July 2021

Data-Oriented Approaches towards Mobile, Network and Secure Systems

Shangqing Zhao
University of South Florida

Data-Oriented Approaches towards Mobile, Network and Secure Systems

by

Shangqing Zhao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Zhuo Lu, Ph.D.
Yao Liu, Ph.D.
Nasir Ghani, Ph.D.
Kwang-Cheng Chen, Ph.D.
Kaiqi Xiong, Ph.D.

Date of Approval:
June 25, 2021

Keywords: Novel signal processing, collision decoding, 802.11, network tomography, network inference, USRP

Dedication

This dissertation is dedicated to my brilliant and virtuous wife, Zheying Sun, and my parents, Laifu Zhao and Manjv Wang, who inspire and support me through this journey.

Acknowledgments

First of all, I would like to specially thank my major advisor Dr. Zhuo Lu, who gives me insightful guidance and significant supports throughout my whole Ph.D. study and research. I could not have done my dissertation without his help. His spirit of research, his patience, his motivation, as well as his immense knowledge have positive and lifelong impacts on my professional development. Therefore, I am very proud to have him as my advisor.

I would like to thank all my committee members, Dr. Yao Liu, Dr. Nasir Ghani, Dr. Kwang-Cheng Chen, Dr. Kaiqi Xiong, for their time and efforts reviewing and discussing my research. Their guidance is immeasurable both during the writing of this dissertation and over the entire course of my time at the University of South Florida.

Thank you to my lab-mates with whom I've had the opportunity to work, and talk. They are Zhengping Luo, Tao Hou, Lei Pi, Zhe Qu, Rui Duan, Song Fang, Tao Wang, Di Zhuang, Sen Wang, Mingchen Li, Mohammed Alrowaily. Last but not the least, I would like to give my special thanks to all my families and friends, all the faculty and staff members in the Department of Electrical Engineering for their help and support during these years.

Table of Contents

List of Tables

List of Figures

Abstract

With the rapid evolvement of information science, data-oriented research has solicited a new philosophy for the future mobile network and security design, since it can not only encourage new designs achieving more efficient and reliable networks, but also pose new challenges towards security designs. In this dissertation, we propose four novel data-oriented designs or frameworks to prompt or calibrate the performance with respect to efficiency, reliability, and security.

In the wireless domain, packet corruption and packet collision are two major threats that jeopardize the performance of a mobile network. To cope with the packet corruption, we propose the STAteful inter-Packet signaL procEssing (STAPLE) framework, which is an inter-packet oriented signal process design for wireless networking. STAPLE transforms the signal processing procedure into a lightweight stateful process that caches in a small-sized memory table physical and link layer header fields as packet state information. The similarity of such information among packets serves as prior knowledge to further enhance the reliability of signal processing and thus improve the wireless network performance. For the packet collision, we present a new design called comb decoding (CombDec) to efficiently resolve RTS collisions without changing the 802.11 standards. We observe that an RTS payload, when treated as a vector in a vector space, exhibits a comb-like distribution; i.e., a limited number of vectors are much more likely to be used than the others due to RTS payload construction and firmware design. This enables us to use sparse recovery such as compressive sensing to resolve RTS collisions.

For the network security design, we revisit network tomography and inference from a data-driven perspective and discover that there are two vulnerabilities. The first one is of

the measurement integrity. By taking advantage of this vulnerability, we develop an attack strategy, called measurement integrity attack, which not only destroys the measuring system, but can even mislead the system to scapegoat other innocent users. The second vulnerability is of the measurement confidentiality since network inference is able to leak network flow information without directly measuring it. To prevent disclosing the flow information, we find that random routing strategies are capable of hiding the flow information. Then, by leveraging the measurement data, we propose a new framework that can systematically study the behavior of different randomized routing protocols, and explore the fundamental reason why randomized routing strategies can prevent information leakage against network inference.

Chapter 1: Introduction

With advances in mobile technologies, computing capabilities, improved access to networks, interconnected devices in different networks, e.g., wireless local area network (WLAN), cellular network, personal area network (PAN), Internet of things (IoT), etc., have become more advanced and integrated into people's life, inevitably generating a large amount of data. The emerging data broadens our horizons when seeking new approaches in the next-generation wireless network and security designs since leveraging data provides many powerful capabilities, such as the predictive sight or continuous improvement. Therefore, data-oriented approaches have become a new dimension in a wide range of areas, including network and security design, in the future. In this dissertation, we propose four data-oriented approaches that renovate or calibrate current network and security designs.

1.1 Wireless Data Based Mobile System Design

Mobile technologies have become one of the dominant ways for people to access the Internet. However, due to the broadcast nature of the wireless channels, the intensive traffic environment can desperately jeopardize the performance within a mobile network. To cope with this issue, we revisit the fundamental part within wireless mobile networks, and deal with two main aspects to improve mobile network system performance: packet corruption and packet collision. The first one is due to the unpredictable wireless fading, and the second one is resulted from concurrent transmissions. We discover that there are indeed opportunities from a data-driven perspective to further improve mobile network efficiency and reliability.

1.1.1 Inter-Packet Based Approach for Packet Corruption

In a wireless network, signal processing is an essential procedure at a receiver during packet reception to resist the packet corruption. It includes three major steps: timing/frequency synchronization, channel estimation, and equalization [1, 2, 3, 4]. These together are generally treated as an independent process, serving as the foundation towards data decoding and higher layer protocol management [5, 6, 7, 8, 9, 10].

Our research finds out that leveraging common protocol information across network packets can substantially improve the signal processing performance in a wireless network. Our motivation is that in a realistic network scenario, packets are not transmitted with uniformly random in-packet settings, such as date rate, packet size, source and destination addresses in the physical (PHY) or medium access control (MAC) fields. There exist common PHY and MAC header fields across different packets. For example in a WiFi network: a station only receives data from the access point (AP), indicating that the packets it cares about always have the same source address; and packets for data-intensive applications usually have the same packet size (i.e., the maximum allowable size). Such commonness of in-packet settings can serve as prior knowledge to improve the signal processing performance.

Hence, if we take signal processing out of its traditional domain and place it in a practical wireless network, there is indeed a lot of prior knowledge that can be used to boost its performance. The underlying challenge is how we are able to harness all possible common information and piece everything together to re-design signal processing with limited overhead for practical use in the wireless network domain.

To deal with these challenges, we propose the STAteful inter-Packet signaL procEssing (STAPLE) framework, which is a generic design to improve the performance of signal processing for wireless networking. Rather than designing signal processing in its traditional domain, we make signal processing stateful within the network domain. In particular, the

key design in STAPLE is that a receiver maintains a small-sized state table, whose entry includes bit values of selected PHY/MAC fields from successfully decoded packets, such as MAC addresses, packet length and data rate, depending on a standard. These bit values constitute the state of a packet.

When a new packet arrives and decoding error happens, STAPLE performs state association (i.e., matching the bit values of its header fields with a known entry to restore the packet to a previous state). If the association succeeds, STAPLE considers the state of the packet is known (i.e., a part, if not all, of PHY/MAC information is known). Then, such known knowledge serves for the same purpose of "training sequence" (also known as "preamble" in many standards [11, 12]). Therefore, the restored state is combined with the original preamble in the packet to form a longer preamble. STAPLE re-performs signal processing on this longer preamble. Because more "prior" information is fetched for signal processing, STAPLE can obtain better frequency and channel estimates, which helps packet decoding and accordingly improves the network link performance.

To the best of our knowledge, there is no comprehensive system study in the literature to utilize the "prior knowledge" extracted from packet headers to improve the signal processing performance in wireless networks. We propose STAPLE as a general software radio system framework, then explore the feasibility and evaluate effectiveness of such a framework. In particular, we design and implement STAPLE with adapted configurations for 802.11a/b/g/n/ac and 802.15.4 on USRP X300 devices. The features of STAPLE are as follows: (i) lightweight processing and low overhead with a very small state table size (as experimental results suggest that it is sufficient to store 2-5 states at a station and 12-20 at the AP); (ii) a universal framework that benefits a wide range of wireless networks; (iii) no modification to any wireless standard; (iv) orthogonality to existing packet data decoding mechanisms (e.g., partial packet recovery [13, 14, 15]).

We conduct comprehensive experiments to show the benefits of STAPLE for different wireless standards. Our main contributions are summarized as follows: (i) We propose to improve signal processing within the network domain. We introduce the concept of the state of a packet as the bit values of selected PHY and MAC fields, and demonstrate that state-associating a packet to a previous state and re-performing signal processing in STAPLE improve the wireless network performance. (ii) We design, implement and configure the STAPLE prototypes for 802.11b/g/n/ac and 802.15.4. STAPLE is lightweight, effective and brings universal performance benefit to the packet reception process for wireless networks. Comprehensive experimental results show that STAPLE improves the packet delivery ratio by up to 20.8% under various conditions.

1.1.2 RTS Data Based Approach for Packet Collision

CSMA/CA is fundamental for WiFi to coordinate multiple nodes to access the wireless channel. RTS/CTS is a widely-used mechanism in WiFi, which uses short RTS packets for fast collision inference, transmission path check as well as combating the hidden terminal problem [11]. In many early WiFi products, RTS/CTS was disabled due to the concern of overhead [16]. With the substantial increase of data demand in recent years, WiFi data packets become longer and longer, and today's WiFi devices send an RTS packet when the size of a data packet exceeds a given threshold. The threshold is usually set to around 2,300 bytes [17, 18, 19, 20, 21] to balance the performance and the overhead. RTS/CTS is also used in advanced WiFi functionalities, such as beamforming and MU-MIMO [11]. In addition, global RTS/CTS [22, 23, 24] has also been proposed for cross-technology communications.

The essence in the RTS/CTS mechanism is to trade a small cost of the RTS collision for a potentially large cost of data collision. In this research, we revisit the RTS collision problem and present the comb decoding (CombDec) system to resolve RTS collisions without

changing the 802.11 standard and thus improve the wireless channel utilization as well as the network throughput.

The observation behind designing CombDec is that the data payload of an RTS packet, when treated as a vector in a vector space, exhibits a comb-like distribution. Specifically, the RTS content consists of 160 bits, which leads to 2^{160} possibilities. We find that the standard-structured data fields in RTS actually result in at most around 2^{21} possible contents in today's WiFi networks. Therefore, the probability distribution of such contents will exhibit a comb-like shape: only up to 2^{21} out of 2^{160} contents having non-zero probabilities.

As a result, we reformulate the RTS collision problem as a weighted sum problem: we consider the received signal due to an RTS collision is the sum of all 2^{21} RTS contents transmitted at the same time, but with different channel weights. An RTS content has a non-zero channel weight if it is actually transmitted, and zero weight otherwise. Then, resolving the collision is equivalent to solving for the channel weight for each RTS content. The key observation to solve the problem is that a vast majority of the 2^{21} channel weights should be zeros because a collision involves only a few RTS packets in a real-world network. In other words, the vector that includes all channel weights is sparse, which opens a path to use sparse recovery [25, 26, 27] to resolve RTS collisions.

One significantly challenging issue around collision resolution based on sparse recovery is the computational complexity because we start from around 2^{21} possibilities of RTS signals to resolve a collision. To cope with this issue, we analyze how a key RTS data field, `Duration` (that specifies the time duration an RTS packet reserves), is constructed in state-of-the-art 802.11 firmware. A comprehensive set of 802.11ac packet traces are also collected to understand the distribution of `Duration` in various scenarios. It is found that today's firmware imposes extra restrictions on `Duration` and is biased towards a limited number of value selections, and the distribution of `Duration` in real-world packets is highly uneven and patterned. Based on this observation, CombDec is designed with two key components: (α,

β)-construction and γ-decimation, which adaptively narrow down the search range in sparse recovery to a set of only hundreds of potential RTS signals, making system design of collision resolution practical for WiFi networks.

We implement CombDec in a 20-node network testbed, and evaluate it in different scenarios. Experimental results demonstrate that our design has both direct and indirect impacts on today's WiFi systems and setups.

- For the direct impact, today's WiFi devices usually adopt a conservative RTS threshold (i.e., around 2300 bytes), which results in 30% - 45% data transmissions initiated by RTS (according to our packet trace collection and analysis). By directly using CombDec with current RTS settings, we find via experiments that CombDec is able to decode 98% of two-RTS collisions and improve the network throughput by up to 23.3%

- For the indirect impact, CombDec offers a new capability of decoding RTS collisions and in fact encourages changing today's WiFi setups for more RTS transmissions. Therefore, by reducing the RTS threshold to zero and letting every device always send RTS before data (indicating that most collisions in the network become RTS collisions), CombDec significantly improves the network throughput by up to 46.6% in experimental evaluations.

The design of CombDec is the first systematic work towards resolving RTS collisions in WiFi networks. It is non-invasive and redefines the role of the RTS functionality and pushes WiFi towards a collision-free environment.

1.2 Measurement Data Based Network Security Design

Accurate and timely monitoring of network performance is vital to ensure a reliable and efficient network environment. However, under some situation, direct observation is not fea-

sible, then network tomography or network inference has emerged as an algorithmic process to transfer end-to-end path measurements into link metric estimates [28, 29, 30, 31, 32, 33, 34, 35, 32, 36, 37, 38]. However, when revisiting network tomography from a data-driven perspective, there are two vulnerabilities. The first one is of the measurement integrity. Specifically, in network tomography, existing studies make a seeing-is-believing assumption that measurements over end-to-end paths between monitors indeed reflect the real performance aggregates over individual links. However, such an assumption does not always hold in the presence of malicious autonomous systems. Therefore, by taking advantage of this vulnerability, we develop an attack strategy, called measurement integrity attack, which not only destroys the measuring system, but can even mislead the system to scapegoat other innocent users. The second vulnerability is of the measurement confidentiality. In particular, network tomography (inference) [39, 40, 41, 42, 43] offers a way to obtain the path information without directly measuring it, which poses severe challenges in many anonymous networks. Therefore, to prevent disclosing the path information, we find that random routing strategies are capable of hiding the path information. Then, by leveraging the measurement data, we systematically study the behavior of different randomized routing protocols, and explore the fundamental reason why randomized routing strategies can prevent information leakage against network inference.

1.2.1 Vulnerability of Measurement Integrity

By nature, network tomography does not directly observe network link metrics, but "tele-measure" them via measurements over end-to-end paths. Each path consists of a few or more links. Existing work mainly focused on algorithm design and applications (e.g., [33, 34, 35, 32, 36, 37, 38]); and some recent papers also considered the problems of placement of monitors and identifiability of link metrics (e.g., [44, 45, 46, 47]). In essence, network tomography can be considered as an algorithmic process to transfer end-to-end measurements

into link metric estimates. Interestingly, most existing studies on network tomography emphasize extracting as much information about link metrics as possible from available measurements, and always make a *seeing-is-believing* assumption that measurements over end-to-end paths between monitors indeed reflect the real performance aggregates over individual links. However, such an assumption does not always hold in the presence of malicious autonomous systems [48, 49], backdoor-infected routers [50], and node-capture attacks [51, 52] as these adversaries actively affect packet forwarding and have become increasingly possible in today's complicated environments. Rather, the assumption renders a serious security problem of measurement integrity during the network tomography process.

By taking advantage of this *seeing-is-believing* vulnerability in network tomography, we develop a new class of attack strategies, called *measurement integrity attacks*. Unlike conventional data integrity problems that are usually protected by standard methods (e.g., encryption and authentication), a key challenge associated with measurement integrity attacks is that the facts (e.g., packet transmission/delivery timings) during network measurement cannot be protected by such standard methods, but can be easily manipulated by malicious attackers. The basic idea of measurement integrity attacks is to intentionally delay or drop packets at malicious nodes to manipulate end-to-end measurements between monitors in a way such that a legitimate node is incorrectly identified by network tomography as the root cause of the problem, thereby becoming a scapegoat. We propose three basic attack strategies with different objectives.

1. Chosen-victim scapegoating, in which attackers target one or more given victims such that these victims are misleadingly identified by network tomography as the root cause of a problem.

2. Maximum-damage scapegoating, in which attackers find the optimal set of victims among all nodes to inflict the maximum damage to the network.

3. Obfuscation, by which network tomography is tricked to produce a substantial number of link estimates beyond the normal status to confuse a network operator.

We analyze the feasibility of these strategies, and present the conditions for detecting and locating such attacks. We also use network datasets to perform simulation experiments to show the success possibility, damage, and the detectability and locatability of such attacks. Our main contributions can be summarized as follows.

- We are the first to investigate the vulnerability in network tomography mechanisms from a security perspective, and reveal that measurement integrity attacks are able to damage the network while substantially misleading network tomography without knowing the global routing knowledge of the network.
- We systematically construct three basic attack strategies, and investigate the feasibility of such attacks, then propose methods to detect and locate measurement integrity attacks.
- We use real-world datasets to evaluate the threats of measurement integrity attacks in network systems with various settings. Experimental results demonstrate that the current practice of network tomography is even vulnerable to a single attacker.

Our work demonstrates that when a measurement integrity attack is successfully launched, network tomography generates misleading and erroneous outputs, based on which failure recovery or mitigation procedures may further exacerbate the damage caused by the attack. As security plays a critically important role in network design and measurement, network tomography should be developed not only for conventional goals such as efficiency and identifiability, but also for security. Hence, existing network tomography methods in various applications need to be revisited to increase attack resilience and adopt necessary detection mechanisms.

1.2.2 Vulnerability of Measurement Confidentiality

In a wireless network, network flow information (i.e., the flow data rates of source-destination pairs on end-to-end paths) is the essential knowledge about the network. Equipped with such knowledge, malicious adversaries will know who is communicating with whom in the network or how much data rate a pair of communicating parties has, and then launch powerful, effective attacks targeting the network [53, 54].

In many large-scale wireless networks, such as wireless sensor network (WSN) [55] or mobile ad-hoc network (MANET) [56], directly observing the end-to-end flow information is not always possible or even infeasible because of the prohibition in anonymous networks or the measurement traffic overhead. To acquire network flow information, adversaries can leverage the method of *network inference* to infer the end-to-end flow rates through eavesdropping on wireless link activities, which is widely feasible in wireless networks because of the broadcast nature of the wireless medium. Network inference was originally designed for the effective network management and diagnosis, therefore most existing studies focus on optimizing the inference performance [57, 42, 41, 58, 59, 60, 43]. However, from the adversary's perspective, such a line of work indeed exposes the vulnerability of leakage of network flow information, in which the adversary can simply obtain such information through wireless link measurements.

How accurately the adversary can obtain the flow information depends on the inference method the adversary uses, network traffic patterns, and routing protocols. Although it has been noted recently [61] that the inference error for an adversary can be maximized if a network can make sure the adversary has no knowledge of how packets are routed in the network, little progress has been made to reveal how a network can indeed meet this goal. Obviously, a larger inference error helps a network to hide its flow information against leakage more. To achieve this, the network can make the routing of packets unpredictable to adversaries. For example, if a deterministic routing protocol like the shortest path routing

[62] is used, it can be very likely that the adversary can accurately infer the flow information from link measurements based on standard network inference methods [63]. On the contrary, if a routing protocol is randomized, such as using Tor [64], the intermediate relay nodes are randomly selected to form an end-to-end path, yielding unpredictable behavior of packet forwarding observed by an outside adversary.

In this work, we study the security benefits of randomized routing protocols against malicious network inference as well as their associated costs. In particular, we focus on analyzing three major templates for randomized protocols based on common routing behaviors in wireless networks.

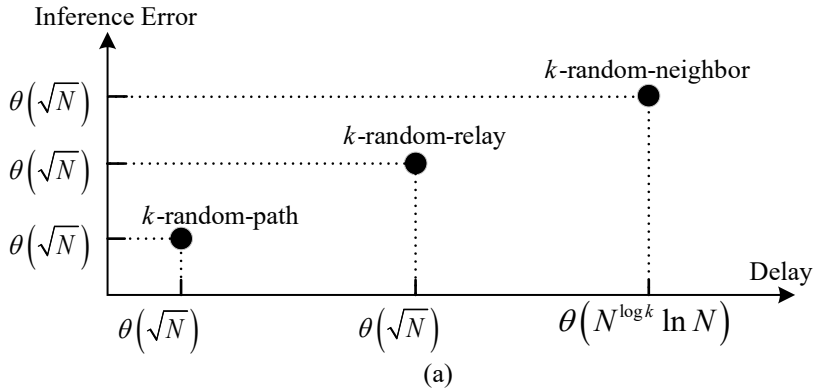
1. k-random-relay: k forwarding nodes (called relays) are selected randomly in a network. The routing path for each packet must contain such relays.

2. k-random-neighbor: in which the next hop is selected randomly. Specifically, for each packet, at each hop, one neighbor is selected randomly from k neighbors with the shortest distances (the distance is measured by the number of hops) to the destination.

3. k-random-path: each packet is sent by one randomly selected path from the k shortest end-to-end paths.

These templates capture the majority of randomized behaviors during packet forwarding that can make the entire routing unpredictable. In this chapter, we use an asymptotic approach to theoretically model these routing templates and measure their induced inference errors. We also analyze the incurred delay cost of three templates. Our major results can be summarized in Fig. 1.1, which depicts the lower bound of inference errors for three templates scaled by the delay cost in asymptotic notations for a network with N nodes and $\sqrt{N}$ active end-to-end flows under the condition that N is sufficiently large.

From Fig. 1.1(a), we see that the inference errors induced by all templates are on the same order of $\sqrt{N}$ with constant difference. Both k-random-relay and k-random-path templates



(a)

	Inference Error	Delay
k-random-relay	$\theta\left(\frac{\sqrt{N}}{(k+2)}\right)$	$\theta\left((k+1)\sqrt{N}\right)$
k-random-neighbor	$\theta\left(\frac{N}{(\ln N)^{-1}+\sqrt{N}}\right)$	$\theta\left(N^{\log k}\ln N\right)$
k-random-path	$\theta\left(\frac{\sqrt{N}(k-1)^2}{2k^2}\right)$	$\theta\left(\sqrt{N}\right)$

(b)

Figure 1.1. Asymptotic inference error and delay performance for a network with N nodes and $\sqrt{N}$ flow, where (a) shows the relationship of them, and (b) provide more exact results.

have delay costs on the same order of $\sqrt{N}$. The k-random-neighbor template incurs a delay cost of $N^{\log k}\ln N$. Our results indicate that k-random-path achieves the inference error with the same order of the other two templates while maintaining the lowest delay cost. But it requires knowing the global path information of the network, and in practice, such information is generally unavailable or prohibited to know. For a large k, the delay of k-random-neighbor is significantly larger than others, but the inference error is still on the same order of the others. As a result, k-random-neighbor with a large k should be avoided. From Fig. 1.1(b), for k-random-relay, both the inference error and the delay are larger than k-random-path by a constant order of magnitude. In addition, k-random-relay

does not require to know the global information, thus a number of real-world applications (e.g., Onion routing or Tor [64]) leverage this template to achieve the anonymity.

