel attack, in which the attackers gather information from the physical implementation of a crypto-system, for example, the attack can get useful information by analyzing power consumption, the timing of the software operation execution or the frequency of the electro magnetic waves.

- **Environment tampering:** In environment tampering, the attackers compromise the integrity of sensor readings by tampering with the environment. For example, an attacker can tamper with the temperature of the environment

around the temperature sensor, or put magnets around a magnetometer.

2.2.2.2 Attacks on network communication stacks

- **Physical layer:** The sensor nodes communicate with each other through wireless radio channels. One important attack on wireless channels is jamming, in which the attackers deliberately use radio noise or signals to disrupt the communications in the network. A common defense against jamming attack is to use some form of spread spectrum communications, e.g., frequency hopping, code spreading, etc. Another attack on the physical layer is eavesdropping, where the attacker can either passively listen to the wireless broadcasts among sensor nodes, or actively send queries to sensor nodes to collect information. Both cryptographic and non-cryptographic methods can be used to defend against the eavesdropping attack. Cryptographic methods include using message encryption to protect message content and using authentication techniques to ensure the message is from the node it claims to be. Non-cryptic methods include generating fake messages into the network that are indistinguishable from the real messages [3].

- **MAC (Medium Access Control) layer:** The MAC layer enables the transmission of message frames through the use of physical channels. It also provides a management interface, controls frame validation, guarantees time slots and handles node associations. Examples of attacks on the MAC layer include deliberately causing collisions with packets in transmission, exhausting nodes'

battery by repeated retransmissions and unfairness in using the wireless channels among neighboring nodes, which could lead to denial of service in sensor networks. A number of solutions have been proposed for defending against these attacks [75].

- **Routing and network layers:** In wireless sensor networks, each node acts as a router, and communicates with others using multi-hop wireless channels. At the same time, routing protocols in sensor networks usually have energy and memory constraints. These unique characteristics could introduce new complex dimensions to the design of routing protocols.

  The most direct attack on the routing and network layers is to target the routing information. By spoofing, altering, or replaying routing information, the attackers would be able to create routing loops, attract or repel network traffic, partition the network, increase end-to-end latency, etc. [43]. For example, a compromised node can spoof or replay an advertisement for an extremely high quality route to the base station, in order to lure traffic from a particular area to go through itself. Such an attack is known as the sinkhole attack. By mounting a sinkhole attack, the attacker could have many opportunities to tamper with application data, or enable many other attacks. For example, the malicious node may refuse to forward certain messages and drop them, ensuring that they are not propagated any further. While suppressing or modifying packets originating from a selected few nodes, it will reliably forward the remaining traffic in order to limit suspicion of its wrongdoing.

By manipulating routing and network layer messages, a malicious node can also mount the Sybil attack, in which a single node presents multiple identities to other nodes in the network. The Sybil attack can significantly reduce the effectiveness of fault-tolerant schemes such as distributed storage, multi-path routing and topology maintenance.

Another important attack on the routing layer is the wormhole attack. In a wormhole attack, the malicious nodes create an illusion that two remote parts of the network are directly connected, by performing a tunneling procedure, i.e., one node will covertly tunnel packets to another colluding node, and the colluding node will replay these packets as it receives them from its physical neighbors. The wormhole attack will affect the shortest path routing and allow the malicious nodes to attract traffic from other parts of the network. The attacker can then analyze network traffic or degrade network performance.

The above mentioned attacks are a few of the important attacks on the routing and network layers. A more complete list can be found in [43].

2.2.2.3 Attacks on application protocols

- **In-network processing:** In-network processing, also known as data aggregation, is an important building block of sensor networks. Many sensor network applications involve a distributed system of sensors measuring the environment from many different points and then somehow aggregating the collected data to form a global view that can be used for the whole network to make

decisions and take actions. The benefits of in-network processing for wireless sensor networks include improved scalability, prolonged lifetime and increased versatility [21]. However, if a few nodes are compromised, they can inject faulty data into the network, which will result in corrupted aggregation. For example, consider a simple scenario where the sensor nodes are deployed to monitor the temperature of the surrounding environment and the aggregation process is to give the average of sensor readings. If a node is compromised and provides much higher or lower readings, it will pull the result in one direction and may lead to false alarms. In more complex applications such as object tracking, the tracking result can deviate arbitrarily even if only one sensor is compromised [81]. In this thesis, we will discuss in Chapter 3 and Chapter 4 how to identify such malicious nodes and increase data integrity during in-network processing in sensor networks.

2.3 Countermeasures

Basic security mechanisms in wireless sensor networks can be divided into three categories: prevention, detection and survivability. This thesis mainly focuses on detection and survivability.

Prevention mechanisms typically rely on cryptographic techniques for authentication and access control. The goal is to prevent the attackers from participating in the communication and compromising message integrity. However, prevention is far from making sensor networks secure as there is always possibility

that powerful attackers may learn the secrets either by breaking the cryptographic techniques or through some other ways.

Detection mechanisms are thus very important to ensure the correct functioning of wireless sensor networks. Intrusion detection is usually considered as the second defense line for protecting a network, since malicious attacks in sensor networks can usually lead to network performance anomalies. For example, the denial of service attack may cause very long transmission latency or high package loss rate. Most of the detection schemes are based on detection of these performance anomalies. However, the constrained resources of wireless sensor networks could bring unique challenges for the design of detection schemes. We will discuss the related issues in Chapter 5 and Chapter 6.

Survivability is the ability of the network to remain in operation despite attacks on the network. We will discuss in Chapter 3 and Chapter 4 how to increase sensor network survivability using the concept of trust.

Part I

Data Trust in Wireless Sensor Networks

Chapter 3

Faulty Data in Sensor Networks

3.1 False data injection

Modern sensor networks provide us with information about phenomena at a much higher level of detail than previously available. In order to make meaningful inference with sensor data, the quality of received data must be ensured. Faulty data in sensor networks can be caused by different reasons. There may be physical sensor problems such as calibration issues, power supply problems, or other unforeseen sensor issues that may cause non-normal sensor readings. However, one of the important sources of false sensor data is due to malicious attacks. The low cost of sensor hardware limits the security design of the nodes. And the unattended operation environment of sensor networks make it possible for attackers to compromise sensors and introduce malicious measurements; the latter is the *false data injection attack* [49]. As a first step to tackle the false data injection attack, we will study an example in the distributed state estimation problem using sensor networks. The case study will clearly demonstrate the severe performance degradation caused by false data injection and illustrate how to defend against the attack.

3.2 Case study: distributed state estimation

In a distributed sensor network, sensors are deployed to monitor some phenomenon and the data gathered by sensors can be analyzed to provide state estimation of the monitored phenomenon. For instance, a surveillance system can track objects based on different kinds of signals measured by sensors, such as acoustic and video signals. The output of state estimation may include positions, speeds, moving directions of the objects, which can be used to guide the response algorithm of the network. If the sensor data are contaminated, the false information can affect the outcome of state estimation and mislead the network to make wrong decisions.

We consider a sensor network composed of n sensors. The monitored phenomenon is modeled by a linear random process $\mathbf{x}(k+1) = A\mathbf{x}(k)+\mathbf{w}(k)$. $\mathbf{x}(k) \in R^m$ is the state vector with dimension m, $\mathbf{w}(k) \in R^m$ is the noise, which accounts for modeling errors, uncertainties or perturbations to the process and is assumed to be Gaussian with zero mean and covariance matrix Q. The initial state $\mathbf{x}_0$ has a Gaussian distribution with mean $\boldsymbol{\mu}_0$ and covariance matrix M_0. Such a linear random process, for example, can represent a moving object in the field monitored by the sensor network. We assume that each sensor can sense this process and the sensing model for each sensor is given by $\mathbf{y}_i(k) = H_i\mathbf{x}(k) + \mathbf{v}_i(k)$, where $\mathbf{y}_i(k) \in R^{p_i}$ is the p_i-dimensional observation made by sensor i, and $\mathbf{v}_i(k) \in R^{p_i}$ is the measurement noise, assumed to be Gaussian with zero mean and covariance V_i. In distributed sensing, the measurements of an individual sensor may only contain partial information of the state, so state estimation requires the cooperation of sensors in the

network.

In distributed state estimation, the goal of each sensor is to compute an accurate estimation of the state $\mathbf{x}(k)$ using its local measurements and the information received from its communication neighbors. Distributed Kalman Filtering (DKF) provides a basic solution for this problem. The main idea of DKF [59, 60] is to use a standard Kalman filter locally by each sensor, together with a consensus step to ensure that the local estimates agree. However, existing DKF algorithms are derived in a benign setting without considerations of possible malicious attacks. Therefore they are not robust against the false data injection attack. In what follows, we use the DKF algorithm introduced in [60] as a basic algorithm to build our robust distributed estimation algorithms.

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with an adjacency matrix $[g_{ij}]$ that specifies the topology of the sensor network, and $\mathcal{N}_i = \{j \in \mathcal{V} | g_{ij} \neq 0\}$ be the communication neighborhood of sensor i, i.e., the subset of sensors with whom i can exchange information. Let $\mathcal{J}_i = \mathcal{N}_i \cup \{i\}$ denote the inclusive neighbors of node i and $\mathbf{y}(k) = [\mathbf{y}_1(k), \ldots, \mathbf{y}_n(k)]^T$ denote the measurements obtained from sensor 1 to sensor n at time k. Then given the observations $\mathbf{y}(1:k) = \{\mathbf{y}(1), \ldots, \mathbf{y}(k)\}$ at all sensors from time 0 to time k, we denote the state estimation at node i as follows,

$$\begin{aligned} \hat{\mathbf{x}}_i(k) &= \mathrm{E}[\mathbf{x}(k)|\mathbf{y}(1{:}k)],\ \bar{\mathbf{x}}_i(k) = \mathrm{E}[\mathbf{x}(k)|\mathbf{y}(1{:}k-1)], \\ \hat{M}_i(k) &= \mathrm{Cov}(\mathbf{x}(k)|\mathbf{y}(1{:}k)),\ \bar{M}_i(k) = \mathrm{Cov}(\mathbf{x}(k)|\mathbf{y}(1{:}k-1)), \end{aligned}$$

where $\hat{\mathbf{x}}_i(k)$ and $\hat{M}_i(k)$ are the estimates of $\mathbf{x}_i(k)$ and the covariance matrix given

observations $\mathbf{y}(1:k)$, and $\bar{\mathbf{x}}_i(k)$ and $\bar{M}_i(k)$ are the corresponding estimations given observations $\mathbf{y}(1:k-1)$.

The basic algorithm in [60] is described in Table 3.1, where ϵ is a small positive constant. For simplicity, we omitted the time index. In each round, node i only needs to broadcast the message $(\mathbf{u}_i, U_i, \bar{\mathbf{x}}_i)$ to its neighboring nodes, where U_i is the locally aggregated data.

Table 3.1: Algorithm I – Basic DKF Algorithm

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i = \boldsymbol{\mu}_0$
2. For each node in the network
3. While new data comes do
4. Locally aggregate data and covariance matrices:
$\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T V_j^{-1} \mathbf{y}_j,\ \mathbf{z}_i = \sum_{j\in\mathcal{J}_i} \mathbf{u}_j,$
$U_j = H_j^T V_j^{-1} H_j,\ S_i = \sum_{j\in\mathcal{J}_i} U_j$
5. Compute the Kalman-consensus estimate:
$\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1},$
$\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i) + \epsilon\hat{M}_i \sum_{j\in\mathcal{N}_i}(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)$
6. Update the state of the Kalman-consensus filter:
$\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i = A\hat{\mathbf{x}}_i$
7. end while
8. end for

We model the false data injection attack using an attack vector

$$\mathbf{a}(k) = [\mathbf{a}_1(k), \ldots, \mathbf{a}_n(k)]^T,$$

which represents the false measurements that the attackers add to the original sensor measurements. In other words, let $\mathbf{y}^a$ represent the vector of measurements that may contain malicious data, then we have $\mathbf{y}^a(k) = \mathbf{y}(k) + \mathbf{a}(k)$. The i^{th} element of $\mathbf{a}(k)$ being non-zero means that the attacker compromises the i^{th} sensor and replaces

its original measurement $\mathbf{y}_i(k)$ with a false measurement $\mathbf{y}_i(k)+\mathbf{a}_i(k)$. The attacker can choose any arbitrary vector as $\mathbf{a}(k)$. To illustrate the effect of this attack on the outcome of state estimation, we study an example where the attack vector is drawn from a multi-variate Gaussian distribution with mean $\boldsymbol{\mu}^a = [\boldsymbol{\mu}_1^a, \ldots, \boldsymbol{\mu}_n^a]^T$ and covariance matrix $V^a = \text{diag}(V_1^a, \ldots, V_n^a)$. The distributed estimation of states $\mathbf{x}(k)$ follows the procedure in Table 3.2, where $\hat{\mathbf{x}}^a(k)$ denotes the estimated states using $\mathbf{y}^a(1:k)$, and $\hat{V}_i = V_i + V_i^a$.

Table 3.2: Algorithm II – DKF under a Gaussian attack vector

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i^a = \boldsymbol{\mu}_0$
2. For each node in the network,
3. While new data exists do
4. Locally aggregate data and covariance matrices:
   $\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T \hat{V}_j^{-1}(\mathbf{y}_j^a - \boldsymbol{\mu}_j^a),\ \mathbf{z}_i = \sum_{j\in\mathcal{J}_i} \mathbf{u}_j,$
   $U_j = H_j^T \hat{V}_j^{-1} H_j,\ S_i = \sum_{j\in\mathcal{J}_i} U_j$
5. Compute the Kalman-consensus estimate:
   $\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1},$
   $\hat{\mathbf{x}}_i^a = \bar{\mathbf{x}}_i^a + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i^a) + \epsilon\hat{M}_i \sum_{j\in\mathcal{N}_i}(\bar{\mathbf{x}}_j^a - \bar{\mathbf{x}}_i^a)$
6. Update the state of the Kalman-consensus filter:
   $\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i^a = A\hat{\mathbf{x}}_i^a$
7. end while
8. end for

Comparing the Algorithms in Table 3.1 and Table 3.2, we can see that when the sensor nodes are not aware of the false data injection attack, even if only one sensor is compromised, the state estimation $\hat{\mathbf{x}}^a(k)$ can deviate from the true state $\mathbf{x}(k)$ arbitrarily or be arbitrarily noisy. This verifies that the false data injection attack can cause serious errors.

3.3 Bayesian detection of faulty data

The energy and cost constraints restrict the deployment of tamper resistant hardware for the whole network. Therefore, once a node is compromised, all the security information stored in the node will be exposed to the attackers. The normal nodes cannot distinguish the attackers by only using cryptographic techniques. Most existing works use *majority voting* based schemes against the false data injection attack. Ye et al. [76] presented a statistical en-route filtering mechanism in which the truthfulness of data from each node is determined via collective decision making by multiple nodes. Zhu et al. [84] proposed an interleaved hop-by-hop authentication scheme where pairwise keys are established between nodes $t + 1$ hops away. In their scheme, up to t compromised nodes can be tolerated. In [71], Shukla et al. presented a secure statistical scheme to distinguish data transience from false injection in a clustered sensor network by utilizing statistical digests sent from each sensor. For additional results, see [79, 77, 61]. All of these works rely on a hierarchical architecture and central coordination among sensors, which limits the sensor network scalability due to the large number of nodes and the volatility of the network. In the following section, we propose a distributed state estimation method with Bayesian learning, which achieves performance similar to a centralized majority voting without causing extra communication overhead.

3.3.1 The algorithm

Our DKF algorithm with Bayesian learning is inspired by the observation, in Algorithm II, that the true system state can be accurately estimated if the attack parameters are known. In reality, the attack parameters are unknown, but we can use Bayesian learning to estimate them and remove the false measurements. Then we can follow the same procedure of Algorithm II to obtain more reliable state estimates.

Bayesian learning usually involves highly complicated computational techniques such as Markov Chain Monte Carlo, which are not suitable for sensor networks due to the resource constraints. We propose an algorithm based on variational Bayesian learning, which provides approximate but efficient estimation of the marginal likelihood of probabilistic models that have latent variables or incomplete data [8].

3.3.1.1 Variational Bayesian EM algorithm

In this section, we briefly introduce the variational Bayesian EM algorithm. Assuming that $\mathbf{y}$ is the observed data, $\mathbf{x}$ represents the latent variables, and $\boldsymbol{\theta}$ is the model parameters, variational Bayesian learning constructs and optimizes a lower bound on the marginal likelihood $f(\mathbf{y})$ using variational calculus. More specifically, $\log f(\mathbf{y})$ is lower bounded by first introducing any distribution, over both latent variables and parameters, which has the same support as $f(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y})$ does, and then

applying Jensen's inequality [8], i.e.,

$$\log f(\mathbf{y}) = \log \int f(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) d\mathbf{x} d\boldsymbol{\theta} \geq \int q(\mathbf{x}, \boldsymbol{\theta}) \log \frac{f(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})}{q(\mathbf{x}, \boldsymbol{\theta})} d\mathbf{x} d\boldsymbol{\theta}.$$

Maximizing the lower bound with respect to $q(\mathbf{x}, \boldsymbol{\theta})$ results in $q(\mathbf{x}, \boldsymbol{\theta}) = f(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y})$, which does not simplify the problem, as evaluating the true posterior distribution $f(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y})$ requires knowing the normalizing factor, i.e., the marginal likelihood. Therefore, a simple factorized approximation $q(\mathbf{x}, \boldsymbol{\theta}) \approx q_{\mathbf{x}}(\mathbf{x}) q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ is used, and the variational Bayesian algorithm iteratively maximizes the above lower bound of $\log f(\mathbf{y})$ with respect to the free distributions $q_{\mathbf{x}}(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$. Elementary variational calculus leads to the following updating rules [8]:

$$\begin{aligned} q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) &\propto \exp[\int \log f(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta}) q_{\boldsymbol{\theta}}^{(t)}(\boldsymbol{\theta}) d\boldsymbol{\theta}], \\ q_{\boldsymbol{\theta}}^{(t+1)}(\boldsymbol{\theta}) &\propto f(\boldsymbol{\theta}) \exp[\int \log f(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) d\mathbf{x}]. \end{aligned}$$

Since $q_{\mathbf{x}}(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ are coupled, the algorithm would iterate these equations until convergence. The variational Bayesian EM algorithm has been studied with a particular class of graphical models with latent variables, namely, the conjugate-exponential models [8], which satisfy the following two conditions:

Condition (1): *The complete data likelihood is that of an exponential family* $p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = f(\mathbf{x}, \mathbf{y}) g(\boldsymbol{\theta}) exp\{\phi(\boldsymbol{\theta})^T u(\mathbf{x}, \mathbf{y})\}$*, where* $\phi(\boldsymbol{\theta})$ *is the vector of natural parameters, and* u*,* f *and* g *are functions that define the exponential family.*

Condition (2): *The parameter prior is conjugate to the complete data likelihood:*

$p(\boldsymbol{\theta}|\boldsymbol{\eta},\boldsymbol{\nu}) = h(n,\boldsymbol{\nu})g(\boldsymbol{\theta})^{\boldsymbol{\eta}}exp\{\phi(\boldsymbol{\theta})^T\boldsymbol{\nu}\}$, *where* $\boldsymbol{\eta}$ *and* $\boldsymbol{\nu}$ *are the hyper-parameters.*

The following theorem [8] has been well established for the conjugate-exponential model.

Theorem 3.3.1. *Given i.i.d. data set* $\mathbf{y} = \{y_1, \ldots, y_n\}$*, for the conjugate-exponential model, at every iteration of the variational Bayesian EM algorithm and at the maxima of* $\mathcal{F}(q_{\mathbf{x}}(\mathbf{x}), q_{\boldsymbol{\theta}}(\boldsymbol{\theta}), \mathbf{y})$*:*

(a) $q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ *is conjugate with parameters* $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta} + n$, $\tilde{\boldsymbol{\nu}} = \boldsymbol{\nu} + \sum_{i=1}^{n} \bar{u}(\mathbf{y_i})$*:*

$$q_{\boldsymbol{\theta}}(\boldsymbol{\theta}) = h(\tilde{\boldsymbol{\eta}}, \tilde{\boldsymbol{\nu}})g(\boldsymbol{\theta})^{\tilde{\boldsymbol{\eta}}}exp\{\phi(\boldsymbol{\theta})^T\tilde{\boldsymbol{\nu}}\}$$

where $\bar{u}(\mathbf{y_i}) = E_{q_{x_i}}[u(x_i, y_i)]$*, and* $E_{q_{x_i}}$ *denotes the expectation under the variational posterior over the latent variables associated with the* i^{th} *data.*

(b) $q_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{n} q_{x_i}(x_i)$ *with*

$$q_{x_i}(x_i) = p(x_i|y_i, \bar{\phi}) \approx f(x_i, y_i)exp\{\bar{\phi}^T u(x_i, y_i)\}$$

where $\bar{\phi} = E_{q_{\boldsymbol{\theta}}}[\phi(\boldsymbol{\theta})]$ *is the expectation of the natural parameters.*

3.3.1.2 DKF with variational Bayesian learning

In our DKF model, the latent variables are the system states to be estimated $\mathbf{x}(1:k) = \{\mathbf{x}(1), \ldots, \mathbf{x}(k)\}$, and the observations are the malicious measurements $\mathbf{y}^a(1:k) = \{\mathbf{y}^a(1), \ldots, \mathbf{y}^a(k)\}$. In general, the attack vector can be modeled by a Gaussian mixture model (GMM), as GMM comprises a finite or infinite num-

ber of different Gaussian distributions that can describe different features of data. To simplify the problem and focus on algorithm performance, we assume that the attack vector is drawn from a single multi-variate Gaussian distribution with parameters $\boldsymbol{\mu}^a$ and V^a and the attack signals at each sensor are uncorrelated, so V^a is a block-diagonal matrix. Let $\hat{V} = diag(V_1 + V_1^a, \ldots, V_n + V_n^a)$ and denote the model parameters by $\boldsymbol{\theta} = \{\boldsymbol{\mu}^a, \hat{V}\}$, the marginal likelihood of the model is

$$f(\mathbf{y}^a(1{:}k)) = \int \prod_{i=1}^{n} f(\mathbf{y}_i^a(1{:}k)|\mathbf{x}(1{:}k), \boldsymbol{\theta}) f(\mathbf{x}(1{:}k)) f(\boldsymbol{\theta}) d\mathbf{x} d\boldsymbol{\theta}.$$

Denoting the Gaussian distribution by $N(\cdot)$, we have

$$\begin{aligned} f(\mathbf{y}_i^a(1:k)|\mathbf{x}(1:k), \boldsymbol{\theta}) &= \prod_{j=1}^{k} N(H_i \cdot \mathbf{x}(j) + \boldsymbol{\mu}_i^a, \hat{V}_i), \\ f(\mathbf{x}(1:k)) &= \prod_{j=1}^{k} N(A \cdot \mathbf{x}(j-1), Q). \end{aligned}$$

Furthermore, we assume that $\boldsymbol{\mu}^a$ and $\hat{V}$ have conjugate priors, so the posterior probability distribution of $\boldsymbol{\theta}$ will be in the same family as the prior probability distribution, i.e.,

$$\boldsymbol{\mu}^a|(\boldsymbol{\mu}^0, \rho^0, \hat{V}) \sim N(\boldsymbol{\mu}^0, \hat{V}/\rho^0), \quad \hat{V}^{-1}|(\gamma^0, B) \sim W(\gamma^0, B^{-1}/\gamma^0),$$

where $\boldsymbol{\mu}^0$, ρ^0, γ^0 and B are hyper-parameters and $W(\cdot)$ represents the Wishart distribution. Since the distributions $f(\mathbf{y}^a(1:k))$, $f(\mathbf{x}(1:k))$ and $f(\boldsymbol{\theta})$ all belong to the exponential family, following Theorem 3.3.1, we can obtain the updating rules for the attack parameters $\tilde{\boldsymbol{\mu}}_i^a = \mathrm{E}[\boldsymbol{\mu}_i^a|\mathbf{y}(1:k)]$ and $\tilde{V}_i(k) = \mathrm{E}[\hat{V}_i|\mathbf{y}(1:k)]$ as

in equations (3.1) and (3.2). For simplicity, we assume B is diagonal, that is, we assume that the measurement noise is uncorrelated in each dimension.

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_i^a(k) &= \frac{\rho^0 \boldsymbol{\mu}_i^0 + \sum_{j=1}^{k} [\mathbf{y}_i(j) - H_i \cdot \hat{\mathbf{x}}_i(j)]}{\rho^0 + k}, \qquad (3.1) \\
\tilde{V}_i(k) &= \frac{[\mathrm{diag}(\rho^0 \boldsymbol{\mu}_i^0 + \sum_{j=1}^{k} \mathbf{y}_i(j) - H_i \hat{\mathbf{x}}_i(j))]^2 + \gamma^0 B_i}{\gamma^0 + k} \\
&+ \frac{\rho^0 \cdot [\mathrm{diag}(\boldsymbol{\mu}_i^0)]^2 + \sum_{j=1}^{k} [\mathrm{diag}(\mathbf{y}_i(j) - H_i \hat{\mathbf{x}}_i(j))]^2}{\gamma^0 + k}, \qquad (3.2)
\end{aligned}$$

where $diag(\mathbf{v})$ represents a diagonal matrix whose diagonal entries are the components of vector $\mathbf{v}$. Equations (3.1) and (3.2) require all data until time k to compute $\tilde{\boldsymbol{\mu}}_i^a(k)$ and $\tilde{V}_i(k)$, so they can not be computed online. We rewrite them in a form using only current observations by collecting sufficient statistics, i.e.,

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_i^a(k+1) &= \frac{\rho(k) \bar{\boldsymbol{\mu}}_i^a(k) + \mathbf{y}_i^a(k) - H_i \hat{\mathbf{x}}_i(k)}{\rho(k) + 1}, \\
\tilde{V}_i(k+1) &= \frac{\gamma(k) \bar{V}_i(k) + \rho(k) [\mathrm{diag}(\bar{\boldsymbol{\mu}}_i^a(k))]^2}{\gamma(k) + 1} \\
&- \frac{(\rho(k) + 1)[\mathrm{diag}(\tilde{\boldsymbol{\mu}}_i^a(k))]^2 + [\mathrm{diag}(\mathbf{y}_i(k) - H_i \hat{\mathbf{x}}_i(k))]^2}{\gamma(k) + 1}, \\
\rho(k+1) &= \rho(k) + 1, \\
\gamma(k+1) &= \gamma(k) + 1.
\end{aligned}$$

Our DKF algorithm with variational Bayesian learning is presented in Table 3.3.

Table 3.3: Algorithm III – DKF Algorithm with Bayesian Learning

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i = \boldsymbol{\mu}_0$, $\bar{V} = B$,
   $\bar{\boldsymbol{\mu}}^a = \boldsymbol{\mu}^0$, $\rho = \rho^0$, $\gamma = \gamma^0$
2. For each node in the network
3. While new data exists do
4. Locally aggregate data and covariance matrices:
   $\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T \bar{V}_j^{-1}(\mathbf{y}_j^a - \bar{\boldsymbol{\mu}}_j^a),\ \mathbf{z}_i = \sum_{j\in\mathcal{J}_i} \mathbf{u}_j,$
   $U_j = H_j^T \bar{V}_j^{-1} H_j,\ S_i = \sum_{j\in\mathcal{J}_i} U_j$
5. Compute the Kalman-consensus estimate:
   $\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1}$,
   $\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i) + \epsilon\hat{M}_i \sum_{j\in\mathcal{N}_i}(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)$
6. Compute the attack parameter estimate:
   $\tilde{\boldsymbol{\mu}}_i^a = (\rho\bar{\boldsymbol{\mu}}_i^a + \mathbf{y}_i^a - H_i\hat{\mathbf{x}}_i)/(\rho+1)$,
   $\tilde{V}_i = (\gamma\bar{V}_i + \rho[\mathrm{diag}(\bar{\boldsymbol{\mu}}_i^a)]^2 - (\rho+1)[\mathrm{diag}(\tilde{\boldsymbol{\mu}}_i^a)]^2 + [\mathrm{diag}(\mathbf{y}_i - H_i\hat{\mathbf{x}}_i)]^2)/(\gamma+1)$
7. Update the state of the Kalman-consensus filter:
   $\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i = A\hat{\mathbf{x}}_i,\ \rho = \rho + 1,$
   $\gamma = \gamma + 1,\ \bar{\boldsymbol{\mu}}_i^a = \tilde{\boldsymbol{\mu}}_i^a,\ \bar{V}_i = \tilde{V}_i$
8. end while
9. end for

3.3.2 Performance evaluation

To evaluate the performance of our DKF algorithm with Bayesian learning, we study the application where sensors are deployed to estimate the position of a moving object in a R^2 plane that approximately moves in circles. The output matrix is $H_i = I_2$ and the state of the process dynamics is 2-dimensional corresponding to the continuous-time system $\dot{\mathbf{x}} = A_0\mathbf{x} + C_0\mathbf{w}$ with $A_0 = [0, -1; 1, 0]$ and $C_0 = 50I_2$. We use the discrete-time model with step-size $\zeta = 0.02$, i.e., $\mathbf{x}(k+1) = A\mathbf{x}(k) + C\mathbf{w}(k)$ with parameters $A = I_2 + \zeta A_0 + \zeta^2 A_0^2/2 + \zeta^3 A_0^3/6$ and $C = \zeta C_0$. The sensor networks used in our experiments consists of 100 randomly located nodes. The attack vectors are drawn from Gaussian distributions with different mean vectors $\boldsymbol{\mu}^a$ and covariance matrices V^a.

There are two main observations in our experiments. First, the algorithm is very robust under attacks that have mean vector $\boldsymbol{\mu}^a = \mathbf{0}$. No matter what is the value of V^a and how many sensors are compromised, as long as there is one sensor telling truth, the network always provides accurate estimates. Second, the algorithm performs as *majority voting* when $\boldsymbol{\mu}^a \neq \mathbf{0}$. Figure 3.1 shows an example. In Figure 3.1, the attacker try to deviate the estimation result by 10

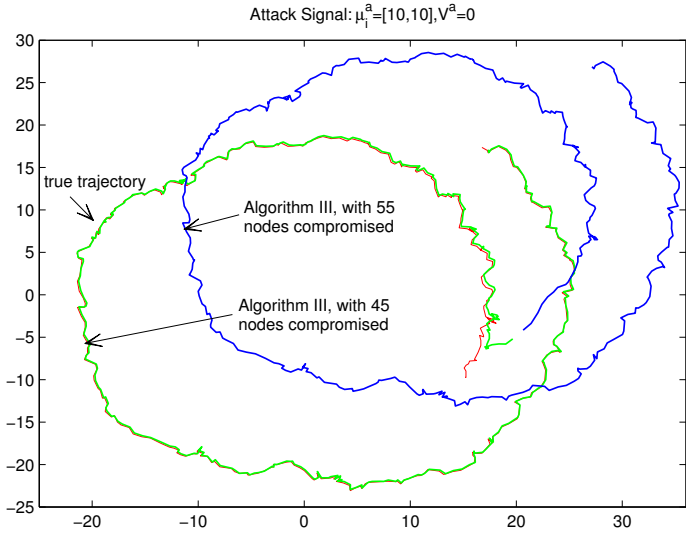


Figure 3.1: Perfomrance of DKF with Bayesian learning

in both directions, i.e., $\boldsymbol{\mu}_i^a = [10, 10]$ and $V^a = \text{diag}([0, 0])$. It's found that the performance is very good when 45 sensors are compromised, but degrades abruptly when 55 sensors are compromised. The reason for the two observations is that in Algorithm III the nodes use all the data received for estimation and resolve the inconsistency among data by adjusting the estimated mean and covariance matrix of the measurement noise. When $\boldsymbol{\mu}^a = 0$, the algorithm handles the perturbation on the covariance matrix well since it knows that large noise covariance matrix indicates low accuracy. But when the perturbation is on the mean value, the algorithm cannot

identify compromised nodes and only the majority voting rule can be used.

One limitation of majority voting is that the network may be totally subverted when more than half of the sensors are compromised. In the next Chapter, we will discuss how to use the concept of trust to overcome this limitation and ensure network robustness even when more than half of the sensors are compromised.