Our main contributions are summarized as follows.

- We are the first to study the vulnerability of the leakage of network flow information from the routing protocol perspective, and reveal that randomized routing protocols help to prevent revealing flow information by inflicting large inference errors. Whereas, a large inference error is always associated with a large delay cost.

- We propose three templates based on routing behavior, i.e., k-random-relay, k-random-neighbor and k-random-path. Then we systematically characterize and model these templates, and investigate the inference error as well as the incurred delay cost of each template. Our theoretical results verify that each randomized template is able to hide the flow information with different capabilities.

- We conduct comprehensive simulations to evaluate the inference error and delay of each template under a practical network inference setup. Experimental results confirm the relationship between the inference error and the delay cost.

1.3 Dissertation Overview

In Chapter 2, we present STAPLE, which is a lightweight, efficient mechanism to improve the signal processing performance for wireless networking. In Chapter 3, we present a new decoding system CombDec that uses (α, β)-construction, γ-decimation and sparse recovery to resolve RTS collisions. In Chapter 4, we present new attack strategies called measurement integrity attacks against network tomography, and use theoretical and experimental results to analyze the feasibility. We also present the conditions to detect and locate these attacks. In Chapter 5, we systematically categorize randomized routing protocols into three tem-

plates, k-random-relay, k-random-neighbor and k-random-path, and theoretically analyze the inference errors and incurred delay costs of them.

Chapter 2: Stateful Inter-Packet Signal Processing for Wireless Networking

2.1 Abstract

Traditional signal processing design (e.g., frequency offset and channel estimation) at a receiver treats each packet arrival as an independent process to facilitate decoding and interpreting packet data. In this chapter, we enhance the performance of this process in the wireless network domain. We propose STAteful inter-Packet signaL procEssing (STAPLE), a framework of *stateful* signal processing residing between the physical and link layers. STAPLE transforms the signal processing procedure into a lightweight stateful process that caches in a small-sized memory table physical and link layer header fields as packet state information. The similarity of such information among packets serves as prior knowledge to further enhance the reliability of signal processing and thus improve the wireless network performance. We implement STAPLE on USRP X300-series devices with adapted configurations for 802.11a/b/g/n/ac and 802.15.4. The STAPLE prototype is of low processing complexity and does not change any wireless standard specification. Comprehensive experimental results show that the benefit from STAPLE is universal in various wireless networks.

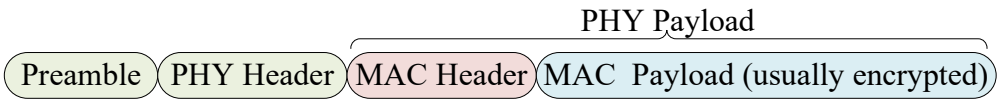


Figure 2.1. Generic packet structure viewed at the PHY and MAC layers.

2.2 Preliminaries

In this section, we describe preliminaries about signal processing and packet decoding in wireless networks.

2.2.1 Packet Specifications in Wireless Networks

In today's wireless networks, elements for data exchange are packets that contain both standard-specified information and user data. Figure 2.1 illustrates a generic packet format viewed at the PHY and MAC layers. As shown in Figure 2.1, a packet consists of the preamble, the PHY header, and the PHY payload that includes the MAC header and payload.

The preamble is specified in a wireless standard and known to the public. It is also called as pilot or training sequence [65] for energy detection, timing/frequency synchronization and channel estimation [66]. The PHY header usually provides specific PHY layer information for decoding the PHY payload. Depending on a wireless standard, such information may include packet length, modulation type, and data rate. The PHY header may also contain a checksum field, such as cyclic redundancy check (CRC) in 802.11b, to verify the successful decoding of the PHY header. The MAC header facilitates the functionality of coordinating multi-access of network nodes to share the wireless channel. It generally includes the source and destination MAC addresses, and packet type information, such as acknowledgement (ACK), beacon, and data packets in 802.11a/b/g/n/ac [11]. MAC payload is the actual data at the MAC layer, including high-layer protocol information and the user's data at the application layer. In today's wireless networks, this payload is usually encrypted [11, 12].

When a packet is transmitted by a sender to a receiver in a wireless network, the increase of the energy level at the receiver triggers the packet reception process.

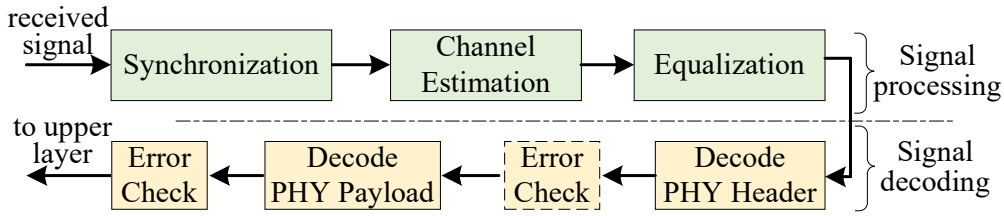


Figure 2.2. Packet reception at a receiver.

2.2.2 Packet Reception in Wireless Networks

The packet reception process consists of two relatively independent procedures: signal processing and signal decoding, as shown in Figure 2.2. The signal processing procedure is used to prepare all necessary pre-decoding steps before decoding the packet data at the receiver. These steps generally happen in sequence as synchronization, channel estimation and equalization. The signal decoding procedure is to first decode the PHY header to obtain essential PHY information, which is then used to decode the PHY payload, including the MAC header and the MAC payload.

2.2.2.1 Signal Processing

Because the radio wave propagating through the wireless channel is a complicated phenomenon characterized by various environmental factors, such as the multi-path fading and the doppler effect [66], the signal processing procedure estimates and compensates those factors before the signal is decoded into data bits. Generally, signal processing contains three major steps: synchronization, channel estimation, and equalization, as shown in Figure 2.2. All these steps are based on the known preamble with length n in the packet, denoted by $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, where x_i is the i-th baseband symbol in the preamble.

Synchronization includes timing synchronization and frequency synchronization. Timing synchronization is triggered by energy detection, and is to find the start position of the first PHY layer symbol in a packet [67] such that the later processing on the symbols in the packet

is aligned. After timing synchronization, the received signal of the preamble in the packet is represented as a vector $\mathbf{s} = [s_1, s_2, \cdots, s_n]$. Frequency synchronization is to estimate and compensate the frequency offset Δf between the transmitter and the receiver [66] from the received preamble signal $\mathbf{s}$ based on the knowledge of the preamble $\mathbf{x}$.

After frequency offset compensation, the received signal vector of the preamble $\mathbf{s} = [s_1, s_2, \cdots, s_n]$ has been compensated to a new vector $\mathbf{s}' = [s'_1, s'_2, \cdots, s'_n]$ [4], based on which channel estimation is performed to estimate the channel state information (CSI). The maximum-likelihood (ML) algorithm that minimizes the estimation error on the flat-fading channel [66] is

$$\hat{h} = (\mathbf{x}^H \mathbf{x})^{-1} \mathbf{x}^H \mathbf{s}', \tag{2.1}$$

where $(\cdot)^H$ and $(\cdot)^{-1}$ denote the matrix conjugate transpose and inverse, respectively. ML channel estimation for frequency-selective channels [3], orthogonal frequency-division multiplexing (OFDM) [1] or multiple-input and multiple-output (MIMO) channels [68] are performed in a similar linear process to (2.1). Note that although the matrix inverse operation in (2.1) is nonlinear, the preamble $\mathbf{x}$ is known to the public and accordingly $(\mathbf{x}^H \mathbf{x})^{-1} \mathbf{x}^H$ in (2.1) can be pre-computed.

Once the receiver obtains the estimated CSI $\hat{h}$, it performs channel equalization [69] to compensate the channel effect to support coherent demodulation. Note that OFDM has been widely adopted in today's wireless standards, such as in 802.11a/g/n/ac [11], because it significantly simplifies the channel equalization process by transforming a frequency-selective wireless channel into multiple parallel flat-fading ones.

2.2.2.2 Signal Decoding

Signal decoding follows immediately once signal processing is finished. As a packet may have different modulation and coding schemes in the PHY header and payload (e.g., in

802.11a/g/n/ac), the receiver first demodulates and decodes the PHY header and obtains the data rate information (i.e., the modulation and error-correction coding information) about the PHY payload, along with other necessary information. As shown in Figure 2.2, if a checksum (e.g., CRC or parity) exists in the PHY header and the decoded header does not pass the error check, the receiver has to drop the packet (when there is no other error recovery option); otherwise, the receiver uses the data rate information in the PHY header to demodulate and decode the PHY payload, obtaining the MAC header and the MAC payload. If the receiver verifies the checksum of the MAC payload, it passes the payload data to the upper layer for protocol management and data delivery.

Demodulation and error-correction decoding during the signal decoding process have been well studied in the wireless communication domain [70, 16].

2.3 Design Motivation behind STAPLE

In this section, we introduce the intuition behind stateful signal processing, then use real-world measurements to validate our intuition. Finally present our basic design of the STAPLE mechanism.

2.3.1 Basic Intuition and Observations

Traditionally, the entire packet reception is treated as an independent process at a receiver. Recent studies have focused on enhancing the signal decoding procedure to improve wireless link performance, such as partial packet recovery [71, 15, 72, 14, 7, 9]. Little attention has been focused on re-designing the signal processing procedure for wireless packet reception, even in the signal processing community, because synchronization, channel estimation and equalization are all well explored in the literature [2, 1, 4, 66, 73, 65].

In this chapter, we take a closer look at the signal processing procedure. Intuitively, such a procedure must be treated as an independent process for each packet arrival. For example,

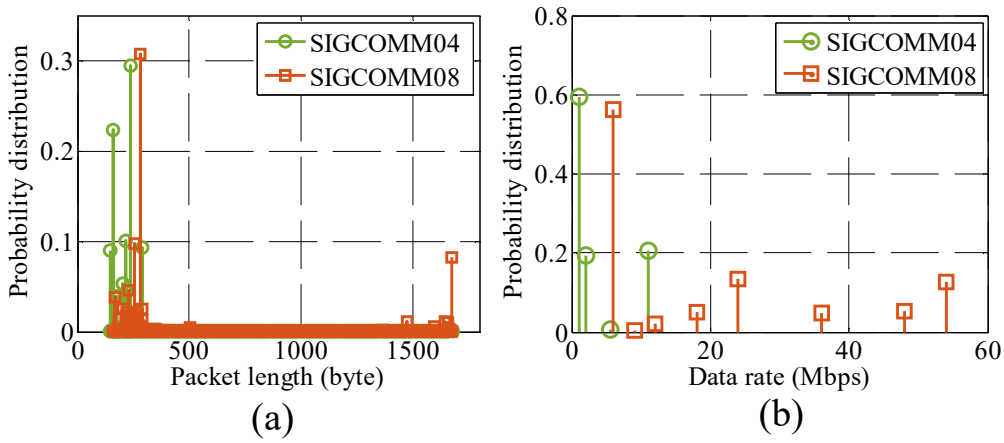


Figure 2.3. Distributions of (a) packet length and (b) data rate in SIGCOMM04/08 datasets.

upon arrival of a packet, the receiver estimates the frequency offset and the CSI from the preamble, compensates their effects on the received signal for decoding. As the frequency offset and the CSI are time-varying and hard to predict [4], the receiver has to estimate and compensate their effects again when a new packet arrives.

It is difficult to further improve a well-established signal processing algorithm in practice. Our design intuition is that if we place signal processing in the network domain, we should be able to boost its performance if we can leverage the common information across different packets. In a wireless network, packets indeed have some common information, which can serve as prior knowledge during packet arrival. We analyze the realistic packet trace datasets SIGCOMM04 [74] and SIGCOMM08 [75] to see whether common information exists between packets in typical WiFi network usages. Figure 2.3(a) shows the packet length distributions in the datasets. We observe that the distribution in each dataset is extremely uneven and consists of two regions: the first region includes small packet lengths (1-250 byes), indicating control or short data packets; and the second is around the maximum allowable packet size (1400-1600 bytes), indicating packets for data-intensive applications. Similarly, uneven distributions of data rates in packets are shown in Figure 2.3(b). Given an AP in SIGCOMM08, we also find that on average 4.2 nodes sent 51.7 packets to the AP within one-second period.

Those packets have the same PHY/MAC field values with high probability. The quantitative evaluation on each of these fields will be detailed in Section 2.5.

2.3.2 Basic Design of STAPLE

The preliminary observations show that there exists common (but maybe scattered) information among packets, which can serve as prior knowledge for signal processing during packet reception. If we look at such prior information solely within the signal processing domain, a traditional way is to perform profiling/training among all PHY/MAC fields in collected packets to build the prior distributions of such fields, and then integrate these distributions into a maximum a posteriori (MAP) framework [66] to improve signal processing.

Nonetheless, such a method faces two practical issues: (i) Profiling empirical distributions is always practically cumbersome due to its heavy dependencies on variable factors, such as physical environments, the number of users, and how they use the network, all of which cannot be easily predicted from time to time. (ii) Different data fields in packets are very likely to exhibit distinct distributions and also have correlations. Integrating these distributions into a unified framework is challenging. Moreover, the design must be lightweight and efficient to ensure low overhead and timely signal processing.

Thus, we avoid designing this method solely in the signal processing domain. Rather, we look at it from the network perspective. A core idea in network management is to maintain a network state, such as the backoff state at the MAC layer or the congestion control state in TCP. When a sender transmits a packet, we can use its header field values to form its state. Formally, we define the state of a packet as follows.

Definition 1 *The state of a packet is a bit sequence concatenated by one or more bit strings in the PHY and MAC header fields in the packet.*

All PHY/MAC header fields in a packet can be combined to represent a state. In this way, the state space of a packet might be very large in theory. In practice, we can carefully

select a few fields that are very likely to have the same value across packets to form the state (as indicated in our packet trace analysis). Hence, our motivation is to introduce state management into signal processing in a network context. Figure 2.4 draws the sketch of STAPLE, which consists of three major steps.

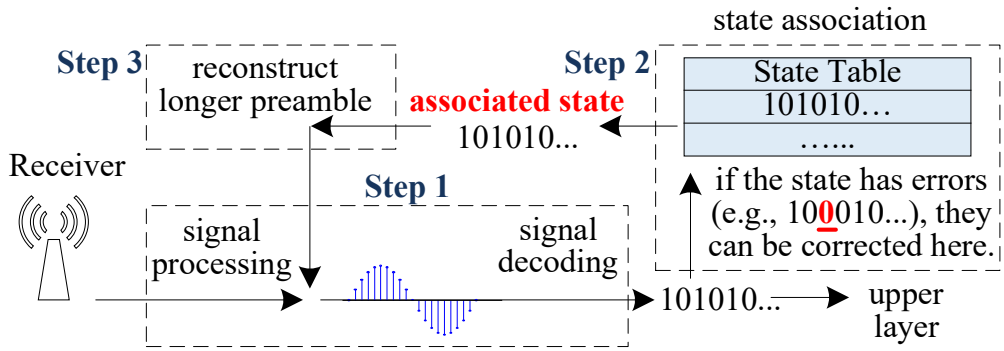


Figure 2.4. Sketch of STAPLE.

1. When a receiver receives a packet, it first goes through signal processing and decoding. If no error happens, the receiver can directly send the data to the MAC and higher layers, finishing the packet reception.

2. When errors happen, STAPLE tries to associate the corrupted packet with a previous state. Specifically, it compares the PHY/MAC fields in the corrupted packet with a state table that stores those fields from successfully decoded packets, which finds the closest state to the packet and recover it into that state. As shown in Step 2 of Figure 2.4, when the state of the packet has errors, they can be then corrected after state association; when the state contains no error and exactly matches a previous one, other parts of the packet, such as the packet payload, should have errors leading to decoding failure. In both cases, the state of the packet is associated.

3. At this point, the PHY/MAC header fields in the packet state can be considered as prior knowledge. This indicates that the information can be combined with the original

preamble in the packet to form a longer preamble to re-perform the signal processing, which can yield better estimates for frequency offset and CSI (as estimation errors in signal processing generally decrease with the preamble length increasing [2, 76, 1, 77]). Then, the receiver re-performs the signal decoding using these better estimates. They can improve the performance of decoding the payload because it usually has a higher modulation/coding rate than the packet header, and accordingly requires more accurate frequency offset and CSI estimates.

STAPLE transforms prior information in a wireless network into the concept of packet state. In this way, prior information is recovered by state association, and is then used to construct a longer preamble. This new preamble is in turn used to re-do the signal processing to obtain better frequency offset and CSI estimates, which is called signal re-processing in this chapter.

2.4 Architecture and Components

In this section, we turn the basic concept of STAPLE into detailed design. We first present the STAPLE architecture, and then elaborate its key components.

2.4.1 STAPLE Architecture

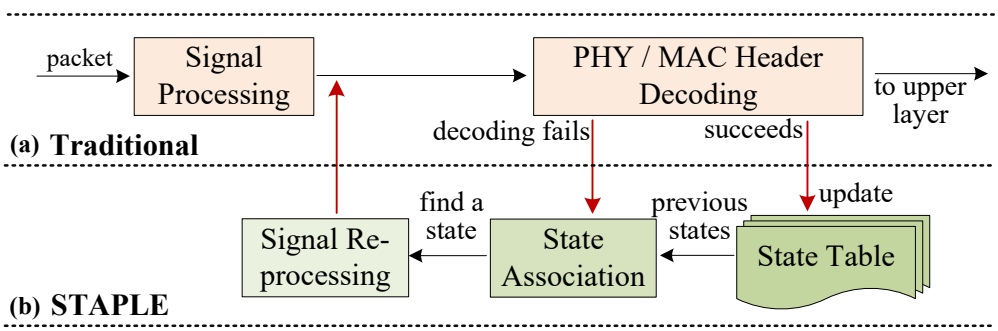


Figure 2.5. Architecture of STAPLE.

STAPLE can be regarded as a module (as shown in Figure 2.5(b)) flexibly attached to the traditional architecture of a wireless receiver (as shown in Figure 2.5(a)).

There are three key components in STAPLE: state association, signal re-processing, and state table management. To provide a generic framework, we decouple the design of STAPLE into two independent parts: (i) designing STAPLE components (i.e., state association and signal re-processing) and (ii) constructing the state of a packet from PHY/MAC fields. In the following, we describe the first part, which is standard-independent. The second part answers what PHY/MAC fields are used to construct the state of a packet and will be discussed based on wireless standards in Section 2.5.

2.4.2 State Association

State association is triggered when the decoding of a packet P fails. From Definition 1, we know that the state of a packet is the bit combination of several selected PHY/MAC fields. Thus, STAPLE can read the state of the packet $S(P)$ by combining the decoded bit values in certain PHY/MAC fields in P, as shown in Figure 2.6. Note that even upon failure of decoding an entire packet, the bit values in its PHY/MAC fields can still be decoded (with potential errors), and then fetched into state association for state look-up and comparison.

After obtaining $S(P)$, STAPLE associates it with a previous state in the state table, which stores a set of previous states, denoted by $\mathcal{S} = \{S_1, S_2, \cdots, S_{|\mathcal{S}|}\}$, where $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$.

As P is a corrupted packet, any bit in P can be erroneous. However, if we can find a state in $\mathcal{S}$ that resembles $S(P)$, it is very likely that $S(P)$ should be equal to that state, but may exhibit another value because of noise or interference during the reception of P. This is from our observations in packet trace analysis: a node is found to send packets to another node with limited variations of bit values in PHY/MAC headers, therefore exhibiting a limited

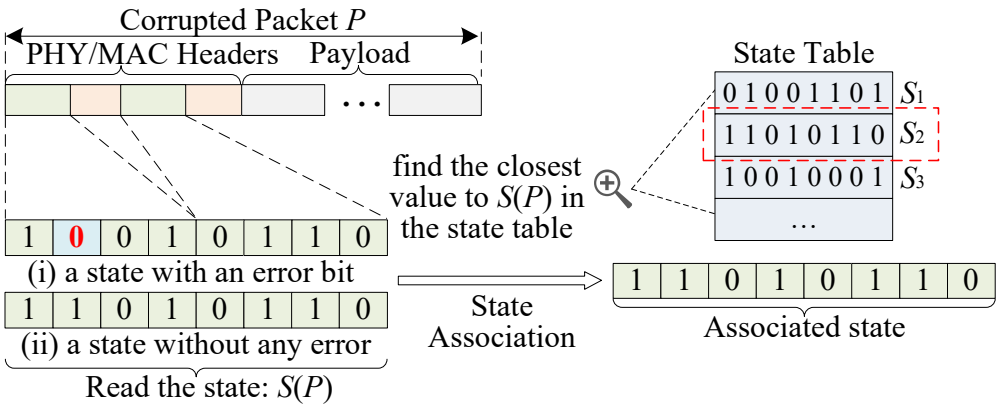


Figure 2.6. State association in STAPLE.

set of most frequent states. If these states are all stored in the table, the state of a corrupted packet is very likely stored in the state table.

As a result, the state association component in STAPLE finds the stored state closest to $S(P)$ in the state table. The measure of "being closest" is by the Hamming distance, which is defined as the number of bit differences between two bit sequences. Accordingly, the association algorithm can be written as

$$\text{Objective}: \quad \hat{S}(P) = \arg\min_{s \in \mathcal{S}} H(s, S(P)) \tag{2.2}$$

$$\text{Subject to}: \quad H(S(P), \hat{S}(P)) < d_{\text{th}}, \tag{2.3}$$

where $H(\cdot\,,\cdot)$ denotes the Hamming distance between two bit sequences, and the objective (2.2) is to re-associate the packet P with a new state $\hat{S}(P) \in \mathcal{S}$ that is the closest to the original state $S(P)$. A constraint (2.3) also exists to make sure that the Hamming distance between $\hat{S}(P)$ and $S(P)$ must be smaller than a threshold d_{th}. This is because the real state of P does not always exist in the state table. A large value of $H(S(P), \hat{S}(P))$ indicates that (i) packet P was transmitted with a completely different state that was never stored in the

state table; (ii) too many bit errors happened because of low signal quality. In both cases, STAPLE stops state association and simply drops the packet P.