Chapter 4

Enhancing Data Integrity using Trusted Cores

4.1 Motivation

Trust has been widely used for many applications, including e-commerce, peer to peer networks, web-based services and distributed computing [39]. Incorporating the concept of trust in wireless sensor networks has recently attracted considerable research attention [24]. While traditional security methods are inadequate or too complicated to protect the sensor networks, trust-based methods provide complementary security mechanisms.

Trust can appear in various ways and meanings in wireless sensor networks. It can refer to the trustworthiness of a sensor, meaning whether the sensor has been compromised. It can also refer to the trustworthiness of the data transmitted by a sensor, or refer to the trustworthiness of a link, e.g., whether it is jammed. The trust information can help different functionality of the network, such as routing, data aggregation, and malicious node detection. There are also various ways to numerically represent trust weights. Both continuous and discrete numerical values are used in the literature. In [53], the trustworthiness of a certificate is described by a continuous value in $[0, 1]$. In [37], trust opinion is represented by a triplet in $[0, 1]^3$, where the elements in the triplet represent belief, disbelief and uncertainty respectively. Trust has also been interpreted as probability, e.g., subjective probability is

used in [38], while objective probability is used in [40].

In our work, we present a special construction of a trust system in sensor networks, namely, the trusted cores. The trusted cores are a special class of nodes that have much higher levels of security than other regular sensor nodes in the network. The motivation is that we may achieve better security for the entire network at the cost of a small subset of highly secured nodes.

4.2 The general model

4.2.1 Computational model for trust

We consider two types of trust in our model: local trust and global trust. The global trust weight assigned to a node is a unique vector, independent of other nodes who are evaluating the node. A local trust weight from one node to another node provides the personalized trust value that depends on the opinion of the evaluating node. To handle the uncertainty on the possible existence of malicious nodes, we assume that each node is in one of two states: normal or malicious. We denote the state space of a node by $\mathcal{S} = \{S_1, S_2\}$, where S_1 is the normal state and S_2 is the malicious state. The true state of node i is denoted by $S(i)$. Then, the local trust opinion of node i on node j is defined by the transition probability matrix:

$$P_{i,j}=\begin{bmatrix} \Pr(S(i)=S_1|S(j)=S_1) & \Pr(S(i)=S_1|S(j)=S_2) \\ \Pr(S(i)=S_2|S(j)=S_1) & \Pr(S(i)=S_2|S(j)=S_2) \end{bmatrix}.$$

The transition probability matrix $P_{i,j}$ is actually a column stochastic matrix, i.e., the sum of elements in each column is equal to 1. We will use $P_{i,j}(m,n)$ to describe the element in the m^{th} row and n^{th} column of matrix $P_{i,j}$. We model $P_{i,j}(m,n)$ by a logistic function that depends on the Euclidean distance between the measurements from two neighboring nodes i and j. We assume that neighboring nodes have similar observations, so the larger the distance between the neighboring measurements, the smaller the local trust values. Let i and j be neighbors that exchange the measurements $\mathbf{y}_i^a(k)$ and $\mathbf{y}_j^a(k)$ at the k^{th} step, and denote $d_{ij}(k) = ||\mathbf{y}_i^a(k) - \mathbf{y}_j^a(k)||_2$. We compute the conditional probability by

$$P_{ij}(1,1) = P_{i,j}(2,2) = f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k)) = \frac{2e^{-d_{ij}(k)/\alpha}}{1 + e^{-d_{ij}(k)/\alpha}}.$$

The reason that we set $P_{ij}(1,1) = P_{i,j}(2,2)$ is that there is no prior information on whether the node is in a normal state or a malicious state, so we treat the two cases as equally likely. The function $f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k))$ is actually a 'soft' step function, i.e., when $d_{ij}(k) = 0$, it equals to 1 and when $d_{ij}(k) = +\infty$, it equals to 0. The parameter α controls the slope and cutoff value of the function.

To accumulate previous experience into the local trust opinion, we update the matrix $P_{i,j}^{(k)}$ at the k^{th} step by

$$P_{i,j}^{(k)} = (1-\beta) \begin{bmatrix} f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k)) & 1 - f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k)) \\ 1 - f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k)) & f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k)) \end{bmatrix} + \beta P_{i,j}^{(k-1)}$$

where β is a time discount factor.

The global trust value for node i is defined to be a 2×1 vector $\mathbf{g}_i$, which represents the probability of node i in one of the two states, i.e.,

$$\mathbf{g}_i = [\Pr(S(i) = S_1), \Pr(S(i) = S_2)]^T.$$

Given node j's global trust value, node i's posterior probability to be in one of the two states can be computed by $P_{i,j} \cdot \mathbf{g}_j$. Assuming a uniform prior for node i's neighboring nodes, the estimation of $\mathbf{g}_i$ can be written as

$$\mathbf{g}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} P_{i,j} \cdot \mathbf{g}_j.$$

As the global trust values for the trusted core nodes are known, we can propagate the global trust from them to other regular nodes in the network. Denote the set of indices of the trusted core nodes by $\mathcal{T}$ and the set of indices of other regular nodes by $\mathcal{U}$. Without loss of generality, we assume that the trusted core nodes are indexed from 1 to t. Let $\mathbf{g}_\mathcal{T} = [\mathbf{g}_1^T, \ldots, \mathbf{g}_t^T]^T$, $\mathbf{g}_\mathcal{U} = [\mathbf{g}_{t+1}^T, \ldots, \mathbf{g}_n^T]^T$, $D_{\mathcal{U}\mathcal{U}} = diag(|\mathcal{N}_{t+1}|, \ldots, |\mathcal{N}_n|)$, $P_{\mathcal{U}\mathcal{U}}$ be the probability transition matrix from $\mathcal{U}$ to $\mathcal{U}$, i.e.,

$$P_{\mathcal{U}\mathcal{U}} = \begin{bmatrix} 0 & P_{t+1,t+2} & \ldots & P_{t+1,n} \\ P_{t+2,t+1} & 0 & \ldots & P_{t+2,n} \\ \vdots & \ddots & \ddots & \vdots \\ P_{n,t+1} & \ldots & P_{n,n-1} & 0 \end{bmatrix},$$

and $P_{\mathcal{UT}}$ be the probability transition matrix from $\mathcal{U}$ to $\mathcal{T}$, i.e.,

$$P_{\mathcal{UT}} = \begin{bmatrix} P_{t+1,1} & P_{t+1,2} & \dots & P_{t+1,t} \\ P_{t+2,1} & P_{t+2,2} & \dots & P_{t+2,t} \\ \vdots & \ddots & \ddots & \vdots \\ P_{n,1} & \dots & P_{n,t-1} & P_{n,t} \end{bmatrix}.$$

Then the computation of the global trust values for nodes in $\mathcal{U}$ can be written in a matrix form, i.e.,

$$\mathbf{g}_{\mathcal{U}} = D_{\mathcal{UU}}^{-1} P_{\mathcal{UU}} \cdot \mathbf{g}_{\mathcal{U}} + D_{\mathcal{UU}}^{-1} P_{\mathcal{UT}} \cdot \mathbf{g}_{\mathcal{T}}. \tag{4.1}$$

It can be shown that the spectral radius $\rho(D_{\mathcal{UU}}^{-1} P_{\mathcal{UU}})$ is less than 1, as long as there exist some paths that connect each regular node to one of the trusted cores. The proof follows a similar procedure as in [15], which is given below.

Assuming that for each regular node $i \in \mathcal{U}$, the number of i's neighbors belonging to $\mathcal{U}$ is denoted by u_i and the number of its neighbors belonging to $\mathcal{T}$ is denoted by t_i, i.e., $|\mathcal{N}_i| = u_i + t_i$. Consider the eigenvalue equation

$$D_{\mathcal{UU}}^{-1} P_{\mathcal{UU}} \cdot \boldsymbol{\mu} = \lambda \boldsymbol{\mu}, \tag{4.2}$$

where $\boldsymbol{\mu}$ is a vector whose i^{th} element is a 2×1 vector. Let $\boldsymbol{\mu}_i$ be the component of $\boldsymbol{\mu}$ that has the largest L_1 norm. Consider the corresponding detailed equations in

equation (4.2), i.e.,

$$\frac{1}{u_i + t_i}(\sum_{j \in \mathcal{N}_i \cap \mathcal{U}} P_{ij}\boldsymbol{\mu}_j) = \lambda \boldsymbol{\mu}_i. \tag{4.3}$$

Since we have

$$\begin{aligned} ||\sum_{j \in \mathcal{N}_i \cap \mathcal{U}} P_{ij}\boldsymbol{\mu}_j||_1 &\leq \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} ||P_{ij}\boldsymbol{\mu}_j||_1 \leq \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} ||P_{ij}||_1 ||\boldsymbol{\mu}_j||_1 \\ &= \sum_{j \in \mathcal{N}_i \cap \mathcal{U}} ||\boldsymbol{\mu}_j||_1 \leq u_i ||\boldsymbol{\mu}_i||_1 \end{aligned}$$

By taking the L_1 norm of equation (4.3) we have $|\lambda| \cdot ||\boldsymbol{\mu}_i||_1 \leq \frac{u_i}{u_i + t_i} ||\boldsymbol{\mu}_i||_1$. Therefore $|\lambda| \leq \frac{u_i}{u_i + t_i}$. If $t_i > 0$, i.e., node i has trusted core(s) as neighbor(s), we have $\lambda < 1$. If $t_i = 0$, we prove $\lambda < 1$ by contradiction. Assuming that $|\lambda| = 1$, we must have $||\boldsymbol{\mu}_j||_1 = ||\boldsymbol{\mu}_i||_1$ in $\sum_{j \in \mathcal{N}_i \cap \mathcal{U}} P_{ij}\boldsymbol{\mu}_j$ of equation (4.3). Then for each $j \in \mathcal{N}_i \cap \mathcal{U}$, we can repeat the above arguments. If there exists a path that connects node i and a trusted core, we will hit a regular node k that has at least one trusted core as a neighbor, i.e., $t_k > 0$, which means that $|\lambda| < 1$. Therefore, for a network that there exist some paths that can connect each regular node to a trusted core node, we have $\rho(D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{U}}) < 1$.

Since $\rho(D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{U}}) < 1$, according to equation (4.1), we have

$$\mathbf{g}_{\mathcal{U}} = (I - D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{U}})^{-1} \cdot D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{T}} \cdot \mathbf{g}_{\mathcal{T}}. \tag{4.4}$$

Using Theorem 4.2.1 [27] in the following statement, we can write equation (4.4) in

an iterative way.

Theorem 4.2.1. *Suppose $b \in R^d$ and $\Delta \doteq M - N \in R^{d\times d}$ is nonsingular. If M is nonsingular and the spectral radius of $M^{-1}N$ satisfies the inequality $\rho(M^{-1}N) < 1$, then the iterates of $x^{(k)}$ defined by $Mx^{(k+1)} = Nx^{(k)} + b$ converge to $x = \Delta^{-1}b$ for any starting vector of x.*

Based on Theorem 4.2.1, the following Jacobi iteration will converge to the solution of equation (4.4), i.e.,

$$\forall i \in \mathcal{U},\ \mathbf{g}_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|}[\sum_{j\in\mathcal{U}, j\in\mathcal{N}_i} P_{i,j}\mathbf{g}_j^{(k)} + \sum_{l\in\mathcal{T}, l\in\mathcal{N}_i} P_{i,l}\mathbf{g}_l], \tag{4.5}$$

which provides the distributed computation of a node's global trust values.

The algorithm of global trust aggregation is described in Table 4.1. The con-

Table 4.1: Global trust aggregation algorithm

1. for each node i do
2. Query all nodes $j \in \mathcal{N}_i$ for $g_j^{(0)}$;
3. do
4. Compute $g_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|}[\sum_{j\in\mathcal{U}, j\in\mathcal{N}_i} P_{i,j}\mathbf{g}_j^{(k)} + \sum_{l\in\mathcal{T}, l\in\mathcal{N}_i} P_{i,l}\mathbf{g}_l]$;
5. Send $g_i^{(k+1)}$ to all nodes $j \in \mathcal{N}_i$;
6. Compute $\delta = |g_i^{(k+1)} - g_i^{(k)}|$;
7. Wait for all nodes $j \in \mathcal{N}_i$ to return g_j^{k+1}
8. until $\delta < \epsilon$
9. end for

vergence speed of the global trust aggregation depends on the graph structure. More specifically, it is related to the second largest eigenvalue of the matrix $D_{\mathcal{U}\mathcal{U}}^{-1}P_{\mathcal{U}\mathcal{U}}$ [31]. Experimental results show that this computation converges very fast, usually in less than 10 iterations. Figure 4.1 shows the convergence of equation (4.5) in several

different networks. These networks are randomly generated with uniform sensor deployment with size 100. In each network, 10 randomly selected sensors are set as the trusted core nodes with global trust value $[0.95, 0.05]^T$.

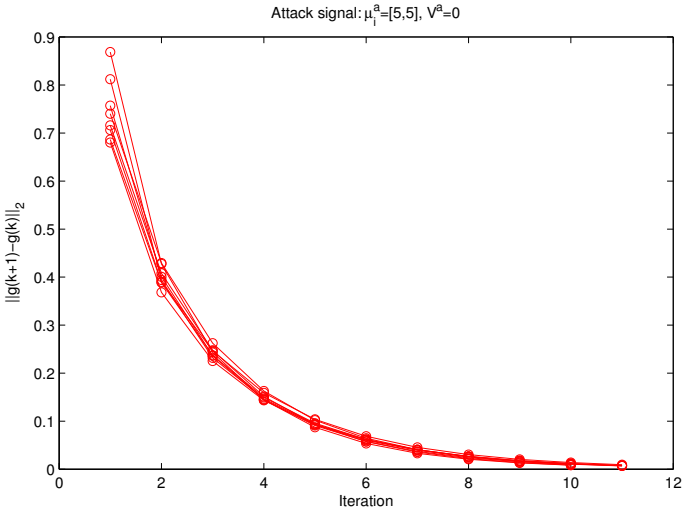


Figure 4.1: Convergence speed of global trust aggregation

4.2.2 Random walk interpretation

Imagine a random walk on the graph. Assuming that each node is in one of two states, GOOD or BAD. Starting from a regular node i in state $S(i)$, we move to a node j in state $S(j)$ with probability $P_{i,j}(S(i), S(j))$. The walk stops when we hit a trusted core. Then $g_i(1)$ is the probability that the random walk, starting from node i, hits a trusted core in GOOD state, while $g_i(2)$ is the probability that the random walk hits a trusted core in BAD state. Intuitively, a node in GOOD state will hit a trusted core in GOOD state with a higher probability, which explains why our trust model works.

4.2.3 Secure trust aggregation

In the previously presented algorithm, each node computes and reports its own trust value. Malicious nodes could easily report false trust values to subvert the system. To combat this, as discussed in [40], first, the trust value of a node should not be computed by and reside at the node itself. We should have a different node in the network computing the trust value of a node. Second, it will be in the interest of malicious peers to return wrong results when they are supposed to compute any node's trust values. To overcome this situation, the trust value of one node should be computed by more than one other nodes. Such schemes are usually called secure score management schemes.

In peer to peer network, secure score management schemes depend on Distributed Hash Tables (DHT), such as EigenTrust [40], PowerTrust [83], etc. However, these DHT protocols typically interconnect nodes independently of their proximity in the physical network topology, which is not suitable for energy constrained sensor networks, as the DHT logical identifier space may actually be far apart and each logical hop within a DHT may cost many packet transmissions. To satisfy the constrained resources in sensor networks, we developed a secure score management scheme based on the Virtual Ring Routing (VRR) [12] protocol.

VRR uses random unsigned integers to identify nodes and organizes the nodes into a virtual ring in order of increasing identifier, with wrapping around zero. To maintain the integrity of the virtual ring with node and link failures, each node maintains a virtual neighbor set *vset* of cardinality r containing the node identifiers

of the $r/2$ closest neighbors clockwise in the virtual ring and the $r/2$ closest neighbors counter clockwise. Each node also maintains a physical neighbor set *pset* with the identifiers of nodes that it can communicate with at the link layer. Fig. 4.2 from [12] illustrates an example of the virtual ring with a 12-bit identifier space with identifiers in base 16, and also the vset of the node with identifier $8F6$ with $r = 4$.

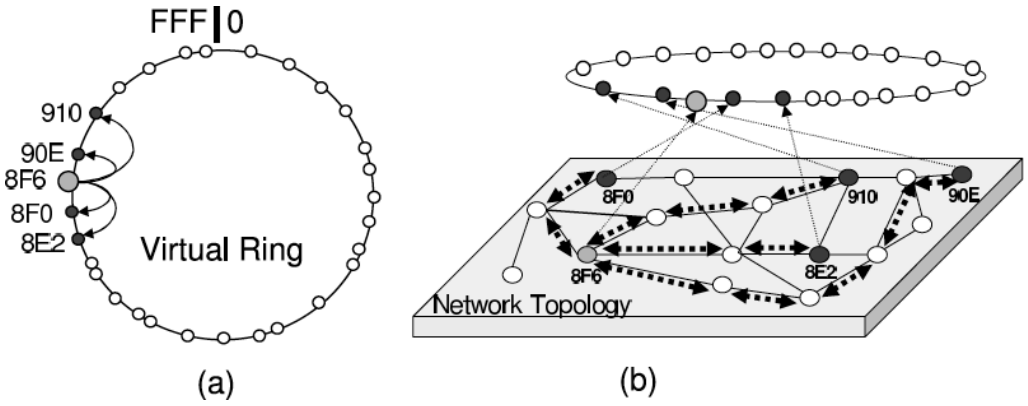


Figure 4.2: Relationship between the virtual ring and the physical network [12]

VRR sets up and maintains routing paths between a node and each of its virtual neighbors, which are called *vset-paths*. It can be shown that if the average vset-path length is p, then a packet is expected to reach a node that has a vset-path to the destination after visiting $O(n/(rp))$ nodes, which will add only a constant stretch if p grows with $\sqrt{n}$, as in wireless ad hoc networks. VRR provides both traditional point-to-point network routing and DHT routing to the node responsible for a hash table key. More details can be found in [12].

We designed our secure score management scheme using the ability of VRR to route messages sent to numerical keys to the node whose identifier is numerically closest to the key. For each regular node i in the network, we assign node j as the score manager of node i if it is the closest successor node of k_i in the virtual ring

space, where k_i is the hash value of the identifier of node i by a predefined hash function h_1. To have multiple score managers for node i, we can use multiple hash functions, namely, $h_1, \ldots, h_M$, and we denote the score managers by $s_1(i), \ldots, s_M(i)$. For each trusted core node k, we let it to be its own score manager.

During the trust aggregation, each trusted core node k submits its global trust value g_k to the score managers of its neighboring nodes. Each regular node i submits the local trust values $P_{i,j},\ j \in \mathcal{N}_i$ to its score managers $s_m(i),\ m = 1, \ldots, M$. Since node i also acts as other nodes' score manager, if we denote the set of these other nodes by $\mathcal{C}_i$, node i also collects local trust values $P_{c,j},\ j \in \mathcal{N}_c$ from all nodes $c \in \mathcal{C}_i$. In each round of trust update, node i will compute the value $g_c^{(k+1)}$ based on $P_{c,j}$ and $g_j^{(k)},\ j \in \mathcal{N}_c$ collected from the score managers of c's neighboring nodes, then send $g_c^{(k+1)}$ back to these score managers. The update will stop if $|g_i^{(k+1)} - g_i^{(k)}| < \epsilon$, where ϵ is a small pre-defined constant. A summary of the secure trust aggregation algorithm is described in Table 4.2.

By using such a secure score management scheme, a node in the network will not be able to find out the node for whom it computes the trust values, so malicious nodes cannot increase the reputation of other malicious nodes. Also by using multiple trust score managers, possible conflicts arising from malicious nodes presenting faulty trust values can be settled using a majority vote.

Table 4.2: Secure trust aggregation algorithm

1. for each regular node i, do
2. submit local trust values $P_{i,j},\ j \in \mathcal{N}_i$ to score managers $s_m(i),\ m = 1, \ldots, M$;
3. collect local trust values $P_{c,j}\ j \in \mathcal{N}_c$ for all nodes $c \in \mathcal{C}_i$;
4. for each node $c \in \mathcal{C}_i$, do
5. repeat
6. compute $g_c^{(k+1)} = \frac{1}{|\mathcal{N}_c|}[\sum_{j \in \mathcal{U}, j \in \mathcal{N}_c} P_{c,j}\mathbf{g}_j^{(k)} + \sum_{l \in \mathcal{T}, l \in \mathcal{N}_i} P_{c,l}\mathbf{g}_l]$;
7. send $g_c^{(k+1)}$ to score mangers $s_m(j),\ j \in \mathcal{N}_c,\ m = 1, \ldots, M$;
8. wait all nodes to return $g_j^{(k+1)},\ j \in \mathcal{N}_c$;
9. until $|g_i^{(k+1)} - g_i^{(k)}| < \epsilon$
10. end
11. end
12. for each trusted core node k, do
13. submit g_k to score managers $s_m(j),\ j \in \mathcal{N}_k,\ m = 1, \ldots, M$;
14. end

4.3 Case study and performance evaluation

We now evaluate our trust aggregation algorithm in the setting of distributed Kalman filtering. To handle the convergence of the distributed Kalman filtering and the trust aggregation process together, we evaluate trust on a smaller time scale than the DKF. That is, we let the trust aggregation algorithm achieve convergence between each interval of state estimate updates in DKF. This is possible since equation (4.5) converges very fast. Moreover, since an abrupt change to a node's trust value does not occur frequently, we can expect that after the first round of trust aggregation, the following rounds will converge much faster.

Let $r_i = 1 \times \mathbf{g}_i(1) + 0 \times \mathbf{g}_i(2)$ be the trust weight of node i, which is also the probability that node i is in the normal state. Figure 4.3 illustrates the performance of the trust aggregation algorithm on distinguishing malicious nodes from normal nodes. We can see that the trust weights for the normal nodes are much larger than

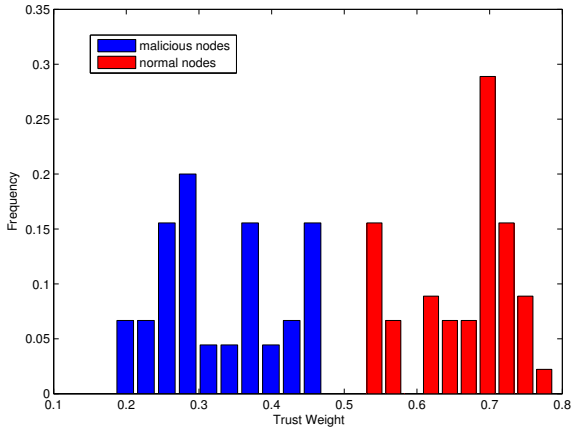


Figure 4.3: Distribution of trust weights for nodes in the network

that of the malicious nodes.

In the trust-aware DKF algorithm, we weight the data sent by a node by its trust weight r_i. Table 4.3 presents the trust-aware DKF algorithm, where $d(i)$ is the degree of node i.

Table 4.3: Algorithm IV – Trust-aware DKF Algorithm

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i = \boldsymbol{\mu}_0$
2. For each node in the network
3. While new data comes do
4. Locally aggregate data and covariance matrices:
$\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T V_j^{-1} \mathbf{y}_j^a,\ U_j = H_j^T V_j^{-1} H_j,$
$\mathbf{z}_i = \frac{(d(i)+1)\sum_{j\in\mathcal{J}_i} r_j \mathbf{u}_j}{\sum_{j\in\mathcal{J}} r_j},\ S_i = \frac{(d(i)+1)\sum_{j\in\mathcal{J}_i} r_j U_j}{\sum_{j\in\mathcal{J}_i} r_j}$
5. Compute the Kalman-consensus estimate:
$\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1},$
$\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i) + \epsilon\hat{M}_i \frac{d(i)\sum_{j\in\mathcal{N}_i} r_j(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)}{\sum_{j\in\mathcal{N}_i} r_j}$
6. Update the state of the Kalman-consensus filter:
$\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i = A\hat{\mathbf{x}}_i$
7. end while
8. end for

In Table 4.3, less trusted nodes have less impact on the algorithm. We can also isolate those less trusted nodes by ignoring data sent from them. In this way, the

algorithm can achieve better performance. We call this algorithm, of isolating less trusted nodes, as Algorithm V. Table 4.4 shows the performance comparison of Algorithm III from the previous chapter (DKF with Bayesian learning), Algorithms IV and V (Trust-aware DKFs). In the experiments, we generate hundreds of networks that consist of 100 uniformly distributed sensors, where 60 of them are randomly selected to be compromised. We measure the algorithm performance by the mean square error till time k, i.e., $e_i^{MSE} = \sqrt{(\sum_{l=1}^{k}(\hat{\mathbf{x}}_i(l) - \mathbf{x}(l))^2)/k}$, where $\hat{\mathbf{x}}_i(l)$ is the state estimation by sensor i at time l. In the experiment, we set $k = 400$. Since the sensors can approximately achieve consensus in DKF [59], the estimated state $\hat{x}_i(l)$ is approximately the same for each sensor i, and so is the value of e_i^{MSE}. Table 4.4 shows the average value of e_i^{MSE}.

Table 4.4: Performance Comparison of Algorithm III, IV, V

μ_i^a, V_i^a	$[5,5], 0$	$[5,5], 2$	$[10,10], 0$	[10,10], 5
Algorithm III	4.7345	3.1572	9.8317	6.8935
Algorithm IV	1.2816	1.6131	3.3624	2.1507
Algorithm V	0.6724	0.5298	1.1997	0.8966

We can see that the trust-aware schemes (Algorithm IV and V) perform better than the DKF algorithm with Bayesian learning (Algorithm III), and Algorithm V performs better than Algorithm IV. The only drawback of Algorithm V is that the isolation of the less-trusted nodes may lead to a disconnected network. Also we observe that when $V_i^a \neq 0$, the algorithms may perform better than the case when $V_i^a = 0$, which means that V_i^a being non-zero in some sense reduces the impact of $\boldsymbol{\mu}_i^a$ on the estimated states.

4.4 Discussion

In this chapter we proposed a secure distributed trust aggregation algorithm that can utilize the existence of trusted cores to enhance data integrity and improve network robustness. The trusted cores are a special class of nodes that have much higher levels of security than other regular nodes. We show that as long as there exist some paths that can connect each regular node to one of the trusted cores, the network is robust even if more than half of the sensors are compromised. Future research can be conducted to study the impact on algorithm performance due to the changes on the number of trusted cores and their positions in the network. Also, different models for the local trust opinion can be developed according to application requirements.

Part II

Sensor Network Monitoring and Anomaly Detection

Chapter 5

Trust-assisted Network Probing and Anomaly Localization

5.1 Introduction

Due to the possibly unattended and hostile operating environment, a sensor network may suffer from system failures due to loss of links and nodes, or malicious intrusions. Therefore, it is critical to continuously monitor the overall state of the sensor network to ensure its correct and efficient functioning. Compared to the diagnosis protocols for the Internet, monitoring and diagnosing wireless sensor networks pose unique challenges due to the low communication bandwidth and the limited resources such as energy, memory and computational power in sensors.

Existing works on anomaly detection and localization in wireless sensor networks can be roughly divided into two categories: centralized and distributed. In centralized approaches, a central controller takes responsibility of monitoring and tracing anomalous behaviors in the network [64]. However, resource constrained sensor networks can not always afford to periodically collect all the measurements in a centralized manner. Distributed approaches address this issue by encouraging local decision making: for example, nodes can rely on neighbor coordination such as Watchdog [52] to detect misbehaving nodes. However, in a hostile environment, a node may not report its status or its neighbors' status honestly, and an intermediate node can intentionally alter its forwarded messages. One possible solution for this

problem would be to use only end-to-end measurements if the end nodes can be trusted. Furthermore, it may be difficult to access individual nodes in large scale sensor networks, and consequently end-to-end data provide valuable information for inferring the internal status of the network. Making inferences using end-to-end measurements has been extensively studied for wired networks such as the Internet [23, 56]. However, these techniques cannot be directly applied to sensor networks due to the constrained resources of sensor networks.

In this chapter, we present a trust-assisted framework for anomaly localization in a heterogenous sensor network with a small set of nodes that have stronger computation and communication capabilities. The set of stronger nodes is responsible for collecting end-to-end measurements through a two-phase probing. In both phases, the historic information on link trustworthiness is explored to achieve a good tradeoff between the probing overhead and the inference accuracy.

5.2 Related Work

Network tomography. Anomaly inference from end-to-end measurements can be generally categorized as a problem of *network tomography*, i.e., inferring a network's internal properties using information derived from end point data. It requires solving a system of equations that relate the end-to-end measurements to link properties such as packet loss rate, link delay, etc.. Most network tomography techniques are developed for wired networks such as the Internet [23, 56, 28]. Duffield et al. [23] formulated the network tomography problem as a set-cover problem and solved it on

a tree topology to identify the most unreliable links of the network. In [56], Nguyen et al. proposed a Boolean algebra based approach to improve inference accuracy by an order of magnitude over previous algorithms. Gu et al. [28] presented an optimal probing scheme for unicast network delay tomography that can provide the most accurate estimation. However, these techniques usually incur high computational complexity and probing overhead, and therefore are not suitable for sensor networks due to the severe resource constraints.

Monitoring sensor networks. Monitoring wireless sensor networks has recently generated a surge of interest from the research community [64, 57, 73]. Ramanathan et al. [64] proposed *Sympathy*, which carefully selects a minimal set of metrics at a centralized sink and uses an empirically developed decision tree to determine the most likely causes of detected failures. Nguyen et al. [57] proposed inference schemes based on maximum likelihood and Bayesian principles, which can isolate links with high loss rates even with routing changes and noisy measurements. Wang et al. [73] formulated the anomaly detection and localization problem as an optimal sequential testing guided by end-to-end data. They proposed a greedy algorithm that can determine an optimal selection sequence of network measurements to minimize testing cost, while the measurements are not limited to end-to-end behaviors. All of the above mentioned work assume the existence of a centralized sink node which is responsible for collecting required measurements, and rarely consider the strict constraint on communication overhead. In our work, we do not use any centralized sink node but exploit a hierarchical network structure to improve bandwidth

and energy efficiency. Furthermore, our work focuses on achieving a good tradeoff between communication overhead and inference accuracy, which provides a more practical and flexible solution for real-world applications.

Trust in sensor networks. We have discussed briefly on the concept of trust in wireless sensor networks in the previous chapter. In this chapter, trust is associated with a communication link, which is computed according to the historic observations of anomaly occurrences on the link. The information of link trustworthiness enables us to achieve an effective tradeoff between algorithm performance and resource consumption in sensor networks.

5.3 A trust-assisted two-phase probing and anomaly localization

5.3.1 Problem formulation

In this section, we describe the network model and define the problem of anomaly localization. We use a two-layer heterogeneous network as illustrated in Fig. 5.1. The lower-layer network is the main network responsible for environment sensing and task execution. It consists of regular Mote-type sensor nodes with severe energy and communication constraints. The upper-layer network is responsible for monitoring the status of the lower-layer network by taking end-to-end measurements. It consists of a set of nodes with stronger computation and communication capabilities. These nodes are also highly trusted, e.g., they may be associated with tamper resistant hardware, so they would not give false information. We found by

experiments that a small number of the stronger nodes would be enough to monitor a large network. The main concern of using such hierarchical structure is to concentrate resource intensive computation and communication tasks only in the upper-layer network, thus to prolong the lifetime of the lower-level network.

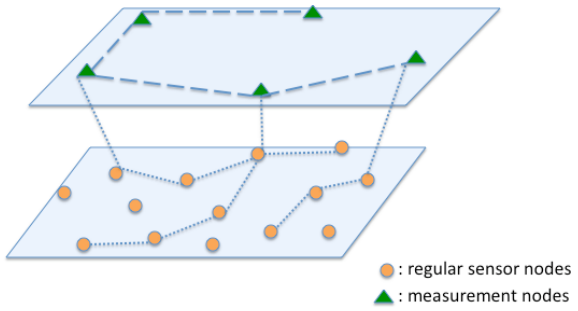


Figure 5.1: A hierarchical network structure

We assume that the anomalies may occur on any links in the monitored network. These network anomalies typically lead to deviations of the end-to-end measurements from the normal case, which can be explored for anomaly detection. We will defer the discussion of the detection techniques in a later chapter. In this chapter, we focus on how to probe the network to collect end-to-end measurements, and given the detection results on individual paths, how to localize anomalous links that are responsible for path anomalies.

Let $\mathcal{P}$ be the set of all end-to-end paths from the stronger nodes to the lower-layer network nodes. Denoting the set of links that appear in $\mathcal{P}$ by $\mathcal{E}$, we use a matrix A of dimension $|\mathcal{P}| \times |\mathcal{E}|$, namely the routing matrix, to represent the information relating paths to links. Each row of A represents a path in $\mathcal{P}$, and each column represents a link in $\mathcal{E}$. The entry a_{ij} equals to 1 if path P_i contains link e_j. Let x_i be the indicator for the anomaly in path P_i, and y_j be the indicator

for the anomaly in link e_j, i.e., $x_i = 1$ indicates that P_i is anomalous, and $y_j = 1$ indicates that link e_j is anomalous. Assuming that $|\mathcal{P}| = n_p$ and $|\mathcal{E}| = n_e$, if the network has no noise, the path P_i is anomalous if any of its links is anomalous, i.e., $x_i = 1 - \Pi_{j=1}^{n_e}(1 - a_{ij}y_j)$, so we have at most n_p constraints on n_e variables. For a sensor network with severe communication constraints, the number of observations on path behaviors may not be sufficient to achieve a unique solution for the y_j's. More generally, a practical network usually has some noise and follows the Noisy-OR model [80], i.e., $P(x_i = 0|\mathbf{y}) = (1 - \rho_j)\prod_j (a_{ij}\rho_{ij})^{y_j}$ where ρ_j is a leak probability representing the probability that path P_j performs as anomalous even if all its links are normal, and ρ_{ij} is an inhibition probability representing the probability that link e_j in path P_i is anomalous but performs as normal. Therefore, exact inference for the locations of the anomalous links may not be possible. We focus on the *maximum a posteriori* estimation of the anomalous links.

A summary of the anomaly localization problem is as follows. We are given the following information: (1) The set of wireless links $\mathcal{E} = \{e_1, \ldots, e_{n_e}\}$; (2) The set of all paths $\mathcal{P} = \{P_1, \ldots, P_{n_p}\}$; (3) The routing matrix $A = [a_{ij}]_{n_p \times n_e}$; (4) Constraints on probing overheads. The objective is to select a subset of paths in $\mathcal{P}$ to collect end-to-end measurements under probing overhead constraints, and find the most probable candidates of anomalous links.

Ideally, to monitor network status, probes should cover all links in the network in order to detect all possible anomalies. However, probing all links at a rate that is sufficiently fast for the detection may cost high communication overhead and cause serious network congestion. In the following section, we propose a trust-

assisted two-phase probing strategy for anomaly localization under communication constraints. In the first phase, a small set of probes is sent to localize suspicious areas and cover as many anomalous links as possible. In the second phase, probes are sequentially selected based on previous probing results and sent only to the suspicious areas to locate individual links responsible for the observed end-to-end anomalous behaviors. In both phases, link trustworthiness information is utilized to achieve the best possible performance under the given communication constraints.

5.3.2 Phase I: narrowing suspicious areas

Our first-phase probing scheme is motivated by the observation that different links might be exposed to different levels of risks, e.g., the attacker may be more likely to target some "important" links in the network. Therefore, the links should be probed with *different frequencies* and *priorities*. The priority of a link can be computed based on its trustworthiness and the obsoleteness of previously collected information on the link. The probes are then selected to cover links of highest priorities and satisfy a given communication constraint. The goal is to narrow the suspicious areas to be examined in the second phase.

Problem Formulation For each link $e_i \in \mathcal{E}$ at time k, we assign a trust value $t_i(k) \in [0, 1)$ to represent the expected probability of e_i being normal. To maintain this trust information, we adapt to our probe selection scenario the Beta reputation system [38], which is well known for its simplicity, efficiency for capturing user trustworthiness and its firm foundation on the theory of statistics.

The Beta reputation system is based on the beta probability density function that can be used to represent probability distributions of binary events. It provides a sound mathematical basis for combining feedback and expressing reputation ratings [38]. In the Beta reputation system, the posterior probability of binary events, i.e., in our case, the probability $p_i(k)$ of whether the link is normal, is represented by a beta distribution with parameter $\alpha_i(k)$ and $\beta_i(k)$, i.e.,

$$f(p_i(k)|\alpha_i(k),\beta_i(k)) = \frac{\Gamma(\alpha_i(k),\beta_i(k))p_i(k)^{\alpha_i(k)-1}(1-p_i(k))^{\beta_i(k)-1}}{\Gamma(\alpha_i(k))\Gamma(\beta_i(k))},$$

where $\Gamma(\cdot)$ is the gamma function.

Given previously observed states of link e_i till time k, we denote the number of events that e_i is normal by $r_i(k)$ and the number of events that e_i is anomalous by $s_i(k)$, then $\alpha_i(k)$ and $\beta_i(k)$ can be represented by $\alpha_i(k) = r_i(k) + 1$ and $\beta_i(k) = s_i(k) + 1$. Note that the variable $p_i(k)$ is a probability variable, and for a given $p_i(k)$ the probability density $f(p_i(k)|\alpha_i(k),\beta_i(k))$ is a second order probability that represents that the first order variable $p_i(k)$ has a specific value. Since $p_i(k)$ is continuous, the second order probability $f(p_i(k)|\alpha_i(k),\beta_i(k))$ for any given value of $p_i(k)$ will be vanishingly small and therefore meaningless [38]. Therefore, we define the link trust value $t_i(k)$ to be the expected value of $p_i(k)$,

$$t_i(k) = E[p_i(k)] = \frac{\alpha(k)}{\alpha(k)+\beta(k)} = \frac{r_i(k)+1}{r_i(k)+s_i(k)+2}.$$

Fig. 5.2 shows the trust values for a link that has 20 normal behaviors first, then

followed by 20 malicious ones. The initial trust value for the link is set to be 0.5.

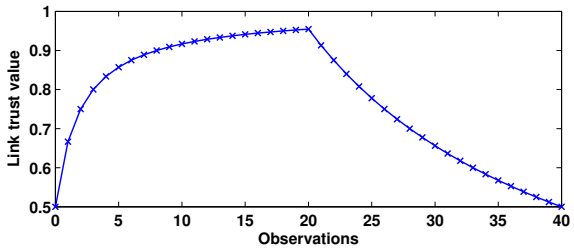


Figure 5.2: Link trust values

Since links may change behaviors over time, old observations are less relevant for the current trust value, thus should be given less weight. Time discounting factors can be introduced to gradually forget the old observations. In our case, we let $\alpha_i(k+1) = \kappa_1\alpha_i(k) + I_i(k+1)$ and $\beta_i(k+1) = \kappa_2\beta_i(k) + 1 - I_i(k+1)$, where $I_i(k+1)$ is the indicator function for whether the $(k+1)^{th}$ observation of link e_i is normal, and $0 < \kappa_1, \kappa_2 \leq 1$ are the forgetting factors. The forgetting factors κ_1 and κ_2 can be set differently, e.g., if we want to punish more on the occurrence of an anomalous event, i.e., decrease trust value, we can set $\kappa_2 > \kappa_1$. For example, Fig. 5.3 shows the trust values for a link that has the same behaviors as in Fig. 5.2, but with $\kappa_1 = 0.5$ and $\kappa_2 = 0.8$. We can see that the link's trust value decreases very fast after only a few malicious behaviors.

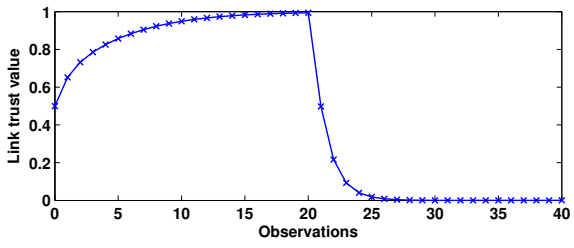


Figure 5.3: Link trust values with forgetting

In our reputation system the trust value of a link only changes after we probe

it, therefore, another important factor that must be taken into account is the obsoleteness of this trust information. For example, if a link has a high trust value, but it is obtained long time ago, this link should still have higher priority to be probed. In this way, the trust information about all the links can be maintained at relatively recent values, i.e., not stale values. We let the obsoleteness of link trust information decrease exponentially with the number of rounds that the link is not probed. Let $A_i(k)$ denote the event that e_i is not probed in the k^{th} interval and $A_i^C(k)$ denote the opposite event. Then we define the link *obsoleteness value* to be

$$o_i(k+1) = \begin{cases} 1 - (1 - o_i(k))e^{-\tau}, & \text{if } A_i(k), \\ 0, & \text{if } A_i^C(k). \end{cases}$$

where $\tau > 0$ is a parameter controlling the fading speed of the information. The more recent a link is probed, the smaller its obsoleteness value is. Let

$$w_i(k) = \rho \cdot (1 - t_i(k)) + (1 - \rho) \cdot o_i(k),$$

with ρ being a parameter that adjusts the relative importance of the trust value vs. the obsoleteness value. Then $w_i(k)$ can be used as a weight that indicates the urgency of probing link e_i. Note that our definition of link weights is only one possible way, the link weights can also be defined in other ways if some other information is known. As long as the link weight information is available, the first-phase probing selection can be formulated as an optimization problem.

Assume that h_i is the length of path P_i, and h_0 is the probing overhead con-

straint such that the number of links traversed by the probes can not be larger than h_0. Let u_i be the indicator function for selecting path P_i, i.e., $u_i = 1$ means that path P_i is selected, and v_j be the indicator function for selecting link e_j, the probe selection problem can be defined as

$$\max \quad z = \sum_{j \in \mathcal{E}} w_j \cdot v_j, \tag{5.1}$$

$$s.t. \quad \forall\ i \in \mathcal{P},\ u_i \in \{0,1\},\ \sum_{i \in \mathcal{P}} h_i u_i \leq h_0, \tag{5.2}$$

$$\forall\ j \in \mathcal{E},\ v_j \in \{0,1\}\ \sum_{i \in \mathcal{P}} a_{ij} u_i - v_j \geq 0. \tag{5.3}$$

Constraint (5.2) represents the probing overhead constraint and constraint (5.3) represents the fact that one link may belong to multiple paths.

The above optimization problem belongs to the class of the budgeted maximum coverage problem [44], which is NP-hard. Khuller et al. [44] proposed a $(1 - 1/e)$ approximation algorithm that achieves a best possible approximation ratio, i.e., the ratio of the objective value obtained by the approximation algorithm to the optimal objective value is lower bounded by $(1 - 1/e)$. However, their method involves an enumeration of all subsets of $\mathcal{P}$ that have cardinality k, where $k \geq 3$, which is prohibitively computationally expensive for an implementation in sensor networks.

To solve the optimization problem efficiently, we propose an approximation algorithm that has very low computational complexity. We also provide a numerical performance bound for our algorithm. In addition, experimental results demonstrate that our algorithm achieves not only low computation overhead but also high approximation factor.

The approximation algorithm. The approximation algorithm and its analysis are based on linear programming duality. To solve the optimization problem, we first relax the integer constraints in (5.2) and (5.3) to pairs of linear constraints $0 \leq u_i \leq 1$ and $0 \leq v_j \leq 1$. It can be further shown that $0 \leq u_i \leq 1$ and $0 \leq v_j \leq 1$ can be equivalently changed to $u_i \geq 0$ and $v_j \leq 1$. Then the dual problem of the original optimization problem can be written as

$$\min_{\lambda,\boldsymbol{\gamma}} \quad \lambda h_0 + \sum_{j \in \mathcal{E}} \gamma_j \tag{5.4}$$

$$\text{s.t. } \forall\, i \in \mathcal{P},\ \lambda h_i + \sum_{j \in \mathcal{E}} a_{ij}\gamma_j \geq \sum_{j \in \mathcal{E}} a_{ij} w_j = \tilde{w}_i, \tag{5.5}$$

where λ, γ_j for $j = 1, \ldots, n_e$, are Lagrange multipliers, and $\tilde{w}_i$ is the sum of the link weights for path P_i.

We denote the set of selected paths as $\mathcal{X}$, the set of links covered by $\mathcal{X}$ as $\mathcal{E}(\mathcal{X})$, and the total hop counts of the paths in $\mathcal{X}$ as $hops(\mathcal{X})$. Note that $hops(\mathcal{X})$ is usually larger than $|\mathcal{E}(\mathcal{X})|$ as paths can have overlapping links. Let $q_0 = \arg\max_{i \in \mathcal{P}} \tilde{w}_i/h_i$, the algorithm starts with the feasible dual solution $\mathcal{X} = \{P_{q_0}\}$, $hops(\mathcal{X}) = h_{q_0}$, $\lambda = \tilde{w}_{q_0}/h_{q_0}$ and $\gamma_j = 0$ for $j \in \mathcal{E}(\mathcal{X})$. Then in each iteration, the basic idea is to reduce the dual objective value while improving the primal objective value. Initially, the objective value of the dual problem is λh_0 and only the q_0^{th} constraint in (5.5) is active. To reduce the dual objective value in each iteration, we choose one inactive constraint in (5.5), say, the i^{th} constraint and make it active. We reduce the value of λ by a constant β and raise the value of γ_j by the same amount β. Let $overlap = |\mathcal{E}(\{P_i\}) \cap \mathcal{E}(\mathcal{X})|$ be the number of overlapping links between P_i and

$\mathcal{X}$, the change to the left side of the i^{th} constraint will be $-h_i \cdot \beta + overlap \cdot \beta < 0$ as $h_i = |\mathcal{E}(\{P_i\})| \geq overlap$. Therefore, the i^{th} constraint can be made active with a properly selected positive value of β. In order to keep the dual solution feasible, all other constraints should not be violated, so the chosen constraint must be associated with the smallest β among all the inactive constraints. For the already active constraints, the change of their left side equations will be $-h_i \cdot \beta + \sum_j a_{ij} \cdot \beta = 0$, so they are still active and not violated. After each iteration, the change of the dual objective value is $\beta \cdot (|\mathcal{E}(\mathcal{X})| - h_0)$. Since $|\mathcal{E}(\mathcal{X})| < hops(\mathcal{X})$, the dual objective value will be reduced as long as the primal solution is feasible, i.e., $hops(\mathcal{X}) \leq h_0$. The algorithm will terminate as soon as the communication constraint is violated, i.e., $hops(\mathcal{X}) > h_0$. Table 5.1 provides a summary of the algorithm.

The probe selection algorithm is performed among the stronger nodes in the higher-layer network. Each of these nodes maintains a partial view of the network, i.e., the information of the links that constitute its monitored paths, which can be extracted from the routing protocol running in the lower-layer network, and the corresponding link weights, which are updated locally by the Beta reputation system. In the algorithm described in Table 5.1, except for the computation of $(\max \tilde{w}_i/h_i)$ in the 1^{st} line and the computation of $(\min \beta_i)$ through lines 4-10 , all other computations are local; i.e., they can be executed locally by individual nodes without exchange of information. The distributed computation of minimum can be achieved easily by maintaining a spanning tree among the stronger nodes [7]. Furthermore, the iterations in the 'While-end' statement through lines 2-19 can be finished in less than h_0/h_{min} rounds, with h_{min} being the length of the shortest path

Table 5.1: A sequential algorithm for first-phase probe selection

Initialization: $q_0 = \arg\max_{i \in \mathcal{P}} \tilde{w}_i/h_i$,
$\mathcal{X} = \{P_{q_0}\}$, $hops(\mathcal{X}) = h_{q_0}$, $\lambda = \tilde{w}_{q_0}/h_{q_0}$, $\gamma_j = 0$.
While $hops(\mathcal{X}) < h_0$,
  $\beta = inf, idx = 0$
  for each $i \notin \mathcal{X}$
    $overlap = |\mathcal{E}(\{P_i\}) \cap \mathcal{E}(\mathcal{X})|$
    $\beta_i = \frac{\lambda \cdot h_i + \sum_{j \in \mathcal{E}} a_{ij}\gamma_j - \tilde{w}_i}{h_i - overlap}$
    if $\beta > \beta_i$
      $\beta = \beta_i$, $idx = i$;
    end
  end
  if $hops(\mathcal{X}) + h_{idx} <= h_0$
    $\lambda = \lambda - \beta$,
    $\forall\ i \in \mathcal{E}(\mathcal{X}), \gamma_i = \gamma_i + \beta$,
    $\forall\ i \notin \mathcal{E}(\mathcal{X}), \gamma_i = \gamma_i$,
    $\mathcal{X} = \mathcal{X} \cup P_{idx}$,
  else
    terminate
  end
end

in $\mathcal{P}$. Since the number of the stronger nodes is small, the communication cost for this algorithm can be kept low.

We also derive the performance bound for the proposed approximation algorithm, as given in Theorem 3. The proof is given in Appendix A.1.

Theorem 5.3.1. *Assuming that the algorithm in Table 5.1 terminates after l_1 iterations and the primal solution is $\mathcal{X} = \{P_{s_1}, \ldots, P_{s_{l_1}}\}$ with objective value z^p, and let us denote the optimal objective value for the primal problem by z^*. Then z^p can be lower bounded by $z^p > \frac{z^*}{\delta(1+r/l_1)}$, where δ is the maximum number of paths in $\mathcal{X}$ that intersect at the same link, and $r = h_{max}/h_{min}$, with h_{max} and h_{min} being the maximum and minimum path lengths.*

Experimental results show that our method can achieve good approximation ratios. Fig. 5.4 shows the approximation ratios for our method and the literature method, where we generate networks with different sizes and the sensor nodes are uniformly distributed. The results are averaged over 100 simulation traces. More detailed descriptions of experimental settings can be found in a later section.

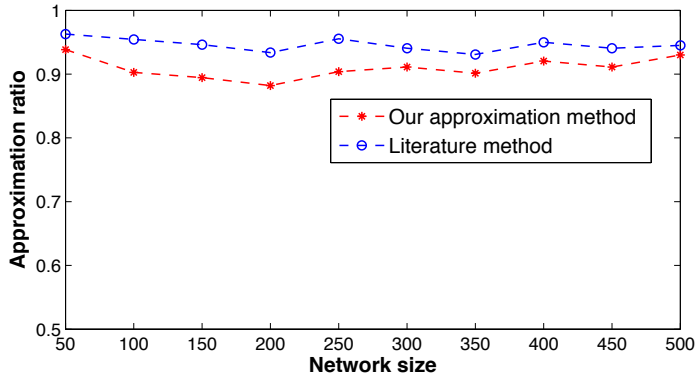


Figure 5.4: Comparison of our method and the literature method

After the probes are selected and sent, the higher layer network nodes will do hypothesis testings to detect anomalous paths based on the collected end-to-end measurements. The detection is based on identifying significant measurement deviations from the normal state. More details can be found in Chapter 6.

5.3.3 Phase II: locating malicious links

The goal of the second-phase probing is to find the individual links responsible for the observed path anomalies. In this phase, additional probes are sequentially selected according to previous observations and the predicted diagnosis quality. This online selection is typically more efficient than its offline counterpart [65], where the offline method attempts to select the set of probes before any observations are made.

The proposed algorithm Denote the second-phase probe selection strategy by π, the states of links in $\mathcal{E}$ by $\mathcal{Y}$, and assuming that the observed path states from the first phase is represented by $\mathbf{x}_1$, then the diagnosis quality of the probe selection strategy π can be represented by $f(\pi) = H(\mathcal{Y}|\mathbf{x}_1) - H(\mathcal{Y}|\mathbf{x}_1, \phi(\pi))$, where $H(\cdot)$ represents the entropy of a random variable, and $\phi(\pi)$ represents the observed path state based on the collected end-to-end measurements by implementing strategy π.

Let $h(\pi)$ be the communication overhead for implementing π, measured by the number of links traversed by the selected probes, and $\tilde{h}_0$ be the communication constraint for the second phase. Then the probe selection problem in this phase can be formulated as to find a policy π^* such that

$$\pi^* = \arg\max_{\pi} E_{\pi}[H(\mathcal{Y}|\mathbf{x}_1) - H(\mathcal{Y}|\mathbf{x}_1, \phi(\pi))], \text{ s.t. } h(\pi) \leq \tilde{h}_0.$$

Solving this problem is equal to solving a finite-horizon Markov Decision Problem (MDP) that has exponential state space [26], which is NP-hard.

A widely used method for solving this type of problem is to use a heuristic greedy approach that iteratively selects the probe that provides the largest reduction in uncertainty at each step [80, 18]. More specifically, let $\mathcal{X}_C$ represent the previously sent probes, including the probes sent in the first phase. Assuming that the observations for $\mathcal{X}_C$ are denoted by $\mathbf{x}_C$, and the probing overhead is $h(\mathcal{X}_C)$, then the next probe is selected to be

$$i = \arg \max_{j: h_j \leq \tilde{h}_0 - h(\mathcal{X}_C)} H(\mathcal{Y}|\mathbf{x}_C) - H(\mathcal{Y}|\mathbf{x}_C, X_j), \tag{5.6}$$

where h_j is the hop count for path P_j and X_j is a random variable representing the unknown state of path P_j. We will first a provide performance bound for this greedy algorithm and then discuss its implementation issues.

Theorem 5.3.2. *Assuming that the obtained diagnosis quality, i.e., the reduced uncertainty on link states, by the greedy algorithm is $\tilde{\Delta}$, and the optimal diagnosis quality is Δ^*, then $\tilde{\Delta}$ can be lower bounded by $\tilde{\Delta} \geq (1 - e^{-l_2 \cdot h_{min}/\tilde{h}_0})\Delta^*$, where l_2 is the number of probes selected by the greedy algorithm, h_{min} is the minimum path length, and $\tilde{h}_0$ is the probing overhead constraint in the second probing phase.*

We now discuss how to implement the greedy algorithm according to equation (5.6). Fig. 5.5 illustrates the graphical model, where $\mathcal{X} = \{X_1, \ldots, X_{n_p}\}$ is the set of the evidence nodes that represent path states which can be observed by probing, and $\mathcal{Y} = \{Y_1, \ldots, Y_{n_e}\}$ is the set of the latent nodes that represent link states which can only be inferred from path observations.

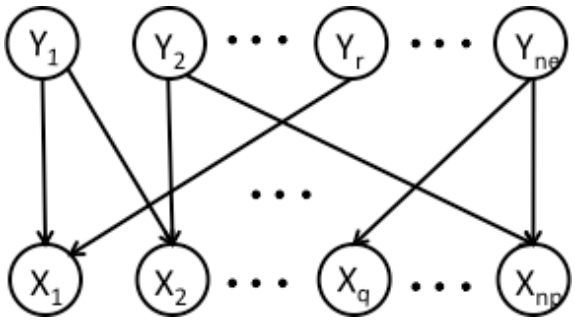


Figure 5.5: Graphical Model: Bayesian Network

By simple algebraic manipulation, it is found that

$$H(\mathcal{Y}|\mathbf{x}_C, X_j) = H(\mathcal{Y}|\mathbf{x}_C) - H(X_j|\mathbf{x}_C) + H(X_j|\mathcal{Y}_{pa_j}, \mathbf{x}_C),$$

where $\mathcal{Y}_{pa_j}$ is the set of the parent nodes of X_j in the graphical model. Then, equation (5.6) becomes

$$i = \arg \max_{j:h_j \leq \tilde{h}_0 - h(\mathcal{X}_C)} H(X_j|\mathbf{x}_C) - H(X_j|\mathcal{Y}_{pa_j}, \mathbf{x}_C). \tag{5.7}$$

Therefore, only $H(X_j|\mathbf{x}_C)$ and $H(X_j|\mathcal{Y}_{pa_j}, \mathbf{x}_C)$ need to be computed for the sequential probe selection. The computations can be carried out using a variant of the Loopy Belief Propagation algorithm (LBP) [9].

LBP [9] is a message passing algorithm for performing inference on graphical models, where it calculates the marginal distribution for each unobserved node, conditioned on any observed nodes. A typical graphical model for performing LBP is a Markov network with pairwise potentials, so that the joint distribution factors according to

$$\prod_{s,t} \Psi(s,t) \prod_{s} \Psi(s),$$

where s, t represent the nodes in the Markov network and $\Psi(\cdot)$ represents the potential function. A graphical model with higher-order potential functions can always be converted to a graphical model with only pairwise potential functions [74].

In a Markov network, the LBP can be expressed as an iterated message-passing process among nodes. Let $\mu^i_{t\to s}(s)$ denote the outgoing message in the i^{th} iteration

from node t to a neighbor node s. Then $\mu^i_{t\to s}(s)$ can be computed as

$$\mu^i_{t\to s}(s) \propto \sum \Psi(t,s) \prod_{u\in nb(t)\setminus s} \mu^{i-1}_{u\to t}(t),$$

where $nb(t)$ represents the set of neighboring nodes of t. The belief, or, marginal probability of node t can be calculated by

$$p^i(t) \propto \Psi(t) \prod_{u\in nb(t)} \mu^i_{u\to t}(t).$$

In practice, the iterated LBP process can often arrive at a reasonable approximation to the correct marginal distributions. The fact that no global coordination is required for LBP makes it suitable for sensor network applications.

We first convert our Bayesian network model in Fig. 5.5 to a Markov network with pairwise potentials, as shown in Fig. 5.6. The conversion process is simple

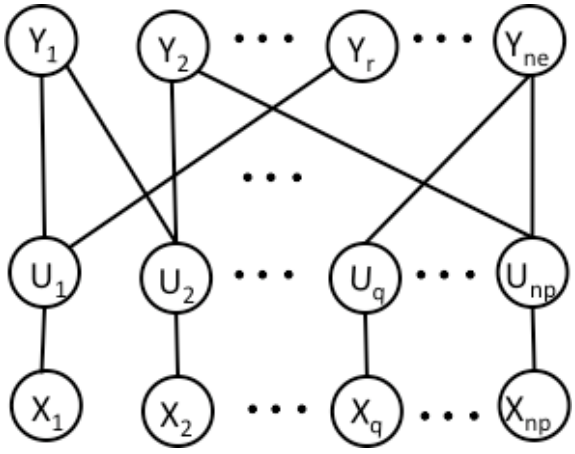


Figure 5.6: Markov Network with pairwise potentials

and typical. For any evidence node X_i in Fig. 5.5, we create a compound node U_i into which the parent nodes of X_i are clustered. This compound node is then con-

nected to X_i and the individual parent nodes. To make the probability distribution identical, we set the potential function $\Psi(U_i, X_i)$ to be the conditional probability $P(X_i|\mathcal{Y}_{pa_i})$ and the potential function between U_i and an individual parent node $Y_j \in \mathcal{Y}_{pa_i}$ to be $P(Y_j)$. For example, in Fig. 5.5 if node Y_1 and Y_r are the only common parent nodes for node X_1, then we create a compound node U_1 and connect it to X_1, Y_1 and Y_r. The corresponding potential functions are $\Psi(U_1, X_1) = P(X_1|Y_1, Y_r)$, $\Psi(U_1, Y_1) = P(Y_1)$ and $\Psi(U_1, Y_r) = P(Y_r)$. In this way, the Bayesian network in Fig. 5.5 can be converted into the Markov network in Fig. 5.6. The joint probability for the Markov network is

$$P(\mathcal{X}, \mathcal{Y}) = \prod_{U_i, X_i} \Psi(U_i, X_i) \prod_{U_i, Y_j} \Psi(U_i, Y_j),$$

which contains only pairwise potentials and is identical to the joint probability corresponding to the Bayesian network shown in Fig. 5.5 .

We then perform sequential probe selection according to equation (5.7) based on a variant of LBP executed on the Markov network. First, the LBP algorithm already provides the marginal distribution $P(X_j|\mathbf{x}_C)$ at node X_j, so $H(X_j|\mathbf{x}_C)$ can be easily computed as $\sum_{x_j} P(x_j|\mathbf{x}_C) \log P(x_j|\mathbf{x}_C)$ at node X_j.

To compute $H(X_j|\mathcal{Y}_{pa_j}, \mathbf{x}_C)$ in equation (5.7), we can write it as

$$\begin{aligned} H(X_j|\mathcal{Y}_{pa_j}, \mathbf{x}_C) &= -\sum_{\mathcal{Y}_{pa_j}} P(\mathbf{y}_{pa_j}, x_j|\mathbf{x}_C) \log P(x_j|\mathbf{y}_{pa_j}) \\ &= -\sum_{\mathcal{Y}_{pa_j}, x_j} P(\mathbf{y}_{pa_j}, x_j|\mathbf{x}_C) \log P(\mathbf{y}_{pa_j}, x_j|\mathbf{x}_C) \\ &+ \sum_{\mathcal{Y}_{pa_j}} P(\mathbf{y}_{pa_j}|\mathbf{x}_C) \log P(\mathbf{y}_{pa_j}|\mathbf{x}_C). \end{aligned} \tag{5.8}$$

The second term in the right side of equation (5.8) can be computed using the marginal distribution in the compound nodes. For example, if $\mathcal{Y}_{pa_j}$ denotes the set of parent nodes for node X_j, then it's straightforward to see that the marginal distribution at the compound node U_j is $P(\mathcal{Y}_{pa_j}|\mathbf{x}_C)$. For the first term in the right side of equation (5.8), we can compute it by piggybacking an additional message in LBP. More specifically, in the i^{th} iteration of LBP, the message from the U_j to X_j is

$$\begin{aligned} \mu^i_{u_j \to x_j} &= \sum_{\mathcal{Y}_{pa_j}} \Psi(U_j, X_j) \prod_{y_i \in \mathcal{Y}_{pa_j}} \mu^{i-1}_{y_i \to u_j} \\ &= P^i(x_j|\mathbf{x}_C). \end{aligned}$$

Therefore, $P(\mathbf{y}_{pa_j}, x_j|\mathbf{x}_C)$ can be obtained at the compound node U_j using

$$P(\mathbf{y}_{pa_j}, x_j|\mathbf{x}_C) = \Psi(U_j, X_j) \prod_{y_i \in \mathcal{Y}_{pa_j}} \mu^{i-1}_{y_i \to u_j},$$

We thus have finished the computation for sequential probe selection according to equation (5.7).

Implementation Issues. There are two main concerns for the implementation of the above mentioned algorithm. First is the mapping from the graphical model to the sensor network, as we should limit messaging across sensors whenever possible in order to save resources. Second is that whenever an additional probe is selected, the beliefs need to be updated, which requires repeated executions of LBP and may lead to very high computational complexity. We first describe how to map the graphical model to the sensor network and then propose a heuristic algorithm to speed up the inference algorithm by utilizing the redundancy in each execution of LBP.

Mapping the graphical model to the sensor network. In the graphical model Fig. 5.6 , the evidence nodes $\mathcal{X}$, compound nodes $\mathcal{U}$, and latent nodes $\mathcal{Y}$ are connected by edges that represent statistical correlations. On the other hand, in the sensor network, sensor nodes are connected through wireless channels, thus constitute a communication graph. The communication graph is in general different from the graphical model, so the graphical model must be mapped to the sensor network and then we can assign inference functions in each individual sensor node. More specifically, we assign each graphical model node to a unique sensor node so each sensor node is associated with a subset of the graphical model nodes and it will be responsible for handling the belief propagation within the subset of graphical model nodes assigned to it, and also to other graphical model nodes assigned to other sensor nodes.

Since the evidence nodes $\mathcal{X}$ correspond to the path state, we can assign each of them to the higher-layer network sensors that monitors the corresponding path.

Each compound node U_i corresponds to an evidence node X_i, so it can be assigned to the same sensor node as X_i. The assignment of the latent nodes is a little more involved as in the graphical model, the compound nodes are coupled through the latent nodes. However, we can decouple the compound nodes by making duplicated copies of the latent nodes. Similar ideas have been used in [69] for mapping a Markov Random Field (MRF) to a sensor network.