It is also possible that the corrupted packet P is associated to a wrong state $\hat{S}(P)$, which can happen when the Hamming distances among some pairs of states stored in the table are very small. STAPLE makes best efforts to associate P with a previous state. When P is indeed associated to a wrong state, STAPLE is not likely to recover P by signal re-processing. But this only reduces the gain from STAPLE and does not degrade the performance as P is already corrupted. Wrong association can be mitigated by carefully choosing PHY/MAC fields to construct the state of a packet, which is detailed in Section 2.5. A smaller threshold d_{th} can also mitigate wrong association from happening. We will evaluate the impacts of d_{th} in experiments in Section 2.6.

Figure 2.6 shows two simple examples of how state association works: (i) when a corrupted packet P has a state with one error bit $S(P) = 10010110$, state association is performed based on (2.2) with threshold $d_{\text{th}} = 2$ to obtain $\hat{S}(P) = S_2 = 11010110$ because their Hamming distance $H(S(P), S_2) = 1$. This indicates that the second bit in $S(P)$ is likely an error, and is corrected from 0 to 1. Then, the state of P is associated to a new one $\hat{S}(P) = 11010110$. (ii) When P has a state without any error bit $S(P) = 11010110$, obviously $\hat{S}(P) = S(P) = S_2 = 11010110$ after state association.

When the header is coded in some standards (e.g., convolutional coding is used for 802.11g/n/ac headers), state association can be performed on the coded data and thus the Hamming distance comparison will not be affected by the coding scheme.

Note that the performance benefits of STAPLE do not come solely from error-correcting a part of the PHY/MAC header in a packet during state association. More importantly, STAPLE helps improve the performance when the payload, not the header, is decoded with errors. This is because the payload usually has a higher modulation/coding rate than the header, therefore requires more accurate frequency offset and CSI estimates that STAPLE

can provide based on a longer preamble formed by the conventional preamble and the state of a packet. Overall, state association serves as the first essential step for this generic framework to improve the signal processing performance.

2.4.3 Signal Re-processing

Given the corrupted packet P as the input, the output of state association is the associated state $\hat{S}(P)$. Then, STAPLE performs signal re-processing, which consists of two steps as shown in Figure 2.7.

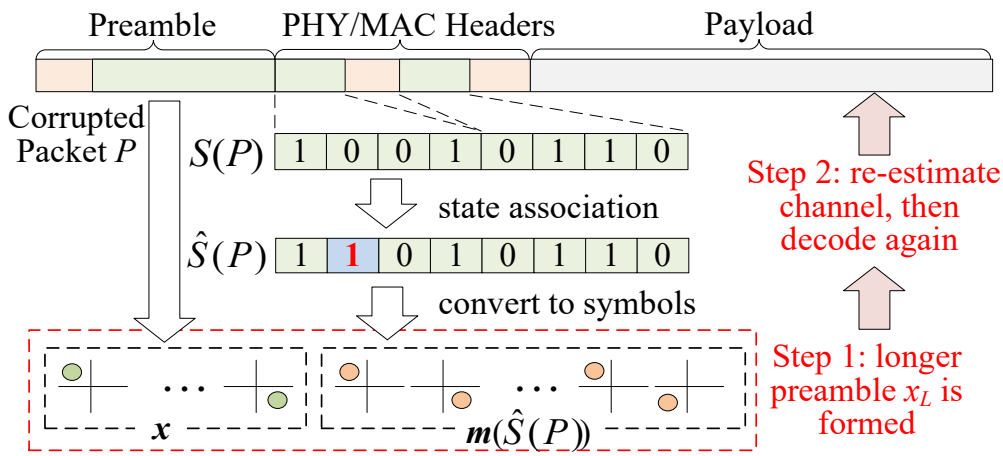


Figure 2.7. Signal re-processing in STAPLE.

First, STAPLE converts $\hat{S}(P)$ that represents the recovered bits from PHY/MAC fields in packet P to physical layer symbols, which is done by re-modulating $\hat{S}(P)$. If a header is interleaved or/and coded (e.g., 1/2 convolutional coding in WiFi signal), STAPLE re-interleaves or/and re-encodes, then re-modulates $\hat{S}(P)$. Denote by $\mathbf{m}(\hat{S}(P))$ the symbol vector of re-modulated symbols from $\hat{S}(P)$. The longer preamble is constructed as $\mathbf{x}_L = \left[\mathbf{x}\ \mathbf{m}(\hat{S}(P))\right]$, where $\mathbf{x}$ is the original preamble described in Section 2.2.2.

Second, given the newly constructed preamble $\mathbf{x}_L$, we can improve both frequency synchronization and channel estimation during signal processing, as shown in Figure 2.2: (i)

Frequency synchronization for low complexity processing must leverage a special repetitive structure (e.g., two same consecutive symbols in 802.11 a/g/n/ac) in the preamble. The longer preamble $\mathbf{x}_L$ does not generally result in such a repetitive structure, leading to a non-convex optimization problem in [1, 2]. (ii) In contrast, channel estimation is usually a linear process (as discussed in Section 2.2.2). A longer preamble in general results in a more accurate CSI estimate.

As performing signal re-processing requires a tradeoff between performance and complexity, and we wish to ensure low complexity in the STAPLE implementation, we choose to perform signal re-processing only based on channel estimation, and avoid solving the non-convex optimization to re-estimate the frequency offset at the receiver. Thus, given $\mathbf{x}_L$, STAPLE re-estimates the CSI as

$$\hat{h}_L = (\mathbf{x}_L^H \mathbf{x}_L)^{-1} \mathbf{x}_L^H \mathbf{s}'_L, \tag{2.4}$$

where $\mathbf{s}'_L$ is the vector combined by the received symbols from the original preamble part and from the PHY/MAC fields in the received signal of packet P at the receiver.

After obtaining a better CSI estimate $\hat{h}_L$, STAPLE re-demodulates and re-decodes the corrupted packet P in order to recover it. The implementation for this step can be optimized to minimize the processing complexity for a wireless standard. We will analyze the implementation complexity for 802.11ac in Section 2.6.2.1.

If there are pilot signals in the payload of the packet P (e.g., pilot subcarriers in 802.11 a/g/n/ac), because more information is obtained by STAPLE, it can yield a better initial channel estimate $\hat{h}_L$, which can be based on to track the channel change using these pilots [65] in a fast fading wireless channel environment.

Although $(\mathbf{x}_L^H \mathbf{x}_L)^{-1}$ is not a linear operation in (2.4), in OFDM equalization for many wireless standards, $\mathbf{x}_L^H \mathbf{x}_L$ is a complex number and $(\mathbf{x}_L^H \mathbf{x}_L)^{-1}$ only incurs a division operation

in the complex domain. In addition, the value of $(\mathbf{x}_L^H \mathbf{x}_L)^{-1}$ can always be pre-computed during state update and stored along with its associated state in the state table.

2.4.4 Table Management Policies

The state table (shown in Figure 2.5(b)) is used to store the states of successfully decoded packets. We design two table update policies.

1. Timestamp-based policy, which uses packet arrival time as an indicator to update the table. Each state is stored with a timestamp. Once the most recent packet is decoded successfully, STAPLE updates the state's timestamp if the state is already in the table, or otherwise replaces the oldest state with this new one.

2. Frequency-based policy, which stores a timestamp and a usage count with each state in the table. Similar to the previous policy, the timestamp is used to indicate the last update time of a state. Once a packet is decoded, STAPLE updates the state's timestamp and increments the usage count if the state is already in the table. Otherwise, STAPLE replaces the state that has the smallest count with the new state. If several states have the same count number, STAPLE chooses the oldest state for replacement. This policy also zeros out the count of a state if the state's timestamp is too old (i.e., larger than a given threshold).

When the table size is sufficiently large, the two policies should have no evident performance difference. However, if the table size is small, the downlink performance may not be affected significantly because a station, as the receiver, always receives data solely from the AP. The uplink performance of the station that transmits more data (than others) to the AP may be boosted because the state of its packets can always be stored in the AP's table. Therefore, the AP should choose a larger table size than a station in practice. The impacts

of state table parameter configurations and policy setups will be evaluated in experiments in Section 2.6.

2.5 Configuring the STAPLE System

We have designed the major components in STAPLE. Another key question left is which PHY/MAC fields constitute the state of a packet, which is standard-dependent. In this section, we configure STAPLE for 802.11a/b/g/n/ac and 802.15.4 by specifying how to construct the state of a packet in such networks.

2.5.1 Methodology for Configurations

The motivation to develop STAPLE is from the observation that there exist most frequently appeared values of PHY/MAC header fields in packets. Thus, we aim to select those fields exhibiting little randomness to form the state of a packet. In particular, we use Shannon entropy [78] per bit to quantitatively measure the randomness of each PHY/MAC field in packet trace data and choose those fields with low entropies. We choose the SIGCOMM04/08 datasets for 802.11b/g and our own datasets STAPLEn/STAPLEac for 802.11n/ac, respectively. These datasets can represent typical WiFi scenarios in a conference, a campus or a residential place.

STAPLEn is a dataset collected from a campus WiFi network and STAPLEac from a residential WiFi network. Unlike 802.11b/g packet capturing, it is not always possible for a third-party to correctly capture and decode all 802.11n/ac data packets because of beamforming between individual stations and the AP [79]. Hence, we place four laptops at different locations and capture packets individually in the networks. We then aggregate all packets to measure the entropy of each header field. We find that the 802.11ac network did not run in the multi-user MIMO (MU-MIMO) mode because the GroupID field in all collected

packets was set to 0. Thus, our collection and measurement can reflect how PHY/MAC fields in packets are set up for routine WiFi use in a non-MU-MIMO 802.11ac network.

Note that wireless standards adopt different coding schemes for the PHY/MAC headers, which can affect the complexity of the signal re-processing component in STAPLE. We categorize a coding scheme into one of three types: plaintext (P), interleaving and coding (I/C) data, and encryption (E). Type-P indicates no coding is used for a header field (e.g., PHY/MAC headers of 802.11b); Type-I/C means that data fields in a header are interleaved and error-correction coded (e.g., MAC headers of 802.11a/g/n/ac); Type-E indicates that data fields in a header are all encrypted (e.g., MAC headers of 802.15.4). Generally, we avoid choosing Type-E fields and select Type-P or Type-I/C fields based on their entropy values for our implementation.

Constructing the state of a packet is not unique. In our STAPLE configurations for 802.11a/b/g/n/ac and 802.15.4, we target a lightweight design that balances between performance benefits and costs of storage and computation in typical wireless networks with normal usages. It is certainly feasible to add additional fields to (or remove existing fields from) our configurations to optimize STAPLE along certain direction.

Because a newer 802.11 standard usually extends and relies on previous ones, we configure STAPLE starting from 802.11b (legacy) to 802.11ac (state-of-the-art) for the sake of clear presentation. Our implementation and evaluation are focused on 802.11ac in Section 2.6. We also minimize the description of a wireless standard, which is well documented.

2.5.2 Configurations for Wireless Standards

2.5.2.1 Configurations for 802.11b

We start from 802.11b, which is a legacy WiFi standard but still backward supported in many WiFi networks. Figure 2.8 shows a typical structure of physical protocol data unit

(PPDU) for 802.11b, including the PHY Layer Convergence Procedure (PLCP) preamble, PLCP header, and PLCP Service Data Unit (PSDU) that contains the MAC header and the MAC payload.

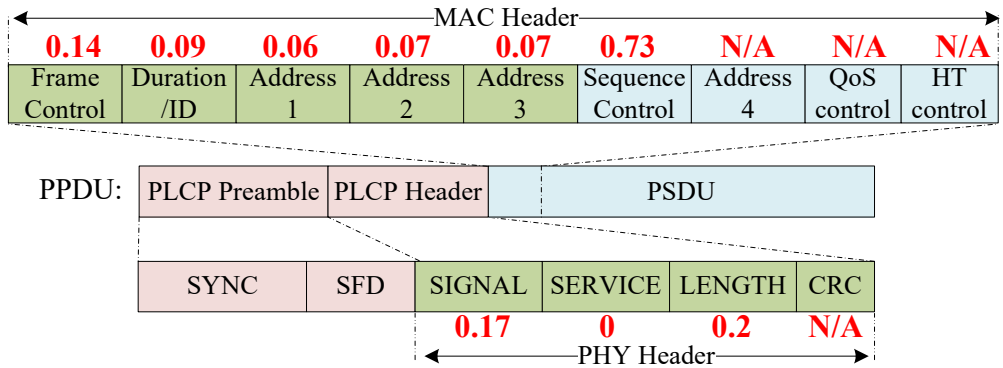


Figure 2.8. Typical format of an 802.11b data packet and measured entropies values.

As shown in Figure 2.8, the PLCP header represents the PHY header including four fields of Type-P: SIGNAL, SERVICE, LENGTH, and CRC, where SIGNAL denotes the modulation scheme, LENGTH is the payload length, SERVICE contains three bits to represent the PHY configuration and the rest of bits are reserved and default to 0. The 802.11b MAC header at the beginning of PSDU consists of a set of fields.

According to the measured entropy in each PHY/MAC field in Figure 2.8, we choose SIGNAL, SERVICE, LENGTH at the PHY header and Frame Control, Duration/ID, Addr.1, Addr.2, and Addr.3 in the MAC header to form the state of a packet because of their low entropies. We also include CRC in the PHY header because CRC directly depends on other fields. Once these fields are known, we can simply calculate the CRC. Including CRC increases the length of the state of a packet, and accordingly can further improve the signal processing reliability because more data is used for signal processing. Thus, for a packet P, its state $S(P)$ is written as $S(P) =$ SIGNAL|SERVICE|LENGTH|CRC|Frame Control|Duration/ID|Addr.1|Addr.2|Addr.3 in our STAPLE configuration for 802.11b.

2.5.2.2 Configurations for 802.11a/g

802.11a/g uses OFDM to combat multipath fading. Figure 2.9 shows a typical format of the PPDU including the PLCP preamble, PLCP header, PSDU and others. Different from 802.11b, all the data except for the PLCP preamble is of Type-I/C in an 802.11a/g packet.

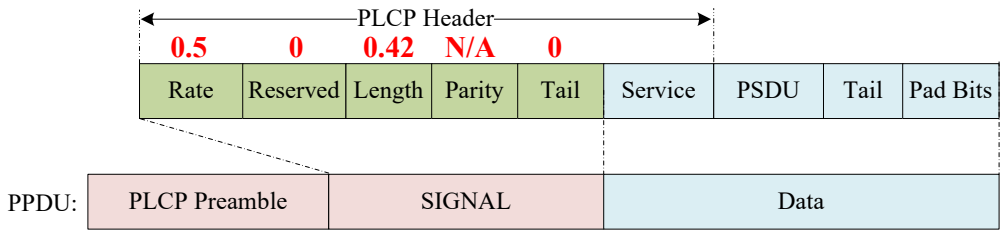


Figure 2.9. Typical structure of an 802.11a/g packet and measured entropies in SIGNAL.

The PLCP header includes the SIGNAL field that occupies one single OFDM symbol and the Service field that exists in the next symbol. SIGNAL contains a number of fields of Type-I/C to represent packet information like the data rate and packet length. We measure the entropy of each field in SIGNAL, also shown in Figure 2.9. All entropies are small therefore we include the entire SIGNAL field in the state of a packet. Note that the Parity field can be directly computed given other fields, therefore we do not need to measure its entropy.

The Service field is used to synchronize the descrambler shown in Figure 2.9. All bits in Service are constantly set to 0. The MAC header at the beginning of PSDU follows Service. The 802.11a/g MAC header structure is exactly the same as that of 802.11b. However, Service and the MAC header together do not exactly occupy one OFDM symbol. This means that they are interleaved and coded with other data then OFDM-modulated, which can be difficult (if not impossible) for STAPLE's signal re-processing to accurately extract Service and parts of the MAC header from interleaved and coded OFDM symbols. Therefore, Service and all MAC fields can be used for state association, but not for signal re-processing in STAPLE. As we aim at a lightweight design, we avoid using any of these fields to reduce the

size of the state table. Therefore, the state $S(P)$ for packet P in our STAPLE configuration for 802.11a/g is $S(P) = \mathsf{SIGNAL}$.

2.5.2.3 Configurations for 802.11n/ac

802.11n and 802.11ac are the state-of-the-art WiFi standards leveraging MIMO technologies with more complicated packet formats.

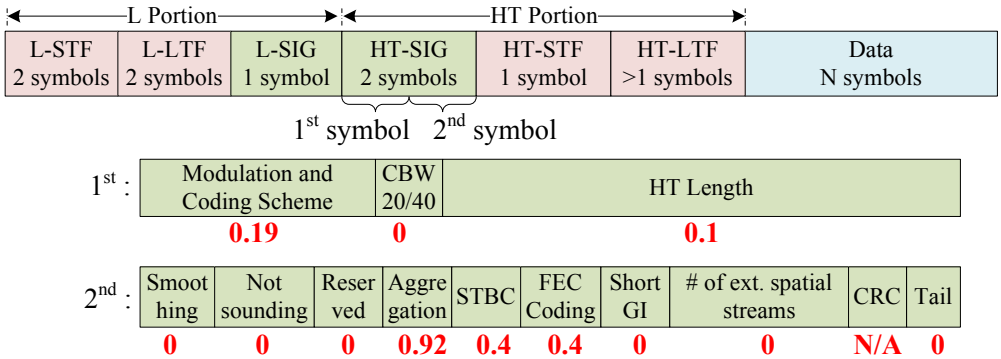


Figure 2.10. Typical packet format in 802.11n and measured entropies of field values.

Figure 2.10 depicts a typical packet structure of 802.11n which contains two portions: legacy (L) and high throughput (HT). The L portion is for backward compatibility, and the HT portion is specified to support MIMO communication. The L-SFT and L-LFT fields at the beginning of the L portion are legacy preambles. The HT-SFT and HT-LFT fields in the HT portion are preambles for MIMO training. The L-SIG of Type-I/C is exactly the same as the SIGNAL field in 802.11a/g. Hence, we use the entire L-SIG field for STAPLE. HT-SIG of Type-I/C is at the beginning of the HT portion and consists of 2 OFDM symbols with a number of fields. We measure the entropy of each field from packet trace data, as shown in Figure 2.10. Most fields have small entropies, therefore we include the entire HT-SIG field. Note that the single-bit Aggregation field indicates whether multiple payloads are combined together into a single packet with a unique PHY header, and it is almost random with entropy close to 1. However, we cannot exclude this bit because it is interleaved and coded

with other fields in exactly one OFDM symbol. We find that the high entropy of this single bit always happens in the downlink. It does not have substantial impact on the performance but the size of the state table needs to be slightly increased at the downlink. Therefore, we still include it to represent the state of a packet. As a result, the state $S(P)$ for packet P in our configuration for 802.11n is $S(P) = \mathsf{L\text{-}SIG|HT\text{-}SIG}$.

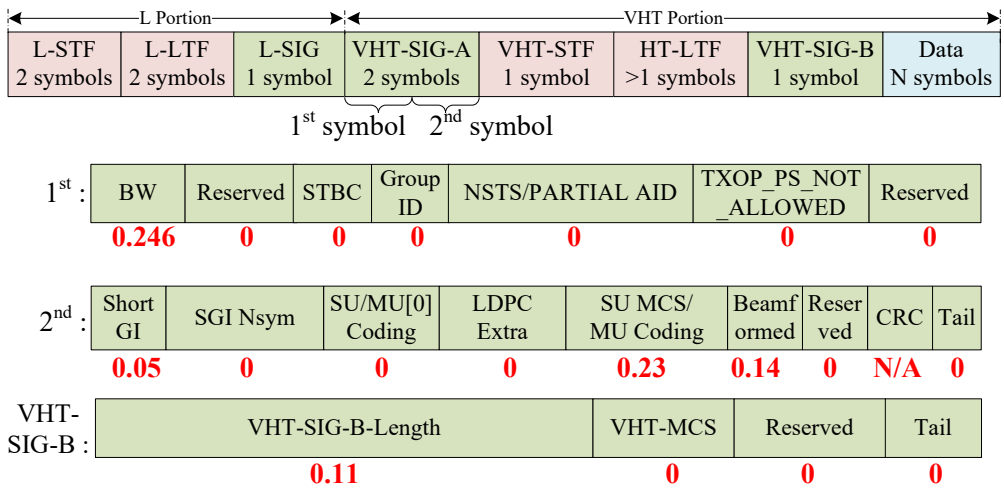


Figure 2.11. Typical packet format in 802.11ac and measured entropies of field values

Figure 2.11 plots the typical packet format in 802.11ac, which has a very similar structure to 802.11n: the HT portion becomes the very HT (VHT) portion. Similarly, based on the measured entropies from packet trace data shown in Figure 2.11, we include L-SIG and VHT-SIG to represent the state of a packet. In addition, we find the entropies of fields in the VHT-SIG-B are also small. Thus, we set the state as $S(P) = \mathsf{L\text{-}SIG|VHT\text{-}SIG|VHT\text{-}SIG\text{-}B}$ in our configuration for 802.11ac.

2.5.2.4 Configurations for 802.15.4

IEEE 802.15.4 is the basis for ZigBee towards low-cost, low-speed ubiquitous communication. We configure STAPLE for offset-QPSK (O-QPSK) based ZigBee, whose typical packet format is shown in Figure 2.12.

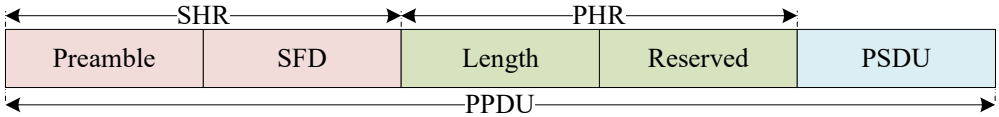


Figure 2.12. Typical 802.15.4 packet format.

As shown in Figure 2.12, the PHY header (PHR) of 802.15.4 contains only two Type-P fields: Length and Reserved, which indicate the length of the packet and reserved bits (set to all 0s), respectively. The MAC header is contained in PSDU, which is however of Type-E. To reduce the complexity, our STAPLE configuration avoids any Type-E data during signal re-processing, therefore choosing to include only Length and Reserved for the state of a packet P, i.e., $S(P) = \mathsf{Length}|\mathsf{Reserved}$.

2.6 Experimental Evaluation

In this section, we implement STAPLE with adapted configurations for 802.11b/g/n/ac and 802.15.4. We first describe our implementation and experiment setups, then focus on presenting experimental evaluation of STAPLE in 802.11ac, and finally describe experimental results on other standards.

2.6.1 Implementation and Setups

2.6.1.1 Platform and Implementation

STAPLE requires modifications of a conventional wireless receiver. We choose the X300 USRP with CBX daughterboards [80] as the implementation platform. We synchronize multiple USRPs using OctoClock-G [80] when performing MIMO experiments with 2 - 8 antennas.

We implement a generic software radio platform that covers the basic designs of 802.15.4 and 802.11b/g/n/ac transceivers. We adopt BPSK, QPSK, and 16QAM for 802.11b/g/n/ac,

and O-QPSK for 802.15.4. For 802.11n/ac, we implement the Alamouti code [81] based MIMO transmission schemes. In 802.11g/n/ac implementations, the header and data payload of a packet use 1/2 and 3/4 convolutional coding schemes, respectively. We also implement a CSMA/CA scheme with uniform random backoff (i.e., the contention window for random backoff is always fixed [82, 83]) at the MAC layer for all standards. Note that it is challenging and time-consuming to implement a comprehensive USRP-based system fully compatible with all 802.11 PHY/MAC standards. To balance the efficiency of fast prototyping and the cost of high-fidelity evaluation, our implementation is a proof of concept one with a subset of fundamental STAPLE-related PHY functionalities, serving the purpose of evaluating the benefits of STAPLE brought to practical wireless scenarios with different system setups and conditions. As mentioned in Section 2.4.3, our STAPLE implementation does not improve frequency offset estimation but improves the channel estimation in signal re-processing.

Also note that we implement a subset of modulation and coding schemes. However, our experiments generate packets according to packet trace data. This means the PHY field that denotes the data rate in some generated packets may indicate a higher rate scheme that we do not implement. In this case, we choose 16QAM as the scheme to modulate and demodulate. This does not give an advantage to STAPLE during performance evaluation, since we still allow all possible values in the data rate field of a packet, which affect state association and state table update of STAPLE in practice.

2.6.1.2 Experimental Setups

We conduct experiments in a realistic indoor environment shown in Figure 2.13 as the scenario reflects how STAPLE can help improve the wireless link performance in an office environment with typical WiFi usages. Network nodes are placed at different locations for performance evaluation with limited transmit power. In our network experiments, packets

are generated with PHY/MAC field values according to the SIGCOMM04/08 and STAPLEn/ac datasets.

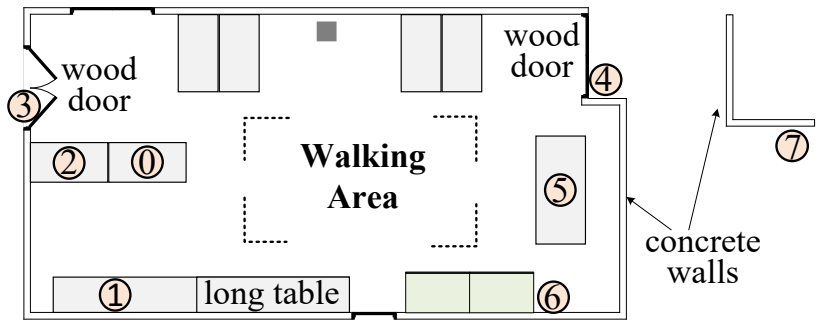


Figure 2.13. Environment for experiments.

In STAPLE, the default threshold for state association (2.2) is set to 12. The default size of the state table for a station in the downlink is set to 5; and that for the AP in the uplink is 15. The default table update policy is frequency-based. In experiments, we change table setups to evaluate the impact of STAPLE parameters on the performance. We use the construction of the packet state for each standard given in Section 2.5.

2.6.1.3 Performance Metrics

Packet dropping due to decoding errors only happens at the PHY layer and STAPLE is a method to improve the PHY layer performance. Thus, it is natural to use the performance metric at the PHY layer to directly see the benefits of STAPLE without involving other potential affecting factors at higher layers. In experiments, we measure the packet delivery ratio (PDR) with and without STAPLE, where PDR is defined as the ratio of the number of successfully decoded PHY packets at a node to the total number of PHY packets transmitted to the node. We measure the benefit of STAPLE by using the performance gain ratio, which

is defined as

$$\text{Performance gain ratio} = \frac{\text{PDR with STAPLE}}{\text{PDR with traditional design}} - 1.$$

2.6.2 STAPLE for 802.11ac Networks

We focus on performance evaluation on 802.11ac because it is a state-of-the-art WiFi standard. We consider two types of scenarios in experiments: single-link evaluation, in which a two-antenna transmitter sends packets to a receiver with $2-8$ antennas; (ii) network evaluation, in which we set up two clock-synchronized USRPs as a single 802.11ac AP, and six USRPs as stations communicating with the AP.

2.6.2.1 Implementation Complexity for 802.11ac

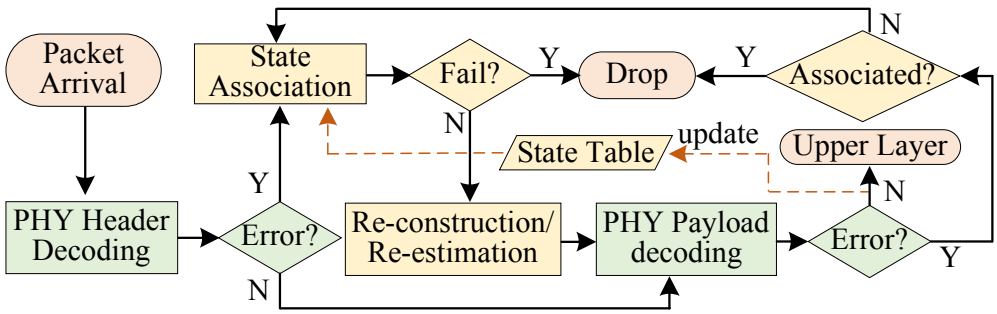


Figure 2.14. Implementation of 802.11ac.

Figure 2.14 shows the flowchart of our STAPLE implementation for 802.11ac. the extra storage due to STAPLE is mainly the memory used for the state table, whose size is 5 states for stations and 15 states for the AP as default values. In terms of extra processing complexity, if errors are detected in the PHY layer, extra procedures including state association, preamble reconstruction, and CSI re-estimation are introduced by STAPLE. All these procedures are linear, which will not incur much overhead. If errors occur in the data payload,

re-decoding the payload is needed. The re-decoding complexity is not linear and depends on the decoding algorithm that is usually polynomial. However, thanks to the PHY payload check, this case does not quite often happen in today's WiFi networks.

Table 2.1. Complexity of STAPLE for 802.11ac.

Situation	**Extra processing**
1) No error in a packet	none
2) Error first detected in PHY header	state association, preamble reconstruction and CSI re-estimation
3) Error not detected in header but payload	extra processing in 2), and re-decoding of PHY payload

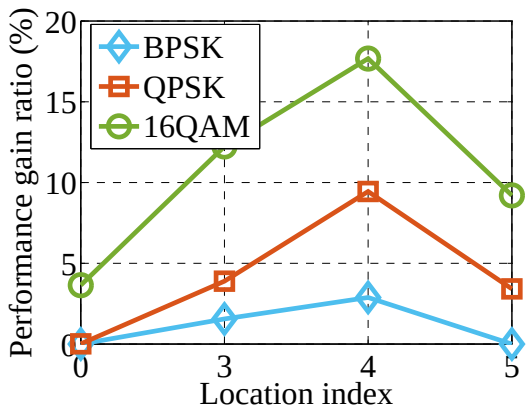


Figure 2.15. Performance gain ratios under different locations.

2.6.2.2 Single-Link Performance Evaluation

In our single-link performance evaluation, we fix the modulation scheme for packets and measure the link performance between two nodes to evaluate how STAPLE helps improve the single-link performance.

For varying locations, we first compare the single-link performance between a traditional receiver and STAPLE at different locations. We fix the transmitter at location 1 as shown in Figure 2.13, and place the receiver at location 0, 3, 4, or 5, which represents the short-distance

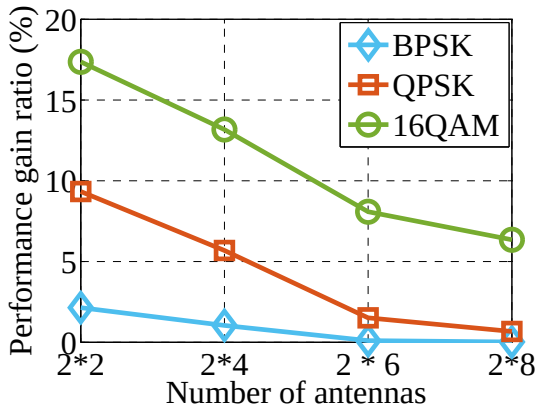


Figure 2.16. Performance gain ratios of uplink and downlink.

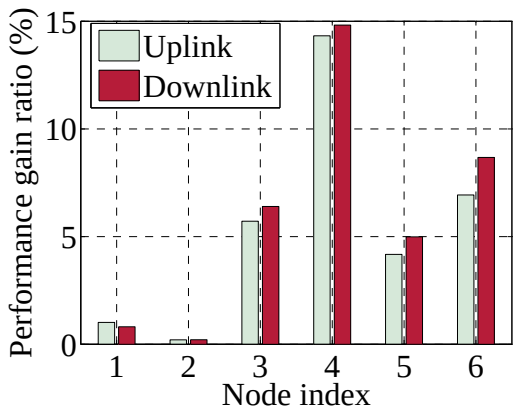


Figure 2.17. Performance gain ratios of STAPLE for 802.11ac.

line of sight (S-LoS), short-distance Non-LoS (S-NLoS), long-distance NLoS (L-NLoS), or long-distance LoS (L-LoS) channel condition, respectively. The transmitter and receiver are both equipped with 2 antennas.

Figure 2.15 depicts the performance gain ratios at different locations under BPSK, QPSK, and 16QAM. We can observe that the performance of STAPLE is uniformly better than the performance of traditional signal reception for most cases. There is no performance improvement at locations 0 and 5 (both LoS cases) for the BPSK modulation because the packet delivery ratio is 100%. We also find that at location 4 (L-NLoS), STAPLE substantially

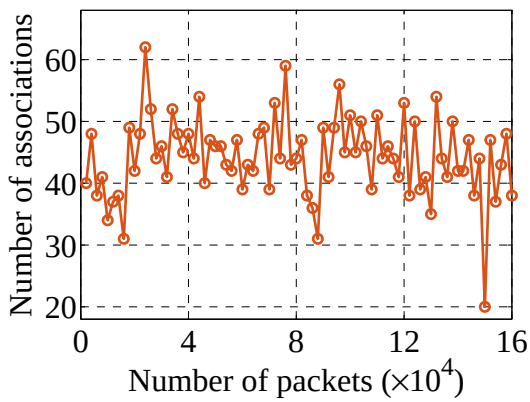


Figure 2.18. No. of state associations for every 2000 packets.

Table 2.2. Packet delivery ratios with different packet lengths (in bytes).

	10	**100**	**500**	**1000**	**1500**
Traditional	99.9%	99.9%	98.3%	93.5%	91.5%
STAPLE	100%	100%	99.9%	99.7%	98.3%

improves the packet delivery ratio for 16QAM from 80% to 95% (a performance gain ratio of 17.6% shown in Figure 2.15). This is because communication with higher data rate is less error-tolerant, thus demanding more accurate CSI estimation, which STAPLE provides. We can conclude from Figure 2.15 that STAPLE improves the performance in NLoS cases more than that in LoS cases.

For varying packet lengths, we then evaluate the effect of packet length on the performance gain of STAPLE. During this experiment, the transmitter always sends packets with a fixed length. Table 2.2 shows the packet delivery ratios on a 2×4 MIMO link from location 1 to location 5 with QPSK modulation for varying packet lengths. It is noted from Table 2.2 that transmitting shorter packets results in better packet delivery ratios. For example, traditional signal reception and STAPLE achieve almost 100% packet delivery ratio when the packet length is no larger than 100 bytes. As the packet length increases, STAPLE gradually

Table 2.3. Performance gain ratios from STAPLE under noisy MIMO channels at location 7.

	BPSK	**QPSK**	**16QAM**
2×4	20.2%	N/A	N/A
2×6	16.6%	20.4%	N/A
2×8	11.4%	18.9%	20.8%

outperforms traditional signal reception, and improves the packet delivery ratio from 91.5% to 98.3% (a performance gain ratio of 7.4%) when the length is 1500 bytes.

For varying numbers of antennas, we also measure the performance benefits from STAPLE for 2×2–8 MIMO links. Figure 2.16 plots the performance gain ratios of STAPLE on 2×2, 2×4, 2×6, and 2×8 MIMO links under different modulation schemes. The transmitter and the receiver are placed at locations 1 and 4, respectively, as shown in Figure 2.13. We can observe from Figure 2.16 that STAPLE generally improves the performance more as the number of antennas decreases. For example, under 16QAM, STAPLE achieves a 17.6% performance gain ratio in the 2×2 scenario and a 6.0% ratio for the 2×8 scenario. Similarly, the performance gain ratio for QPSK reduces from 9.4% to only 0.6% when the number of receive antennas increases from 2 to 8. This is due to more link reliability from more antennas.

We then move the receiver to location 7 in Figure 2.13 to test the performance in an L-NLoS MIMO channel. Table 2.3 shows the performance gain ratios of STAPLE on 2×4, 2×6, and 2×8 MIMO links. We find that location 7 is beyond the limit of the 2×4 MIMO link under QPSK and 16QAM, and beyond the limit of the 2×6 MIMO link under 16QAM. In these cases, the receiver cannot reliably decode any packet with or without STAPLE. In other cases, we always observe that STAPLE substantially improves the link reliability, with the maximum gain ratio achieved at 20.8% for the 2×8 MIMO link under 16QAM. Consequently, we conclude that STAPLE brings more benefits in noisy channel environments.

2.6.2.3 Network Performance Evaluation

Next, we evaluate the performance gain of STAPLE in an 802.11ac network environment. In our testing network, the AP and stations are equipped with 4 and 2 antennas, respectively. In the single-link evaluation, we always fix the modulation scheme for a packet to test the performance of a single link. In our network performance evaluation, all the values in the PHY/MAC fields of a packet (e.g. packet size and modulation scheme) are generated according to 802.11ac packet trace data. This creates a much more dynamic environment especially in uplink scenarios. We place the AP at location 0, and also place other stations, named nodes 1–6, at locations 1–6, respectively, as illustrated in Figure 2.13.

For the overall performance, Figure 2.17 shows the overall performance gain ratios under uplink and downlink scenarios. Similar to the single-link case, STAPLE always improves the packet delivery ratio. We find in Figure 2.17 that there are very slight performance gains for nodes 1 and 2. This is because these two nodes have very good LoS channels to the AP and corrupted packet at these nodes are mostly due to collisions in the network. STAPLE is not a mechanism to resolve packet collisions. If there is no collision, the single-link packet delivery ratios for nodes 1 and 2 are close to 100%. Thus, there is little room for STAPLE to improve the performance.

As Figure 2.17 illustrates, the maximum performance gain ratios for the uplink and downlink are 14.3% and 14.8%, respectively, both observed at node 4, which is the furthest away from the AP in the network. This shows the capability of STAPLE to boost the link performance under noisy conditions.

We also demonstrate the dynamics of state associations inside STAPLE at the AP. We compute the number of successful state associations for every 2000 packets received. Figure 2.18 plots this number as a function of the total number of received packets at the AP. It is seen that most of the time, STAPLE needs to perform signal re-processing on 30-60

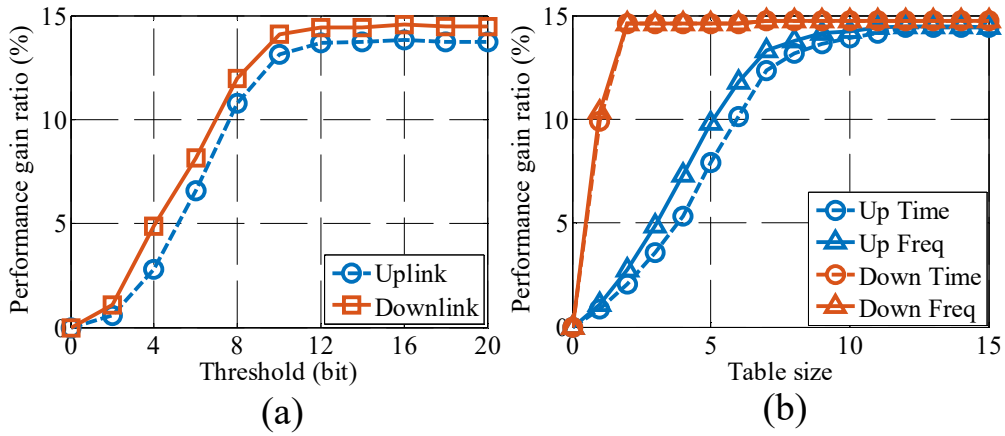


Figure 2.19. The impacts of (a) threshold in state association and (b) state table size.

packets for every 2000 packets, and packet corruption events are uniformly distributed over time in our testing environment.

For the impacts of threshold in state association, in state association (2.2), a pre-set threshold d_{th} is needed to associate a corrupted packet with a previous state. We test the impact of different thresholds on the performance gain ratio of STAPLE at node 4. Figure 2.19(a) illustrates the uplink and downlink performance gain ratios with different thresholds. We can observe that the performance gain ratio of STAPLE first increases then remains approximately the same as the threshold d_{th} keeps increasing. This means that a larger d_{th} is desirable to maximize the performance gain from STAPLE in practice. But a larger d_{th} may indicate that STAPLE tries to recover heavily corrupted packets that may be not recoverable with signal re-processing, incurring more wrong associations and processing overhead. Therefore, the optimal value of d_{th} for STAPLE to balance the performance gain and complexity is 12-20 bits, as observed in Figure 2.19(a).

For the impacts of state table setups, we also evaluate the impacts of the state table size and update policy on the performance gain of STAPLE at node 4 in the network.

Figure 2.19(b) shows the comparison of performance gain ratios of timestamp-based and frequency-based table update policies for different table sizes. From Figure 2.19(b), the frequency-based policy exhibits slightly better performance than the timestamp-based policy. The difference is negligible for both uplink and downlink when the table size is large.

We also see from Figure 2.19(b) that the state table size has more impacts on the performance benefits of STAPLE. For the downlink, as a station only cares about the packets from the AP, a small table maintaining 2-5 states is sufficient to increase the performance gain ratio to approximately 15%. For the uplink, the AP requires a larger table size because it has to communicate with a number of nodes. Figure 2.19(b) shows that a table with the size of 12-15 states at the AP is sufficient for the testing network to achieve a nearly 15% performance gain ratio.

2.6.3 STAPLE for Other Standards

During our experiments with 802.11ac, we find that STAPLE always improves the packet delivery by up to 20.8% and 15% in single-link and network evaluations, respectively. In the following, we show the benefits of STAPLE brought to 802.11b/g/n and 802.15.4.

- For 802.11b/g/n network setups, we use the previous network scenario and default STAPLE parameters to set up the 802.11b/g/n testing network with the AP placed at location 0, and nodes 1-6 placed at locations 1-6, respectively, as shown in Figure 2.13.

- For 802.15.4 network setups, the network topology is the same as the one in 802.11 scenarios: the network coordinator is placed at location 0, and nodes 1-6 placed at locations 1-6, respectively. Nodes 1-6 are communicating with the coordinator in the network. We obtain two ZigBee datasets [84] and [85]. The packets in [84] have 7 different lengths and packets in [85] always have the same length. We generate data packets according to [84]. The state table size is set to 5 for all nodes.

For experimental results, Figure 2.20 shows the performance gain ratios of STAPLE in the uplink and downlink scenarios, respectively. Similar to Figure 2.17, nodes 1 and 2 gain slight benefits from STAPLE because of their good LoS channels to the AP; and node 4 has the highest performance gain ratios among all nodes.

It is observed at node 4 that the highest performance gain ratio (15.9% in uplink or 16.6% in downlink) is achieved under 802.11b and the lowest (4.9% in uplink or 5.8% in downlink) is under 802.15.4. This is because the packet state sizes in 802.11b and 802.15.4 are the largest and smallest in our STAPLE configurations (in Section 2.5.2), respectively. A large packet state size means that STAPLE can construct a longer preamble for signal re-processing, therefore generally indicating better performance gain. The experimental results from Figure 2.20 also demonstrate that STAPLE can benefit a wide range of wireless networks.

2.6.4 Discussions and Limitations

2.6.4.1 Discussions

In our performance evaluation, the downlink performance is found always slightly better than the uplink performance. This is because all nodes in the network transmit packets with limited power, leading to a minor hidden terminal problem. For example, as shown in Figure 2.13, with the AP being placed at location 0, node 1 (at location 1) has a small probability of not accurately sensing the transmission from node 4 (at location 4), and vice versa. This only results in minor asymmetric uplink and downlink performance, and does not affect the evaluation of the performance gains from STAPLE.

During our experiments, the size of the state table in STAPLE is chosen and evaluated empirically according to the distributions of PHY/MAC header fields in real-world 802.11 datasets. Therefore, the choice of 2-5 states for stations and 12-20 for the AP should be

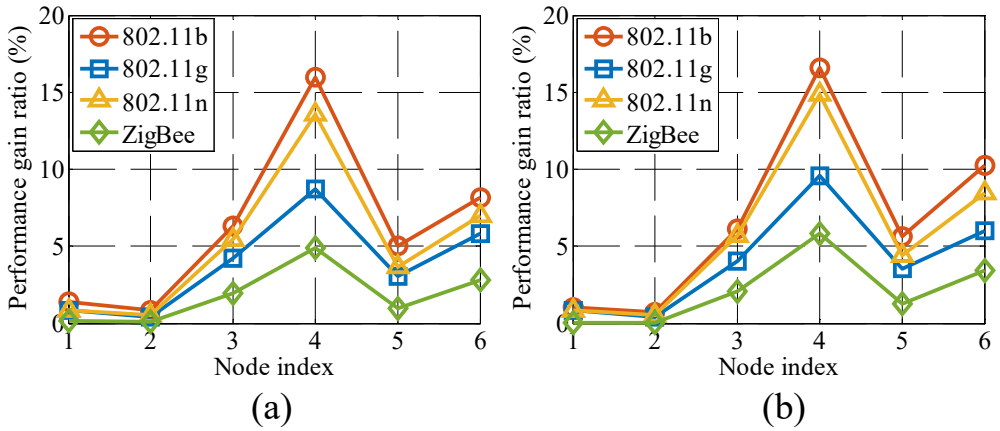


Figure 2.20. Network-level performance gain ratios: (a) uplink and (b) downlink.

practically suitable for a normal WiFi network environment. The table size may have to be increased in a network with a large number of users using data-intensive applications. An under-designed table size may only reduce the performance gain of STAPLE, but does not degrade the network performance. In addition, the advantage of STAPLE is that the downlink requires much smaller table size than the uplink, which significantly benefits network users.