Assuming that in the Markov network Fig. 5.6, the compound nodes U_p and U_q are coupled through a latent node Y_r, i.e. Y_r is a common neighboring node for both U_p and U_q, the corresponding sensor nodes assigned to U_p and U_q, denoted by S_p and S_q, are assumed to be connected through a path consisting of the sensor nodes $(S_{r_1}, \ldots, S_{r_n})$. In order to assign the latent node Y_r to sensor nodes, we add new latent variables $(Z_{r_1}, \ldots, Z_{r_n})$ to the original graphical model, remove the link from Y_r to U_q, and add new links for $(Y_r, Z_{r_1}), (Z_{r_1}, Z_{r_2}), \ldots, (Z_{r_{n-1}}, Z_{r_n}), (Z_{r_n}, X_q)$ with

$$\Psi(Z_{r_1}, Y_r) = \boldsymbol{I}(Z_{r_1} = Y_r),$$

$$\Psi(Z_{r_{k+1}}, Z_{r_k}) = \boldsymbol{I}(Z_{r_{k+1}} = Z_{r_k}),\ k = 1, \ldots, n-1,$$

$$\Psi(X_q, Z_{r_n}) = P(X_q | Y_r),$$

where $\boldsymbol{I}(\cdot)$ is the indicator function. The newly added variables $(Z_{r_1}, \ldots, Z_{r_n})$ are assigned to the sensor nodes $(S_{r_1}, \ldots, S_{r_n})$ and the latent variable Y_r is assigned to the sensor S_p. Fig. 5.7 illustrates the process.

All coupled pairs of the compound nodes can be decoupled in this way, and

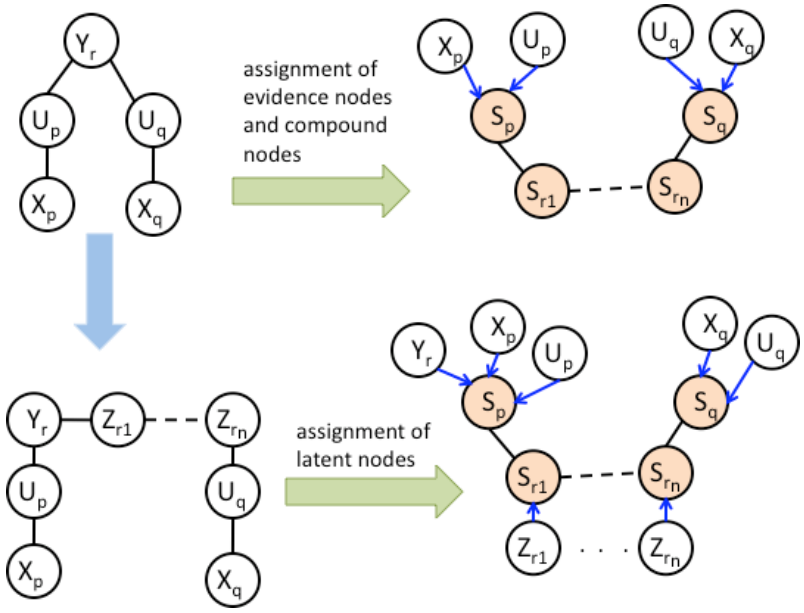


Figure 5.7: An example of mapping an MRF to a sensor network

the latent nodes can be assigned to corresponding sensor nodes.

Belief Updating. Belief updating is the most computationally intensive part in the second-phase probe selection algorithm as the beliefs need to be updated each time when one additional probe is collected. In [80], Zheng et al. also utilized belief propagation for online probe selection, however, the computation complexity is pretty high in their implementation due to the repeated executions of belief propagation. In [18], Cheng et al. proposed to improve algorithm efficiency based on an observation of approximated conditional independence of probes, but their method does not hold true in the online scenario where each probe is sequentially selected according to previous observations.

We propose a heuristic algorithm that exploits the redundancy in the repeated executions of LBP to reduce computational complexity. It is based on the observation that adding one evidence at a time may only affect a small region in the

graphical model. Therefore, messages should be updated only in that region. A similar principle is used for expanding frontier belief propagation (EFBP) in [55]. However, EFBP focuses on the case where the choice of evidence variables is fixed but the evidence changes, while in our case the choice of evidence variable is not fixed, i.e., one more evidence is added each time.

Our algorithm starts with one run of the full LBP algorithm to select the first probe given the observations from the first-phase probing. Then for each time when an additional probe is sent and the corresponding path state x_i is observed, the message from x_i to its neighboring nodes in the graphical model, will be updated and sent if and only if it differs by ϵ from the previous one, when x_i last participated in belief propagation. Similarly, if the neighboring node receives a new message, it will update and send messages to its own neighbors if and only if the new message differs by ϵ from the last one it received. Table 5.2 summarizes the algorithm, where $h(x_i)$ represents the hop count of the path x_i, $\tilde{h}_0$ is the communication constraint for the second phase, and $\mu_{s\to t}^{[k,i]}$ represents the message from node s to node t in the i^{th} iteration of the k^{th} run of our approximated LBP.

In most cases, the effect of adding one evidence dies out very quickly, so the number of message passing is greatly reduced compared to the full LBP algorithm. It is found by experiments that our heuristic approximation algorithm can achieve similar performance as the repeated executions of full LBP, while the speed is more than one order of magnitude faster.

We now provide a performance bound for the difference of the estimated marginal probabilities using LBP and our heuristic approximation method. In our

Table 5.2: An improved LBP based algorithm for second-phase probe selection

Initialization: perform LBP to select the first probe x_1, obtain initial belief messages $\mu_{\cdot\to\cdot}^{[0]}$,
$k = 1,\ h_k = h(x_1)$,
$converged = false$, $\mathcal{S}^{[k,0]} = \{x_1\}$,
while $h_k <= \tilde{h}_0$
  $i = 1$
  while $converged == false$
    $converged = true,\ \mathcal{S}^{[k,i]} = \emptyset$
    for $s \in \mathcal{S}^{[k,i-1]}$
      for $t \in \mathcal{N}(s)$
        compute $\mu_{s\to t}^{[k,i]}$ based on new received message
        if $|\mu_{s\to t}^{[k,i]} - \mu_{s\to t}^{[k,i-1]}| > \epsilon$
          $converged = false$,
          send $\mu_{s\to t}^{[k,i]}$ to t
          $\mu_{s\to t}^{[k,i-1]} = \mu_{s\to t}^{[k,i]},\ \mathcal{S}^{[k,i]} = \mathcal{S}^{[k,i]} \cup t$
        end if
      end for
    end for
    $i = i + 1$
  end while
  $k = k + 1$
  select the k^{th} probe x_k, $h_k = h_{k-1} + h(x_k)$
  $\mathcal{S}^{[k,0]} = \{x_k\}$
end while

approximation method, the belief message $\mu_{s\to t}^i$ from node s to node t in the i^{th} iteration could be different from the true value $\tilde{\mu}_{s\to t}^i$ by at most ϵ. This difference can be treated as a multiplicative error $e_{s\to t}^i$, i.e., $\mu_{s\to t}^i = \tilde{\mu}_{s\to t}^i \cdot e_{s\to t}^i$, where $e_{s\to t}^i$ is bounded by

$$1 - \epsilon/\tilde{\mu}_{s\to t}^i \le e_{s\to x_i}^i \le 1 + \epsilon/\tilde{\mu}_{s\to t}^i.$$

In [34], Ihler et al. introduced the concept of *dynamic range* of a function, defined

to be $d(f) = \sup_{a,b} f(a)/f(b)$. In our case, the dynamic range of $e^i_{s\to t}$ is bounded by

$$d(e^i_{s\to t}) \leq \frac{1+\epsilon/\tilde{\mu}^i_{s\to t}(a)}{1-\epsilon/\tilde{\mu}^i_{s\to t}(b)} = \delta(e^i_{s\to t}).$$

Then, according to the Theorem 15 in [34], we have the following theorem.

Theorem 5.3.3. *Denote p^*_t as the marginal probability of node t in the graphical model of Fig. 5.6 estimated by LBP at convergence, and p^n_t as the marginal probability of node t using our approximation method after n iterations. Then we have*

$$\log d(p^n_t/p^*_t) \leq \sum_{u\in nb(t)} \log v^n_{ut},$$

where v^n_{ut} is defined by the iteration

$$\begin{aligned} \log v^{i+1}_{ts} &= \log \frac{d(\Psi(t,s))^2\gamma^i_{ts}+1}{d(\Psi(t,s))^2+\gamma^i_{ts}} + \log \delta(e^i_{t\to s}), \\ \log \gamma^i_{ts} &= \sum_{u\in nb(t)\setminus s} \log v^i_{ut}, \end{aligned}$$

with the initial condition $v^1_{ut} = \delta(e^i_{s\to t})\cdot d(\Psi(u,t))^2$, where $d(\Psi(t,s))$ is the dynamic range for the potential function $\Psi(t,s)$.

Proof. The theorem is a direct application of Theorem 15 in [34] with $d(e_{s\to t}) \leq \delta(e^i_{t\to s})$ calculated for our case. □

5.4 Evaluations

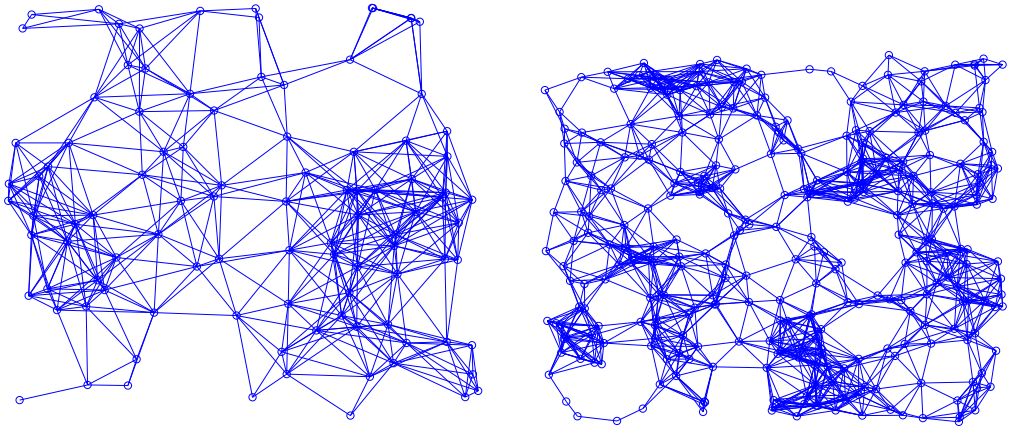
5.4.1 Experiment setup

To evaluate the performance of the proposed algorithm, we generate a set of different networks. The sensor nodes are uniformly deployed in a square field. Each node is assumed to have identical transmission radii. If an edge between two nodes is not greater than the radii, the two nodes are 1-hop neighbors. This is a widely used model called "unit disk model" [47]. We employed the model here due to its simplicity, so we can focus more on the performance of our algorithm. Table 5.3 summarizes the parameters for the evaluated networks with different sizes. For each type of networks, we averaged the parameters and experimental results over 100 simulations. Fig. 5.8 shows some example network topologies used in our experiment. In the generated networks, a small number, i.e., $5 \sim 7$ of the nodes are randomly selected as the higher-level stronger nodes to collect the end-to-end measurements. The monitored paths from the stronger nodes to other regular sensor nodes follow the shortest path calculation. Initially, all the monitored paths are considered as the suspicious paths in the network, i.e., they constitute the candidate probe set.

The focus of our experiments is to evaluate the probe selection algorithm for anomaly localization. We explicitly do not focus on the accuracy of the detection scheme on individual path measurements. For this reason, rather than creating network anomalies endogenously by simulating possible malicious attack scenarios, we assume that the ground truth for the malicious links is known in advance. The

Table 5.3: Network parameters

Net size	Avg. node degree	Num. of M nodes	Num. of links	Num. Num. paths	Avg. path lengths
100	5.36	5	268	485	3.30
200	5.06	5	506	985	5.21
300	4.68	5	702	1779	5.30
400	5.09	5	1049	1985	6.39
500	6.27	7	1568	3472	7.83



(a) Network size 100 (b) Network size 300

Figure 5.8: Example network topologies

effect of the performance of any employed detection scheme is captured as noise in the Noisy-OR model. The occurrence of an anomaly on each link in the network is modeled as a Bernoulli process. The probability associated with each Bernoulli process can be adjusted for different anomaly densities, i.e., to make the anomaly density higher, the success probability of the Bernoulli process will be increased. For simplicity, we assume that once an anomaly occurs on a link, the link remains anomalous until being detected.

An individual simulation experiment proceeds as follows. In the first phase, a

small set of probes is selected and sent, given the communication constraint. Based on the anomalous path measurements discovered by these probes, we reduce the suspicious area for the second phase probing. More specifically, the candidate suspicious paths for the second phase will only contain those paths that are intersecting with the discovered anomalous paths in the first phase. We then perform a second phase probing, in which, the probes are sequentially selected to maximize the predicted diagnosis quality under the constraint on probing overhead. For each additional probe selection, our approximate LBP algorithm is performed. When the probing overhead constraint is violated, we stop sending probes and infer the locations of malicious links that are responsible for all the observed path anomalies during the two phase probing. The inference is a maximum a posteriori estimation based on LBP.

We implement our algorithm in MATLAB and report results for the two probing phases under different experimental settings, including different network sizes, communication constraints and anomaly densities. Since existing works typically do not consider a strict communication constraint, we design and implement a simple baseline algorithm that can work under communication constraints and compare our trust-assisted algorithm with the baseline algorithm. The baseline algorithm is assumed to be not aware of the link trust information. It follows the same procedures as the trust-assisted algorithm but assumes that each link is equally likely to be anomalous with probability 0.5.

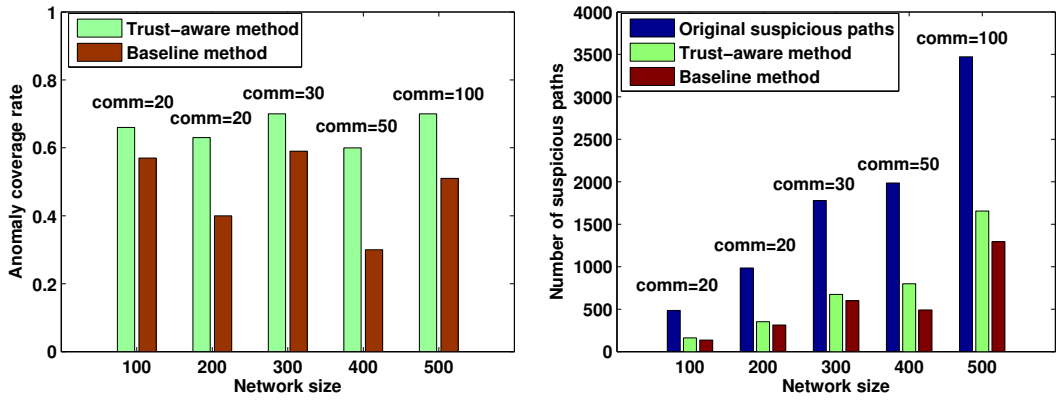
5.4.2 Experimental results

First-phase probing. The first-phase probing generates a set of candidate probes for the second phase probing. The goal is to narrow the suspicious area in the network while cover as many anomalous links as possible given the communication constraint. To measure the probe selection performance for this phase, we propose two metrics based on the generated candidate probe set.

The first metric is the anomaly coverage rate, which is computed as the ratio of the number of anomalous links in the generated candidate probe set to the number of all anomalous links in the network. It measures our expectation that the candidate probe set should cover as many anomalous links as possible. The second metric is the suspicious area measured by the size of the candidate probe set. With smaller size of candidate probe set, the second phase probing can be faster. These two metrics, to some level, measure two conflicting properties of the candidate probe set, i.e., a large candidate probe set usually contains more anomalous links. However, we want to optimize the tradeoff between the two metrics. The ideal case would be to have a small candidate probe set but cover all the anomalous links. Considering the randomness of anomaly occurrences in the network, this is not always possible. We show that our trust-assisted algorithm can achieve relatively good performance on optimizing the tradeoff between the two metrics, when compared to the original candidate probe set and the one generated by the baseline method.

Fig. 5.9(a) shows the anomaly coverage rates for our trust-assisted method and the baseline method in different networks under the same communication con-

straints. Fig. 5.9(b) shows the size of the corresponding candidate probe sets. The x-axis in the two figures correspond to the network size while other network parameters are shown in table 5.3. The number above the bars are the corresponding communication constraints used for the first phase, in terms of the number of links traversed by the selected probes. For larger network sizes, a larger communication constraint is necessary for a fair comparison. Anomaly densities in these networks are around 20%, i.e., about 20% of the links in the networks are anomalous.



(a) Anomaly coverage rate

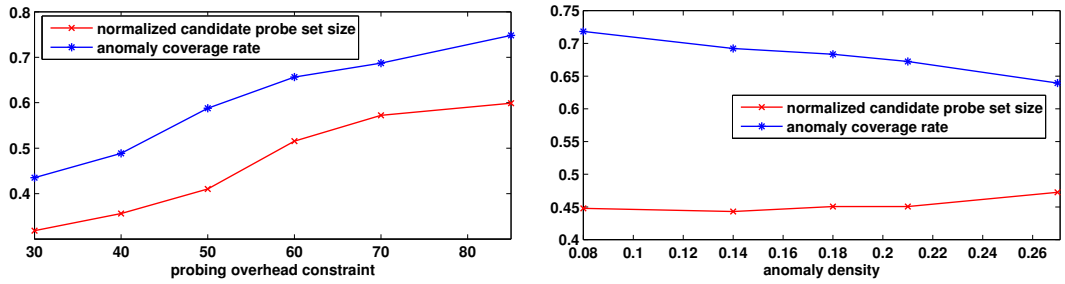
(b) Candidate probe set size

Figure 5.9: Comparison of the trust-aware method and the baseline method

From Fig. 5.9(a) we can see that the trust-assisted method can cover more anomalous links than the baseline method given the same communication constraints, while Fig. 5.9(b) demonstrates that both the trust-aware algorithm and the baseline algorithm have scaled down the suspicious areas: the number of candidate probes have been greatly reduced. We also note that the baseline method may achieve a slightly smaller probe set; because it can only cover much fewer anomalous links.

We next investigate the effects of different probing overhead constraints and

anomaly densities for the first phase probing. Fig. 5.10(a) shows the anomaly coverage rate and candidate probe set size using our trust-assisted selection method when the probing overhead constraint varies. For illustration purposes, we show the two metrics together in one figure, where candidate probe set size is normalized over the original probe set size. Fig. 5.10(b) shows the results when the probing overhead constraint is fixed to be 50 but the anomaly density varies. Network size is 400 for both figures.



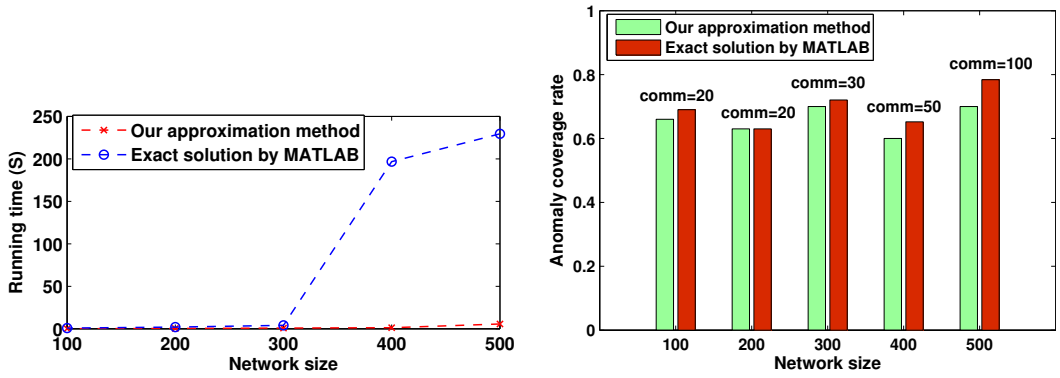
(a) Probing overhead constraint varies (b) Anomaly density varies

Figure 5.10: Probe selection performance

We can see that when the probing overhead constraint increases, the candidate probe set size and the anomaly coverage rate both increase, which is as expected. However, the anomaly density does not affect the first-phase probe selection too much, which is probably due to the randomness of anomaly occurrence on the links. Given the same probing overhead constraint, the increase of anomaly density only slightly degrades the performance, in terms of a slightly larger candidate probe size and a slightly smaller anomaly coverage rate.

Another concern for the first-phase probe selection, as mentioned before, is the computational complexity of the algorithm. We compare our approximation method implemented in MATLAB to the exact method provided by MATLAB in-

teger programming solver, in terms of both speed and performance. Fig. 5.11(a) shows the running time of our method and the exact solution for different networks. Fig. 5.11(b) shows the anomaly coverage rate for the two methods. Since the sizes of candidate probe set selected by the two methods are within very small difference, we do not show them here.



(a) Running time (b) Anomaly coverage rate

Figure 5.11: Comparison of the approximation method and the exact solution

In Fig. 5.11(a), we note that for small networks, the exact solution is only about 3 times slower than the approximation method. This is because the communication constraint for these networks is also small, so the budgeted maximum coverage problem has a small scale. However, the approximation method starts to provide much superior performance over the exact method when the network size becomes larger, e.g., for network size larger than 300, the approximation method is more than one order of magnitude faster, while reduction on the anomaly coverage rate is less than 0.1. Considering the much larger running time of the exact solution, e.g, in the scale of minutes, the performance reduction of the approximation method is acceptable.

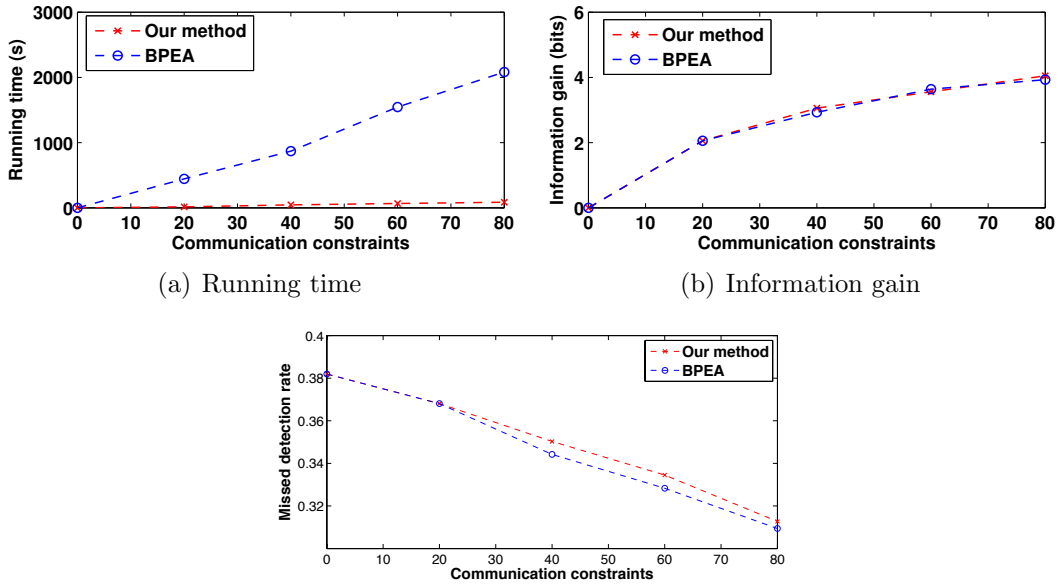
Second-phase probing For the second-phase probing, the goal is to find the individual links that are responsible for the observed path anomalies. The performance is thus evaluated in terms of the missed detection rate (MDR) and false alarm rate (FAR). The missed detection rate represents the percentage of anomalous links in the network that are not identified as anomalous, while the false alarm rate represents the percentage of normal links that are recognized as anomalous. Denote $\mathcal{E}$ as the set of monitored links, $\mathcal{E}_A$ as the set of anomalous links, and $\mathcal{E}_D$ as the set of anomalous links that are correctly identified. Then the MDR and FAR are computed as

$$MDR = 1 - \frac{|\mathcal{E}_A \cap \mathcal{E}_D|}{|\mathcal{E}_A|},\ FAR = \frac{|\mathcal{E}_D \cap (\mathcal{E} - \mathcal{E}_A)|}{|\mathcal{E} - \mathcal{E}_A|}.$$

The second-phase probing involves the repeated executions of the approximated LBP like algorithm for sequential probe selection. We have implemented our algorithm on top of the belief propagation algorithm from Kevin Murphy's Bayes Net toolbox [54] in MATLAB. To demonstrate the effectiveness of our approximation method on improving algorithm efficiency, we also implemented the BPEA (Belief Propagation for Entropy Approximation) algorithm proposed by Zheng et al. [80], which, to the best of our knowledge, provides the best inference accuracy in the literature. In BPEA, repeated executions of the LBP algorithm is also used for online probe selection, however, it did not exploit the redundancy in each execution of LBP, so it has very high computational complexity.

Fig. 5.12 shows the performance comparison of our method and the BPEA

algorithm in a network with 400 nodes. Other network parameters are shown in the corresponding row in Table 5.3. The anomaly density is around 30%. The candidate probe set is generated from the first phase probe selection given communication overhead at 50 hops. Fig. 5.12(a) compares the running time of our method and the BPEA method. Fig. 5.12(b) compares the information gain in bits for the two methods. The information gain measures the reduction on uncertainty for the maximum probability estimation if the selected probes are sent, computed using the conditional entropy. Fig. 5.12(c) shows the missed detection rates for the two methods. The false alarm rate is very small for both methods, i.e., around 0.04, and sending more probes did not change the false alarm rate much, so we don't show it here. The x-axis in the three figures corresponds to the communication constraint for the second phase, measured by the number of hops traversed by the probes.



(a) Running time

(b) Information gain

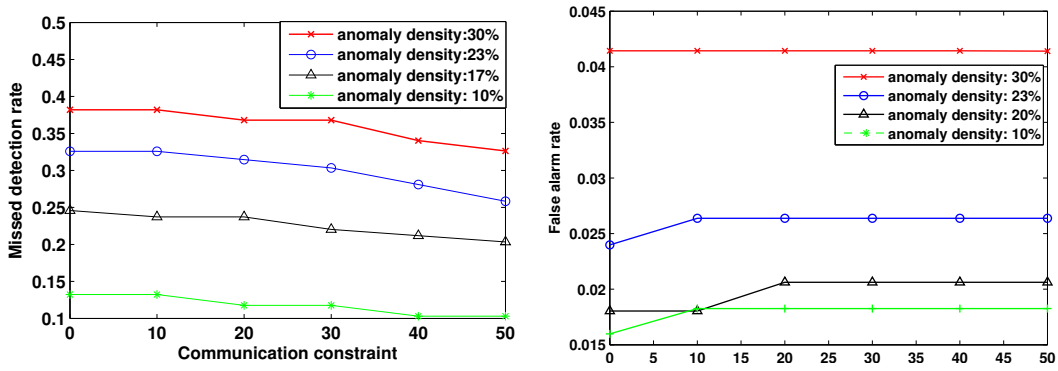
(c) Missed detection rate

Figure 5.12: Comparison of BPEA and our approach

We can see that our approximation method indeed speeds up the probe selec-

tion process a lot, i.e., more than one order of magnitude than BPEA. In the case that the communication constraint is 80 hops, BPEA needs about 35 minutes, which is not realistic for real-time probe selection, while our approach only needs one and a half minutes. The information gain obtained by the two methods are almost the same for given communication constraints, while the missed detection rate is also similar.

Next we change the anomaly densities in the network. Fig. 5.13 shows the corresponding changes of MDR and FAR. It is observed that when the anomaly density reduces, both MDR and FAR become lower. The reason is that, when there are few anomalous links, the probes sent can concentrate more on the area near these anomalous links, thus increase the inference accuracy. When the number of anomalous links increases, the uncertainty on their locations becomes larger, and using the same amount of probes will not be enough.

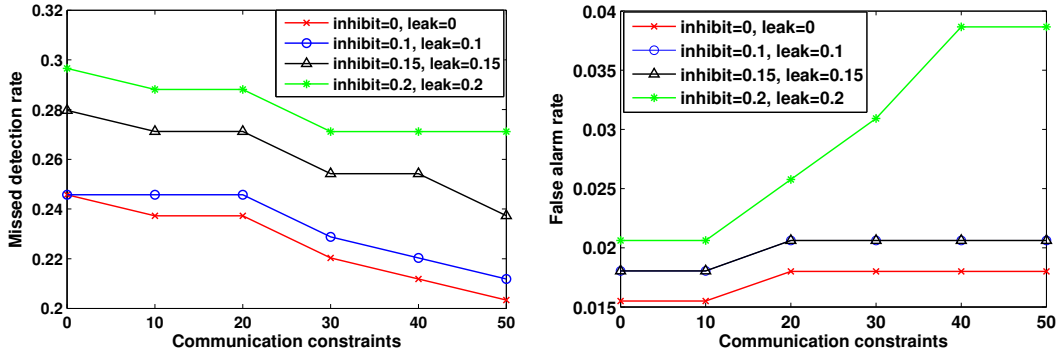


(a) Missed detection rate (b) False alarm rate

Figure 5.13: Performance when anomaly density varies

The inference accuracy on the locations of anomalous links is also affected by the noise level in the network, i.e., the leakage probability and the inhibition probability in the Noisy-OR model, where the leakage probability captures the situation

that a path performs as anomalous even if all its constituting links are normal, and the inhibition probability measures the situation that some links in the path are anomalous but the path performs as normal. We investigate the impact of different noise levels on the performance of our algorithm. Fig. 5.14 compares the MDR and FAR when the inhibition and leakage probability varies. The anomaly density is around 17%. Generally speaking, the performance degrades when more noise is injected into the network, which is as expected. It is also found that for low noise level, e.g., $inhibit < 0.1, leak < 0.1$, the performance does not change much.



(a) Missed detection rate (b) False alarm rate

Figure 5.14: Performance under different levels of noise

5.5 Discussions

In this Chapter, we presented a trust-assisted framework for anomaly detection and localization in resource constrained wireless sensor networks using end-to-end traffic measurements. In contrast to most prior work that focus on detection and localization accuracy, our focus is on the tradeoff between inference accuracy and the resource consumption in sensor networks. Especially, we are interested in the case that there is a strict constraint on the communication bandwidth for anomaly de-

tection and localization. We proposed a hierarchical network structure and designed an efficient two-phase probing scheme that utilize link trust information to achieve a good tradeoff between inference accuracy and probing overhead. Simulation results demonstrate the efficiency and effectiveness of our algorithms.

Chapter 6

Analysis of Long Range Dependent Traffic Effects on Anomaly Detection

6.1 Introduction and related work

In Chapter 5, we discussed the problem of network monitoring and anomaly localization. The results of anomaly localization not only depends on the probing strategy and location inference algorithm, but also highly depends on the anomaly detection techniques applied on the collected network traffic measurements. In this chapter, we will discuss anomaly detection techniques for Long Range Dependent (LRD) traffic in sensor networks. Especially, recent studies have shown that node mobility along with spatial correlation of the monitored phenomenon in sensor networks can lead to LRD traffic [73], which can cause high false alarms if using traditional anomaly detection methods, as traditional anomaly detection methods usually assume that the traffic measurements are either independent or Short Range Dependent (SRD).

Anomaly detection methods can be generally classified in two categories: signature-based and statistics-based [22]. Signature-based detection methods use attack signatures to identify anomalies. The attack signatures are collected based on historical observations under the same attack, thus it can not be applied to detect unknown

anomalies. Statistics-based detection methods overcome this drawback by only modeling normal network traffic and treat everything that falls outside the normal scope as anomalies. A typical statistics-based anomaly detection usually consists of the following steps [22]: first, collect network measurements and model the normal traffic as a reference, then, apply a decision rule to detect whether current network traffic deviates from the reference. In the decision rule, some statistical distance between the analyzed traffic and the reference is computed, then it must be decided whether the distance is large enough to trigger an alarm.

The modeling of the normal traffic can be based on various statistical characteristics of the data. For example, Lakhina et al. [48] proposed to use principal component analysis to identify an orthogonal basis along which the network measurements exhibit the highest variance. The principal components with high variance model the normal behavior of a network, whereas the remaining components of small variance are used to identify and classify anomalies. Schereer et al. [63] used a non-Gaussian long-range dependent process to model network traffic, which can provide several statistics such as the marginal distribution and the covariance to characterize the traffic. Zhang et al. [78] proposed a spatio-temporal model using traffic matrices that specify the traffic volumes between origin and destination pairs in a network. Anomalies are detected by finding significant differences from historical observations. Spectral density [17, 33] and covariance [36], have also been used for modeling normal network traffic.