For 802.11ac, our packet collection and evaluation are based on the non-MU-MIMO mode. The MU-MIMO mode does not affect the uplink and is for the downlink only [11]. In addition, MU-MIMO related PHY fields (e.g., GroupID) usually remain constant if a user does not change his/her location dramatically. Thus, a low-overhead STAPLE design is expected to be still effective when used for the MU-MIMO mode in 802.11ac.

2.6.4.2 Limitations

In this chapter, configurations for STAPLE are based on packet trace analysis for typical wireless network scenarios (e.g., a typical residential WiFi network). When a network has a large number of users, the size of the state table should be increased at least for the uplink

in order to accommodate more users' packet information. As a result, configurations for STAPLE may need to be adapted accordingly for the scenarios with a large number of users and intensive network usages.

STAPLE can be considered as a way to push the performance limit of signal processing for wireless networking. Experimental results show that overall, STAPLE achieves moderate performance improvements in different evaluation scenarios. Thus, it is important to minimize the complexity of the STAPLE implementation and configuration for a practical scenario to avoid excessive energy overhead, which is proportional to the processing complexity.

STAPLE is designed as a general software radio architecture to improve the signal processing performance on a wireless link. STAPLE is shown effective to improve the link performance under long-distance or severe channel fading scenarios. However, STAPLE cannot resolve any packet corruption due to collisions in wireless networks. This may limit its use in the scenarios with congested network traffic under good channel conditions, where the wireless packet collisions are the dominate performance bottleneck.

2.7 Related Work

In this section, we discuss existing works related to the research efforts in this chapter.

2.7.1 Wireless Signal Processing

Timing/frequency synchronization and channel estimation have been well explored in the signal processing and wireless communication communities under a wide range of algorithmic settings and channel conditions [2, 1, 4, 73, 86, 77, 87], thereby establishing a relatively mature research area. Essential signal processing procedures for traditional packet reception remain generally unchanged. STAPLE takes signal processing out of the traditional domain and places it in the network domain, then demonstrates a new perspective of re-designing

the signal processing procedure to improve the performance during packet reception at a wireless receiver.

2.7.2 Partial Packet Recovery

STAPLE is related to partial packet recovery (PPR) that attempts to recover a corrupted packet using a number of approaches, such as automatic repeat request (ARQ), forward error correction (FEC) [88, 14, 8, 10, 89, 72, 7, 9, 6, 90]. In particular, hybrid-ARQ (HARQ) [72, 7, 9] has become a widely-used technique to combine the information from multiple packets to improve the wireless link performance. STAPLE differs from H-ARQ in the following aspects: (i) H-ARQ focuses on repairing corrupted packets with the cooperation from the transmitter. STAPLE is designed as a new architecture of signal processing only residing at the receiver without any cooperation at the receiver; (ii) H-ARQ combines the retransmitted packet (including the redundant code) with the originally corrupted packet for error correction to help decoding. On the contrary, STAPLE does not directly touch upon signal decoding, but leverages low-entropy packet header information to construct a longer preamble to improve the signal processing accuracy, which in turn enhances the decoding performance.

To the best of our knowledge, STAPLE is designed as a new way to provide better signal processing performance, which is orthogonal to H-ARQ and other existing partial packet recovery mechanisms. Therefore, STAPLE can be integrated with these mechanisms to further improve the wireless network performance.

2.7.3 Link quality improvement

Research efforts [16, 91, 70, 90, 92, 93] have been devoted to improving the wireless link quality. For example, [16, 91] focus on resolving the problem of packets collisions. Channel prediction is proposed in [94] to facilitate rate selection in 802.11 networks. An architecture

is designed in [95] to flexibly support multiple radio frequency link chains. In addition, a few recent efforts focus on improving the MU-MIMO performance from various aspects [96, 68, 97, 98]. STAPLE is also related to [70] that leverages protocol signatures to improve the packet decoding performance based on modifications to the 802.11 PHY layer.

In contrast, STAPLE focuses on a different domain to improve the signal processing performance during packet reception and does not modify any standard. Hence, STAPLE is also complementary to these efforts and is broadly applicable to wireless networks.

2.8 Summary

This chapter presents STAPLE, which is a lightweight, efficient mechanism to improve the signal processing performance for wireless networking. The core idea of STAPLE is to collect common header information to form the state of a packet, and use state association to recover a corrupted packet back to a previous state for constructing a longer preamble to further enhance the reliability of signal processing. We implement STAPLE on USRP X300 devices with adapted configurations for 802.11a/b/g/n/ac and 802.15.4. Experimental results show that STAPLE improves the wireless network performance under various conditions.

Chapter 3: Comb Decoding towards Collision-Free WiFi

3.1 Abstract

Packet collisions happen every day in WiFi networks. RTS/CTS is a widely-used approach to reduce the cost of collisions of long data packets as well as combat the hidden terminal problem. In this chapter, we present a new design called comb decoding (CombDec) to efficiently resolve RTS collisions without changing the 802.11 standard. We observe that an RTS payload, when treated as a vector in a vector space, exhibits a comb-like distribution; i.e., a limited number of vectors are much more likely to be used than the others due to RTS payload construction and firmware design. This enables us to reformulate RTS collision resolution as a sparse recovery problem. We create algorithms that carefully construct the search range for sparse recovery, making the complexity feasible for system design and implementation. Experimental results show that CombDec boosts the WiFi throughput by 33.6% – 46.2% in various evaluation scenarios.

3.2 Motivation and Design Intuition

In this section, we introduce the motivation and key idea of reformulating the problem of packet collisions.

3.2.1 Packet Collision and Resolution

We use a noise-free, flat-fading uplink scenario as a simple motivating example: Alice and Bob send their packets to the AP at the same time. Alice's and Bob's packets consist of

L time-domain baseband symbols, represented by vectors $\mathbf{x}_A \in \mathcal{X}$ and $\mathbf{x}_B \in \mathcal{X}$, respectively, where $\mathcal{X} \subset \mathcal{C}^{L\times 1}$ denotes the set of all possible baseband symbol vector for $\mathbf{x}_A$ and $\mathbf{x}_B$, and $\mathcal{C}^{L\times 1}$ is the L-dimensional complex vector space. Note that throughout this chapter, a vector is by default a column vector instead of a row vector, unless otherwise specified.

Then, the received signal at the AP can be written as

$$\mathbf{y} = h_A\mathbf{x}_A + h_B\mathbf{x}_B, \tag{3.1}$$

where $h_A, h_B \in \mathcal{C}$ ($\mathcal{C}$ denotes the complex plane) are the channel gains from Alice and Bob to the AP, respectively.

If we look at the collision (3.1) and assume that Alice's and Bob's signals go through similar channel conditions to the AP, Alice's or Bob's signal will have an SNR around 0dB due to mutual interference. Simply given (3.1), the AP is less likely to recover Alice's or Bob's signal due to two major reasons.

- If the AP adopts a traditional decoding design, it cannot decode a signal with SNR around 0dB, because an acceptable SNR is usually 10dB or above [99] for WiFi.
- Although multi-user detection [100] has been developed as a vital solution to decode multiple user's signals, this technique in general requires that users employ distinct spread spectrum codes [101] to differentiate themselves at the signal level. Nonetheless, there is no such code design in WiFi.

Apparently, additional information is needed to resolve the collision (3.1). Our key observation is that the RTS packet format itself provides valuable information for collision resolution in a WiFi network.

3.2.2 Anatomy of RTS in WiFi

The RTS/CTS mechanism lets a sender reserve the channel by sending an RTS packet first. Once the receiver replies with a CTS packet, the sender transmits the data packet. The RTS packet specifies a network allocation vector (NAV), which is the total time duration it wants to reserve, including the time durations of the CTS, the data and the ACK.

The data payload in an RTS packet consists of 20 bytes or 160 bits, and we denote it as an RTS data vector $\mathbf{b} \in [0,1]^{160}$, where $[0,1]^{160}$ is the space for all vectors with length 160, whose element is either 0 or 1. At the PHY layer, the data vector $\mathbf{b}$ is interleaved, coded and modulated into a signal vector $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is the set of all values of $\mathbf{x}$. These processes together can be denoted as a one-to-one function mapping $f : [0,1]^{160} \to \mathcal{X}$, which converts the RTS data vector $\mathbf{b}$ to the RTS signal vector $\mathbf{x} = f(\mathbf{b})$. As f is one-to-one correspondence, $|\mathcal{X}| = 2^{160}$ ($|\cdot|$ denotes the cardinality, or the number of elements, of a set).

We observe that all RTS data vectors in $[0,1]^{160}$ are not equally probable in the real world because all data fields in RTS are well structured and specified.

- `FrameControl` contains 2 bytes specified in 802.11.
- `Duration` is the 2-byte NAV in microseconds. The last bit is set to 0 and thus it holds up to 2^{15} values.
- `RA` and `TA` (6 bytes each) are the destination and source addresses, respectively. As today's WiFi is widely used for Internet access, stations communicate mostly with the AP. The AP knows that `RA` in an RTS packet sent to it is its own address and `TA` should be the address of one of its stations. Suppose that a dense network can support 2^6 stations (e.g., Linksys EA8500 firmware supports up to 51 stations [102]) and the number of possible values of `TA` in RTS is 2^6.

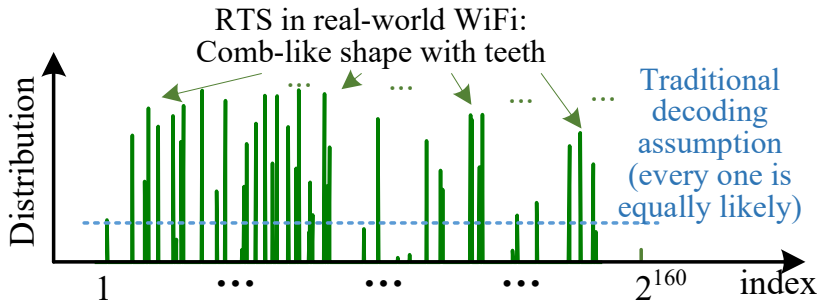


Figure 3.1. Example: distribution of RTS signal vectors.

- `FCS` (4 bytes) is for error detection and relies on other data fields. It provides no additional information.

Thus, from the AP's perspective, the number of RTS data vectors of interest is 2^{15} (from `Duration`) $\times$ 2^6 (from `RA/TA`) $= 2^{21}$ in the full RTS vector space $[0,1]^{160}$. As a result, the number of RTS signal vectors of interest is also 2^{21} in the signal vector space $\mathcal{X}$ with $|\mathcal{X}| = 2^{160}$. If we index all vectors in $\mathcal{X}$ from 1 to 2^{160} and measure via the probability distribution how each vector is likely to be seen in the real world, we will obtain a comb-shaped distribution similar to the example shown in Figure 3.1. We call an RTS signal vector a *tooth vector* if the probability that it can be seen at the AP is positive. Although the index goes from 1 to 2^{160} in Figure 3.1, the number of tooth vectors should be no less than 2^{21} according to our analysis. The comb-shaped distribution is in evident contrast to the traditional decoding assumption (also illustrated in Figure 3.1) that all potential values of a signal vector are equally probable [66]. This opens a door for us to go beyond traditional decoding to resolve RTS collisions.

3.2.3 Idea of Collision Resolution

Based on the observation from Figure 3.1, we present the basic idea regarding how to resolve an RTS collision.

3.2.3.1 Problem Reformulation

Denote by $\mathcal{M} = \{\mathbf{m}_i\}_{i\in[1,M]}$ ($M = |\mathcal{M}|$) the set of tooth vectors. Define the comb matrix $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_M]$ (i.e., each column in $\mathbf{M}$ is a tooth vector). The AP can pre-construct the comb matrix $\mathbf{M}$ by inserting all possible RTS to $\mathbf{M}$ to form the columns. The AP's goal is to find exactly which ones in $\mathbf{M}$ are actually involved in the collision.

Mathematically, we reformulate the collision problem to an equivalent one: assume that all tooth vectors in $\mathbf{M}$ are transmitted to the AP, but they go through different wireless channels. In particular, the tooth vectors involved in the actual collision go through the wireless channels with realistic channel gains, but those not involved in the collision go through the channels with zero channel gains. For example, in Figure 3.2, Alice, Bob and other users have different tooth vectors. The received signal $\mathbf{y}$ is considered as the sum of all these tooth vectors weighted by different channel gains. The channel gain weight of a tooth vector is the realistic channel gain if it is indeed transmitted, and zero otherwise. If Alice's transmitted signal is $\mathbf{m}_1$, the channel gain weight g_1 for $\mathbf{m}_1$ is the real channel gain between Alice and the AP, and the weights for the rest of Alice's tooth vectors are all zeros (e.g., $g_2 = 0$ as Alice transmits $\mathbf{m}_1$ not $\mathbf{m}_2$). As such, the received signal $\mathbf{y}$ can be reformulated as

$$\mathbf{y} = \sum_{i=1}^{M} \mathbf{m}_i g_i = \underbrace{[\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_M]}_{\text{comb matrix } \mathbf{M}} \underbrace{[g_1, g_2, \cdots, g_M]}_{\text{channel gain weight vector } \mathbf{g}}^{T}, \tag{3.2}$$

where $\mathbf{g}$ is called the channel gain weight vector and $\cdot^T$ denotes the matrix transpose. Based on (3.2), collision resolution is equivalent to solving for unknown $\mathbf{g}$ given $\mathbf{y}$ and $\mathbf{M}$. Then, the tooth vectors actually involved in the collision correspond to non-zero elements in the solved $\mathbf{g}$.

3.2.3.2 Solution Based on Reformulation

There are two key observations on the reformulation in (3.2).

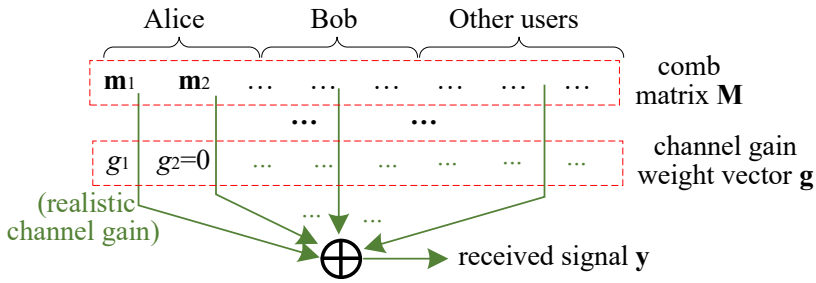


Figure 3.2. Reformulation of network collision.

- The unknown channel gain weight vector $\mathbf{g}$ is sparse in a real-world network because of two reasons: (i) There is no self-collision. As shown in Figure 3.2, if Alice sends a tooth vector $\mathbf{m}_1$, we have $g_1 \neq 0$; and any other tooth vector belonging to Alice will have a zero channel gain weight (e.g., $g_2 = 0$) since there is no way Alice transmits both $\mathbf{m}_1$ and $\mathbf{m}_2$. (ii) Because of the random backoff in WiFi, a collision is likely caused by only several users transmitting at the same time. Thus, $\mathbf{g}$ should include only several non-zero elements.

- The comb matrix $\mathbf{M}$ should exhibit a nearly random matrix property. Each tooth vector $\mathbf{m}_i$ in $\mathbf{M}$ is mapped from an RTS data vector through interleaving and error-correction coding. Their main purpose is to scramble and re-map all bits into a larger bit space in a (nearly) random way such that the error-correction performance approaches the random coding performance in Shannon's capacity [78].

Given the sparse property and nearly random matrix property, the channel gain weight vector $\mathbf{g}$ should be recovered with high probability by $\mathcal{L}_1$-norm minimization according to the theory of compressive sensing [103, 104, 105]. Thus, resolving an RTS collision leads to

the following optimization.

$$\begin{aligned} \text{Given:} \quad & \text{comb matrix } \mathbf{M} \text{ and received signal } \mathbf{y}, \\ \text{Objective:} \quad & \mathbf{g}_{\text{solution}} = \arg\min \|\mathbf{g}\|_1, \text{ subject to } \mathbf{y} = \mathbf{M}\mathbf{g}, \end{aligned} \tag{3.3}$$

where $\|\mathbf{g}\|_1$ is the $\mathcal{L}_1$-norm of $\mathbf{g}$ (i.e., $\|\mathbf{g}\|_1 = \sum_{g_i \in \mathbf{g}} |g_i|$).

The theoretical framework lays out a promising path towards resolving RTS collisions. Although the $\mathcal{L}_1$-norm minimization in (3.3) can be solved by many efficient algorithms [106, 107, 108, 27, 26], directly applying (3.3) to collision resolution incurs an unbearable cost because the comb matrix $\mathbf{M}$ consists of up to $2^{21} = 2,097,152$ tooth vectors according to our initial analysis in Section 3.2.2. Thus, significant challenges exist to make collision resolution meaningful and practical.

3.3 Construction of Comb Matrix

The initial step towards our CombDec design is to find a way to reduce the size of the comb matrix $\mathbf{M}$ (that can include 2^{21} tooth vectors) for a low-cost solution. In this section, we analyze the 802.11 standard, firmware and packet traces to show that there is a feasible way to significantly reduce the size of 2^{21}. Then, we design two algorithms, (α, β)-construction and γ-decimation, to reduce 2^{21} to only a few hundreds, while maintaining the high performance for collision resolution. The use of (α, β)-construction and γ-decimation clears the major hurdle towards system implementation.

3.3.1 Standard and Firmware Based Analysis

As aforementioned in Section 3.2.2, $\mathbf{M}$ consists of 2^{21} tooth vectors because of `Duration` and `RA/TA` fields in RTS. The `Duration` field specifies the 15-bit NAV in microseconds with $2^{15} = 32,768$ potential values, which largely contribute to the size of $\mathbf{M}$. 802.11 specifies that

the NAV is computed as one data packet duration, plus one CTS, one ACK, and three SIFS durations. Given a network setup, SIFS, CTS and ACK durations are usually fixed (e.g., SIFS is fixed to be 16 μs in 802.11ac at 5GHz). Thus, the value space of the NAV depends dominantly on the value space of the time duration of a data packet. In 802.11, the time duration of a data packet is bounded by aPPDUMaxTime, the maximum time duration of a data packet (in μs), and indirectly bounded by aPSDUMaxLength, the maximum payload length of a data packet (in bytes). As aPPDUMaxTime for 802.11ac is $5,484\mu s$, the space size of the NAV is reduced from 32,768 to at most 5,484 in the 802.11ac network.

Today's WiFi chipsets may still avoid transmitting a long packet (close to $5,484\mu s$) due to the cost consideration or hardware limitations. Hence, drivers implement their own packet length constraints on a data packet, which is usually less than the standard-defined aPSDUMaxLength or aPPDUMaxTime. We perform comprehensive code analysis on WiFi drivers and find that different vendors indeed pose different constraints on their own chipset, further limiting the value selection of the NAV. The detailed firmware analysis can be found in Appendix A.

As a result, we can leverage these constraints to further reduce the value space of the NAV in RTS. However, many WiFi drivers (in particular 802.11ac ones) are still proprietary and distributed in the binary form. It is not practical to study every WiFi chipset/firmware and optimally minimize the value space of the NAV in RTS packets. In what follows, we analyze real-world packet traces to develop a generic way to narrow down the value space.

3.3.2 Packet Trace Based Analysis

The key to reducing the size of the comb matrix is through shrinking the value space of NAV in RTS. We have shown that a WiFi driver can restrain the value space of NAV. Moreover, implementation-dependent rate control and data aggregation in proprietary WiFi drivers are not likely to produce uniformly distributed NAV values, but may be more biased

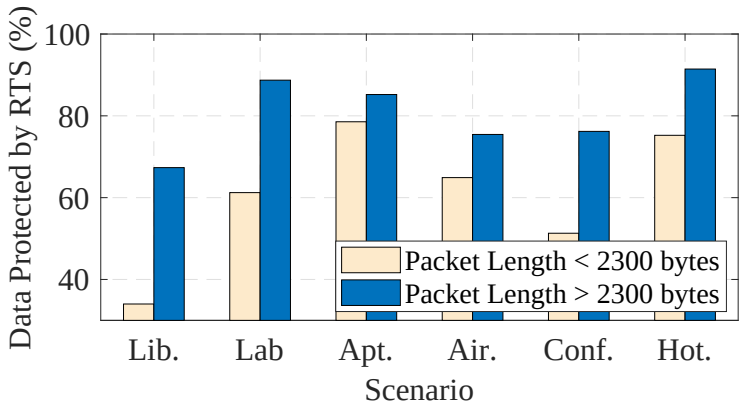


Figure 3.3. Percentages of data packets protected by RTS.

towards certain selections and yield a NAV distribution similar to Figure 3.1. Our objective is to collect massive packet traces to understand the NAV distribution. Then, we create generic algorithms to select those NAVs that are the most likely to be seen for constructing the comb matrix $\mathbf{M}$.

The first step towards understanding the NAV distribution in real-world RTS packets is to collect a substantially large number of packet traces for analysis. As no set of 802.11ac packet data is publicly available, we conducted our own measurements and collected in total 1.3 TB packet trace data with 2.33 billion packets in realistic environments, including (i) a public library (65.21 GB), (ii) three academic conferences (31.14 GB), (iii) five residential communities (65.69 GB), (iv) three major-brand hotels (109.48 GB), (v) four major US airports (88.5 GB), (vi) a university research lab (938.81 GB). The library, conference, and airport data traces were measured only within the business hours (i.e., 9am–5pm). Note that the payloads of all data packet were removed after the collection to avoid the privacy concern.