Besides these methods, wavelet transform is another popular technique used for analyzing network traffic, especially for the LRD traffic. Abry et al. [2] proposed

a wavelet-based tool for analyzing LRD time series and a related semi-parametric estimator for estimating LRD parameters. Barford et al. [6] assume that the low frequency band signal of a wavelet transform represents the normal traffic pattern. They then normalize both medium and high frequency band signals to compute a weighted sum of the two signals. An alarm is raised if the variance of the combined signal exceeds a pre-selected threshold. In [45], Kim et al. used a wavelet-based technique for de-noising and separating queueing delay caused by network congestions from various other delay variations. Zuraniewski et al. [85] have combined wavelet transforms and change-point detection algorithms to detect the instants that the fractality changes noticeably. The key feature of these wavelet-based methods lies in the fact that the wavelet transform can turn the LRD that exists among the data samples into a short memory structure among the wavelet coefficients [1]. In our work, we build a wavelet-domain multi-level hidden Markov model for the LRD network traffic. The merit of our method is the model's mathematical tractability and its capability of capturing data dependencies.

To measure the deviation of the analyzed traffic from the reference model, several statistical distances can be used, including simple threshold, mean quadratic distance [63], and entropy [29]. Entropy is a measure of the uncertainty of a probability distribution. It can be used to compare certain qualitative differences of probability distributions. Gu et al. [29] used a maximum entropy technique to estimate the reference traffic model and compute a distance measure related to the relative entropy of the analyzed network traffic with respect to the reference. Nychis et al. [58] thoroughly evaluated entropy-based metrics for anomaly detection. In our

detection scheme, we apply the symmetric relative entropy as a distance measure. The online EM algorithm can efficiently compute the symmetric relative entropy between the current HMM model and a previous estimated one. Anomalies are then detected as abrupt changes in the symmetric relative entropy measurements.

6.2 Long range dependent traffic in sensor networks

A time series $\{x(t)\}_{t=1}^{N}$ is considered to be long range dependent if its autocorrelation function $\rho(k)$ decays at a rate slower than an exponential decay. Typically, $\rho(k)$ asymptotically behaves as ck^{2H-2} for $0.5 < H < 1$, where $c > 0$ is a constant and H is the Hurst parameter. The intensity of LRD is expressed as the speed of the decay for the autocorrelation function and is measured by the Hurst parameter, i.e., as $H \to 1$, the dependence among data becomes stronger. It can be shown that $\sum_{k=1}^{\infty} \rho(k) = \infty$. Intuitively, LRD implies that the process has infinite memory, i.e., individually small high-lag correlations have an important cumulative effect. This is contrast to the conventional Short Range Dependent (SRD) processes which are characterized by an exponential decay of the autocorrelations resulting in a summable autocorrelation function. LRD is an important property for traffic modeling as it is likely to be responsible for the decrease in both network performance and quality of service [63] .

For empirical time series, a number of time and frequency domain estimators have been developed [72] for detecting LRD and quantifying its intensity, including the aggregated variance plot, R/S analysis, periodogram analysis and Whittle's

estimator. The variance-time plot examines the decay of the sample variance at increasing time aggregation levels which, for LRD time series, should be slower than the reciprocal of the sample size. The R/S method examines the growth in the rescaled range of the partial sums on the standard deviations of the time series, as a function of the sample number in the time-aggregated series. The periodogram method is based on the discrete Fourier transform and is an estimate of the power spectral density of a discrete process which, for LRD time series, should exhibit power-law behavior for frequencies close to the origin. Whittle's estimator is a maximum likelihood type estimator which is applied to the periodogram of the time series. It requires the empirical series to be a Gaussian process and the underlying form of the series must be provided, usually FGN or ARFIMA. We use these four estimators for analyzing the behavior of network traffic in wireless sensor networks.

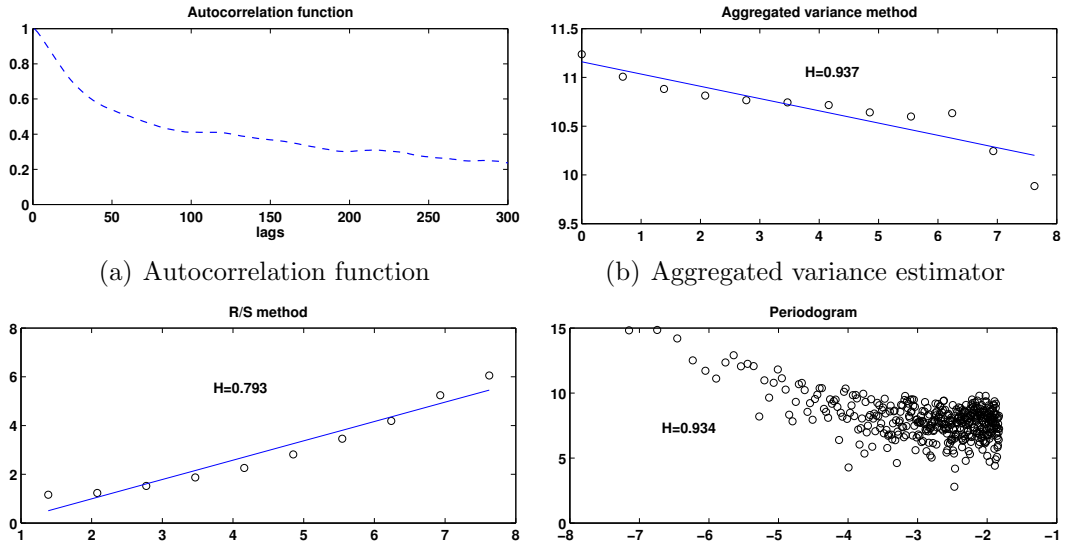
A wireless sensor network operates on the IEEE 802.15.4 standard. It has been shown recently that the traffic generated from a single mobile node in a wireless sensor network can be represented by an ON/OFF process $X(t)$, where the probability density function of the ON period τ_a can be approximated by a truncated Pareto distribution [73]

$$f_{\tau_a}(x) = \frac{\gamma_a x^{-(\gamma_a+1)}}{t_{min}^{-\gamma_a} - t_{max}^{-\gamma_a}},$$

with $t_{min}(t_{max})$ denoting the minimum (maximum) ON time and γ_a denoting the tail index. The value of γ_a depends on the variability of node mobility pattern and the spatial correlation of the monitored phenomena by the network. Using this traffic model, we analyze the dynamic behaviors of sensor network traffic measurements,

such as the packet round trip time, the number of received packets per second, etc., by conducting experiments using Network Simulator 2 (NS-2). To determine whether the collected data traces have the LRD property, we apply the aggregate variance estimator, R/S estimator, Periodogram estimator and the Whittle estimator using the SELFIS tool [41].

Fig. 6.1 illustrates one representative result using the collected packet round trip time between a source-destination pair. Fig. 6.1(a) shows the autocorrela-



(a) Autocorrelation function

(b) Aggregated variance estimator

(c) R/S estimator

(d) Periodogram estimator

Figure 6.1: Autocorrelation function and LRD estimators for packet round-trip time in a wireless sensor network

tion function of the collected data samples. Fig. 6.1(b), 6.1(c) and 6.1(d) correspond to the Hurst parameter estimations using the aggregate variance estimator, R/S estimator, and Periodogram estimator respectively. The Whittle's estimator is a maximum likelihood estimator so it is not shown in the figure. The estimated Hurst parameter using the Whittle's estimator is 0.781 with 95% confidence interval $[0.760 - 0.801]$. We can see that LRD do exist in the collected network measure-

ments. It is also observed that the aggregate variance estimator is more consistent with the periodogram estimator, while the R/S estimator is more consistent with Whittle estimator for this trace. Varied estimates among different LRD estimators are frequently reported in literature [10, 42]. We stress that here we do not examine estimator accuracy but the existence of LRD in the collected data.

Similar LRD properties are observed in several other network measurements such as the number of packets received per second, link throughput, etc. Since traditional anomaly detection methods for sensor networks usually assume that network measurements are either independent or SRD, the existence of LRD can lead to high false alarm rates for these methods. Incorporating LRD traffic characteristics accurately in the design of anomaly detection schemes for sensor networks is critical and important.

6.3 Anomaly detection using wavelet domain hidden Markov model

Wavelet transforms have been popular [2, 6] for analyzing autocorrelated time series due to their capability to compress multi-scale features and approximately de-correlate the time series. They can provide compact information about a signal at different locations in time and frequency. Our traffic model is in the wavelet domain. More specifically, we build a Hidden Markov Model (HMM) for the wavelet transformed network measurements. The basic idea for transform domain models is that a linear invertible transform can often restructure a signal, generating transform coefficients whose structure is simpler to model.

6.3.1 Wavelet domain hidden Markov model

In wavelet transform (decomposition), the measurements $x(t), t = 1, \ldots, N$ are decomposed into multiple scales by a weighted sum of a certain orthonormal basis functions,

$$x(t) = \sum_{k=1}^{N} a_{L,k}\phi_{L,k}(t) + \sum_{m=1}^{L}\sum_{k} d_{m,k}\psi_{m,k}(t),$$

where $\phi_{L,k},\ \psi_{m,k}$ are the orthonormal basis, $a_{L,k},\ d_{m,k}$ are the approximation and detail coefficients. The approximation coefficients $a_{L,k}$ provide the general shape of the signal, while the detail coefficients $d_{m,k}$ from different scales provide different levels of details for the signal content, with $d_{1,k}$ providing the finest details and $d_{L,k}$ providing the coarsest details. The locality and multi-resolution properties enable the wavelet transform to efficiently match a wide range of signal characteristics from high-frequency transients and edges to slowly varying harmonics.

In our work, we apply the Discrete Wavelet Transform (DWT) to measured network traffic. A DWT is a wavelet transform for which the basis functions are discretely sampled. DWT can be explained using a pair of quadrature mirror filters, which includes a high pass filter $h[n]$ and a low pass filter $g[n]$ [51]. Efficient methods have been developed for decomposing a signal using a family of wavelet basis functions based on convolution with the corresponding quadrature mirror filters. A 2-level discrete wavelet transform using the corresponding quadrature mirror filters is illustrated in Fig. 6.2.

However, the wavelet transform cannot completely decorrelate real-world sig-

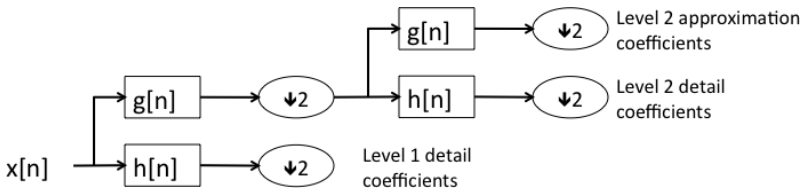


Figure 6.2: 2-level discrete wavelet transform

nals, i.e., a residual dependency always remains among the wavelet coefficients. A key factor for a successful wavelet-based algorithm is an accurate joint probability model for the wavelet coefficients [20]. A complete model for the joint probability density function would be too complicated, if not impossible, to obtain in practice, while modeling the wavelet coefficients as independent is simple but disregards the inter-coefficient dependencies. To strike a balance between the two extremes, we use a Hidden Markov Model (HMM) to capture the remaining dependency among the wavelet coefficients. It is based on two properties of the wavelet transform [20, 50]: first is the *Clustering* property, meaning that if a particular wavelet coefficient is large/small, the adjacent coefficients are very likely to also be large/small; second is the *Persistence* property, meaning that large/small values of wavelet coefficients tend to propagate across scales.

For the wavelet coefficients $d_{j,k},\ j = 1, \ldots, L$ and $k = 1, \ldots, n_j$, where L is the decomposition level and n_j is the number of wavelet coefficients in scale j, we assume that each $d_{j,k}$ is associated with a hidden state $s_{j,k}$. We then use a hidden

Markov model to characterize the wavelet coefficients through the factorization

$$
\begin{aligned}
&P(\{d_{1,i}, s_{1,i}\}_{i=1}^{n_1}, \ldots, \{d_{L,i}, s_{L,i}\}_{i=1}^{n_L}) \\
&= p(s_{L,1}) \prod_{j=2}^{n_L} p(s_{L,j}|s_{L,j-1}) \prod_{i=1}^{L-1} p(s_{i,1}|s_{i+1,1}) \\
&\cdot \prod_{i=1}^{L-1} \prod_{j=2}^{n_i} p(s_{i,j}|s_{i,j-1}, s_{i+1,\lceil j/2 \rceil}) \prod_{i=1}^{L} \prod_{j=1}^{n_i} p(d_{i,j}|s_{i,j}).
\end{aligned} \tag{6.1}
$$

This factorization involves three main conditional independence assumptions: first, conditioned on the states at the previous coarser scale $i+1$, the states at scale i form a first order Markov chain; second, conditioned on the corresponding state at the previous coarser scale $i+1$, i.e., $s_{i+1,\lceil j/2 \rceil}$, and the previous state at the same scale, i.e., $s_{i,j-1}$, the state $s_{i,j}$ is independent of all states in the coarser scales; third, the wavelet coefficients are independent of everything else given their hidden states. The three independence assumptions are critical for deriving the inference algorithms for this wavelet domain HMM. Fig. 6.3 illustrate a hidden Markov model for a 3-level wavelet decomposition.

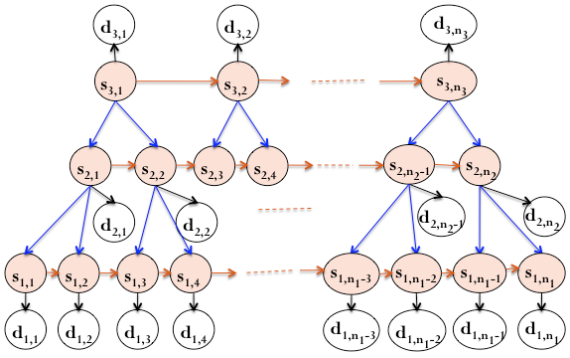


Figure 6.3: HMM for 3-level wavelet decomposition

6.3.2 Model estimation: backward-forward decomposition

6.3.2.1 The Expectation-Maximization (EM) algorithm

Denote the set of wavelet coefficients by $\mathcal{D} = \{\{d_{L,i}\}_{i=1}^{n_L}, \ldots, \{d_{1,i}\}_{i=1}^{n_1}\}$ and their hidden states by $\mathcal{S} = \{\{s_{L,i}\}_{i=1}^{n_L}, \ldots, \{s_{1,i}\}_{i=1}^{n_1}\}$ respectively, where n_i is the number of wavelet coefficients in the i^{th} scale. The parameters of the HMM include the following three probability distributions: first is the initial probability for the state $s_{L,1}$, i.e., $\pi_k = P(s_{L,1} = k),\ k \in \mathcal{K}$, where $\mathcal{K}$ represents the domain of the hidden states; second is the two types of state transition probabilities, i.e.,

$$\pi_{k_1|k_2}^{i,1} = P(s_{i,1} = k_1 | s_{i+1,1} = k_2) \text{ for } i < L,$$

$$\pi_{k_1|k_2,k_3}^{i} = P(s_{i,j} = k_1 | s_{i,j-1} = k_2, s_{i+1,\lceil j/2 \rceil} = k_3);$$

and third is the conditional probability of the wavelet coefficient given its hidden state, i.e., $P(d_{i,j}|s_{i,j} = k)$, which can be modeled by a mixture Gaussian distribution. For simplicity and presentation clarity, we use a single Gaussian distribution to capture $P(d_{i,j}|s_{i,j} = k)$, i.e, $P(d_{i,j}|s_{i,j} = k) \sim \mathcal{N}(\mu_k^i, \sigma_k^i)$, where μ_k^i and σ_k^i are the mean and the standard deviation for the state k in the i^{th} scale. The extension to a mixture Gaussian distribution is straightforward. These model parameters, denoted by $\boldsymbol{\theta} = \{\pi_k, \pi_{k_1|k_2}^{i,1}, \pi_{k_1|k_2,k_3}^{i}, \mu_k^i, \sigma_k^i\}$, can be estimated from the real data using the maximum likelihood criterion. Due to the intractability of direct maximization of the likelihood function, we apply an Expectation Maximization (EM) algorithm to estimate the parameters.

The EM algorithm provides an maximum likelihood estimation of model parameters by iteratively applying an E-step and an M-step. In the E-step, the expected value of the log likelihood function $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = E_{\mathcal{S}|\mathcal{D},\boldsymbol{\theta}^{(t)}}[\log P_{\boldsymbol{\theta}}(\mathcal{S},\mathcal{D})]$ is computed. Then in the M-step, the parameters that maximizes $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ are computed, i.e., $\boldsymbol{\theta}^{(t+1)} = arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$.

To implement the two steps, we define the following posterior probabilities,

$$\begin{aligned}
\gamma_k^{i,j} &= P(s_{i,j} = k|\mathcal{D}), \\
\gamma_{k_1,k_2}^{i,1} &= P(s_{i,1} = k_1, s_{i+1,1} = k_2|\mathcal{D}), \text{ for } i < L \\
\gamma_{k_1,k_2,k_3}^{i,j} &= P(s_{i,j} = k_1, s_{i,j-1} = k_2, s_{i,\lceil j/2 \rceil} = k_3|\mathcal{D}).
\end{aligned}$$

According to equation (6.1), maximizing $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ using the Lagrange multiplier method leads to the following estimate of $\boldsymbol{\theta}$,

$$\begin{aligned}
&\pi_k = \gamma_k^{L,1}, \quad \pi_{k_1|k_2}^{i,1} = \frac{\gamma_{k_1,k_2}^{i,1}}{\sum_{l\in\mathcal{K}} \gamma_{l,k_2}^{i,1}}, \quad \pi_{k_1|k_2,k_3}^{i} = \frac{\sum_{j=2}^{n_i} \gamma_{k_1,k_2,k_3}^{i,j}}{\sum_{l\in\mathcal{K}} \sum_{j=2}^{n_i} \gamma_{l,k_2,k_3}^{i,j}}, \\
&\mu_k^i = \frac{\sum_{j=1}^{n_i} \gamma_k^{i,j} d_{i,j}}{\sum_{j=1}^{n_i} \gamma_k^{i,j}}, \quad (\sigma_k^i)^2 = \frac{\sum_{j=1}^{n_i} \gamma_k^{i,j} (d_{i,j} - \mu_k^i)^2}{\sum_{j=1}^{n_i} \gamma_k^{i,j}}.
\end{aligned}$$

The computation of the posterior probabilities $\gamma(\cdot)$ is more involved. Using a brute force computation by direct marginalization will take $O(N \cdot |\mathcal{K}|^N)$ operations, where $|\mathcal{K}|$ represents the cardinality of set $\mathcal{K}$ and N is the length of the input signal. However, by exploiting the sparse factorization in equation (6.1) and manipulating the distributive property of '+' and '×', we are able to design a forward-backward

decomposition algorithm to compute these posterior probabilities with computational complexity $O(N \cdot |\mathcal{K}|^{L+1})$, where L is the wavelet decomposition level and is much smaller than N.

6.3.2.2 Forward-backward decomposition

Our algorithm extends the classical forward-backward decomposition algorithm for a one-level hidden Markov model to our multi-level case. The key idea is to only maintain L appropriate hidden states in both the forward and backward variables for computational efficiency.

Forward decomposition. Let

$$\begin{aligned}
\mathcal{S}_{i,j} &= \{s_{L,\lceil 2^{i-L}j\rceil}, \ldots, s_{i+1,\lceil 2^{-1}j\rceil}, s_{i,j}, s_{i-1,2(j-1)}, \ldots, s_{1,2^{i-1}(j-1)}\}, \\
\mathcal{D}_{i,j} &= \{d_{L,k\le\lceil 2^{i-L}j\rceil}, \ldots, d_{i+1,k\le\lceil 2^{-1}j\rceil}, d_{i,k\le j}, d_{i-1,k\le 2(j-1)}, \ldots, d_{1,k\le 2^{i-1}(j-1)}\},
\end{aligned}$$

we define the forward variable to be $\alpha_{i,j} = P(\mathcal{S}_{i,j}, \mathcal{D}_{i,j})$. Denote $[\alpha_{1,2^{h-1}j}] = f(h, \alpha_{h,j})$ for $h, j \in Z^+$, to be a dynamic programming algorithm with input parameters $(h, \alpha_{h,j})$ and output parameter $\alpha_{1,2^{h-1}j}$. The pseudo-code for computing the forward variables using dynamic programming is shown in Table 6.1. Its correctness can be proved using the three conditional independence assumptions in our HMM, which is shown in Appendix B.1. For simplicity and presentation clarity, in Table 6.1 we assume that the input data length N is a power of 2, and denote the conditional probability $P(d_{i,j}|s_{i,j})$ by $g_1(d_{i,j})$, and $P(s_{i,j}|s_{i,j-1}, s_{i+1,\lceil j/2\rceil})$ by $g_2(s_{i,j})$.

Table 6.1: Algorithm for computing forward variables

Initialization: $\alpha_{L,1} = P(s_{L,1}, d_{L,1})$
For $k_L = 1$ to $2^{-L}N$
$\alpha_{1,2^{L-1}k_L} = f(L, \alpha_{L,k_L})$
$\alpha_{L,k_L+1} = g_1(d_{L,k_L+1}) \sum_{s_{L,k_L}} [g_2(s_{L,k_L+1}) \cdot \alpha_{1,2^{L-1}k_L}]$
end

function $[\alpha_{1,2^{h-1}j}] = f(h, \alpha_{h,j})$
If $h == 2$,
$\alpha_{1,2j-1} = g_1(d_{1,2j-1}) \sum_{s_{1,2j-2}} [g_2(s_{1,2j-1}) \cdot \alpha_{2,j}]$
$\alpha_{1,2j} = g_1(d_{1,2j}|s_{1,2j}) \sum_{s_{1,2j-1}} [g_2(s_{1,2j}) \cdot \alpha_{1,2j-1}]$
else
$\alpha_{h-1,2j-1} = g_1(d_{h-1,2j-1}) \sum_{s_{h-1,2j-2}} [g_2(s_{h-1,2j-1}) \cdot \alpha_{h,j}]$
$\alpha_{1,2^{h-2}(2j-1)} = f(h-1, \alpha_{h-1,2j-1})$
$\alpha_{h-1,2j} = g_1(d_{h-1,2j}) \sum_{s_{h-1,2j-1}} [g_2(s_{h-1,2j}) \cdot \alpha_{1,2^{h-2}(2j-1)}]$
$\alpha_{1,2^{h-2}(2j)} = f(h-1, \alpha_{h-1,2j})$
End

There is some implementation issue for the algorithm in Table 6.1, namely, the numerical under- or over-flow of $\alpha_{i,j}$ as $P(\mathcal{S}_{i,j}, \mathcal{D}_{i,j})$ becomes smaller and smaller with the increasing size of the observations $\mathcal{D}_{i,j}$. Therefore, it is necessary to scale the forward variables by positive real numbers to keep the numerical values within reasonable bounds. One solution is to use a scaled version $\bar{\alpha}_{i,j} = \frac{\alpha_{i,j}}{c_{i,j}}$, where $c_{i,j} = \sum_{\mathcal{S}_{i,j}} \alpha_{i,j}$. In this way, $\bar{\alpha}_{i,j}$ represents the probability $P(\mathcal{S}_{i,j}|\mathcal{D}_{i,j})$ and $c_{i,j}$ represents the probability $P(d_{i,j}|\mathcal{D}_{i,j} \backslash d_{i,j})$. It is straightforward to prove that both $c_{i,j}$ and $\bar{\alpha}_{i,j}$ do not depend on the number of observations. The algorithm for computing $(\bar{\alpha}_{i,j}, c_{i,j})$ can be obtained by adding a normalization step after each update of $\alpha_{i,j}$ for the algorithm in Table 6.1. A by-product for the *scaled* forward decomposition

algorithm is that the log-likelihood $\log P(\mathcal{D})$ can be easily computed as

$$\log P(\mathcal{D}) = \sum_{i=1}^{L} \sum_{j=1}^{n_i} \log c_{i,j}.$$

Backward decomposition. Letting $\mathcal{D}_{i,j}^c = \mathcal{D} - \mathcal{D}_{i,j}$, we define the backward variable to be $\beta_{i,j} = P(\mathcal{D}_{i,j}^c | \mathcal{S}_{i,j})$. It can be computed using a similar dynamic programming algorithm as the one in Table 6.1. To avoid the numerical under- or over-flow problem, instead of computing $\beta_{i,j}$, we compute a scaled version $\bar{\beta}_{i,j}$ as is shown in Table 6.2. The scaled backward variable $\bar{\beta}_{i,j}$ represents the probability $\frac{P(\mathcal{D}_{i,j}^c | \mathcal{S}_{i,j})}{P(\mathcal{D}_{i,j}^c | \mathcal{D}_{i,j})}$. The correctness of the algorithm in Table 6.2 can be verified using the three conditional independence assumptions in our HMM, which is shown in Appendix B.2.

Computing posterior probabilities. Since $\bar{\alpha}_{i,j} = P(\mathcal{S}_{i,j} | \mathcal{D}_{i,j})$ and $\bar{\beta}_{i,j} = \frac{P(\mathcal{D}_{i,j}^c | \mathcal{S}_{i,j})}{P(\mathcal{D}_{i,j}^c | \mathcal{D}_{i,j})}$, we have $\bar{\alpha}_{i,j} \cdot \bar{\beta}_{i,j} = P(\mathcal{S}_{i,j} | \mathcal{D})$ according to the Markovian property of our HMM. Then the posterior probability $\gamma(\cdot)$ can be computed as

$$\begin{aligned}
\gamma_k^{i,j} &= \sum \bar{\alpha}_{i,j} \cdot \bar{\beta}_{i,j}, \\
\gamma_{k_1,k_2}^{i,1} &= \sum \bar{\alpha}_{i,1} \cdot \bar{\beta}_{i,1}, \\
\gamma_{k_1,k_2}^{L,j} &= \sum \bar{\alpha}_{1,2^{L-1}(j-1)} \cdot \bar{\beta}_{L,j} \cdot \frac{g_1(d_{L,j}) g_2(s_{L,j})}{c_{L,j}}, \\
\gamma_{k_1,k_2,k_3}^{i,j} &= \begin{cases} \sum \bar{\alpha}_{1,2^{i-1}(j-1)} \bar{\beta}_{i,j} \cdot \frac{g_1(d_{i,j}) \cdot g_2(s_{i,j})}{c_{i,j}}, \text{if j is even,} \\ \sum \bar{\alpha}_{i+1,\lceil j/2 \rceil} \bar{\beta}_{i,j} \cdot \frac{g_1(d_{i,j}) \cdot g_2(s_{i,j})}{c_{i,j}}, \text{if j is odd.} \end{cases}
\end{aligned}$$

Table 6.2: Algorithm for computing the scaled backward variables

Initialization: $\bar{\beta}_{1,N/2} = 1$

For $k_L = 2^{-L}N$ to 1

$\bar{\beta}_{L,k_L} = f(L, \bar{\beta}_{1,2^{L-1}k_L})$

$$\bar{\beta}_{1,2^{L-1}(k_L-1)} = \sum_{s_{L,k_L}} \frac{g_1(d_{L,k_L})g_2(s_{L,k_L})\bar{\beta}_{L,k_L}}{c_{L,k_L}}$$

end

function $[\bar{\beta}_{h,j}] = f(h, \bar{\beta}_{1,2^{h-1}j})$

If $h == 2$,

$$\bar{\beta}_{1,2j-1} = \sum_{s_{1,2j}} \frac{g_1(d_{1,2j})g_2(s_{1,2j}) \cdot \bar{\beta}_{1,2j}}{c_{1,2j}}$$

$$\bar{\beta}_{2,j} = \sum_{s_{1,2j-1}} \frac{g_1(d_{1,2j-1})g_2(s_{1,2j-1}) \cdot \bar{\beta}_{1,2j-1}}{c_{1,2j-1}}$$

else

$\bar{\beta}_{h-1,2j} = f(h-1, \bar{\beta}_{1,2^{h-2}2j})$

$$\bar{\beta}_{1,2^{h-2}(2j-1)} = \sum_{s_{h-1,2j}} \frac{g_1(d_{h-1,2j})g_2(s_{h-1,2j}) \cdot \bar{\beta}_{h-1,2j}}{c_{h-1,2j}}$$

$\bar{\beta}_{h-1,2j-1} = f(h-1, \bar{\beta}_{1,2^{h-2}(2j-1)})$

$$\bar{\beta}_{h,j} = \sum_{s_{h-1,2j-1}} \frac{g_1(d_{h-1,2j-1})g_2(s_{h-1,2j-1}) \cdot \bar{\beta}_{h-1,2j-1}}{c_{h-1,2j-1}}$$

End

Without confusion, we omit the variables under $\sum$ for the above equations. The correctness of these equations is proved in Appendix B.3.

6.3.3 Anomaly detection by tracking model variations

A first thought on the anomaly detection problem is to treat the anomalies as abrupt changes in the HMM model data and then apply change-point detection methods to detect these abrupt changes as anomalies. This is also the general routine used in the literature [22]. However, it is found that directly applying change-point detection methods to the HMM modeled data is computationally expensive.

We designed here a lightweight anomaly detection scheme based on detecting the structural changes of the estimated HMM.

Difficulty of applying change-point detection methods directly on the HMM modeled data. An anomaly detection problem can be formulated as a hypotheses testing problem, i.e., given finite samples $\mathcal{Y}_{1:N} = \{y_1, y_2, \ldots, y_N\}$, testing between two hypotheses,

$$H_0 : \text{for } 1 \leq k \leq N, p_{\boldsymbol{\theta}}(y_k|\mathcal{Y}_{1:k-1}) = p_{\boldsymbol{\theta}_0}(y_k|\mathcal{Y}_{1:k-1}),$$

$$H_1 : \exists \text{ unknown } 1 \leq t_0 \leq N, \ s.t.$$
$$\begin{cases} \text{for } 1 \leq k \leq t_0 - 1, & p_{\boldsymbol{\theta}}(y_k|\mathcal{Y}_{1:k-1}) = p_{\boldsymbol{\theta}_0}(y_k|\mathcal{Y}_{1:k-1}), \\ \text{for } t_0 \leq k \leq N, & p_{\boldsymbol{\theta}}(y_k|\mathcal{Y}_{1:k-1}) = p_{\boldsymbol{\theta}_1}(y_k|\mathcal{Y}_{1:k-1}), \end{cases}$$

where $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ represent the model parameters for the normal network traffic and the abnormal network traffic respectively. Since $\boldsymbol{\theta}_1$ usually can not be known in advance, the hypothesis H_1 is composite (i.e., $\boldsymbol{\theta}_1 \in \{\boldsymbol{\theta} : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0\}$). The generalized likelihood ratio test (GLR) [16] is one of the most popular change-point detection method for solving this type of hypothesis testing problem. The GLR test can be written as

$$\begin{aligned} g_k &= \max_{1 \leq j \leq k} \sup_{\boldsymbol{\theta}_1} S_j^k \\ S_j^k &= \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\theta}_1}(y_i|\mathcal{Y}_{1:i-1})}{p_{\boldsymbol{\theta}_0}(y_i|\mathcal{Y}_{1:i-1})} \\ t_0 &= \min\{k : g_k \geq h\} \end{aligned}$$

It is known that the likelihood of an HMM belongs to the so called *locally asymptotic normal* family [14], and the GLR statistic $\sup_{\boldsymbol{\theta}_1} S_j^k$ can be approximated by the second-order expansion of S_j^k at $\boldsymbol{\theta}_0$ without the computation of $\sup_{\boldsymbol{\theta}_1}$ over all possible $\boldsymbol{\theta}_1$'s. However, the computation of this second-order expansion involves computation of the Fisher information matrix of $\ln p_{\boldsymbol{\theta}_0}(y_i|\mathcal{Y}_{1:i-1})$, which, in our case, would require an update of an $L|\mathcal{K}|^3 \times L|\mathcal{K}|^3$ matrix each time when a new data sample arrives. This is not computationally realistic for our application, especially in the resource constrained wireless sensor networks.

In the next subsections, we design a lightweight algorithm for anomaly detection by detecting structural changes of the estimated HMM. An important requirement for anomaly detection is to make the decision making process online. Therefore, we first develop an online EM algorithm for HMM model estimation.

An online discounting EM algorithm The online discounting EM algorithm is derived based on the so called *limiting EM algorithm* [13]. We first briefly present the limiting EM algorithm. Let $\mathbf{x}$ denote the hidden states and $\mathbf{y}$ denote the observations. If the joint probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ belongs to an exponential family such that

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = h(\mathbf{x}, \mathbf{y}) \exp(\langle \phi(\boldsymbol{\theta}), ss(\mathbf{x}, \mathbf{y}) \rangle - A(\boldsymbol{\theta}))$$

where $\langle \cdot \rangle$ denotes the scalar product, $ss(\mathbf{x}, \mathbf{y})$ is the sufficient statistic for $\boldsymbol{\theta}$ and $A(\boldsymbol{\theta})$ is some log-partition function. If the equation $\langle \nabla_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}), ss \rangle - \nabla_{\boldsymbol{\theta}} A(\boldsymbol{\theta}) = 0$ has

a unique solution, denoted by $\boldsymbol{\theta} = \bar{\boldsymbol{\theta}}(ss)$, then the limiting EM algorithm obeys the simple recursion

$$ss^{t+1} = E_{\boldsymbol{\theta}^*}[E_{\bar{\boldsymbol{\theta}}(ss^t)}[ss(\mathbf{x}, \mathbf{y})|\mathbf{y}]],$$

where $\boldsymbol{\theta}^*$ represents the true model parameter . Since $E_{\boldsymbol{\theta}^*}[E_{\boldsymbol{\theta}}[ss(\mathbf{x}, \mathbf{y})|\mathbf{y}]]$ can be estimated consistently from the observations by $\frac{1}{N}\sum_{t=1}^{N} E_{\boldsymbol{\theta}}[ss(x_t, y_t)|y_t]$, an online EM algorithm can be obtained by using the conventional stochastic approximation procedure [13],

$$\hat{ss}^{t+1} = \gamma_{t+1} E_{\bar{\boldsymbol{\theta}}(\hat{ss}^t)}[ss(x_{t+1}, y_{t+1})|y_{t+1}] + (1 - \gamma_{t+1})\hat{ss}^t,$$

where γ_{t+1} is a time discounting factor. The estimation of model parameters can then be derived from the sufficient statistic $\hat{ss}$. It is proved in [13] that under suitable assumptions, this online EM algorithm is an asymptotically efficient estimator of the model parameter $\boldsymbol{\theta}^*$.

It is not difficult to see that the joint probability distribution of our HMM model, i.e., $P(\mathcal{S}, \mathcal{D})$, satisfies the above mentioned conditions. For each wavelet coefficient $d_{i,j}$, we have the following sufficient statistics for computing the HMM model parameters,

$\tau_{l,k}^{i,j} = \sum_{m=1}^{n_l^{i,j}} P(s_{l,m} = k, \mathcal{S}_{i,j}|\mathcal{D}_{i,j})$,

$\hat{\tau}_{l,k}^{i,j} = \sum_{m=1}^{n_l^{i,j}} P(s_{l,m} = k, \mathcal{S}_{i,j}|\mathcal{D}_{i,j}) \cdot d_{l,m}$,

$\bar{\tau}_{l,k}^{i,j} = \sum_{m=1}^{n_l^{i,j}} P(s_{l,m} = k, \mathcal{S}_{i,j}|\mathcal{D}_{i,j}) \cdot d_{l,m}^2$,

$\tau_{l,k_1,k_2,k_3}^{i,j} = \sum_{m=2}^{n_l^{i,j}} P(s_{l,m} = k_1, s_{l,m-1} = k_2, s_{l+1,\lceil m/2 \rceil} = k_3, \mathcal{S}_{i,j}|\mathcal{D}_{i,j})$,

where $l \in \{1, \ldots, L\}$ is the scale index, $k \in \mathcal{K}$ is the hidden state index, and $n_l^{i,j}$ represents the number of observed wavelet coefficients in scale l after $d_{i,j}$ arrives, i.e.,

$$n_l^{i,j} = \begin{cases} \lceil 2^{i-l} j \rceil & \text{if } l \geq i, \\ 2^{l-i} j & \text{if } l < i. \end{cases}$$

It is straightforward to prove that the HMM model parameters $\{\pi^l_{k_1|k_2,k_3}, \mu^l_k, \sigma^l_k\}$ can be updated using $\{\tau^{i,j}_{l,k}, \hat{\tau}^{i,j}_{l,k}, \bar{\tau}^{i,j}_{l,k}, \tau^{i,j}_{l,k_1,k_2,k_3}\}$ as follows,

$$\pi^l_{k_1|k_2,k_3} = \frac{\sum_{\mathcal{S}_{i,j}} \tau^{i,j}_{l,k_1,k_2,k_3}}{\sum_{k_1} \sum_{\mathcal{S}_{i,j}} \tau^{i,j}_{l,k_1,k_2,k_3}}, \tag{6.2}$$

$$\mu^l_k = \frac{\sum_k \sum_{\mathcal{S}_{i,j}} \hat{\tau}^{i,j}_{l,k}}{\sum_k \sum_{\mathcal{S}_{i,j}} \tau^{i,j}_{l,k}}, \tag{6.3}$$

$$(\sigma^l_k)^2 = \frac{\sum_k \sum_{\mathcal{S}_{i,j}} \bar{\tau}^{i,j}_{l,k}}{\sum_k \sum_{\mathcal{S}_{i,j}} \tau^{i,j}_{l,k}} - (\frac{\sum_k \sum_{\mathcal{S}_{i,j}} \hat{\tau}^{i,j}_{l,k}}{\sum_k \sum_{\mathcal{S}_{i,j}} \tau^{i,j}_{l,k}})^2. \tag{6.4}$$

The other HMM parameters π^k and $\pi^{i,1}_{k_1|k_2}$ can be updated using sufficient statistics defined as $\tilde{\tau}^{i,j}_{L,k} = P(s_{L,1} = k, \mathcal{S}_{i,j}|\mathcal{D}_{i,j})$ and $\tilde{\tau}^{i,j}_{l,1} = P(s_{l,1} = k_1, s_{l+1,1} = k_2, \mathcal{S}_{i,j}|\mathcal{D}_{i,j})$. We omit the related computations here, as they are similar.

The next step on the design of our online EM algorithm is to obtain recursive (online) updates of the sufficient statistics $\tau^{i,j}_{l,k}$, $\hat{\tau}^{i,j}_{l,k}$, $\bar{\tau}^{i,j}_{l,k}$ and $\tau^{i,j}_{l,k_1,k_2,k_3}$. According to the Markovian property of our HMM, the online updates of the sufficient statistic can be computed by following a similar dynamic programming procedure as the one for computing the scaled forward variables $\bar{\alpha}_{i,j}$. Recall that $\bar{\alpha}_{i,j}$ is computed by adding a normalization step after $\alpha_{i,j}$ is computed in the algorithm in Table 6.1.

The sufficient statistics are updated once $\bar{\alpha}_{i,j}$ is updated. For illustration purposes, we show here how to update the sufficient statistics when $\alpha_{h,2j-1}$ in Table 6.1 is computed. Updates for the other cases are similar.

For $l \in \{1, \ldots, L\}$, let $\gamma_{h-1,2j-1}$ be a time discounting factor, and δ^l_{h-1} be the Dirac Delta function such that

$$\delta^l_{h-1} = \begin{cases} 1 & \text{if } l = h-1, \\ 0 & \text{if } l \neq h-1. \end{cases}$$

Define

$r^l_{h-1,2j-1} = \delta^l_{h-1} \cdot \gamma_{h-1,2j-1} \cdot \bar{\alpha}_{h-1,2j-1}$,

$t^l_{h-1,2j-1} = (1 - \delta^l_{h-1}\gamma_{h-1,2j-1})\frac{g_1(d_{h-1,2j-1})}{c_{h,j}}$,

$q^l_{h-1,2j-1} = \delta^l_{h-1}\gamma_{h-1,2j-1}\frac{g_1(d_{h-1,2j-1})g_2(s_{h-1,2j-1})\bar{\alpha}_{h,j}}{c_{h-1,2j-1}}$,

where $g_1(\cdot)$ and $g_2(\cdot)$ are defined as in Table 6.1. We have the following equations for updating the sufficient statistics,

$$\tau^{h-1,2j-1}_{l,k} = r^l_{h-1,2j-1} + t^l_{h-1,2j-1} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1})\tau^{h,j}_{l,k}, \tag{6.5}$$

$$\hat{\tau}^{h-1,2j-1}_{l,k} = r^l_{h-1,2j-1} d_{h-1,2j-1} + t^l_{h-1,2j-1} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1})\hat{\tau}^{h,j}_{l,k}, \tag{6.6}$$

$$\bar{\tau}^{h-1,2j-1}_{l,k} = r^l_{h-1,2j-1} d^2_{h-1,2j-1} + t^l_{h-1,2j-1} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1})\bar{\tau}^{h,j}_{l,k}, \tag{6.7}$$

$$\tau^{h-1,2j-1}_{l,k_1,k_2,k_3} = q^l_{h-1,2j-1} + t^l_{h-1,2j-1} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1}) \cdot \tau^{h,j}_{l,k_1,k_2,k_3}. \tag{6.8}$$

The correctness of these updates are proved in Appendix B.4 using the conditional independence assumptions in our HMM.

In summary, the online discounting EM algorithm works as follows. Each time a new wavelet coefficient arrives, the sufficient statistics are updated accordingly, e.g., when $d_{h,2j-1}$ arrives, updating the sufficient statistics using equations (6.5), (6.6), (6.7), and (6.8). After a minimum number n_{min} of wavelet coefficients are observed, where n_{min} is small, i.e., $n_{min} = 20$ might be enough, the HMM model parameters are updated according to equation (6.2, 6.3, 6.4).

Change-point detection on model variations To measure the structural changes of the estimated HMM models over time, we use the concept of the symmetric relative entropy to define a model variation score [32]. Denote the model at time $t-1$ and t by P_{t-1} and P_t respectively, then the model variation score is defined to be

$$v_t = \lim_{n\to\infty} \frac{1}{n} D(P_t||P_{t-1}) + \lim_{n\to\infty} \frac{1}{n} D(P_{t-1}||P_t),$$

where $D(p||q)$ represents the relative entropy of distribution p to q, and n represents the length of the input data. It is natural to let $n \to \infty$ as we can then compare the two models under the stationary states in the limit of $n \to \infty$.

It is proved in Appendix B.5 that

$$\lim_{n\to\infty} \frac{1}{n} D(P_t||P_{t-1}) = \sum_{i=1}^{L} \frac{1}{2^i} D((\pi^i_{k_1|k_2,k_3})^t||(\pi^i_{k_1|k_2,k_3})^{t-1})$$
$$+ \sum_{i=1}^{L} \frac{1}{2^i} \sum_k (\pi^i_k)^t D(\mathcal{N}((\mu^i_k)^t, (\sigma^i_k)^t)||\mathcal{N}((\mu^i_k)^{t-1}, (\sigma^i_k)^{t-1})),$$

where $\pi^i_k = P(s_{i,j} = k)$. Therefore, besides the probability distributions $\pi^i_{k_1|k_2,k_3}$ and

$\mathcal{N}(\mu_k^i, \sigma_k^i)$ provided by the online EM algorithm, the computation of $\lim_{n\to\infty} \frac{1}{n} D(P_t||P_{t-1})$ also involves the probability distributions $\pi^i_{k_1,k_2,k_3} = P(s_{i,j} = k_1, s_{i,j-1} = k_2, s_{i+1,\lceil j/2 \rceil} = k_3)$ and π_k^i. The estimation of π_k^i and $\pi^i_{k_1,k_2,k_3}$ can be obtained from the sufficient statistics $\tau_{i,k}^{l,m}$ and $\tau_{i,k_1,k_2,k_3}^{l,m}$ as follows,

$$\pi_k^i = \sum_{\mathcal{S}_{l,m}} \tau_{i,k}^{l,m}, \quad \pi^i_{k_1,k_2,k_3} = \sum_{\mathcal{S}_{l,m}} \tau^{l,m}_{i,k_1,k_2,k_3}.$$

The other relative entropy $\lim_{n\to\infty} \frac{1}{n} D(P_{t-1}||P_t)$ can be computed similarly. We can see that the symmetric model variation score actually captures two types of changes. The first is the changes in the transition probabilities of the hidden states while the second is the changes in the generation pattern of the observed data from a fixed state. By using the symmetric relative entropy as a distance measure between two HMM models, it is expected that not only the changes of the data generation pattern will be detected, but also the changes in the hidden states can also be detected.

6.4 Evaluations

6.4.1 Numerical evaluations

In this section, we numerically evaluate the statistical properties of the proposed anomaly detection scheme. Two types of LRD time series, including the Fractional Gaussian Noise (FGN) and the Autoregressive Fractionally Integrated Moving Average (ARFIMA) model, are used for generating data. We then inject two types of model variations as anomalies. First is the mean level shift, i.e., a

step function with a constant amplitude is imposed on the original signal. Second is to vary model parameters for the data generation process, including the standard deviation and the Hurst parameter for FGN and ARFIMA. The duration for the normal state and the anomaly state is generated from exponential distributions with different mean values. The performance of the detection scheme is evaluated by the detection latency and the Receiver Operating Characteristic (ROC) curve, which is a plot of the detection rate versus the false alarm rate at different thresholds.

The selection of the wavelet basis used in our scheme is based on a balance between its *time localization* and *frequency localization* characteristics [6]. Long filters usually have poor time localization, which can lead to excessive blurring in the time domain, thus may miss strong but short-duration changes in the time series. In contrast, short filters have good time localization but poor frequency localization, which can lead to the appearance of large wavelet coefficients when no significant event is occurring and can cause high false alarms rates if detection is based on a simple threshold. In our scheme, we build a hidden Markov model for the wavelet coefficients and the detection is based on HMM structural changes rather than a simple threshold, thus the sensitivity of the filter's frequency localization capability on detection performance is significantly reduced. In our experiments, we found that the D2 (Haar wavelets) and D4 wavelets from the Daubechies family wavelets can give us relatively good performance. Hence we use the Haar wavelets for all the experiments in this thesis.

For performance comparison, we implement a baseline method adapted from [6], in which only the mean and variance of the wavelet coefficients is used for

anomaly detection. More specifically, for the wavelet coefficients in each scale, the method computes the mean and variance over a time window with fixed length. Any abrupt changes in the mean and variance values are treated as anomalies.

Detection on Mean level shift. We first discuss the detection performance on mean level shifts in the synthetic LRD time series. Fig. 6.4 shows one representative example. The top figure illustrates the time series, which is generated from an ARFIMA model with Hurst parameter 0.9 and length 2^{15}. The standard deviation for the generated data sequence is set to 1. The mean level shift occurs at the first quarter of the time and ends at the middle with intensity 0.75, which is less than the standard deviation. The bottom two figures show the corresponding model variation scores computed by our online EM algorithm with 5-level and 4-level wavelet decomposition respectively. The x axis represents the aggregated time due to the wavelet decomposition and the y axis represents the model variation score.

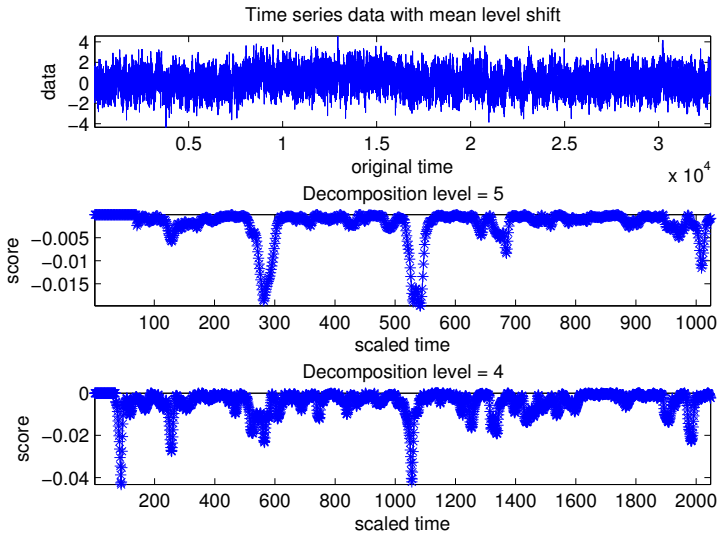


Figure 6.4: Effects of different decomposition levels

From Fig. 6.4, we can see that the visual inspection of the injected mean level shift from the time series directly can be difficult. However, in the tracked model variation scores, there are abrupt changes of the model variation scores at the two time locations where the mean level shift starts and ends. These two abrupt changes suggest where the injection starts and ends. Especially when the wavelet decomposition level is 5, these are the only 2 abrupt changes that exist. When the wavelet decomposition level is 4, there are some false alarms. We further compare the effects of the wavelet decomposition level on detection performance. Fig 6.5 shows the Receiver Operating Characteristic (ROC) curves using different decomposition levels. The ROC curves are obtained over 1000 randomly generated FGN time series with standard deviation 1. The mean level drift starts at different random time points with length 2^{13}. For a fair comparison, all of them have the same injected intensity of 0.75.

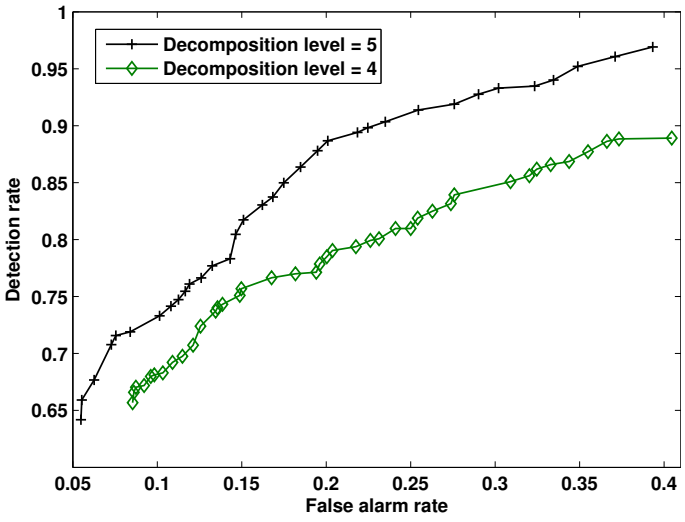


Figure 6.5: ROC curves for different decomposition levels

We can see that a higher decomposition level can reduce the false alarms while

achieving the same detection rate. However, an L-level decomposition has a 2^L time aggregation scale, i.e., it transforms the data samples within a 2^L time window to the wavelet domain so the wavelet coefficients within that window are time-indistinguishable. Therefore, a higher level decomposition would often give longer detection latency than that of a smaller level decomposition. In our experiments, we found that a 5-level wavelet decomposition can give a reasonably good balance between detection accuracy and latency.

The intensity of the injected mean level shift also affects the detection performance. Fig. 6.6 shows the ROC curve and detection latency for the injected mean shift with different intensities. Each curve is obtained over 1000 simulation traces with the 5-level wavelet decomposition. As is expected, for higher injected mean level shifts, the detection becomes much easier, in terms of lower false alarms, higher detection rates, and smaller detection latency. For example, the detection for injected mean level shift with intensity 1.25, which is only slightly higher than the standard deviation, is very accurate and fast.

Detection on changes of data generation model. We next evaluate the ability of our algorithm to detect the second type of anomalies, i.e., anomalies that change model parameters for the data generation process, including the Hurst parameter and the data variance.

Fig. 6.7 shows a typical example of the performance on detecting the Hurst parameter changes. The top figure shows the generated data, which is initially from an ARFIMA model with Hurst parameter 0.9 and standard deviation 1. From the

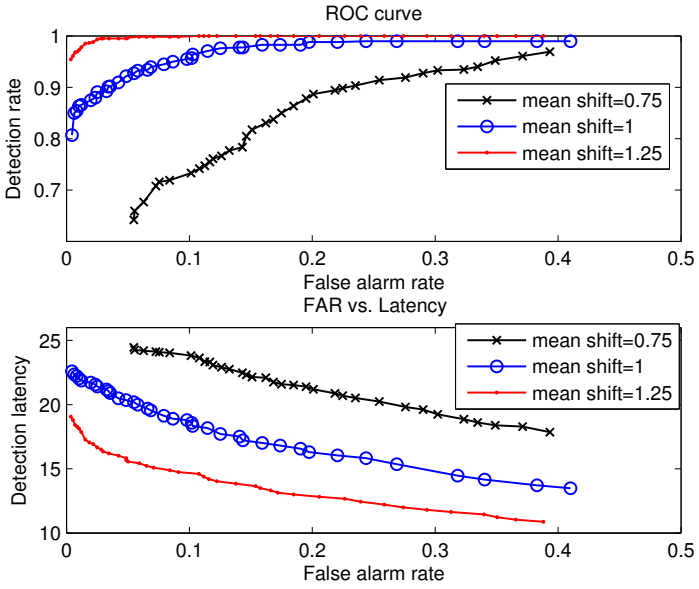


Figure 6.6: Effects of different injected intensities

first quarter to the middle of the time series, it changes to an ARFIMA model with Hurst parameter 0.7 (the standard deviation is unchanged). The bottom figure shows the model variation scores tracked by our online EM algorithm, from which we clearly see that two abrupt changes exist around the time when the Hurst parameter change starts and ends.

Fig. 6.8 shows the ROC curve and detection latency for the detection on Hurst parameter changes. Three types of scenarios are simulated, including the change of Hurst parameter from 0.9 to 0.8, 0.7 and 0.6 respectively. Each curve is obtained from 1000 simulation traces. It can be seen that when the Hurst parameter changes more, the detection becomes easier, i.e., the false alarm rate is lower, the detection rate is higher, and the detection latency is smaller.

We next evaluate the performance of our algorithm on the detection of data variance changes. Fig. 6.9 shows detection results over 1000 simulation traces for

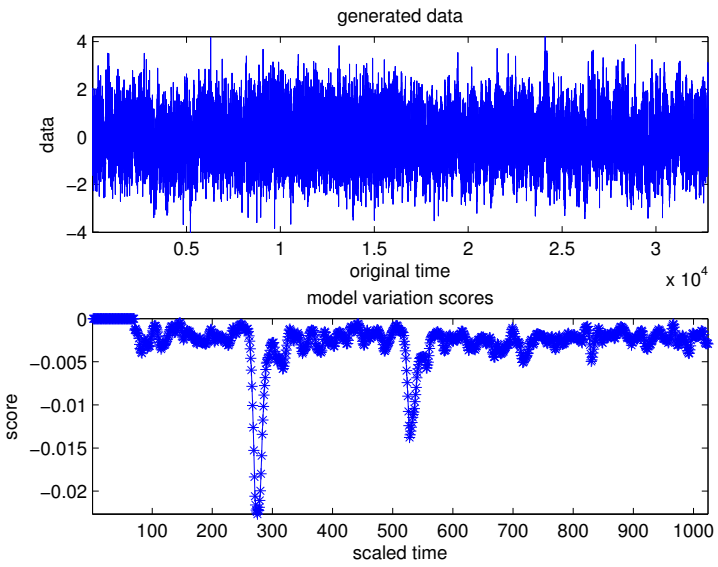


Figure 6.7: Detection of changes on Hurst parameter

data variance changing from 1 to 1.2 and 1.5 respectively. As expected, the detection of variance changed from 1 to 1.5 is much easier than that from 1 to 1.2.

Performance comparison to baseline method. We also compare the performance of our algorithm to the baseline method. It is observed that our method can always beat the baseline method. For example, Fig. 6.10 shows the ROC curves and detection latency for our method and the baseline method on the detection of Hurst parameter changing from 0.9 to 0.7. The results are obtained over 1000 simulations. While achieving the same false alarm rate, our method has a higher detection rate and smaller detection latency. Similar results are observed for the detection of other types of injected anomalies. These experiments verify our expectation that the hidden Markov model can capture more characteristics of the wavelet domain data than using only the first and second order statistics.

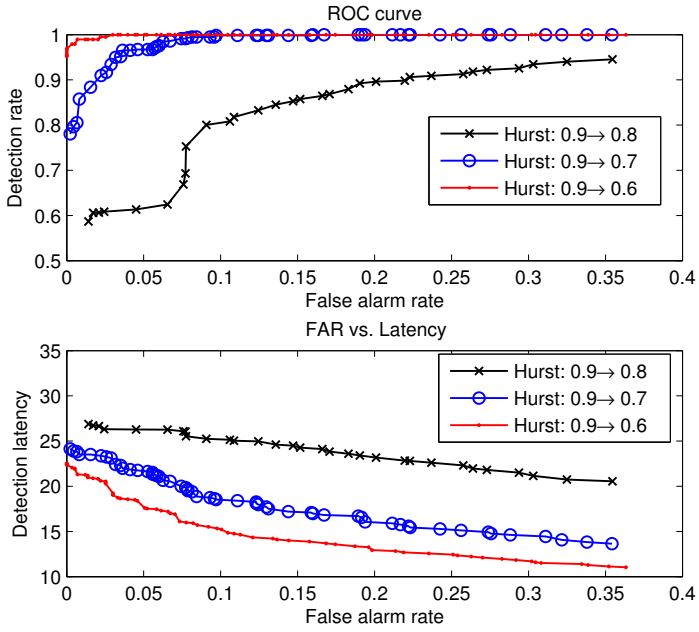


Figure 6.8: Detection of changes on Hurst parameter

6.4.2 NS-2 simulation studies

We next create anomaly scenarios in wireless sensor networks using the NS-2 simulator. More specifically, we simulate the wormhole attack in the routing layer and evaluate the proposed detection scheme. Note that our detection scheme is a general method, i.e., it is not only designed for the wormhole attacks, but can adapt to detect any other attacks that will cause the network traffic to deviate from its normal state.

Application to wormhole detection. The wormhole attack is collaborative attack on the routing layer. During a wormhole attack, the malicious nodes perform a tunneling procedure to form a wormhole where one node receives packets and covertly tunnels them to another colluding node, and then the colluding node re-

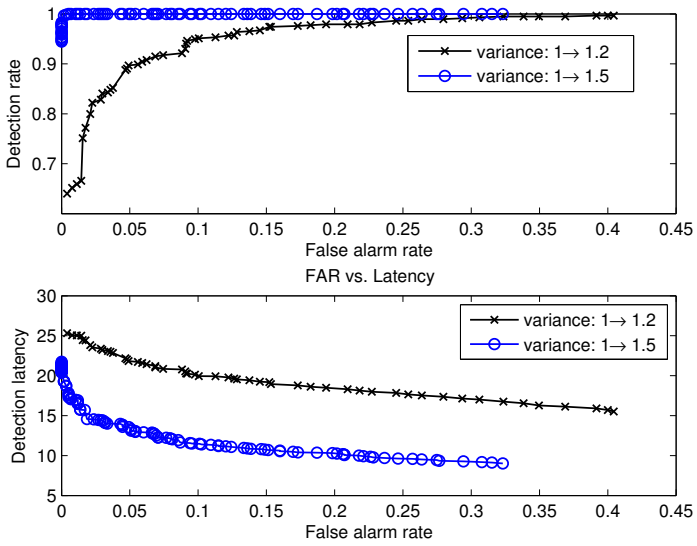


Figure 6.9: Detection of changes on data variance

plays these packets as if it received them from its physical neighbors. Based on different covert communication mechanisms used for tunneling, wormhole attacks can be classified as in-band wormholes and out-band wormholes. The in-band wormhole connects the purported neighbors via multi-hop tunnels over the existing wireless medium while the out-band wormhole attack uses an external communication medium such as a wired link or a long-range wireless transmission channel. Wormhole attacks can affect shortest path routing calculations and allow the attacking nodes to attract and route traffic from other parts of the network to go through them, thus create artificial traffic choke points that can be utilized at an opportune future time to either analyze network traffic or to degrade network performance.

We evaluate the case of in-band wormholes. In-band wormholes are important for several reasons. First, because they do not require additional specialized hardware, they can be launched from any node in the network; as a result, they may

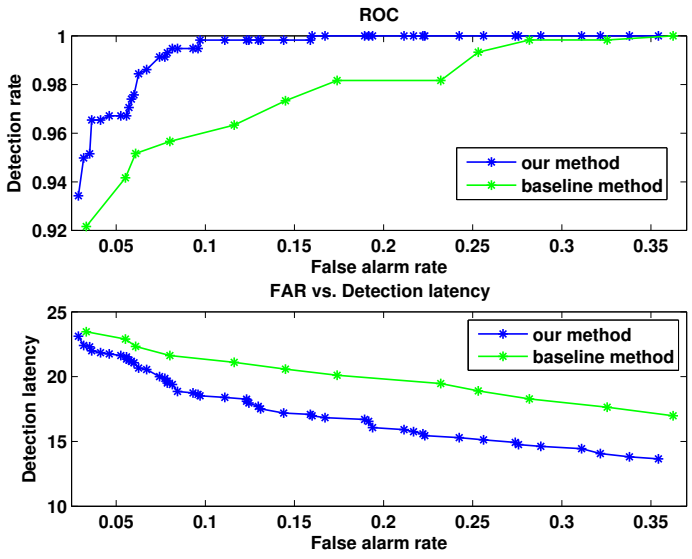


Figure 6.10: Comparison with the baseline method

be more likely to be used by real adversaries. Second, unlike out-band wormholes, which actually add channel capacity to the network, in-band wormholes continuously consume network capacity (i.e., waste bandwidth) thereby inherently causing service degradation. Fig. 6.11 shows an example of a 3-hop in-band wormhole. By 'n-hop wormhole', we mean that the actual path length between the two wormhole endpoints is n-hop but the routing protocol is fooled to consider it as only 1 hop.

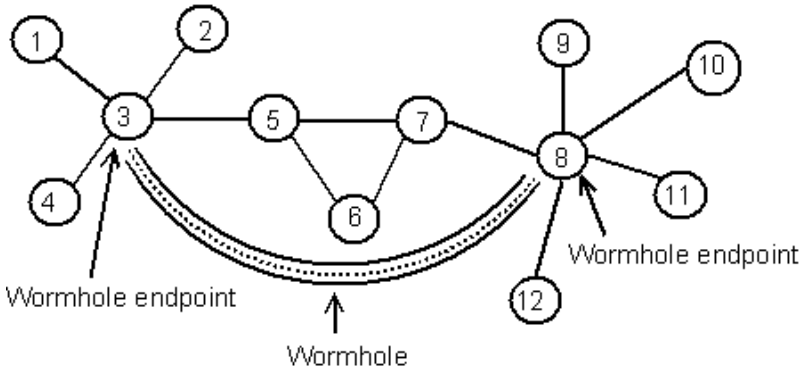


Figure 6.11: An in-band wormhole

In the in-band wormhole attack, the multi-hop tunneling process can cause

the transmission delay along a path to deviate from its normal state. For example, in Fig. 6.11 the attacker fools the routing protocol to treat the 3-hop path 3-5-7-8 as a 1-hop path 3-8. The transmission delay of a 3-hop path, however, would be different from a real 1-hop path due to different path lengths, not to mention that the wormhole attack can introduce additional congestion in the path due to the attraction of traffic from other parts of the network. The difficulty of detection lies in the fact that traffic variability such as normal network congestion may lead to high false alarm rates.

We create the in-band wormholes in networks containing 50 nodes in a 1000×1000 square field using NS-2. Different simulation scenarios are considered, including networks that have different levels of background traffic and wormholes that have different length.

Fig. 6.12 shows three typical scenarios of the collected packet round trip time between a source-destination pair that was attracted by a 4-hop wormhole, where the wormhole starts at time around 2500. The top figure (scenario 1) corresponds to the case when the background traffic is relatively light, so there is no congestion in the wormhole or other places of the network. The packet round trip time becomes slightly higher after the wormhole attack starts. The middle one (scenario 2) shows the case when the background traffic becomes heavier. Since more traffic was attracted by the wormhole, leading to some level of congestion inside the wormhole, traffic going through the wormhole have much longer round trip time than its previous normal state. This case is the easiest case for wormhole detection. In the bottom figure (scenario 3) the background traffic becomes much heavier, in

which case the network congestion causes large traffic variation even when there is no wormhole attack. It is the most difficult case for wormhole detection.