Figure 3.3 shows the the percentages of data packets that are protected by RTS among all data packets collected in different scenarios. Although a typical RTS threshold is set to around 2300 bytes [17, 18, 19, 20, 21], we see from Figure 3.3 that data packets of less

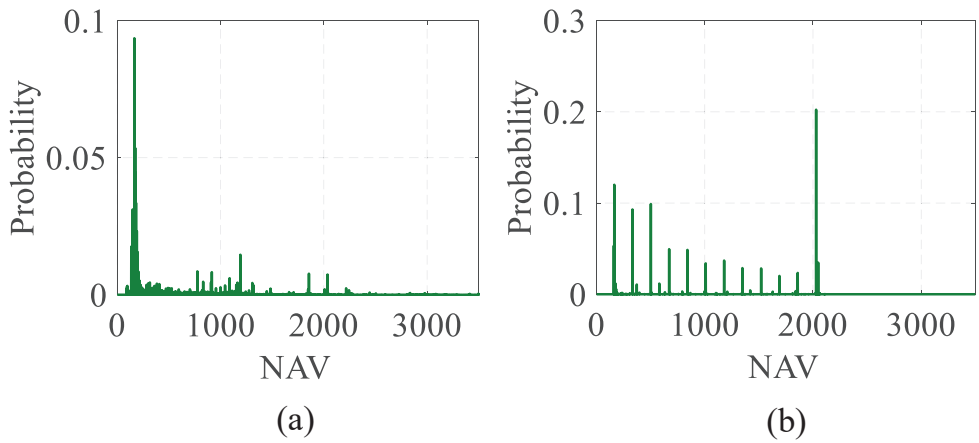


Figure 3.4. Distribution of NAVs from (a) the airport and (b) Belkin devices in all datasets.

than 2300 bytes are still likely to be protected by RTS. For example, in the hotel scenario, 78.55% data packets are initiated by RTS even when their lengths are less than 2300 bytes. Moreover, around 70% - 90% data packets of over 2300 bytes are protected by RTS in different scenarios. Overall, we observe from Figure 3.3 that RTS is still intensively used in today's Wi-Fi networks.

Looking into the NAV values in RTS packets, we observe that these values are unevenly distributed. For example, Figure 3.4(a) shows the NAV distribution of RTS packets in the airport dataset, which reveals that many NAV values (particularly around 230 μs) are much more likely to be observed than the others. The NAV distribution also depends on a WiFi driver. For example, Figure 3.4(b) plots the distribution of the Belkin WiFi driver measured from RTS packets sent by Belkin devices (recognized by MAC addresses) in all datasets. The figure shows that the distribution is quite patterned, indicating that the driver has several NAV levels to construct payloads. We observe uneven or patterned distributions in all datasets and offer a more detailed analysis in Appendix A.

Thus, if we only choose the most likely NAV values (instead of all possible values) to construct the comb matrix M, the size of M should be substantially reduced. In addition,

our design should not be device/firmware specific. For example, it may be possible to select NAV values based on the pattern of Belkin devices in Figure 3.4(b). But this method is too cumbersome because we have to examine the behaviors of all different WiFi devices. Our strategy is to use an online algorithm that actively computes the NAV distribution of a device, and then selects the most likely NAV values from the computed distribution to construct the comb matrix $\mathbf{M}$.

3.3.3 The (α, β)-Construction Algorithm

To select the most likely NAV values to construct $\mathbf{M}$, a node (either the AP or a station) should store the distribution of the NAV values in RTS packets from every other node in a network. In addition, the node should keep updating the storage to account for nodes joining or leaving the network. To this end, we propose the (α, β)-construction algorithm running at individual nodes to construct $\mathbf{M}$.

3.3.3.1 Algorithm Design

For a node that runs the algorithm, it records the frequency (i.e. the number of appearances) of a NAV value in RTS packets transmitted by every other node in the network. When a new RTS packet with a NAV value is decoded, the node will increase the frequency of that NAV value by one in its local storage. There are two key factors α and β in the algorithm.

- For the coverage factor α, for every other node in the network, the algorithm selects the α NAV values with the highest frequencies to form the tooth vectors and the comb matrix.

- For the forgetting factor β, the algorithm decreases the frequencies of all NAV values by β every minute. The minimum frequency is always set to be zero. And if the frequencies

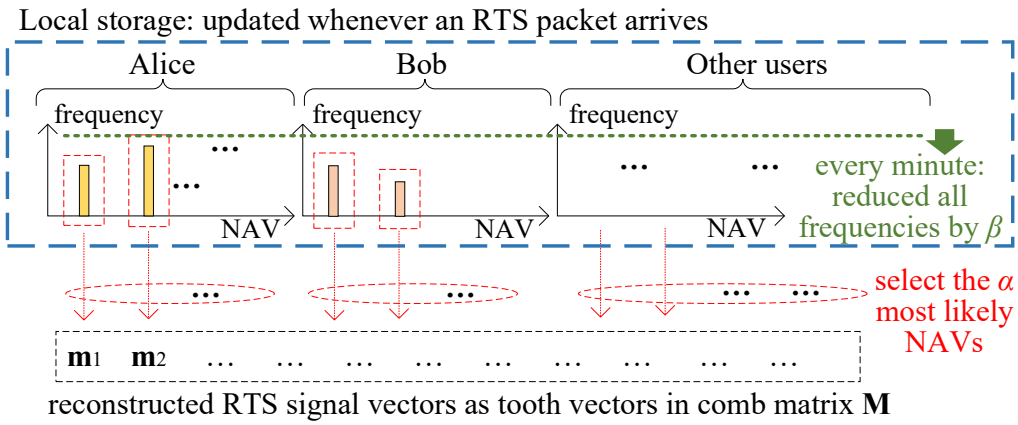


Figure 3.5. (α, β) construction running at the AP.

of all NAV values associated with a node become zero, the node's information will be all removed from the storage as it is considered no longer active or out of the network.

The (α, β)-construction algorithm works differently for the AP and stations. Figure 3.5 shows how it works at the AP: the AP stores the frequency of each possible NAV value from each station. When a collision happens at the AP, it knows the collision must be due to at least two stations (which could be Alice, Bob, or others) transmitting to it. Thus, the AP selects the α most likely NAV values from Alice, constructs an RTS data vector using each of these values, together with Alice's MAC address as `TA` and its own MAC address as `RA`, then maps each RTS data vector by function f (including interleaving, error-correction coding, modulation, IFFT) in Section 3.2.2 to a signal vector (which is a tooth vector in the comb matrix $\mathbf{M}$). Then, the AP repeats the same process for Bob and all other stations to finally obtain the full $\mathbf{M}$.

A station's construction of the comb matrix differs from the AP. For example, as shown in Figure 3.6, Alice observes a collision (a) when two other stations Bob and Carol are transmitting to the AP, (b) when one other station Bob is transmitting to the AP and at the same time the AP is transmitting to a third station Carol, or (c) when the AP is

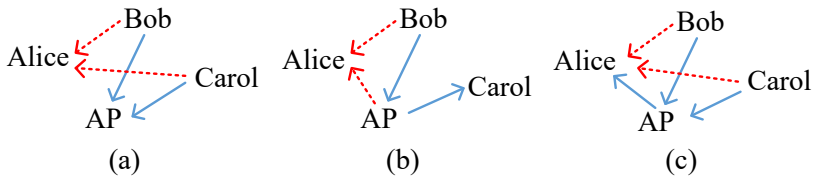


Figure 3.6. Different collision scenarios in Alice's view.

transmitting to Alice while Bob and Carol are transmitting to the AP. In all three cases, collision resolution is only meaningful for Alice in case (c) because the collision in case (c) includes the AP's signal intended for Alice. Even when Alice successfully resolves cases (a) and (b), Alice only knows that there is no signal of interest and then stops. Therefore, the construction is sufficient for Alice as long as case (c) can be resolved by Alice. According to case (c), a station should construct the comb matrix by using RTS signal vectors from the AP to itself and other RTS signal vectors from other stations to the AP. The construction process is similar to Figure 3.5.

3.3.3.2 Selections of (α, β) and Cost Evaluations

The size of the resultant comb matrix $\mathbf{M}$ depends on both α and β. In particular, α is the number of tooth vectors from one node, and β in fact determines how many nodes will be used in constructing $\mathbf{M}$ because (i) all frequencies are decreased by β every minute and (ii) a node will be removed from the construction when all its frequencies become zero. The algorithm with a larger β forgets nodes faster, thereby reducing the number of nodes used for constructing $\mathbf{M}$.

The performance of (α, β)-construction can be evaluated by the miss rate, defined as the probability that when an RTS packet arrives at a node, $\mathbf{M}$ constructed by the algorithm at the node does not include the NAV value in the RTS packet. A large α and a small β are able to reduce the miss rate, but at the same time increase the size of $\mathbf{M}$, incurring more cost.

Table 3.1. Miss rate and average size of the comb matrix.

α=600 β=10	Miss rate	Ave. # of nodes	# of tooth vectors in $\mathbf{M}$
Library	6.9 %	12.8	7680
Conferences	3.6 %	11.6	6960
Apartments	1.5 %	6.5	3900
Hotels	0.7 %	10.3	6180
Airports	8.8 %	18.8	11280
Lab	5.7 %	6.3	3780

Our objective is to find the pair of (α, β) to balance the miss rate and the complexity for general WiFi scenarios. To this end, we simulate a WiFi network in each of the packet datasets, replay all collected packets to simulate RTS arrivals at each node, and measure the miss rate of the algorithm with different values of α and β. Table 3.1 shows one selection of (α, β)=$(600, 10)$ for all scenarios that achieves a good balance between the miss rate and the size of $\mathbf{M}$ (measured by α multiplying the average number of nodes used for constructing $\mathbf{M}$). We can see that all miss rates are below 9% with around 4,000–12,000 tooth vectors in $\mathbf{M}$.

3.3.4 The γ-Decimation Algorithm

Through 802.11 standard analysis, firmware analysis, packet trace analysis and (α, β)-construction, we have dramatically shrink the size of $\mathbf{M}$ from the initial 2,097,152 tooth vectors to 12,000 or fewer vectors. All these push the optimization in (3.3) to the practice. However, finding the $\mathcal{L}_1$-norm minimization with $12,000$ vectors in (3.3) still incurs a substantial cost. We propose γ-decimation to further reduce such a cost while maintaining the high performance.

Denote by M the number of tooth vectors in $\mathbf{M}$ constructed by (α, β)-construction. The basic idea of the γ-decimation algorithm ($\gamma > 1$ is called decimation rate) is to select, based

on the received signal vector $\mathbf{y}$, M/γ vectors out of all M tooth vectors in $\mathbf{M}$ to form a decimated comb matrix $\mathbf{M}'$.

The design intuition is that the received signal $\mathbf{y}$ contains only several tooth vectors in $\mathbf{M}$ that we aim to find out. If we compute the correlation between $\mathbf{y}$ and each tooth vector $\mathbf{m}_i$ in $\mathbf{M}$, defined as $C(\mathbf{y}, \mathbf{m}_i) = \|\mathbf{m}_i^H \mathbf{y}\|_2$ ($\cdot^H$ denotes conjugate transpose and $\|\cdot\|_2$ denotes the $\mathcal{L}_2$-norm), we then obtain M correlation values. Due to the property of correlation, we should observe a high correlation value if a tooth vector is indeed included in $\mathbf{y}$, and a low correlation value otherwise. However, due to channel noise and limited length of tooth vectors, some tooth vectors not in $\mathbf{y}$ may also exhibit high correlation values. But it is not necessary to exactly identify which tooth vectors with high correlation values are indeed in $\mathbf{y}$ at this stage, γ-decimation just chooses M/γ tooth vectors that have the highest correlation values to form the decimated comb matrix $\mathbf{M}'$. It is very likely that tooth vectors involved in the collision are included in $\mathbf{M}'$ as long as M/γ is sufficiently large. We provide theoretical analysis for the performance of γ-decimation and show that all RTS signals involving a collision will survive the decimation and be included in $\mathbf{M}'$ with high probability in Appendix A.

As a result, the final comb matrix for (3.3) is constructed as follows: (i) (α, β)-construction constructs the comb matrix $\mathbf{M}$ with M tooth vectors (this steps ensures a low miss rate), (ii) γ-decimation decimates $\mathbf{M}$ into the decimated comb matrix $\mathbf{M}'$ with only M/γ tooth vectors (this steps ensures that tooth vectors involving the actual collision are preserved with high probability), and (iii) the decimated comb matrix $\mathbf{M}'$ is used in (3.3) for collision resolution to finally identify which tooth vectors are included in the collided signal $\mathbf{y}$.

To make (3.3) feasible for today's systems, $\mathbf{M}'$ should have only hundreds of tooth vectors. As (α, β)-construction maintains up to around $12,000$ vectors (shown in Table 3.1), the decimation rate γ should be around 12 or more.

3.3.5 Complexity Analysis

In the following, we estimate the computational complexity and storage complexity of CombDec.

3.3.5.1 Computational Complexity

We evaluate CombDec's complexity by comparing it with a benchmark, which is the complexity to decode a typical 802.11 data packet with 40MHz bandwidth, 3/4 convolutional coding, and 64QAM. We describe the major computational operations involved in the benchmark and CombDec.

- The benchmark complexity is proportional to the data payload length, denoted by N bytes. Decoding a data packet requires the FFT and Viterbi algorithms. The packet contains $0.0247N$ OFDM symbols plus 4 PHY headers symbols. The FFT on each symbol requires $64\log(128)$ complex multiplications and $128\log(128)$ complex additions [109]. In addition, the Viterbi algorithm incurs $4^K(10.67N + 252)$ real additions and $2^K(10.67N+252)$ real comparisons, where K is the constraint length of convolutional code [110].

- In CombDec, suppose (α, β)-construction maintains M tooth vectors of length L, the correlation of the received signal with each of the tooth vector in γ-decimation needs a total of ML complex additions and multiplications. Among all correlation values, selecting the M/γ largest values incurs $M + M\log(M)/\gamma$ comparisons by using the max heap tree approach [111]. The complexity of $\mathcal{L}_1$ minimization depends on an iterative algorithm and is bounded by the cubic polynomial complexity [107]. In order to estimate an exact computational cost, we use simulations to run the primal-dual interior-point algorithm [25] to solve the $\mathcal{L}_1$-norm minimization in M/γ tooth vectors and compute the average numbers of additions, multiplications and comparisons.

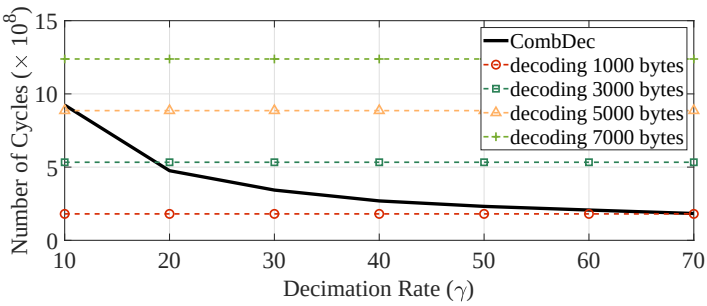


Figure 3.7. Number of cycles: CombDec vs benchmark.

As the complexity involves different types of operations, including additions, multiplications and comparisons. We need to have a unified cost utility metric to measure the overall costs of CombDec and the benchmark. We use the number of general CPU cycles as the utility metric [112]. In particular, one real addition or comparison is counted as one cycle and one multiplication is 4 cycles [112]; and a complex operation is 4 times the number of cycles incurred by its real counterpart [109]. Note that an algorithm or a computational operation can be specifically optimized on a particular signal processing software or hardware platform. Our estimation using CPU cycles is not intended to be exactly accurate for an implementation platform, but serves as an approximate way to demonstrate what computational complexity level CombDec is at when compared with the benchmark.

Figure 3.7 shows the total numbers of cycles incurred by CombDec and the benchmark. In Figure 3.7, we let (α, β)-construction form $M = 12,000$ tooth vectors to accommodate the airport scenario in Table 3.1 and set a typical constraint length $K = 7$ in the Viterbi Algorithm. It is observed from Figure 3.7 that choosing γ to be in [20, 50] leads to the complexity of CombDec roughly equivalent to decoding a packet with 1000–3000 bytes, which makes CombDec ready for system implementation.

3.3.5.2 Storage Complexity

CombDec also incurs a storage cost. First, CombDec for a device must store the frequency of each NAV value for any other active device in the network. The cost of storing all NAV frequencies is equal to the NAV space size multiplying the average number of active devices. There are 5,484 possible NAV values in 802.11ac and around 18.8 active nodes under (α, β) construction for the airport scenario (shown in Table 3.1). Hence, the storage cost is 5484 $\times$ 18.8 $\times$ 1 = 101 KB when the frequency value is a one-byte integer. Second, after γ-decimation, all tooth vectors should be stored for $\mathcal{L}_1$-minimization. The storage cost is the number of tooth vectors multiplying the length of a tooth vector. When $\gamma \in [20, 50]$, the number of decimated tooth vectors in the airport scenario is 240 to 600. If a typical RTS packet is transmitted at 12 Mbps (4 64-subcarrier OFDM symbols) and the element in tooth vector is a 4-byte complex number, the length of a tooth vector is $64 \times 4 \times 4 = 1$ KB. Thus, the cost of storing all these tooth vectors is in the range of $[240, 600]$ KB.

Overall, the major storage cost is around [341, 701] KB. This cost is also reasonable for today's WiFi systems. For example, Qualcomm's 802.11ac chipset IPQ4018 has an on-chip memory of 256 MB [113] and related APs cost as low as tens of dollars [114, 115].

3.4 CombDec System Design

The (α, β)-construction and γ-decimation algorithms pave the way for a feasible system solution to (3.3). In this section, we present CombDec system design. We first introduce the system architecture, then describe each key component.

We design CombDec as an independent decoder in addition to the traditional 802.11 decoder. Figure 3.8 shows four major components in CombDec: the prologue module, (α,β)-construction, γ-decimation, collision resolution, and the epilogue module. As shown in Figure

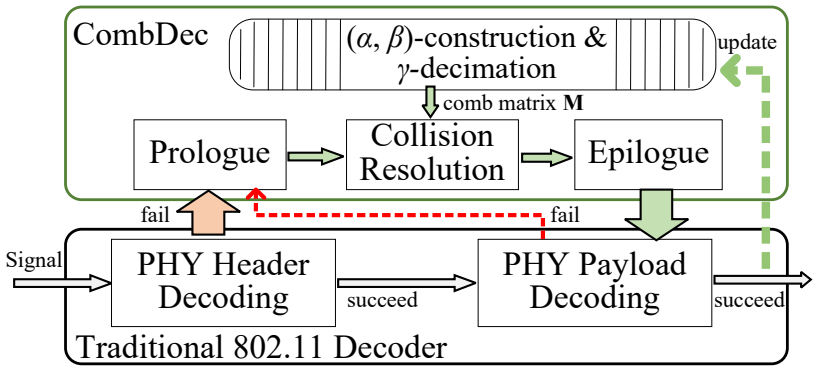


Figure 3.8. Architecture of CombDec decoder.

3.8, CombDec is triggered only when decoding of either the PHY header or the PHY payload fails. In the following, we present the designs of individual CombDec Components.

3.4.1 The Prologue Module

The prologue module does all pre-processing before collision resolution.

3.4.1.1 Combining Multi-path Signal Components

The received signal may include a number of multi-path components with different time offsets. To improve the performance of CombDec under multi-path fading, we use the maximum ratio combining (MRC) [66] to combine the multi-path signal components. Specifically, denote by $s(n)$ the n-th time-domain symbol in the received signal. Based on the 802.11 packet preamble, we use a matched filer [16, 116] to detect the time offset and estimate the channel gain of each signal component. Given K components found, we use the MRC [66] to coherently sum all K time-shifted copies of $s(n)$ and obtain $s'(n) = \sum_{k=1}^{K} h_k^* s(n+\delta_k)$, where k-th component having time offset δ_k and channel gain h_k, and h_k^* is the complex conjugate of h_k. Note that signal components may be from different senders in a collision, CombDec does not differentiate them in the prologue module.

3.4.1.2 Collision Recognition and Early Stop

Next, CombDec decides if $s'(n)$ can be potentially resolved. There are three types of collisions: RTS-only, RTS-data (involving at least one RTS packet and one data packet), and data-only collisions. CombDec will stop if the collision belongs to data-only collision. Note that the recognition does not need to be 100% accurate, it simply provides a way to exclude obvious data-only collisions that cannot be resolved by CombDec.

In WiFi networks, the beginnings of collided packet transmissions are roughly aligned because of CSMA/CA (if we do not consider the hidden terminal problem). Thus, we measure the time durations of different power levels of the received signal to identify the type of a collision. According to 802.11, the data rate for RTS is selected by a station from a limited set of basic rates defined by the AP (e.g., 12 Mbps and 24 Mbps are widely observed in our packet traces). Thus, a RTS time duration can be measured by a very limited number of OFDM symbol durations, e.g., 3 (or 4) OFDM symbol durations for 24 (or 12) Mbps. We record the time duration l_i, from the beginning of the signal $s'(n)$, to the position in $s'(n)$ where the i-th signal power change happens and is larger than a threshold Δ. When the power level reaches the noise floor, we stop and obtain a set of time durations $\mathcal{L} = \{l_i\}_{i \in [1, |\mathcal{L}|]}$. We consider a collision as (i) RTS-only if each value in $\mathcal{L}$ is close to one of the RTS durations corresponding to the basic rate set defined by the AP, (ii) RTS-data if one value is close to an RTS duration and any other value is not close to any of the RTS durations, and (iii) data-only otherwise. Figure 3.9(a) shows an example of RTS-only collision, where two measured time durations occupy 3 and 4 OFDM symbol durations, respectively, which is recognized as the collision of two RTS packets with different rates.

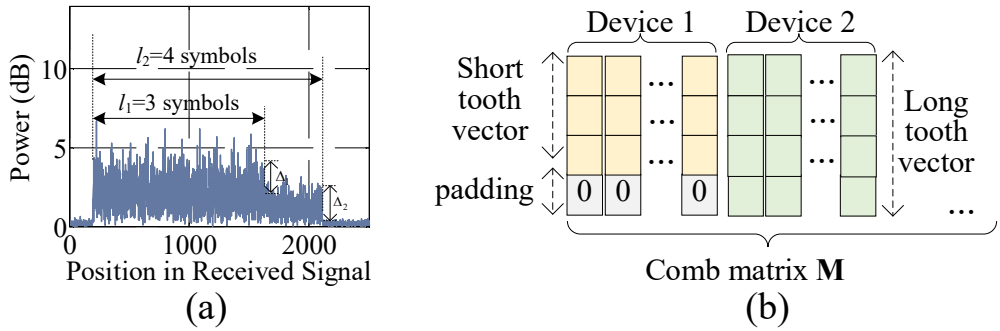


Figure 3.9. Power level changes in the received signal and padding in matrix construction.

3.4.2 The Collision Resolution Module

The prologue module outputs either RTS-only or RTS-data signals for collision resolution, (α, β) construction will construct every tooth vectors for the comb matrix. As devices in a network may use different basic rates to transmit RTS packets (leading to different lengths of tooth vectors), CombDec zero-pads the shorter tooth vectors with higher data rates to form the complete comb matrix $\mathbf{M}$ used for collision resolution. Figure 3.9(b) shows an example of constructing the comb matrix $\mathbf{M}$ by zero-padding shorter tooth vectors. Then, $\mathbf{M}$ is γ-decimated to $\mathbf{M}'$, which is used in the primal-dual algorithm [25] that solves the $\mathcal{L}_1$-minimization in (3.3). Note that if a collision is RTS-data, the module resolves the collided RTS signal vectors by treating any data signal as the noise, and then leaves the potential decoding of data for the epilogue module.

3.4.3 The Epilogue Module

The collision resolution module yields a small set of potential RTS signal vectors involved in the collision, denoted by $\mathcal{R}$. There are still important questions left: (i) Which RTS in $\mathcal{R}$ a receiver should choose to reply with CTS? (ii) Can we decode the data in the presence of an RTS-data collision? (iii) Moreover, we also need an error detection mechanism to ensure the collision is correctly resolved because errors may happen in CombDec. We first describe

how CombDec chooses data or RTS. The AP and stations have different decision making processes.

3.4.3.1 Decision Making at AP

The AP observes a collision when multiple stations transmit to the AP at the same time.

- Choosing the RTS with the largest NAV: if the collision is RTS-only, the AP chooses the RTS of the largest NAV to reply with CTS. Note that we intend to maximize the channel utilization in this way. A more advanced policy (e.g., considering the utilization and fairness) can be designed and adopted going beyond the main scope of this chapter.

- Choosing data over RTS: If the collision is RTS-data, the AP first decodes all RTS packets in the collision resolution module, then proceeds to decode the data. If the data is successfully decoded, the AP chooses data over RTS and sends back the ACK to the sender of the data. This is because data packets are usually longer than RTS packets. Giving priority to data should improve the channel utilization.

3.4.3.2 Decision Making at Stations

A station observes a collision when multiple other nodes (either other stations or the AP) transmit at the same time. As shown in Figure 3.6(c), a station only cares about the AP's signal transmitted to it.

- If $\mathcal{R}$ includes an RTS vector from the AP to the station, the collision is due to the AP transmitting to the station and at the same time at least one other station transmitting to the AP, the station always sends CTS to the AP.

- Otherwise, there is no RTS in $\mathcal{R}$ intended for the station. If the collision is RTS-data, the station proceeds to decode the data because the data may be intended for it; otherwise, the station stops.

3.4.3.3 Error Checking and Data Decoding

Once a decision is made (choosing RTS or data), we must ensure there is no error with the chosen RTS or we must proceed to decode the data. How to proceed with error checking and data decoding? We find that a traditional 802.11 receiver, as shown in Figure 3.8, already has a PHY payload decoder with the cyclic redundancy check (CRC) mechanism. Thus, we should leverage the existing architecture to perform the decoding and error checking to minimize the complexity of CombDec.

As a result, if CombDec decides to act on a particular RTS signal or to decode a data signal, it uses interference cancelation to remove all other signals from the collided signal. Specifically, if a decision is to decode the data, CombDec removes all RTS signal vectors in $\mathcal{R}$ from the received signal $s'(n)$ and write the resultant signal as $s'_c(n) = s'_c(n) - \sum_{i=1}^{|\mathcal{R}|} r_i(n)g_i$, where $r_i(n)$ and g_i are the n-th element and the channel gain weight of the i-th RTS tooth vector in $\mathcal{R}$, respectively. Similarly, if the decision is to act on the j-th RTS vector (i.e., choose the j-th RTS in $\mathcal{R}$ to reply with CTS), CombDec removes all other RTS tooth vectors from the $s'(n)$ and the resultant signal becomes $s'_c(n) = s'_c(n) - \sum_{i=1, i\neq j}^{|\mathcal{R}|} r_i(n)g_i$.

Finally, the signal $s'_c(n)$ goes from CombDec into the traditional PHY payload decoder, which performs decoding and error checking, then passes the correct RTS or data to the MAC layer for protocol processing (e.g., replying with CTS). In addition, the MAC address and data rate information of a correct RTS packet is stored and its NAV value is also updated in (α, β)-construction as shown in Figure 3.8.

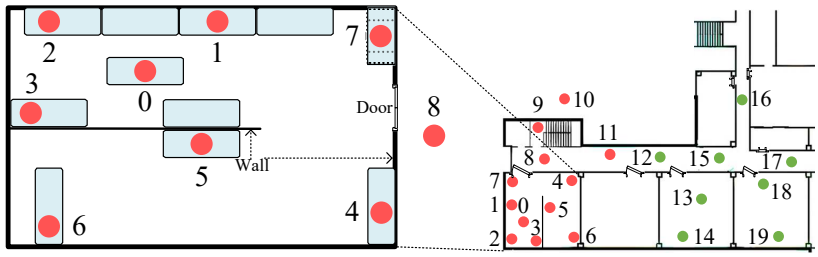


Figure 3.10. Environment for experiments.

3.5 Evaluation

In this section, we evaluate the performance of CombDec. We first introduce the experimental setups, then measure the performance benefits CombDec brings to WiFi networks.

3.5.1 Setups

3.5.1.1 Testbed Implementation

We have implemented the prototype of CombDec on 20 USRP X310/300 devices. Each device is equipped with two UBX-160 daughterboards and two VERT 2450 antennas. We implement a basic 802.11ac PHY-MAC architecture with 20-MHz settings: 64 OFDM subcarriers (including 48 data subcarriers), BPSK, QPSK, 16QAM, and 64QAM modulations, Alamouti code based MIMO, and convolutional coding at the PHY layer, and CSMA/CA scheme with an initial contention window size of 8 [11] at the MAC layer. Control packets including RTS, CTS, and ACK are also implemented.

3.5.1.2 Experimental Settings

We aim to measure the performance of CombDec in a realistic indoor environment inside a campus building shown in Figure 3.10. Network nodes are placed at various locations and

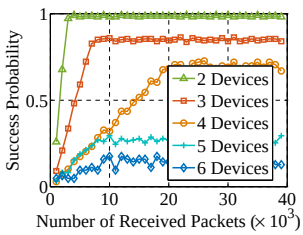


Figure 3.11. Success probabilities under 2-6 transmitters.

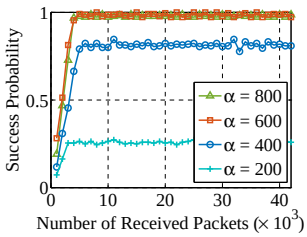


Figure 3.12. Impact of value of α in (α, β)-construction.

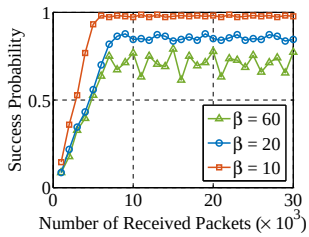


Figure 3.13. Impact of value of β in (α, β)-construction.

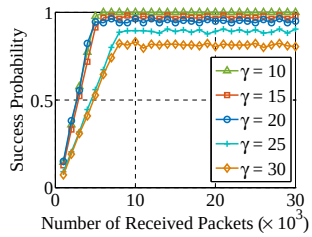


Figure 3.14. Impact of value of γ in γ-decimation.

transmit packets whose contents are generated according to our collected 802.11ac packet traces.

Note that we do not implement the 256QAM modulation as we found no single packet using a data rate associated with 256QAM in all collected packet traces. This does not severely affect the performance evaluation since 256QAM is intended only for very high SNR conditions.

We use the following default settings for experiments (unless otherwise specified): (i) all nodes are saturated; i.e., they always have packets to transmit; (ii) α=600, β=10, and γ=20 for CombDec; (iii) the airport dataset is used to generate packets as it represents the most crowded condition in all datasets; (iv) all nodes have the same transmit power.

3.5.1.3 Evaluation Metrics

We use the following metrics to evaluate the real-time performance of CombDec.

- Success probability of collision resolution is defined as the probability that CombDec recovers exactly all collided RTS signals. The recovery will be considered as a failure if CombDec recovers only a subset of collided RTS signals or mis-identifies an RTS signal not involved in the collision.

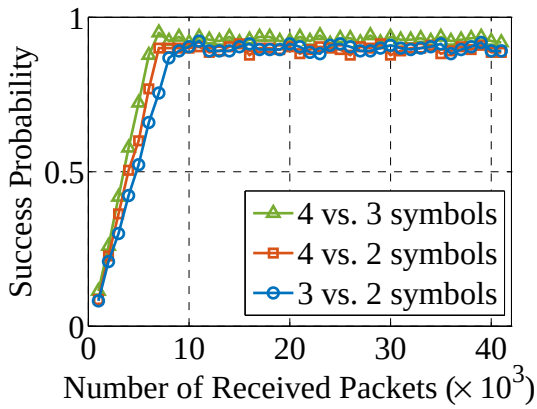


Figure 3.15. Resolving collisions with different rates.

- Normalized throughput (or utilization efficiency), is defined as the percentage of the time duration on the wireless channel that is used to deliver data packets. An ideal network should have a normalized throughput of 1. However, control signals (e.g., RTS/CTS) and collisions deteriorate the throughput. We aim to show how much channel utilization efficiency CombDec can improve via resolving RTS collisions.

- Throughput gain ratio, defined as the ratio between the increased normalized throughput from traditional 802.11 decoding to CombDec and the normalized throughput under traditional decoding. Throughput gain ratio can directly reflect how CombDec improves the network performance.

3.5.2 Success Probability

We first evaluate the success probability of CombDec for collision resolution. In this evaluation, multiple nodes only transmit RTS packets to the AP placed at location 0 in Figure 3.10, where the success probability is measured.

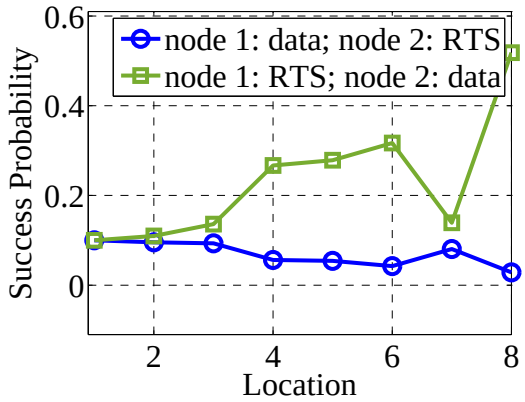


Figure 3.16. Resolving RTS-data collisions at different locations.

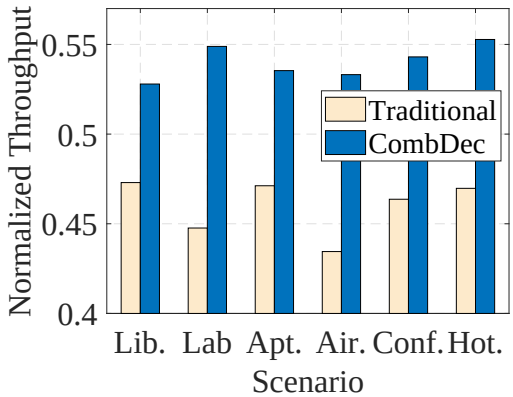


Figure 3.17. Throughputs under different scenarios.

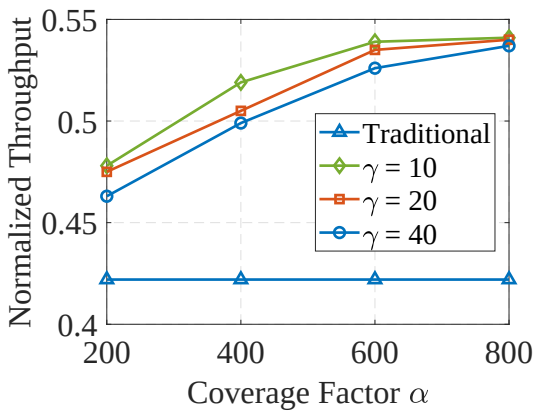


Figure 3.18. Throughputs with different α and γ.

3.5.2.1 Impact of Number of Transmissions

We evaluate CombDec's ability to resolve collisions due to two and more transmissions. To this end, we place multiple transmitters at location 1, which send RTS packets at the same time to the AP at location 0. Figure 3.11 shows the success probabilities that the AP resolves the collision under 2–6 transmitters. The success probability is computed for every 1,000 packets received. It is noted from Figure 3.11 that as the total number of received packets at the AP increases, CombDec gradually gains the NAV information in RTS packets, and thus the success probability also increases and finally remains stable. It is also observed that CombDec is able to resolve 98% of two-node collisions and 86% of three-node collisions. The performance degrades when the number of transmitters increases. However, a collision caused by 5 or more WiFi nodes is much less frequent because of the random backoff in CSMA/CA.

3.5.2.2 Values of α and β

We evaluate the impacts of α and β on the success probability. We place two nodes at location 1 that transmit RTS packets at the same time to the AP at location 0. Figure 3.12 shows that as α goes from 200 to 600, the success probability increases from 0.28 to 0.98; and further increase of α will not substantially improve the success probability. Figure 3.13 shows the impact of β on the success probability. We observe that when β increases from 10 to 60 (i.e., CombDec forgets the history faster), the success probability reduces from 0.98 to 0.77. From both figures, we can see that the uniform selection of $\alpha = 600$ and $\beta = 10$ yield very high performance for the airport scenario, and accordingly are also suitable for other less crowded scenarios.

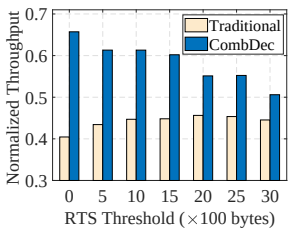


Figure 3.19. Throughputs with different thresholds

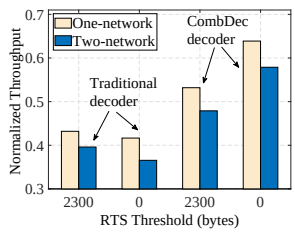


Figure 3.20. Throughputs in collocated networks.

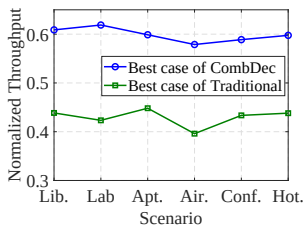


Figure 3.21. Throughputs in collocated networks.

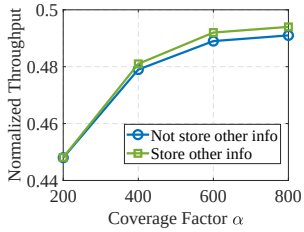


Figure 3.22. Storing information of other network.

3.5.2.3 Value of γ

We adopt the same setup in Figure 3.13 to evaluate the impact of γ. Figure 3.14 illustrates that gradually increasing γ does not severely decrease the success probabilities. For example, when γ becomes 30, the success probability reduces to 0.831. This indicates that adjusting γ can smoothly balance the performance and the implementation cost.

3.5.2.4 Impact of Zero-Padding

In each of our packet datasets, we find that a majority of RTS packets are transmitted at the same data rate; however, RTS packets with different rates do exist. These packets have different lengths of 2, 3, or 4 OFDM symbols. Therefore, a minority of collisions involving RTS packets with different rates. CombDec uses zero-padding to solve this issue as discussed in Section 3.4. We measure the ability of CombDec to resolve such a type of collisions. Hence, we place two nodes at location 1 sending RTS packets with different lengths to the AP. Figure 3.15 shows that the success probabilities remain approximately the same when RTS packets have different lengths in a collision. From Figure 3.15, we conclude that CombDec has no difficulty in resolving this type of collisions.

3.5.2.5 RTS-Data Collisions

CombDec also attempts to resolve an RTS-data collision via canceling the RTS signal from the received signal and then performing decoding. To evaluate such an ability, we place one node (node 1) at location 1 to send RTS to the AP, and place another node (node 2) at one of locations 1–8 to send data to the AP for the first round of experiments, and then let node 1 send data and node 2 send RTS for the second round. The first and second rounds represent the scenarios in which the receiving power of RTS is greater than and less than that of data, respectively. We consider the collision is resolved when both RTS and data packets are decoded successfully. Figure 3.16 depicts the success probability of collision resolution as node 2's location changes. The figure shows that generally, the success probability is higher when the receiving power of RTS is higher than that of data. This is because CombDec first treats any data packet as the noise to recover any RTS packet from the receiving signal. The results demonstrate that CombDec, primarily designed to handle RTS-only collisions, is capable of resolving RTS-data collisions in some scenarios.

3.5.3 Network Performance Evaluation

We then evaluate the benefits of CombDec for the network performance. Note that it is impossible to measure the network performance with CombDec under different setups at the same time because the resolution of a collision directly affects follow-on network dynamics. We have to measure the performance under different setups over non-overlapping measurement periods. Therefore, we conduct experiments during off-business hours to minimize the impact of environmental factors on different measurement periods.

3.5.3.1 Single Network Scenario

We first consider a single-network scenario where 12 nodes are placed at locations 0–11, in which the AP is at location 8, as shown in Figure 3.10. The network does not run in the MU-MIMO mode (i.e., the AP does not transmit data via MU-MIMO to multiple stations in the downlink). The RTS threshold is set to be 2,300 bytes for all nodes (i.e., RTS is triggered only when a data packet to be transmitted has a length over 2,300 bytes). The value of 2,300 is typical for today's WiFi products (e.g., the default value in Cisco APs is 2347 [17]).

For the throughput improvement, Figure 3.17 demonstrates the comparisons of normalized throughputs under traditional 802.11 decoding and CombDec. Note that the throughput performance is always measured at the AP. We can see that CombDec is able to uniformly boost the performance of traditional 802.11 decoding. For example, the normalized throughput for the airport scenario increases from 0.43 to 0.53, leading to a throughput gain ratio of (0.53 - 0.43)/0.43 = 23.3%. In all different scenarios, we observe that the throughput gain ratio under CombDec is 11.6%–23.3%.

For the improvement by tuning α and γ, we aim to find if we can improve the performance by tuning α and γ, which are important factors to balance the performance and complexity. Figure 3.18 shows the normalized throughputs for various α and γ values in the airport scenario. The figure shows that keeping increasing α and decreasing γ do not always lead to evident improvement. For example, when α goes from 600 to 800 and γ decreases from 20 to 10, the throughput under CombDec only increases from 0.491 to 0.494.

For the improvement by reducing RTS threshold, it is still possible to further improve the network performance. Our observation is that traditionally, an RTS collision is considered not resolvable; therefore, many WiFi devices are conservative to set the RTS threshold. The transmission of a data packet triggers an RTS transmission only when its data size is

greater than the threshold. In all previous experiments, the threshold is set as a typical value of 2,300 for today's networks. Now CombDec has the capability of decoding RTS collisions; therefore, it can be beneficial to encourage more RTS transmissions by reducing the threshold. Figure 3.19 compares the normalized throughputs under traditional 802.11 decoding and CombDec for different RTS thresholds. The figure shows that reducing the threshold generally decreases the throughput performance under traditional decoding; however and interestingly, it substantially boosts the performance under CombDec. The best case for traditional decoding is to set the threshold as 2000–2500, resulting in a normalized throughput of 0.456. By contrast, the best case for CombDec is to remove the threshold and let everyone always send RTS before data, yielding a higher throughput of 0.657. The throughput gain ratio is computed as (0.657-0.456)/0.456 = 44.08%. This encouraging result shows that CombDec has an immediate impact on today's practice of setting the RTS threshold, and significantly reducing this threshold can push WiFi towards a collision-free environment.

3.5.3.2 Collocated Networks

Next, we place a new network close to the single network used in previous experiments. In the new network, 8 nodes are placed at locations 12–19 and the AP is at location 15, as shown in Figure 3.10. The two networks use the same frequency and thus interfere with each other. We call the APs in the original and new networks AP 1 and AP 2, respectively.

Figure 3.20 shows the throughput performance under different settings in the airport scenario. In Figure 3.20, the one-network performance is the performance measured at AP 1 in the previous single-network scenario (without the new network); and the two-network performance is measured as the average of the throughputs measured at AP 1 and AP 2. We can observe from Figure 3.20 that when the new network is placed, the throughput performance degrades due to mutual interference. CombDec still performs better

than traditional 802.11 decoding. In the two-network scenario, the best case for CombDec is setting the RTS threshold to 0 (which is also beneficial to solving the hidden terminal problem), yielding a throughput of 0.579; and the best case for traditional decoding has a throughput of 0.396. The throughput gain ratio is thus (0.579 - 0.395)/0.395 = 46.6%, which is also a substantial throughput improvement. Figure 3.21 compares the best case throughput performance between CombDec (removing the RTS threshold) and traditional coding (setting the threshold to 2,300) in different scenarios. It can be seen that the throughput gain ratio of CombDec is 33.6% – 46.2%.

As discussed in Section 3.3, CombDec is designed to only store information of its own network. In the two-network scenario, it is possible to enhance the performance of CombDec by letting (α, β)-construction store the information (including MAC addresses, NAVs, and RTS rates) of the other network. Figure 3.22 shows that storing other network information can further yet slightly improve the throughput performance of CombDec with a fairly large α.

3.6 Related Work

3.6.1 Interference Cancelation and Mitigation

In the literature, successive interference cancelation (SIC) has been proposed to decode collisions by using either pre-coded signatures or different receiving powers [117, 118, 119, 120, 121, 122, 123, 124]. The time offsets in different packet collisions (e.g., in the presence of hidden terminals) has also been leveraged to resolve collisions [16, 125, 91]. In addition, interference cancelation was widely studied in the full-duplex mode [121, 126, 127, 128, 129]. In cross-technology communication, corrupted packets may also be decoded by detecting the interference type [130, 131]. A number of studies have also proposed interference alignment and nulling with or without channel state information [132, 133]. These approaches cannot

be readily adapted to regular WiFi scenarios considered in this chapter, where RTS packets collide at the beginning of each transmission.