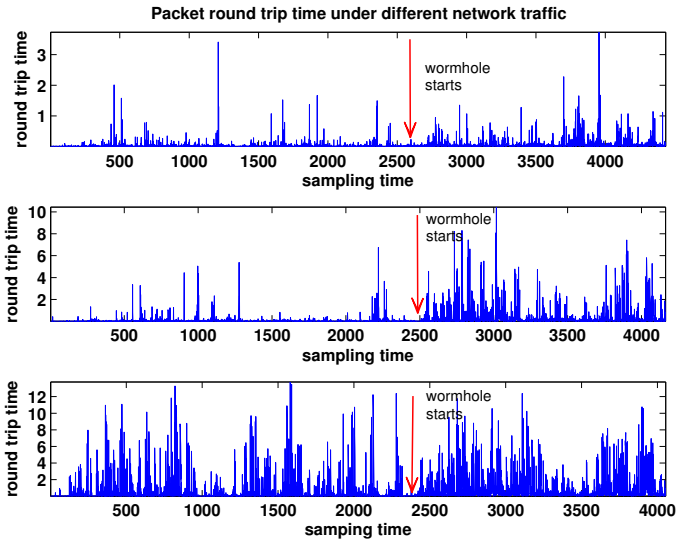


Figure 6.12: Different traffic scenarios under wormhole attack

Fig. 6.13 presents the ROC curves for detection of the 4-hop wormhole under different levels of background traffic corresponding to the three described scenarios. The results are obtained over 100 simulation runs. In the worst case, i.e., scenario 3, our algorithm performs much better than a random guess, e.g., when the false alarm rate reaches 0.4, the detection rate is around 0.8. The results are satisfactory considering that the end-to-end packet round trip time is the only traffic profile we used for detection. We expect that if our detector is combined with methods based on other characteristics of the network traffic, better detection performance can be achieved. Fig. 6.14 shows performance comparison of our method with the baseline method for scenario 3, in which our method achieves better performance.

We also create wormholes with different lengths. Fig. 6.15 shows the detection

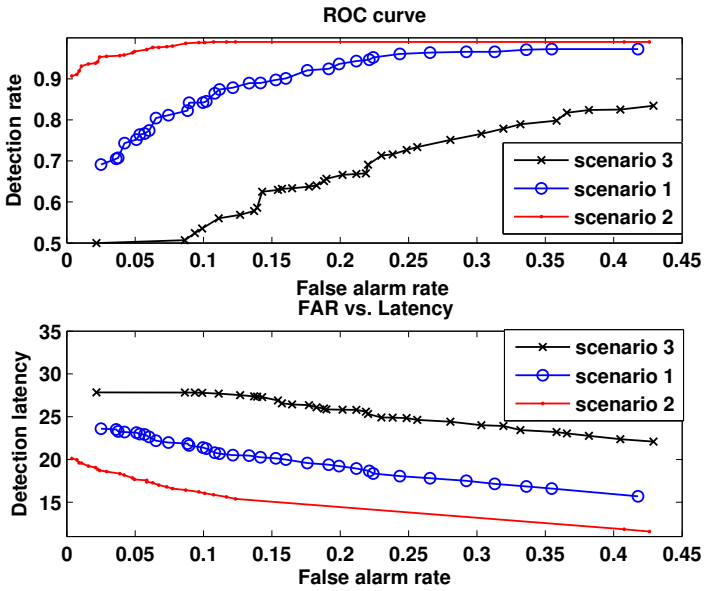


Figure 6.13: Detection of changes on data variance

performance of a 4-hop wormhole and a 6-hop wormhole, which are averaged over 100 simulation traces. For illustration purpose, we only show here the case corresponding to scenario 3, which is the most difficult detection scenario. As is expected, the detection of a longer wormhole is easier than that of a shorter one. Therefore, although a longer wormhole can attract more traffic to go through it, thus possibly cause more damages, it suffers a higher risk to be detected.

6.5 Discussion

In this chapter, we studied the anomaly detection problem in wireless sensor networks. As discovered by recent works, the traffic in wireless sensor networks can have the similar long range dependency (LRD) property as for the wireline and wireless 802.11 network. The existence of LRD could significantly increase the difficulty of network anomaly detection. To reduce the effect of LRD on anomaly

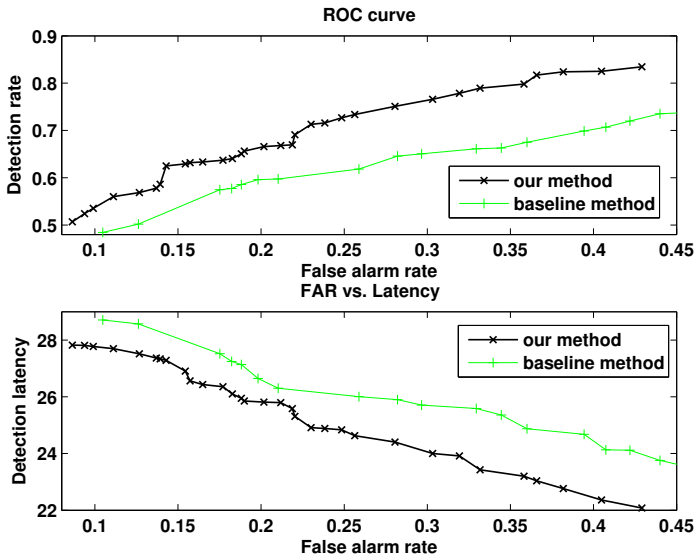


Figure 6.14: Comparison with the baseline method

detection performance, we proposed a wavelet-domain hidden Markov model for capturing the normal network traffic. The wavelet transform is able to turn the long range dependency that exists among the sample data into a short memory structure among its wavelet coefficients. The HMM in the wavelet-domain is used to capture the remaining dependency among the wavelet coefficients, thus model the traffic variability more accurately. Network anomalies are then detected as abrupt changes in the tracked HMM model structures. The performance of our algorithm is evaluated by extensive simulations, including numerical experiments in Matlab and network simulation studies in NS-2, which show promising results. Future work can be conducted to further optimize the wavelet domain HMM to improve detection performance.

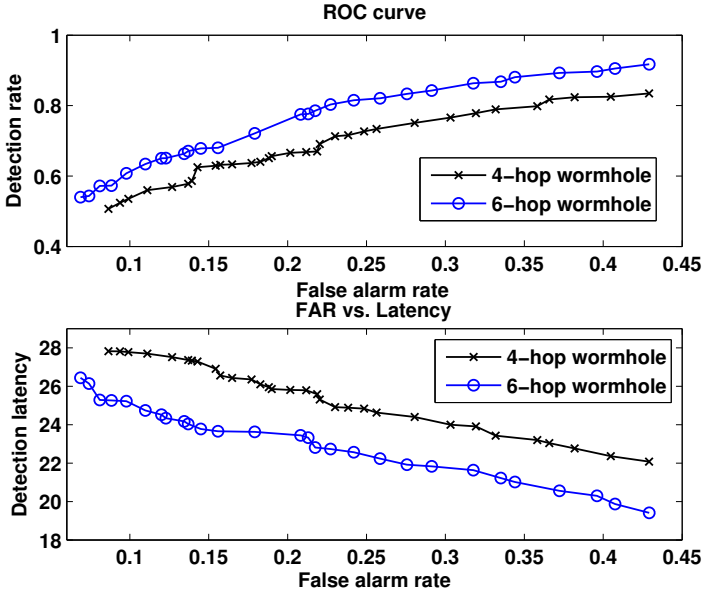


Figure 6.15: Detection of wormholes with different lengths

Part III

Analytical Tools for Security and Trust

Chapter 7

Game Theoretic Modeling

7.1 Motivation and related work

Ad hoc networks including wireless sensor networks rely on the mutual cooperation among nodes to achieve network-wide goals. However, due to resource constraints, these nodes may not be willing to cooperate as cooperation consumes their own resources. A large amount of research have been devoted to studying the collective behaviors of such selfish nodes [67, 68, 30]. By modeling the interactions among nodes as a game, the Nash equilibria are used to study the operating points of the network. It is shown that such operating points usually have inferior performance when compared to the social optimum, which is the operating point that all nodes are subject to centralized control [30]. To encourage cooperation from the self-interested nodes in the Nash equilibria, various incentive mechanisms have been proposed in the literature [70, 5, 62]. One promising technique is to establish trust relations among nodes. Most literature works focus on the design of different aspects of the trust schemes, such as trust evaluation, trust information storage and retrieval. However, few efforts have been made on quantitatively analyzing the efficiency of establishing trust for improving node cooperation. In this Chapter, we provide a game theoretic analysis as a first step towards this end.

We model the trust relations among nodes as a trust-weighted network and

study a simple game in which the user's payoff depends on the contributions from itself and its neighbors, weighted by the trust values associated with each neighbor. This game can capture the efforts of nodes to coordinate across the network and the effects of trust relations on nodes' efforts. We analyze the characteristics of the Nash equilibrium and social optimum of this game and show that they have a close relationship to the Bonacich centralities of nodes in the trust-weighted network. We then evaluate the efficiency of the game which measures the effect of establishing trust for improving node cooperation. The efficiency of the game is measured by the price of anarchy [68], which is defined to be the ratio of the payoff at the worst Nash equilibrium to that of the social optimum. It is demonstrated that the node centralities in the trust-weighted network are very important for determining the game efficiency. Moreover, motivated by the observation that nodes with heterogeneous resources have different willingness to cooperate, we exploit the potential of improving game efficiency by introducing heterogeneous resources. Critical issues on this aspect include where and how many resources to introduce. We provide both experimental and mathematical analysis and show that the resources should be introduced according to node centralities in the trust-weighted network.

Most literature works on establishing trust relations in ad hoc networks have focused on the design of the trust models. For example, Seredynski et al. [70] proposed a trust model for the routing service in ad hoc networks. Their model encourages cooperation among nodes through the incentives they receive: more cooperation gives a node higher trust which results in better service for itself. Baras and Jiang [5] proposed a model based on cooperative games, in which the nodes can

form coalitions in order to maximize their payoffs. Their scheme utilizes a feedback relation among the strategy, the payoff and the trust level. Additional works on game theoretic models on trust and cooperation can be found in [35, 46, 82]. While the solutions put forth in these references address important issues, most of them did not consider the relation between game efficiency and the structure of the trust relations among nodes. In this paper, we propose an initial analysis of this aspect.

Our work is partly motivated by the work in [19], which investigated the effect of network topology in a model of content contribution in peer-to-peer networks with linear quadratic payoffs. They identified the relation between Bonacich centralities and Nash equilibria within their game model and discussed a network-based policy for improving the equilibrium performance by exclusion of a single player. However, no trust relations are considered in their model, and the effect of network topology on game efficiency is not analyzed. In our work, we use a more general payoff function, and focus the analysis on game efficiency and trust relations among nodes.

7.2 Game model and efficiency analysis

7.2.1 Formal description of the model

To investigate the effect of establishing trust for improving node cooperation in Nash equilibria, we study a game in which the network participants can experience a marginal increase in payoff from neighbors' contributions. This type of relations among neighbors are very common in practice. We model the trust relations among nodes as a directed graph $G = \{V, E\}$ associated with a trust matrix $T = [t_{ij}]$,

where $t_{ij} \in [0,1]$ represents the trust opinion from i to j. Due to the asymmetric property of trust relations, t_{ij} may not be equal to t_{ji}. The set of i's neighbors, denoted by $\mathcal{N}_i$, is the set of users that have trust relations with i, i.e., $\mathcal{N}_i = \{j \in V | t_{ij} \neq 0\}$. The payoff of a user depends on the contributions from itself and its neighbors. Each user simultaneously chooses a contribution level $x_i \geq 0$. The effect of neighbors' contribution on the node is captured by a parameter $\delta \in (0,1]$. Since nodes are connected by their trust relations, the actual contributions received by node i are weighted by the trust values associated with each neighbor. In summary, the contribution that node i receives from its neighbors is $\delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j$. We assume that each user receives benefits according to a benefit function $b(x_i + \delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j)$. Assuming that the cost for making a unit contribution is c, then the benefit function satisfies the conditions that $b(0) = 0$, $b'(0) > c$ and $b'(+\infty) < c$, as no user will make any effort if $b'(0) < c$ and all users will make infinite efforts if $b'(+\infty) > c$. The users are rational and risk averse, so the benefit function $b(\cdot)$ is concave [25]. Denoting as usual profile of all users other than i by $\mathbf{x}_{-i}$, then the payoff for user i is

$$u_i(x_i, \mathbf{x}_{-i}) = b(x_i + \delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j) - c \cdot x_i, \tag{7.1}$$

where $x_i \in [0, +\infty)$. Actually, since $b'(0) > c$ and $b'(+\infty) < c$, there exists some $w > 0$ such that $b'(w) = c$, then the range of effort x_i becomes $[0, w]$.

This simple form of payoff can capture the efforts of nodes to coordinate across the network and the effects of trust relations on nodes' efforts. Moreover, it allows us to focus on the structure of the trust-weighted network.

7.2.2 Nash equilibria and efficiency analysis

Before we introduce the analysis of the Nash equilibria for this game, we first introduce a network centrality measure put forth by Bonacich [11]. Given a scalar α and a network with adjacency matrix A such that $(I - \alpha A)$ is invertible, the vector of *Bonacich centralities* for the nodes is defined to be $\mathbf{h}(\alpha, A) = (I - \alpha A)^{-1} A \cdot \mathbf{1}$, where $\mathbf{1}$ is the vector of all 1. Let $a_{ij}^{(k)}$ be the element in the i^{th} row and j^{th} column of the matrix A^k, the kth power of A, and let $\rho(A)$ be the spectral radius of A, i.e., the largest absolute value of A's eigenvalues , then if $|\alpha| < \rho(A)$, we have $h_i(\alpha, A) = \sum_{k=0}^{\infty} \alpha^k \sum_j a_{ij}^{(k+1)}$, where $h_i(\alpha, A)$ is the Bonacich centrality for node i. We can see that $a_{ij}^{(k)}$ actually accounts for the total weight of all paths of length k from user i to user j and α acts as a decay factor that scales down the relative weight of longer paths. If $\alpha < 0$, $h_i(\alpha, A)$ puts positive weights on node i's neighbors, but negative weights on the neighbors' neighbors, and so on. In other words, the larger weights the node's neighbors put on their other neighbors, the less central is the node.

We now analyze the Nash equilibria of the defined game. Due to the concavity of the benefit function and linearity of cost, there are no mixed equilibria for this game. More specifically, a user would always get a higher payoff by playing the average of a set of contribution levels than playing a mixed strategy over this set of contribution levels. The existence of pure Nash equilibria is guaranteed by the standard fixed point argument since the payoff function is concave and the strategy set is compact and convex [66].

At the equilibria of this game, the strategy x_i^{ne} of user i should satisfy the following conditions,

$$\begin{cases} x_i^{ne} = w - \delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j^{ne}, & \text{if } w > \delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j^{ne}, \\ x_i^{ne} = 0, & \text{if } w \leq \delta \sum_{j \in \mathcal{N}_i i} t_{ij} x_j^{ne}. \end{cases} \tag{7.2}$$

where w is the point where $b'(w) = c$. From equation (7.2) we can see that a user will make up any shortfall from its neighbors' contributions to reach w, and exerts no contribution if the weighted sum of its neighbors' contributions has reached w. For small values of δ, Theorem I in [4] implies that if $\delta < 1/(1 + \rho(U - I - T))$, where U is a matrix whose entries are all 1s and I is the identity matrix; then there exists a unique interior solution for (7.2). Denote the unique solution by $\mathbf{x}^{ne}$. Then since $\mathbf{x}^{ne}$ is interior, according to equation (7.2) we have $(I + \delta T) \cdot \mathbf{x}^{ne} = w \cdot \mathbf{1}$. Due to the existence and uniqueness of $\mathbf{x}^{ne}$, we know that $(I + \delta T)$ is invertible and $\mathbf{x}^{ne} = w \cdot (I + \delta T)^{-1} \cdot \mathbf{1} = w \cdot (\mathbf{1} - \delta \cdot \mathbf{h}(-\delta, T))$. Therefore, in the Nash equilibrium, nodes that have higher Bonacich centrality, i.e., with a higher value of $h_i(-\delta, T)$, exert less contributions. To simplify the notation, we use $\mathbf{h}$ to represent $\mathbf{h}(-\delta, T)$ in the rest of the paper.

Next we analyze the social optimal point of the game, which is the network operating point when all users are subject to centralized control and the social welfare achieves its maximum value. We define the social welfare to be the sum of

all individual payoffs, i.e.,

$$SW(\mathbf{x}) = \sum_{i=1}^{n} [b(x_i + \delta \cdot \sum_{j \in \mathcal{N}_i} t_{ij} x_j) - c \cdot x_i], \ x_i \geq 0. \tag{7.3}$$

Since $SW(\mathbf{x})$ is concave, and the inequality constraints are convex, the Kuhn-Tucker conditions are the necessary and sufficient conditions for optimality. Therefore, the social optimum $\mathbf{x}^{so}$ satisfies the following conditions,

$$\begin{cases} \text{if } x_i^{so} > 0, \ b'(x_i^{so} + \delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j^{so}) + \delta \sum_{j \neq i} t_{ji} b'(x_j^{so} + \delta \sum_{k \in \mathcal{N}_j} t_{jk} x_k^{so}) - c = 0, \\ \text{if } x_i^{so} = 0, \ b'(x_i^{so} + \delta \sum_{j \in \mathcal{N}_i} t_{ij} x_j^{so}) + \delta \sum_{j \neq i} t_{ji} b'(x_j^{so} + \delta \sum_{k \in \mathcal{N}_j} t_{jk} x_k^{so}) - c \leq 0. \end{cases}$$

The social optimum $\mathbf{x}^{so}$ has the following property.

Lemma 7.2.1. *Let $\boldsymbol{v} = [v_1, \ldots, v_n]$ be the vector that satisfies*

$$[b'(v_1), \ldots, b'(v_n)]^T = c \cdot (I + \delta \cdot T)^{-1} \cdot \boldsymbol{1} = c \cdot (\boldsymbol{1} - \delta \cdot \boldsymbol{h}),$$

and define $\boldsymbol{x}^* = (I + \delta T)^{-1} \cdot \boldsymbol{v}$. If $\boldsymbol{x}^*$ is non-negative, we have $\boldsymbol{x}^{so} = \boldsymbol{x}^*$. Otherwise, $\boldsymbol{x}^{so}$ is different from $\boldsymbol{x}^*$ but we have $SW(\boldsymbol{x}^{so}) < SW(\boldsymbol{x}^*)$.

Proof. By simple manipulation of liner algebra, we know that $\mathbf{x}^*$ is the solution for the following problem

$$\max_{\mathbf{x}} \sum_{i=1}^{n} [b(x_i + \delta \sum_{j \neq i} t_{ij} x_j) - c \cdot x_i], \tag{7.4}$$

which is only different from the maximization problem in (7.3) by dropping the non-

negative constraints on x_i's; so if $\mathbf{x}^*$ is non-negative, it is also the optimal solution for (7.3). Otherwise, $\mathbf{x}^*$ is not the optimal solution for (7.3), but the maximal objective value of (7.4) is never smaller than that of (7.3), so we have $SW(\mathbf{x}^{so}) < SW(\mathbf{x}^*)$ $\square$

From the above analysis, we can see that both the Nash equilibrium and the social optimum of this game is closely related to the Bonacich centralities of the nodes in the trust-weighted network. To describe the difference between the maximum social welfare $SW(\mathbf{x}^{so})$ and the social welfare at the Nash equilibrium $SW(\mathbf{x}^{ne})$, we compute the *Price of Anarchy* (PoA) for this game. The PoA measures the price that the users will pay if they play selfishly in a decentralized manner rather than play according to a centralized control. It is defined as the ratio of the social optimal welfare to the welfare of the worst Nash equilibrium. Since the Nash equilibrium of this game is unique when $\delta < 1/(1+\rho(U-I-T))$, and we have $SW(\mathbf{x}^{so}) \leq SW(\mathbf{x}^*)$, the PoA of this game can be upper bounded by

$$PoA \leq \frac{SW(\mathbf{x}^*)}{SW(\mathbf{x}^{ne})} = \frac{\sum_{i=1}^{n}[b(x_i^* + \delta \sum_{j\in\mathcal{N}_i} t_{ij}x_j^*) - c\cdot x_i^*]}{\sum_{i=1}^{n}[b(x_i^{ne} + \delta \sum_{j\in\mathcal{N}_i} t_{ij}x_j^{ne}) - c\cdot x_i^{ne}]}.$$

Lemma 7.2.2. *The PoA of this game can be upper bounded by* $PoA \leq \sum_i h_i \cdot v_i / \sum_i h_i \cdot w$.

Proof. Since $b(\cdot)$ is a concave function, $b^{'}(w) = c$, $(I + \delta \cdot T)\cdot \mathbf{x}^* = \mathbf{v}$ and $(I + \delta \cdot T)\cdot \mathbf{x}^{ne} = \mathbf{w}$, we have

$$\begin{aligned} & b(x_i^* + \delta \sum_{j\in\mathcal{N}_i} t_{ij}x_j^*) - b(x_i^{ne} + \delta \sum_{j\in\mathcal{N}_i} t_{ij}x_j^{ne}) \\ = \;& b(v_i) - b(w) \leq (v_i - w)\cdot b'(w) = (v_i - w)\cdot c. \end{aligned}$$

Let $\mathbf{c} = c \cdot \mathbf{1}$ and $\mathbf{w} = w \cdot \mathbf{1}$, then the PoA can be written as,

$$\begin{aligned}
PoA &\leq \frac{\sum_{i=1}^{n}[b(w) + c \cdot (v_i - w) - c \cdot x_i^*]}{\sum_{i=1}^{n}[b(w) - c \cdot x_i^{ne}]} \\
&= 1 + \frac{\sum_{i=1}^{n}[c \cdot (v_i - w) - c \cdot (x_i^* - x_i^{ne})]}{\sum_{i=1}^{n}[b(w) - c \cdot x_i^{ne}]} \\
&< 1 + \frac{\mathbf{c}^T(\mathbf{v} - \mathbf{w}) - \mathbf{c}^T(I + \delta T)^{-1}(\mathbf{v} - \mathbf{w})}{n \cdot w \cdot b'(w) - \mathbf{c}^T(I + \delta T)^{-1}\mathbf{w}} \\
&= 1 + \frac{\delta \mathbf{c}^T(I - (I + \delta T)^{-1})(\mathbf{v} - \mathbf{w})}{\mathbf{c}^T(I - (I + \delta T)^{-1})\mathbf{w}} \\
&= 1 + \frac{\delta \mathbf{c}^T T(I + \delta T)^{-1}(\mathbf{v} - \mathbf{w})}{\delta \mathbf{c}^T T(I + \delta T)^{-1}\mathbf{w}} \\
&= \frac{\mathbf{1}^T \cdot T(I + \delta T)^{-1}\mathbf{v}}{\mathbf{1}^T \cdot T(I + \delta T)^{-1}\mathbf{w}} = \frac{\sum_{i=1}^{n} h_i \cdot v_i}{\sum_{i=1}^{n} h_i \cdot w}
\end{aligned} \tag{7.5}$$

□

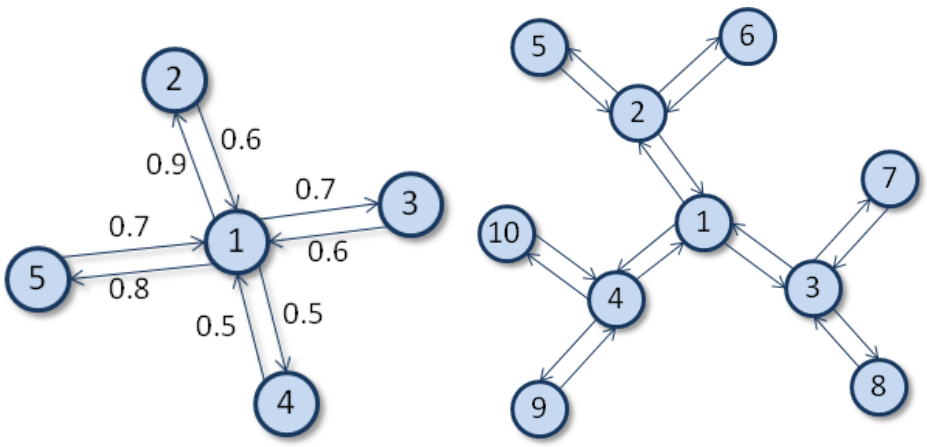
Lemma 7.2.2 shows that the upper bound of the PoA depends on the Bonacich centralities of nodes in the trust-weighted network. Since w is always smaller than v_i and a node with higher Bonacich centrality has a higher value of v_i, the larger the variance of the Bonacich centralities, the lower the upper bound of the game efficiency. An intuitive explanation for the low efficiency is that although the contributions from users with higher Bonacich centrality would benefit more users compared to the contributions from users with lower centrality, those users tend to exert less contribution in the Nash equilibrium because they can rely more on their neighbors.

7.2.3 Heterogeneity and efficiency improvement

To improve game efficiency, one way is to increase the contributions from the nodes that have higher Bonacich centralities. Observing that nodes who have more resources will have higher willingness to contribute, we propose to introduce more resources to those nodes that have higher Bonacich centralities, so they may contribute more at the Nash equilibrium.

We model the different resources in nodes by a discount factor on cost, i.e., after introducing more resources, the cost for user i to make contribution x_i is now $(1-\gamma_i)\cdot c\cdot x_i$, where $\gamma_i \in [0,1)$ depends on the amount of resources introduced. If a node has more resources, the cost for it to make the same level of contributions will be smaller than nodes with fewer resources. There are two important issues that need to be addressed, i.e., the amount and the position of heterogeneous resources to be allocated. We first study two example networks in Figure 7.1(a) and 7.1(b), and then provide mathematical analysis for these two issues.

The links in Figure 7.1(a) and 7.1(b) represent the trust relations among nodes. In Figure 7.1(a) trust values are displayed with the links. In Figure 7.1(b) we assume all the trust values are 1, in order to focus on the underlying structure of the network. We use the benefit function $b(x) = -x^2 + 2x, x \in [0,1)$ and let the cost coefficient be $c = 1$. The benefit function is chosen to satisfy the concavity constraint. The parameter δ is set to 0.2 for Figure 7.1(a) and 0.1 for 7.1(b), in order to satisfy the constraint $\delta < 1/(1+\rho(U-I-T))$.



(a) Network I (b) Network II

Figure 7.1: Two Example Networks

For Figure 7.1(a), the Bonacich centrality measurement for the nodes is

$$\mathbf{h} = [2.7400, 0.2712, 0.2712, 0.2260, 0.3164]^T.$$

The Nash equilibrium of the game is achieved at

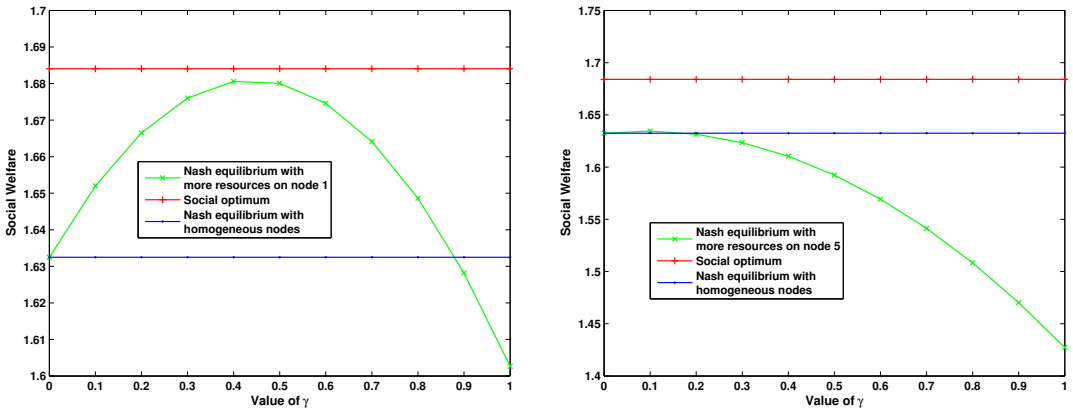
$$\mathbf{x}^{ne} = [0.2260, 0.4729, 0.4729, 0.4774, 0.4684]^T.$$

As expected, the center node, which has much higher Bonacich centrality than others, contributes much less in the Nash equilibrium. However, the social optimum point will be

$$\mathbf{x}^{so} = [0.5037, 0.4667, 0.4667, 0.4722, 0.4611]^T,$$

in which the center node makes the highest contributions. To improve game efficiency, we introduce heterogeneous resources for the nodes. Figure 7.2(a) and Figure 7.2(b) illustrate the social welfare at the various Nash equilibria when introducing

more resources to nodes 1 and 5. Obviously, the social welfare can be improved when node 1 is supplied by an appropriate level of additional resources. But introducing more resources to node 5 actually reduces the social welfare. Therefore, we should design carefully where to introduce the heterogeneous resources. Also, it is observed in Figure 7.2(a) that the social welfare does not monotonically increase with the increasing of resources at the center node, which poses another challenge for the design, i.e., to determine the optimal amount of additional resources.



(a) Introducing more resources to node 1 (b) Introducing more resources to node 5

Figure 7.2: Efficiency Improvement for Network I

For the network in Figure 7.1(b), although node 1 appears to be the most 'central' in the network, it actually has lower Bonacich centrality than nodes 2, 3 and 4. This is because Bonacich centrality puts negative weights on even length paths. The actual Bonacich centrality is 2.2105 for node 1, 2.6316 for nodes 2, 3 and 4, and 0.7368 for nodes 5, 6, 7, 8, 9, 10. Figure 7.3 shows the social welfare improvement by introducing more resources to node 1 (NE1), node 2 (NE2) and node 1, 2, 3, 4 together (NE3). We observe that the game efficiency is not improved much by introducing more resources for node 1 or node 2 only, but is significantly

improved by introducing more resources for node 1, 2, 3 and 4 together. That is, not only where the additional resources should be introduced but also how many nodes should be allocated with additional resources need to be carefully designed.

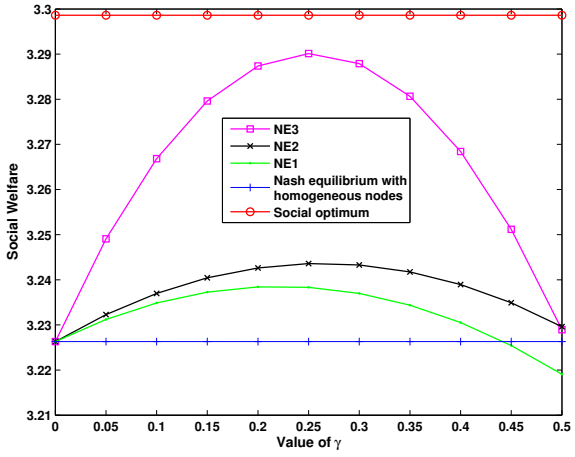


Figure 7.3: Efficiency Improvement for Network II

Next we mathematically analyze the possible improvement of game efficiency by introducing heterogeneous resources. Let $\tilde{\mathbf{c}} = (1 - \boldsymbol{\gamma}) \cdot c$ be the cost for users to make a unit effort after we introduce additional resources to them, where $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]$ is the vector of cost discount factors for users. Assuming $\tilde{\mathbf{w}}$ is the vector that satisfies $[b'(\tilde{w}_1), \ldots, b'(\tilde{w}_n)]^T = \tilde{\mathbf{c}}$, note that due to the concavity of $b(\cdot)$, the value of $\tilde{w}_i$ increases with the increase of γ_i , the strategy for user i at the new Nash Equilibrium, denoted by $\tilde{\mathbf{x}}^{ne}$, becomes

$$\begin{cases} \tilde{x}_i^{ne} = \tilde{w}_i - \delta \sum_{j \in \mathcal{N}_i i} t_{ij} \tilde{x_j}^{ne} & \text{if } \delta \sum_{j \in \mathcal{N}_i} t_{ij} \tilde{x_j}^{ne} < \tilde{w}_i, \\ \tilde{x}_i^{ne} = 0 & \text{if } \delta \sum_{j \in \mathcal{N}_i} t_{ij} \tilde{x_j}^{ne} \geq \tilde{w}_i. \end{cases} \tag{7.6}$$

Since equation (7.6) is also the Kuhn-Tucker condition for the following maximiza-

tion problem

$$\max_{\mathbf{x}} \ \mathbf{x}^T \cdot \tilde{\mathbf{w}} - \frac{1}{2}\mathbf{x}^T(I + \delta T)\mathbf{x}, \forall\ i,\ x_i \geq 0, \tag{7.7}$$

and $(I + \delta T)$ is invertible, there is a unique solution to problem (7.7) and thus a unique Nash equilibrium for the new game. Furthermore, if $(I + \delta T)^{-1}\tilde{\mathbf{w}}$ is non-negative, we have $\tilde{\mathbf{x}}^{ne} = (I + \delta T)^{-1}\tilde{\mathbf{w}}$. Otherwise, the Nash equilibrium will have some i with $\tilde{x}_i^{ne} = 0$. The non-negativeness of $(I + \delta T)^{-1}\tilde{\mathbf{w}}$ depends on the choice of $\tilde{\mathbf{c}}$ and the benefit function $b(\cdot)$.

Lemma 7.2.3. *When $(I+\delta T)^{-1} \cdot \tilde{\boldsymbol{w}}$ is non-negative, thus being the Nash equilibrium for the new game, the improvement of social welfare at the Nash equilibrium can be bounded by*

$$SW(\tilde{\boldsymbol{x}}^{ne}) - SW(\boldsymbol{x}^{ne}) \geq \sum_{\{i:\gamma_i>0\}} (\tilde{w}_i - w)(\delta \cdot h_i - \gamma_i) \cdot c \tag{7.8}$$

$$SW(\tilde{\boldsymbol{x}}^{ne}) - SW(\boldsymbol{x}^{ne}) \leq \sum_{\{i:\gamma_i>0\}} \delta \cdot c \cdot (\tilde{w}_i - w) \cdot h_i, \tag{7.9}$$

and the improvement on the PoA value can be bounded by

$$\frac{SW(\tilde{\boldsymbol{x}}^{ne})}{SW(\boldsymbol{x}^{ne})} \leq \frac{\sum_i \tilde{w}_i h_i}{\sum_i w h_i}. \tag{7.10}$$

Proof. Since $b(\cdot)$ is concave, $(I + \delta T) \cdot \tilde{\mathbf{x}}^{ne} = \tilde{\mathbf{w}}$, and $(I + \delta T) \cdot \mathbf{x}^{ne} = \mathbf{w}$, the improvement of social welfare at the equilibrium by introducing heterogeneity can be written as

$$
\begin{aligned}
S\tilde{W}^{ne} - SW^{ne} &= \sum_{i=1}^{n}[b(\tilde{w}_i) - b(w) + c(x_i^{ne} - \tilde{x}^{ne})] \\
S\tilde{W}^{ne} - SW^{ne} &\geq \tilde{\mathbf{c}}^T(\tilde{\mathbf{w}} - \mathbf{w}) + \mathbf{c}^T(\mathbf{x}^{ne} - \tilde{\mathbf{x}}^{ne}) \\
&= (\tilde{\mathbf{w}} - \mathbf{w})^T(\tilde{\mathbf{c}} - (I + \delta T)^{-1}\mathbf{c}) \\
&= \sum_{i=1}^{n}(\tilde{w}_i - w)(\delta \cdot h_i - \gamma_i) \cdot c, \\
S\tilde{W}^{ne} - SW^{ne} &\leq \mathbf{c}^T(\tilde{\mathbf{w}} - \mathbf{w}) + \mathbf{c}^T(\mathbf{x}^{ne} - \tilde{\mathbf{x}}^{ne}) \\
&= (\tilde{\mathbf{w}} - \mathbf{w})^T(\delta T(I + \delta T)^{-1}\mathbf{c}) \\
&= \sum_{i=1}^{n} \delta \cdot c \cdot (\tilde{w}_i - w) \cdot h_i.
\end{aligned}
$$

Following a similar procedure as in the proof of Lemma 7.2.2, we have

$$
\frac{SW(\tilde{\mathbf{x}}^{ne})}{SW(\mathbf{x}^{ne})} \leq \frac{\delta\tilde{\mathbf{w}}^T T(I + \delta T)^{-1} \cdot \mathbf{c}}{\delta \mathbf{w}^T T(I + \delta T)^{-1}\mathbf{w}} = \frac{\sum_i \tilde{w}_i h_i}{\sum_i w h_i}.
$$

□

The upper bound on the social welfare improvement in (7.9) is monotonically increasing with the increase of cost discount factor. But from the lower bound in equation (7.8), we can see that increasing the cost discount factor to the nodes that have higher Bonacich centralities could increase the value of the term $(\tilde{w}_i - w)$ but decrease the value of the term $(\delta \cdot h_i - \gamma_i)$, so the social welfare improvement may not monotonically increase, which is consistent with the observation in Figure 7.2(a). Also, by equation (7.10) we can expect that the higher the variance of the Bonacich centralities of nodes in the trust-weighted network, the more room we have to improve game efficiency by introducing heterogeneous resources.

When the non-negativeness of $(I + \delta T)^{-1}\tilde{\mathbf{w}}$ is not satisfied, the Nash equilibrium of the new game will have some i such that $\tilde{x}_i^{ne} = 0$, which is a much more involved case.

7.3 Discussion

We studied the effects of trust relations among users on the equilibria in network games. We considered a game in which each user locally interacts with its neighbors, where the local interactions are affected by their trust relations. We characterized the unique Nash equilibrium in this game using Bonacich centrality measures, and proved that the game efficiency, measured in terms of the price of anarchy, is also highly related to this centrality measure. The more spread the Bonacich centralities among nodes, the lower the game efficiency. Motivated by the observation that users that have more resources will have higher willingness to contribute, we also proposed to improve game efficiency by introducing heterogeneous resources according to nodes' centralities.

In future work, this static game can be extended to repeated rounds to capture the evolution of trust and cooperation among nodes. The potential of heterogeneity on improving Nash equilibrium quality can be further explored when the trust relations change dynamically.

Appendix A

Proofs for theorems in Chapter 5

A.1 Proof for performance bound for the approximation algorithm in first phase probing

Proof. The primal objective value z^p can be written as $z^p = \sum_{j \in \mathcal{E}(\mathcal{X})} w_j$ according to (5.1), which is lower bounded by $\frac{1}{\delta} \sum_{i \in \mathcal{X}} \tilde{w}_i$. Since the paths in $\mathcal{X}$ have active constraints in the dual problem, we have $[\sum_{i \in \mathcal{X}} \tilde{w}_i = \sum_{i \in \mathcal{X}} (\lambda h_i + \sum_j a_{ij} \gamma_j) < \delta \cdot z^p$. On the other hand, the dual objective value z^d can be written as $z^d = \lambda h_0 + \sum_{j \in \mathcal{E}} \gamma_j$. Since $\gamma_j = 0$ for $j \notin \mathcal{E}(\mathcal{X})$, we have

$$z^d = \lambda(h_0 - \sum_{i \in \mathcal{X}} h_i) + \sum_{i \in \mathcal{X}} (\lambda h_i + \sum_j a_{ij} \gamma_j) < \lambda h_{max} + \delta \cdot z^p.$$

Since $\lambda \sum_{i \in \mathcal{X}} h_i < \delta \cdot z^p$ and $|\mathcal{X}| = l_1$, we have

$$\lambda h_{max} < \frac{\delta \cdot z^p}{l_1 \cdot h_{min}} h_{max}.$$

Then z^d can be upper bounded by $z^d < \delta \cdot z^p \cdot (1 + r/l_1)$ and we have the optimal objective value z^* satisfying

$$z^* < z^d < \delta(1 + r/l_1) z^p.$$

□

A.2 Proof for performance bound on the greedy algorithm in second phase probing

Proof. First, the diagnosis quality $f(\pi)$, measured in the reduction of estimation uncertainty, satisfies the adaptive monotonicity property defined in [26], i.e. getting more probing observations can never decrease the diagnosis quality. Second, it also satisfies the adaptive submodularity property, i.e., when there are fewer probes that have been sent, probing one additional path can provide the same or more information than the case of probing the same additional path but there are more probes that have been sent. In other words, for $\mathcal{X}_A \subseteq \mathcal{X}_B \subset \mathcal{X}$ and $X_i \in \mathcal{X} \backslash \mathcal{X}_B$, we have

$$\begin{aligned} & f(\mathcal{X}_A, X_i) - f(\mathcal{X}_A) = H(\mathcal{Y}|\mathcal{X}_A) - H(\mathcal{Y}|\mathcal{X}_A, X_i) \\ & \geq f(\mathcal{X}_B, X_i) - f(\mathcal{X}_B) = H(\mathcal{Y}|\mathcal{X}_B) - H(\mathcal{Y}|\mathcal{X}_B, X_i), \end{aligned}$$

which is due to the fact that

$$\begin{aligned} & H(\mathcal{Y}|\mathcal{X}_A) - H(\mathcal{Y}|\mathcal{X}_A, X_i) - (H(\mathcal{Y}|\mathcal{X}_B) - H(\mathcal{Y}|\mathcal{X}_B, X_i)) \\ & = \sum_{\mathcal{X}_B} \sum_{X_i} P(\mathcal{X}_B, X_i) \log \frac{P(\mathcal{X}_B, X_i)}{P(\mathcal{X}_B) P(\mathcal{X}_i|\mathcal{X}_A)} \\ & = D_{KL}(P(\mathcal{X}_B, X_i) || P(\mathcal{X}_B) P(\mathcal{X}_i|\mathcal{X}_A)) \geq 0, \end{aligned}$$

where $D_{KL}(\cdot)$ represents the Kullback-Leibler divergence. Following Theorem 3 in [26], we have $\tilde{\Delta} \geq (1 - e^{-l_2/k})\Delta^*$, where l_2 is the number of steps for the greedy algorithm and k is that of the optimal algorithm. In our case, the number of steps in the greedy algorithm is equal to the number of probes selected by the greedy algorithm, and the number of steps for an optimal algorithm can be upper bounded by $k \leq \tilde{h}_0/h_{min}$, so we have $\tilde{\Delta} \geq (1 - e^{-l_2 \cdot h_{min}/\tilde{h}_0})\Delta^*$. □

Appendix B

Proofs for theorems in Chapter 6

B.1 Proof for the correctness of the forward decomposition

To prove the correctness of the forward decomposition algorithm, we need to prove the correctness of the following results for $h \geq 2$:

$$\alpha_{h-1,2j-1} = g_1(d_{h-1,2j-1}) \sum_{s_{h-1,2j-2}} [g_2(s_{h-1,2j-1}) \cdot \alpha_{h,j}], \tag{B.1}$$

$$\alpha_{h-1,2j} = g_1(d_{h-1,2j}) \sum_{s_{h-1,2j-1}} [g_2(s_{h-1,2j}) \cdot \alpha_{1,2^{h-2}(2j-1)}]. \tag{B.2}$$

By the definition of $\mathcal{S}_{i,j}$ and $\mathcal{D}_{i,j}$, we have

$$\begin{aligned}
\mathcal{S}_{h-1,2j-1} &= \{\mathcal{S}_{h,j} \backslash s_{h,j}\} \cup s_{h-1,2j-1}, \\
\mathcal{D}_{h-1,2j-1} &= \{\mathcal{D}_{h,j} \backslash d_{h,j}\} \cup d_{h-1,2j-1}, \\
\mathcal{S}_{h-1,2j} &= \{\mathcal{S}_{1,2^{h-2}(2j-1)} \backslash s_{1,2^{h-2}(2j-1)}\} \cup s_{h-1,2j}, \\
\mathcal{D}_{h-1,2j} &= \{\mathcal{D}_{1,2^{h-2}(2j-1)} \backslash d_{1,2^{h-2}(2j-1)}\} \cup d_{h-1,2j}.
\end{aligned}$$

Then it is trivial to prove the correctness of equation (B.1),(B.2) according to the Markovian property of the proposed HMM model.

B.2 Proof for the correctness of the scaled backward decomposition

To prove the correctness of the scaled backward decomposition algorithm, we need to prove the following results for $h \geq 2$:

$$\bar{\beta}_{1,2^{h-2}(2j-1)} = \sum_{s_{h-1,2j}} \frac{g_1(d_{h-1,2j}) g_2(s_{h-1,2j}) \cdot \bar{\beta}_{h-1,2j}}{c_{h-1,2j}}, \tag{B.3}$$

$$\bar{\beta}_{h,j} = \sum_{s_{h-1,2j-1}} \frac{g_1(d_{h-1,2j-1}) g_2(s_{h-1,2j-1}) \cdot \bar{\beta}_{h-1,2j-1}}{c_{h-1,2j-1}}. \tag{B.4}$$

Note that $\bar{\beta}_{i,j} = \frac{P(\mathcal{D}^c_{i,j}|\mathcal{S}_{i,j})}{P(\mathcal{D}^c_{i,j}|\mathcal{D}_{i,j})}$ and $c_{i,j} = P(d_{i,j}|\mathcal{D}_{i,j}\backslash d_{i,j})$, and by the definition of $\mathcal{S}_{i,j}$ and $\mathcal{D}_{i,j}$, we have

$$\begin{aligned}
\mathcal{S}_{1,2^{h-2}(2j-1)} &= \mathcal{S}_{h-1,2j}\backslash s_{h-1,2j}, \\
\mathcal{D}_{1,2^{h-2}(2j-1)} &= \mathcal{D}_{h-1,2j}\backslash d_{h-1,2j}, \\
\mathcal{D}^c_{1,2^{h-2}(2j-1)} &= \mathcal{D}^c_{h-1,2j} \cup d_{h-1,2j}, \\
\mathcal{S}_{h,j} &= \mathcal{S}_{h-1,2j-1}\backslash s_{h-1,2j-1}, \\
\mathcal{D}_{h,j} &= \mathcal{D}_{h-1,2j-1}\backslash d_{h-1,2j-1}, \\
\mathcal{D}^c_{h,j} &= \mathcal{D}_{h-1,2j-1} \cup d_{h-1,2j-1}.
\end{aligned}$$

Therefore, according to the Markovian property of the proposed HMM model, we have for the right side of equation (B.3) that

$$\begin{aligned}
&\frac{g_1(d_{h-1,2j})g_2(s_{h-1,2j}) \cdot \bar{\beta}_{h-1,2j}}{c_{h-1,2j}} \\
&= \frac{P(s_{h-1,2j}, \mathcal{D}^c_{h-1,2j} \cup d_{h-1,2j}|\mathcal{S}_{h-1,2j}\backslash s_{h-1,2j})}{P(\mathcal{D}^c_{h-1,2j} \cup d_{h-1,2j}|\mathcal{D}_{h-1,2j}\backslash d_{h-1,2j})} \\
&= \frac{P(s_{h-1,2j}, \mathcal{D}^c_{1,2^{h-2}(2j-1)}|\mathcal{S}_{1,2^{h-2}(2j-1)})}{P(\mathcal{D}^c_{1,2^{h-2}(2j-1)}|\mathcal{D}_{1,2^{h-2}(2j-1)})},
\end{aligned}$$

which proves the equality in equation (B.3). The proof for equation (B.4) can be derived similarly.

B.3 Proof for the computation of the posterior probabilities

We need to prove the following equations:

$$\gamma^{i,j}_{k_1,k_2,k_3} = \begin{cases} \sum \bar{\alpha}_{1,2^{i-1}(j-1)} \bar{\beta}_{i,j} \cdot \frac{g_1(d_{i,j}) \cdot g_2(s_{i,j})}{c_{i,j}}, \text{for j even,} \\ \sum \bar{\alpha}_{i+1,\lceil j/2 \rceil} \bar{\beta}_{i,j} \cdot \frac{g_1(d_{i,j}) \cdot g_2(s_{i,j})}{c_{i,j}}, \text{for j odd.} \end{cases}$$

$$\gamma^{L,j}_{k_1,k_2} = \sum \bar{\alpha}_{1,2^{L-1}(j-1)} \cdot \bar{\beta}_{L,j} \cdot \frac{g_1(d_{L,j}) g_2(s_{L,j})}{c_{L,j}}.$$

We will only prove here the case when j is even for $\gamma^{i,j}_{k_1,k_2,k_3}$. The proof for the other two equations can be derived similarly.

First note that when j is even, we have

$$\begin{aligned} \mathcal{S}_{1,2^{i-1}(j-1)} &= \mathcal{S}_{i,j} \backslash s_{i,j}, \\ \mathcal{D}_{1,2^{i-1}(j-1)} &= \mathcal{D}_{i,j} \backslash d_{i,j}, \\ \mathcal{D}^c_{1,2^{i-1}(j-1)} &= \mathcal{D}^c_{i,j} \cup d_{i,j}. \end{aligned}$$

Then we can know that

$$\begin{aligned} & \bar{\alpha}_{1,2^{i-1}(j-1)} \bar{\beta}_{i,j} \cdot \frac{g_1(d_{i,j}) \cdot g_2(s_{i,j})}{c_{i,j}} \\ &= \frac{P(\mathcal{S}_{1,2^{i-1}(j-1)} | \mathcal{D}_{1,2^{i-1}(j-1)}) P(\mathcal{D}^c_{1,2^{i-1}(j-1)}, s_{i,j} | \mathcal{S}_{1,2^{i-1}(j-1)})}{P(\mathcal{D}^c_{1,2^{i-1}(j-1)} | \mathcal{D}_{1,2^{i-1}(j-1)})} \\ &= P(s_{i,j}, \mathcal{S}_{1,2^{i-1}(j-1)} | \mathcal{D}), \end{aligned}$$

from which the value of $\gamma^{i,j}_{k_1,k_2,k_3}$ can be derived by an appropriate summation.

B.4 Proof for the correctness of the equations for recursively updating the sufficient statistics

The proof is based on the Markovian property of our HMM model. We prove here the updating equation for $\tau_{l,k}^{h-1,2j-1}$ when the new data $d_{h-1,2j-1}$ arrives, i.e.,

$$\tau_{l,k}^{h-1,2j-1} = r_{h-1,2j-1}^{l} + t_{h-1,2j-1}^{l} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1}) \tau_{l,k}^{h,j}.$$

The proof of updating equations for the other sufficient statistics can be derived similarly.

For $l \neq h-1$, we have $n_l^{h,j} = n_l^{h-1,2j-1}$ and $r_{h-1,2j-1}^{l} = 0$. By manipulating the Markovian property of the proposed HMM model, it is easy to prove that

$$t_{h-1,2j-1}^{l} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1}) \tau_{l,k}^{h,j} = \sum_{m=1}^{n_l^{h-1,2j-1}} P(s_{l,m} = k, \mathcal{S}_{h-1,2j-1} | \mathcal{D}_{h-1,2j-1}) = \tau_{l,k}^{h-1,2j-1}.$$

If $l = h-1$, we have $n_l^{h,j} = 2j-2$ and

$$r_{h-1,2j-1}^{l} = \gamma P(s_{h-1,2j-1} = k, \mathcal{S}_{h-1,2j-1} | \mathcal{D}_{h-1,2j-1}),$$

where γ is the time discount factor. We also have

$$t_{h-1,2j-1}^{l} \sum_{s_{h-1,2j-2}} g_2(s_{h-1,2j-1}) \tau_{l,k}^{h,j} = (1-\gamma) \sum_{m=1}^{2j-2} P(s_{h-1,m} = k, \mathcal{S}_{h-1,2j-1} | \mathcal{D}_{h-1,2j-1}),$$

therefore, according to the limiting EM algorithm, the updating equation for $\tau_{l,k}^{h-1,2j-1}$

is exactly the form shown in (6.5).

B.5 Proof for the computation of the relative entropy

The proof is as follows.

$$
\begin{aligned}
& \lim_{n\to\infty} \frac{1}{n} D(P_t||P_{t-1}) \\
= & \lim_{n\to\infty} \frac{1}{n} \sum_{\mathcal{S},\mathcal{D}} P_t(s_{L,1}) \prod_{i=1}^{L-1} P_t(s_{i,1}|s_{i+1,1}) \cdot \prod_{i=1}^{L} \prod_{j=2}^{n_i} P_t(s_{i,j}|s_{i,j-1}, s_{i+1,\lceil j/2 \rceil}) \\
& \cdot \prod_{i=1}^{L} \prod_{j=1}^{n_i} P_t(d_{i,j}|s_{i,j}) \cdot [\sum_{i=1}^{L} \sum_{j=1}^{n_i} \log \frac{P_t(d_{i,j}|s_{i,j})}{P_{t-1}(d_{i,j}|s_{i,j})} + \log \frac{P_t(s_{L,1})}{P_{t-1}(s_{L,1})} \\
& + \sum_{i=1}^{L-1} \log \frac{P_t(s_{i,1}|s_{i+1,1})}{P_{t-1}(s_{i,1}|s_{i+1,1})} + \sum_{i=1}^{L} \sum_{j=2}^{n_i} \log \frac{P_t(s_{i,j}|s_{i,j-1}, s_{i+1,\lceil j/2 \rceil})}{P_{t-1}(s_{i,j}|s_{i,j-1}, s_{i+1,\lceil j/2 \rceil})}] \\
= & \sum_{i=1}^{L} \frac{1}{2^i} \sum_{k_1,k_2,k_3} (\pi^i_{k_1,k_2,k_3})^t \log \frac{(\pi^i_{k_1|k_2,k_3})^t}{(\pi^i_{k_1|k_2,k_3})^{t-1}} \\
& + \sum_{i=1}^{L} \frac{1}{2^i} \sum_{k} (\pi^i_k)^t D(\mathcal{N}((\mu^i_k)^t, (\sigma^i_k)^t)||\mathcal{N}((\mu^i_k)^t, (\sigma^i_k)^{(t)})).
\end{aligned}
$$

Bibliography

[1] P. Abry, H. Helgason, and V. Pipiras. Wavelet-based analysis of non-gaussian long-range dependent processes and estimation of the Hust parameter. *Lithuanian Mathematical Journal*, 51:28–302, 2011.

[2] P. Abry and D. Veitch. Wavelet analysis of long range dependent traffic. *IEEE Transactions of Information Theory*, 44:2–15, 1998.

[3] M. Anand, Z. Ives, and I. Lee. Quantifying eavesdropping vulnerability in sensor networks. In *Proceedings of the 2nd International Workshop on Data Management for Sensor Networks*, pages 3–9, Aug. 2005.

[4] C. Ballester, A. Calvo-Armengol, and Y. Zenou. Who's who in networks. *Econometrica*, 74(5):1403–1417, 2006.

[5] J. Baras and T. Jiang. Cooperation, trust and games in wireless networks. In *book chapter in Advances in Control, Communication Networks, and Transportation Systems: In Honor of Pravin Varaiya, E.H. Abed (Ed.)*, 2006.

[6] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, pages 71–82, 2002.

[7] M. Bawa, H. Garcia-molina, A.Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Computer Science Dept., Stanford University, 2003.

[8] M. J. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian Statistics*, 7, 2003.

[9] C. Bishop. *Pattern Recognition and machine learning*. Springer, 2006.

[10] T. Bohnert and E. Monteiro. A comment on simulating LRD traffic with Pareto ON/OFF sources. In *Proceedings of ACM CoNEXT*, pages 228–229, 2005.

[11] P. Bonacich. Power and centrality: A family of measures. *The American Journal of Sociology*, 92(5), 1987.

[12] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual Ring Routing: network routing inspired by DHTs. In *Proceedings of ACM SIGCOMM*, pages 351–362, Oct. 2006.

[13] O. Cappe. Online EM algorithm for hidden Markov models. *Journal of Computational and Graphical Statistics*, 20(3):728–749, Sept. 2011.

[14] O. Cappe and E. Moulines. *Inference in the hidden Markov models*. Springer, 2005.

[15] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[16] J. Chen and A. Gupta. *Parametric statistical change point analysis*. Birkhauser Verlag, 2000.

[17] C. Cheng, H. Kung, and K. Tan. Use of spectral analysis in defense against DoS attacks. In *Proceedings of IEEE Global Communications Conference (Globecom)*, volume 3, pages 2143 – 2148, Nov. 2002.

[18] L. Cheng, X. Qiu, L. Meng, Y. Qiao, and R. Boutaba. Efficient active probing for fault diagnosis in large scale and noisy networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1 –9, March 2010.

[19] J. Corbo, A. Calvo-Armengol, and D. Parkes. The importance of network topology in local contribution games. In *Proceedings of 3rd International Workshop on Internet and Network Economics (WINE)*, pages 388–395, 2007.

[20] M. Crouse, R. Nowak, and R. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46(4):886 –902, apr 1998.

[21] J. Deng, R. Han, and S. Mishra. Security support for in-network processing in wireless sensor networks. In *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 83–93, Oct. 2003.

[22] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting hidden anomalies using sketch and non-Gaussian multiresolution statistical detection procedures. In *Proceedings of ACM SIGCOMM Workshop on Large-Scale Attack Defense*, pages 145–152, 2007.

[23] N. Duffield. Network tomography of binary network performance characteristics. *IEEE Transactions on Information Theory*, 52(12):5373 –5388, Dec. 2006.

[24] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks*, 4:15:1–15:37, June 2008.

[25] C. Gollier. *The Economics of Risk and Time*. MIT Press, 2001.

[26] D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *Proceedings of International Conference on Learning Thoery (COLT)*, pages 333–345, 2010.

[27] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopskins University Press, 1996.

[28] Y. Gu, G. Jiang, V. Singh, and Y. Zhang. Optimal probing for unicast network delay tomography. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1 –9, March 2010.

[29] Y. Gu, A. McCallum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, pages 32–32, 2005.

[30] G. Hardin. The tragedy of the commons. *Science*, Dec. 1968.

[31] T. Haveliwala, S. Kamvar, and A. Kamvar. Technical report: The second eigenvalue of the Google matrix. Technical report, Stanford, 2003.

[32] S. Hirose and K. Yamanishi. Latent variable mining with its applications to anomalous behavior detection. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 70–86, July 2009.

[33] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM*, pages 99–110, 2003.

[34] A. T. Ihler, J. W. Fisher, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, 2005.

[35] J. J. Jaramillo and R. Srikant. DARWIN: distributed and adaptive reputation mechanism for wireless ad hoc networks. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, pages 87–98, 2007.

[36] S. Jin and D. Yeung. A covariance analysis model for DDoS attack detection. In *Proceedings of IEEE International Conference on Communications (ICC)*, volume 4, pages 1882 – 1886, June 2004.

[37] A. Josang. A logic for uncertain probabilites. *International Journal of Uncertainty, Fuziness and Knowledge-Based Systems*, 9:279–311, June 2001.

[38] A. Josang and R. Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[39] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Sys.*, 43:618–644, March 2007.

[40] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 640–651, 2003.

[41] T. Karagiannis and M. Faloutsos. SELFIS: A tool for self-similarity and long-range dependence analysis. In *Proceedings of the 1st Workshop on Fractals and Self-Similarity in Data Mining: Issues and Approaches*, 2002.

[42] T. Karagiannis, M. Molle, and M. Faloutsos. Understanding the limitations of estimation methods for long-range dependence. Technical report, UC Riverside, 2006.

[43] C. Karlof and D. Wagner. Secure routing in sensor networks: attacks and countermeasures. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, May 2003.

[44] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70:39–45, April 1999.

[45] M. Kim, T. Kim, Y. Shin, S. Lam, and E. J. Powers. A wavelet-based approach to detect shared congestion. *Networking, IEEE/ACM Transactions on*, 16(4):763 –776, Aug. 2008.

[46] K. Komathy and P. Narayanasamy. Best neighbor strategy to enforce cooperation among selfish nodes in wireless ad hoc networks. *Computer Communications*, 30:3721–3735, Dec. 2007.

[47] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 63–72, 2003.

[48] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using trafc feature distributions. In *Proceedings of ACM SIGCOMM*, pages 217–228, 2005.

[49] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pages 21–32, 2009.

[50] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710 –732, July 1992.

[51] S. Mallet. A theory of multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674 –693, July 1990.

[52] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (Mobicom)*, pages 255–265, 2000.

[53] U. Maurer96. Modelling a public-key infrastructure. In *Proceedings of European Symposium on Research in Computer Security*, pages 325–350, 1996.

[54] K. P. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33, 2001.

[55] A. Nath and P. Domingos. Efficient belief propagation for utility maximization and repeated inference. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.

[56] H. Nguyen and P. Thiran. The boolean solution to the congested IP link location problem: Theory and practise. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 2117 –2125, May 2007.

[57] H. X. Nguyen and P. Thiran. Using end-to-end data to infer lossy links in sensor networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1 –12, April 2006.

[58] G. Nychis, V. Sekar, D. Andersen, H. Kim, and H. Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pages 151–156, 2005.

[59] R. Olfati-Saber. Distributed Kalman filter with embeded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference (CDC-ECC 05)*, pages 8179 – 8184, Dec. 2005.

[60] R. Olfati-Saber. Distributed Kalman filtering for sensor networks. In *Proceedings of IEEE Conference on Decision and Control*, pages 5492 –5498, 2007.

[61] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (Sensys)*, pages 255–265, 2003.

[62] D. Qian, C. Zhou, and J. Zhang. Cooperation enforcement in ad hoc networks with penalty. In *Proceedings of International Conference on Mobile Ad-hoc and Sensor Systems*, Nov. 2005.

[63] L. Rabiner. Non-Gaussian and long memory statistical characterisations for Internet traffic with anomalies. *IEEE Transactions on Dependable and Secure Computing*, 4:56–70, 2007.

[64] N. Ramanahan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 255–267, 2005.

[65] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. *IEEE Transactions on Neural Networks*, 16(5):1088 –1109, Sept. 2005.

[66] J.B. Rosen. Existence and uniqueness of equilibrium points for concave n-person game. *Econometrica*, 33(3):520–534, 1965.

[67] T. Roughgarden. Designing networks for selfish users is hard. In *Proceedings of the 42nd IEEE symposium on Theory of Computing*, 2001.

[68] T. Roughgarden and E. Tardos. How bad is selfish routing? *J. ACM*, 49:236–259, March 2002.

[69] J. Schiff, D. Antonelli, A. G. Dimakis, D. Chu, and M. J. Wainwright. Robust message-passing for statistical inference in sensor networks. In *Proceedings of the International Symposium on Information Processing for Sensor Networks (IPSN)*, pages 109–118, 2007.

[70] M. Seredynski, P. Bouvry, and M. A. Klopotek. Modelling the evolution of cooperative behavior in ad hoc networks using a game based model. In *Proceedings of IEEE Symposium on Computational Intelligence and Games*, pages 96 –103, April 2007.

[71] V. Shukla and D. Qiao. Distinguishing data transience from false injection in sensor networks. In *Proceedings of IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 41–50, June 2007.

[72] M. S. Taqqu, V. Everovsky, and W. Willinger. Estimators for long-range dependence: an empirical study. *Fractals*, 3:785–798, 1995.

[73] B. Wang, W. Wei, W. Zeng, and K. R. Pattipati. Fault localization using passive end-to-end measurement and sequential testing for wireless sensor networks. In *Proceedings of IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1 –10, June 2009.

[74] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Comput.*, 12:1–41, Jan. 2000.

[75] A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35:54–62, 2002.

[76] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 2446–2457, March 2004.

[77] Z. Yu and Y. Guan. A dynamic en-route scheme for filtering false data injection in wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 1 –12, April 2006.

[78] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-Temporal compressive sensing and Internet traffic matrices. In *Proceedings of ACM SIGCOMM*, pages 267–278, 2009.

[79] Y. Zhang, J. Yang, and H. Yu. Interleaved authentication for filtering false reports in multipath routing based sensor networks. In *Proceedings of IEEE International Paralllel and Distributed Processing Symposium (IPDPS)*, April 2006.

[80] A. Zheng, I. Rish, and A. Beygelzimer. Efficient test selection in active diagnosis via entropy approximation. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[81] S. Zheng, T. Jiang, and J. S. Baras. Robust state estimation under false data injection in distributed sensor networks. In *Proceedings of IEEE Global Communications Conference (Globecom)*, pages 1–5, Dec. 2010.

[82] S. Zhong and F. Wu. On designing collusion-resistant routing schemes for non-cooperative wireless ad hoc networks. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, pages 278–289, 2007.

[83] R. Zhou and K. Hwang. PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460 –473, April 2007.

[84] S. Zhu, S. Setia, S. Jajodia, and P. Ning. Interleaved hop-by-hop authentication against false data injection attacks in sensor networks. *IEEE Transaction on Sensor Networks*, 3, Aug. 2007.

[85] P. Zuraniewski and D. Rincon. Wavelet transforms and change-point detection algorithms for tracking network traffic fractality. In *Proceedings of the 2nd Conference on Next Generation Internet Design and Engineering*, 2006.