3.6.2 Multi-user Detection

CombDec is related to multi-user detection that attempts to decode multiple users' signals from the overlapped signal [134, 135, 136]. In cellular networks, CDMA has been widely used to assign distinct spread spectrum codes to different users [101]. However, there is no such code design in RTS packets. Constructive interference [135] is able to receive multiple synchronized transmissions. Nevertheless, it requires all packets have the same content, which is impossible for RTS signals. The work in [137] applied the time division technique to the byte level such that multiple users can share the same packet. This method needs a strict coordination among all users. A multi-user system is built in [134] through sharing multiple channels to users who are allowed to duplicate the signal into these channels. Applying these designs to WiFi requires modification of the standard; in contrast, CombDec is a non-invasive design.

3.6.3 Improving WiFi Performance

Substantial efforts have been devoted to improving the WiFi link performance [138, 139, 140, 141, 142, 143, 144, 145, 146]. For example, [138, 139, 141] focused on optimizing the user selection algorithm in MU-MIMO and [147, 146, 142] aimed to improve the beamforming related techniques. Different algorithms were also investigated to improve the rate adaptation in WiFi [143, 144]. Recently, a rapid picocell switching has also been proposed for wireless transit networks [145]. CombDec is orthogonal to these studies that aim to improve WiFi performance in different aspects. We show that CombDec makes it possible to resolve RTS collisions and pushes WiFi towards a collision-free environment.

3.7 Summary

This chapter provides a systematic study to resolve RTS collisions in WiFi networks. Our core contribution is a new decoding system CombDec that uses (α, β)-construction, γ-decimation and sparse recovery to resolve RTS collisions. CombDec does not require changing the 802.11 standard and redefines the role of the RTS functionality in WiFi. We show via system implementation and extensive evaluation that CombDec has a beneficial impact on WiFi networks and substantially improves the throughput performance by 33.6% – 46.2% in various scenarios.

Chapter 4: Measurement Integrity Attacks against Network Tomography

4.1 Abstract

Network tomography is an important tool to estimate link metrics from end-to-end network measurements. An implicit assumption in network tomography is that observed measurements indeed reflect the aggregate of link performance (i.e., *seeing is believing*). However, it is not guaranteed today that there exists no anomaly (e.g., malicious autonomous systems and insider threats) in large-scale networks. Malicious nodes can intentionally manipulate link metrics via delaying or dropping packets to affect measurements. Will such an assumption render a vulnerability when facing attackers? The problem is of essential importance in that network tomography is developed towards effective network diagnostics and failure recovery.

In this chapter, we demonstrate that the vulnerability is real and propose a new attack strategy, called *measurement integrity attack*, in which malicious nodes can substantially damage a network (e.g., delaying packets) and at the same time maliciously manipulate end-to-end measurement results such that a legitimate node is misleadingly identified as the root cause of the damage (thereby becoming a scapegoat) under network tomography. We formulate three basic attack approaches and show under what conditions attacks can be successful. We also reveal conditions to detect and locate such attacks in a network. Our theoretical and experimental results show that simply trusting measurements leads to measurement integrity vulnerabilities. Thus, existing methods should be revisited accordingly for security in various applications.

4.2 Models and Problem Statement

In this section, we first review network tomography and introduce the basic idea behind measurement integrity attacks. Then, we state our research problems. All notations are defined in Table 4.1.

Table 4.1. Notations used throughout the chapter.

$\mathbf{A}^T$	The transpose of matrix $\mathbf{A}$.
$\mathbf{A}^{-1}$	The inverse of matrix $\mathbf{A}$.
$\|\mathbf{a}\|_1$	The $\mathcal{L}$-1 norm of vector $\mathbf{a} = [a_1, a_2, \cdots, a_n]^T$, i.e., $\|\mathbf{a}\|_1 = \sum_{i=1}^{n} \|a_i\|$.
$\mathbf{x} \succeq \mathbf{y}$	Componentwise larger than or equal to, i.e., $x_i \geq y_i$ for every index i and pair of $x_i \in \mathbf{x}$ and $y_i \in \mathbf{y}$.
$\mathbf{0}$	All-zero vector.
$\|\mathcal{A}\|$	The cardinality of set $\mathcal{A}$.

4.2.1 Network Models and Assumptions

We consider a connected network with a known topology denoted by graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V} = \{v_i\}_{i\in[1,|\mathcal{V}|]}$ and $\mathcal{L} = \{l_i\}_{i\in[1,|\mathcal{L}|]}$ represent the sets of nodes and links, respectively. There is at most one link between nodes v_i and v_j for $i \neq j$ and no link for $i = j$ (i.e., no self-loop). Link l_i is associated with a link metric x_i. We assume that link metrics are additive, i.e., the overall measurement metric of an end-to-end path is the sum of individual link metrics over the path. For example, delay metrics are additive; and packet delivery or loss ratios are also additive in the logarithmic form [31, 148, 47].

Throughout this chapter, we adopt similar assumptions in the literature for network tomography (e.g., [45, 46, 47]): (i) a network operator chooses a number of nodes in the network as monitors, which send probe packets between each other to monitor the additive metric of each individual link; (ii) the network operator will collect all measurement results from monitors, and then perform network tomography for monitoring and diagnosis purposes.

In addition, we adopt the assumption that the monitors can control the routing of probe packets over a path as long as the path starts and ends at different monitors. Although end nodes usually have no control of the routing path of a common IP packet, network tomography relies on such a controllable routing assumption (e.g., [45, 46, 47]). It is known from existing studies (e.g., [149, 150]) that controllable routing served for network measurement can be generally supported in (i) networks under common administration, (ii) networks with strict (or loose) source routing, such as wireless networks with ad-hoc on demand distance vector (AODV) routing, or (iii) certain software-defined network (SDN) scenarios where monitors, with the help of the SDN controller, can decide paths of measurement packets. How exactly controllable routing is designed for network tomography is complementary to the work in this chapter that focuses on exploiting the network tomography process and launching measurement integrity attacks.

4.2.2 Network Tomography and Formulation

Network tomography [36] is an algorithm to estimate link metrics from end-to-end measurements. Theoretically, we form the link metrics as a column vector $\mathbf{x} = [x_1, x_2, \cdots, x_{|\mathcal{L}|}]^T$. To efficiently estimate $\mathbf{x}$, monitors first select a set of measurement paths between each other, denoted by $\mathcal{P} = \{P_i\}_{i\in[1,|\mathcal{P}|]}$. Then, they send probe packets over the paths in $\mathcal{P}$ to obtain the path measurement metrics, which are denoted as a column vector $\mathbf{y} = [y_1, y_2, \cdots, y_{|\mathcal{P}|}]^T$. As a path measurement metric in $\mathbf{y}$ is usually the sum of multiple link metrics in $\mathbf{x}$ (e.g., a path delay is the sum of multiple link delays), it has been shown [44] that the linear relation between $\mathbf{x}$ and $\mathbf{y}$ can be represented as

$$\mathbf{y} = \mathbf{R}\mathbf{x}, \tag{4.1}$$

where $\mathbf{R} = (R_{i,j})$ is called the routing or measurement matrix whose entry $R_{i,j}$ has value 1 if link $l_j \in \mathcal{L}$ is present on path $P_i \in \mathcal{P}$, and value 0 otherwise. Network tomography in essence inverts the linear system in (5.1) to solve for $\mathbf{x}$ given $\mathbf{R}$ and $\mathbf{y}$. Existing studies on selecting or placing monitors (e.g., [46, 45]) ensure that $\mathbf{R}$ is revertible (or full column rank) and the solution to (5.1) can be obtained as

$$\hat{\mathbf{x}} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{y}. \tag{4.2}$$

The estimate $\hat{\mathbf{x}}$ is expected to have values close to the real link metric vector $\mathbf{x}$, and will be used as decisive information for link status monitoring, network diagnostics or further failure recovery.

4.2.3 Motivation and Basic Idea

Network tomography does not directly measure network link performance, but deduces such performance from the aggregate measurements observed by monitors. Therefore, the reliability of network tomography relies on an implicit assumption that measurements over end-to-end paths indeed reflect the real performance aggregates over individual links. However, such probe packets may go through malicious autonomous systems [48, 49], intentional bandwidth throttling systems [151], backdoor-infected routers [50] or attack-captured nodes [51, 52] that can intentionally or maliciously cause negative impacts on end-to-end measurements. Thus, such an assumption may not always hold in today's complicated network environments.

Suppose that some nodes in the network are malicious and intend to cause damage. A straightforward attack is that they delay or drop all packets routed to them. However, it is easy for the network operator to detect that the links connecting to these nodes suffer long delay or high loss under network tomography. Therefore, a much more important question

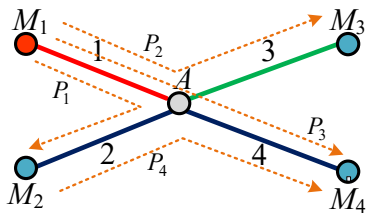


Figure 4.1. A simple network example, where $M_1 - M_4$ are monitors, and M_1 is malicious.

is whether it is possible for these malicious nodes to launch attacks and at the same time mislead network tomography.

To demonstrate the idea of such an attack, we consider a naive scenario shown in Fig. 4.1, where nodes M_1, M_2, M_3 and M_4 are monitors that perform network tomography to estimate link metrics, and the number on each edge denotes the link index. These monitors choose 4 paths listed in Fig. 4.1 for end-to-end measurement. Assume that node M_1 is malicious, which means that it can adversely affect the performance of link 1 to damage the network, such as delaying packets passing through link 1. Note that monitors do not need to enumerate all possible paths between them. They only need to choose a sufficient number of paths to ensure identifiability in network tomography (e.g., [46, 47]). Fig. 4.1 shows such an example with 4 paths chosen.

From Fig. 4.1, the attacker M_1 associated with link 1 presents on paths 1-3. If the attacker M_1 simply delays or drops all packets going through link 1, it is very easy to be identified by network tomography as the root cause. Instead, our proposed attack strategy is that the attacker can try to delay or drop packets along certain directions to mislead tomography. Specifically, in Fig. 4.1, the attacker M_1 is on all measurement paths containing link 3 (i.e., path 2). If the attacker M_1 only does the damage on path 2, and do nothing on other paths (e.g., paths 1, 3), the induced measurements under the network tomography algorithm (4.2) will show that path measurements containing link 3 always suffer long delay or high loss, while the others appear to be normal.

For example in Fig. 4.1, we have the system (5.1) as

$$\begin{aligned} &\text{Path } 1 : y_1 = x_1 + x_2 \\ &\text{Path } 2 : y_2 = x_1 + x_3 \\ &\text{Path } 3 : y_3 = x_1 + x_4 \\ &\text{Path } 4 : y_4 = x_2 + x_4. \end{aligned} \tag{4.3}$$

Clearly, the routing matrix $\mathbf{R}$ in (4.3) satisfies the full column rank requirement. Now suppose an ideal case that the network is congestion free (i.e., almost 0ms delay on every link), and the attacker only damages path 2 by inflicting extra 1000ms delay on link 1. Then, we have the observed path measurement vector $\mathbf{y} = [0, 1000, 0, 0]^T$. According to (5.1), the estimated link metrics become $\hat{\mathbf{x}} = [0, 0, 1000, 0]^T$. Such result indicates that

1. The estimated link vector $\hat{\mathbf{x}}$ shows that no anomaly happens on link 1; therefore, the anomaly of link 1 due to attacker M_1 can be successfully concealed by such an attack strategy against the network tomography.

2. This unavoidably misleads the network operator to believe that link 3 or its end-node M_3 must have some issues.

Therefore, we call such an attack strategy *measurement integrity attack* and call link 3 or nodes M_3 a *scapegoat* in the case.

4.2.4 Problem Statement

From the example in Fig. 4.1, we can consider the measurement integrity attack as a potential attack to hide the real identities of attackers and make some legitimate nodes or links the scapegoats. Many questions can be raised concerning the feasibility of such an attack strategy in the above example: How can M_1 damage the network to launch a feasible

attack? Can M_1 make other links, such as link 2, the scapegoat? Is it possible to detect or locate such an attack?

Before theoretically addressing these issues, we first introduce several necessary definitions. Considering a network $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, we define $\mathcal{V}_m \subseteq \mathcal{V}$ as the malicious nodes set (called attackers), which control a set of links $\mathcal{L}_m \subseteq \mathcal{L}$. Then the attackers can launch the attack by inconsistently inflicting damage on particular paths $\mathcal{P}_m \subseteq \mathcal{P}$, where every entry $P_i \in \mathcal{P}_m$ travels at least one link $l_j \in \mathcal{L}_m$.

According to previous definitions, we unfold our major problems into two aspects as follows.

1. For the attack strategy, network tomography infers the link metrics based on (4.2). Therefore, in order to mislead network tomography, the routing matrix $\mathbf{R}$ is necessary to be known by attackers. However, for most networks, the path information is generally encrypted in network or higher layers, so that the attacker cannot obtain it. Therefore the first challenge is how to launch measurement integrity attacks to delay or drop packets on paths from $\mathcal{P}_m$ in a way such that another set of links $\mathcal{L}_s$ is identified as the root cause and $\mathcal{L}_s \cap \mathcal{L}_m = \varnothing$ (where $\varnothing$ denotes the empty set).

2. For the detection strategy, from the defenders' perspective, they know the global routing matrix $\mathbf{R}$ and the manipulated path measurement vector, but they do not know the true malicious links $\mathcal{L}_m$. Therefore, another challenge is how to detect if measurement integrity attacks exist and how to localize which links have real problems.

We assume all nodes, including norm nodes and monitors, can be malicious in $\mathcal{V}_m$, because they are not dedicated nodes with special protection, but normal nodes representing sources and destinations on measurement paths in the network. A large number of nodes are usually required to be chosen as monitors to ensure identifiability in network tomography [47, 46].

4.3 Attack Strategies

In this section, we formally address the measurement integrity problem. In particular, we categorize measurement integrity attacks into three basic strategies, and then formulate them and discuss their impacts.

4.3.1 Network Link States

The network operator uses network tomography to identify an abnormal link by checking its link metric exhibiting long delay or high loss. Under measurement integrity attacks, a normal link may be misleadingly identified as abnormal. To facilitate formulating such attacks, we first define the normal and abnormal states of a network link.

Definition 2 *Define the state space of a link as* $\mathcal{S} = \{\textsf{normal}, \textsf{abnormal}, \textsf{uncertain}\}$. *Let the state of link* $l_i \in \mathcal{L}$ *be a function* $S : \mathcal{L} \to \mathcal{S}$ *such that* $S(l_i) =$ *abnormal if* l_i*'s link metric* x_i *is larger than an upper bound* b_u *(i.e.,* $x_i > b_u$*), and* $S(l_i) =$ *normal if* x_i *is less than a lower bound* b_l *(i.e.,* $x_i < b_l$*), and* $S(l_i) =$ *uncertain otherwise (i.e., when* $x_i \in [b_l, b_u]$*). In particular, the state* $S(l_i)$ *satisfies*

$$S(l_i) = \begin{cases} \textsf{normal} & x_i < b_l, \\ \textsf{uncertain} & b_l \leq x_i \leq b_u, \\ \textsf{abnormal} & x_i > b_u. \end{cases}$$

Remark 1 *The state of* uncertain *indicates that some links may be in an intermediate state that cannot be clearly classified to* abnormal *or* normal. *There is no standardized definition to clarify all problematic conditions in practical network diagnostics. For example, in an enterprise network, a link can be considered* abnormal *if the link delay is larger than a few seconds, and considered* normal *if the delay is tens of milliseconds (ms). However, when the link delay is hundreds of milliseconds (e.g., 150ms), it really depends on the network operation*

rules in the organization to decide the state of the link. As a result, we introduce the state of uncertain *to accommodate this intermediate state. We also note that our three-state scenario can be easily transitioned into the two-state scenario by setting a single threshold* $b = b_u = b_l$ *in Definition 2.*

With Definition 2, we can say that one of the goals for measurement integrity attacks is to make sure that the links associated with attackers are identified as normal; at the same time, some innocent links are, however, identified as abnormal.

4.3.2 Attack Manipulation Vector and Inflicted Damage

Apparently, except for generating misleading results, a major goal of attackers is to cause damage to the network. Therefore, we also need to measure the damage due to the attacks. The first thing towards measuring the attack damage is to determine what attackers can manipulate. By nature, attackers can affect any end-to-end path that goes through them, accordingly manipulating the end-to-end measurement vector observed at monitors. For example, in Fig. 4.1, node M_1 can obviously affect any data flow going through links 1(e.g., delaying or dropping packets).

Denote by $\mathbf{y}'$ and $\mathbf{y}$ the end-to-end measurement vectors with and without the attack, respectively. Without loss of generality, we can always write

$$\mathbf{y}' = \mathbf{y} + \mathbf{m}, \tag{4.4}$$

where $\mathbf{y}$ reflects the real end-to-end performance, and $\mathbf{m}$ is called the attack manipulation vector that denotes the damage (e.g., intentional delay or packet dropping ratio) inflicted by attacks over all paths. For example, when an end-to-end path has a delay of 10ms, an attacker on the path can incur an extra delay of 1000ms for every packet, making the observed end-to-end measurement 1010ms; and the extra delay of 1000ms can be controlled by the

attacker and will be an entry in $\mathbf{m}$ to represent the damage to the network. Accordingly, each entry in $\mathbf{m}$ reflects the performance degradation induced by the attacker on each path in the network.

All entries in $\mathbf{m}$ should be non-negative in that attackers should not boost, but degrade the network performance, i.e., $\mathbf{m} \succeq \mathbf{0}$, where $\succeq$ means "componentwise greater than or equal to" defined in Table 4.1. For example, attackers can intentionally postpone forwarding packets on paths going through them, thus incurring more delay on those paths. But they are never expected to reduce the delay, because it is in contrast to the attacker's goal to damage the network and it may be technically infeasible for them to further reduce the delay at will. In addition, for the measurement paths that contain no attacker, the corresponding entries in $\mathbf{m}$ must be zero, indicating that attackers cannot manipulate the measurements on these paths. For example, in Fig. 4.1, attacker M_1 is not on path 4, and thus cannot manipulate the measurement of path 4. We formally define these constraints of $\mathbf{m}$ as follows.

Constraint 1 *The attack manipulation vector* $\mathbf{m} = \{m_i\}_{i\in[1,|\mathcal{P}|]}$ *satisfies (i)* $\mathbf{m} \succeq \mathbf{0}$*; and (ii)* $m_i = 0$ *when there exists no such node* $v \in \mathcal{V}_m$ *that is on path* $P_i \in \mathcal{P}$*, where* $\mathcal{V}_m$ *and* $\mathcal{P}$ *denote the sets of malicious nodes and measurement paths, respectively.*

Under the Constraint 1, attackers will attempt to maximize the damage to the network. In the following, we define the damage as total performance degradation over all paths.

Definition 3 *The damage of the measurement integrity attack is measured by* $\|\mathbf{m}\|_1$*, i.e., the* $\mathcal{L}_1$ *norm of attack manipulation vector* $\mathbf{m}$*.*

Remark 2 *Definition 3 defines the damage metric of the attack as the total sum of all entries, representing the total performance degradation over all paths. The larger the value of* $\|\mathbf{m}\|_1$*, the more damage the attack brings. We can also change the damage metric to the average performance degradation or to any other form. For the sake of simplicity, we always*

use the damage metric in Definition 3 for formulation and analysis, and note that the change of the damage metric (e.g., to accommodate the metrics of packet loss or delivery ratios) is a straightforward extension in mathematical manipulations.

4.3.3 Partial Information

According to (4.2), in order to manipulate the inferred link metrics $\hat{\mathbf{x}}$, the global routing information lying in the routing matrix $\mathbf{R}$ and the original overall path measurement vector $\mathbf{y}$ should be known by attackers. For example, in Fig. 4.1, if M_1 knows the linear system (4.3) and $\mathbf{y} = [0, 0, 0, 0]^T$, then M_1 can scapegoat link 3 by introducing $\mathbf{m} = [0, 1000, 0, 0]$. However, in practice, from the security perspective, both $\mathbf{R}$ and $\mathbf{y}$ are less likely open to every node in most networks. Therefore, before introducing the attack strategy, we need to clarify what information that attackers know while launching attacks.

Definition 4 *Attackers know the routing information and path measurement vector on paths that go through them (i.e., paths in $\mathcal{P}_m$). Specifically, for each path $P_i \in \mathcal{P}_m$, the routing information $\mathbf{r}_i$ (i.e., ith row of the routing matrix $\mathbf{R}$), and the true path measurement metric $y_i \in \mathbf{y}$ are known by attackers. The partiality factor β is defined as the ratio between the number of paths in $\mathcal{P}_m$ and the total number of paths, i.e., $\beta = |\mathcal{P}_m|/|\mathcal{P}|$.*

Remark 3 *The partiality factor β is a ratio indicating how much information that attackers know. $\beta = 1$ denotes attackers completely know $\mathbf{R}$ and $\mathbf{y}$, and $0 < \beta < 1$ means attackers only know several rows of $\mathbf{R}$ and associated several entries of $\mathbf{y}$. In fact, β is also positively related to the number of attackers in a network. For example, $\beta = 1$ denotes attackers are present on all paths in a network, and $\beta = 0$ means there is no attacker in the network.*

Remark 4 *In many networks, attackers may acquire the path measurement vector and the routing information of paths that go through them. For example, in AODV, the routing*

information is available during the routing discover process, and the path measurement vector can be obtained if the attacker is the source or destination node of a path.

According to Definition 4, to clearly model the partiality, we split $\mathbf{R}$ and $\mathbf{y}$ into two parts, i.e., $\mathbf{R} = [\mathbf{R}_d^T, \mathbf{R}_r^T]^T$ and $\mathbf{y} = [\mathbf{y}_d^T, \mathbf{y}_r^T]^T$, where $\mathbf{R}_d$ and $\mathbf{y}_d$ denote the parts that attackers know, and $\mathbf{R}_r$ and $\mathbf{y}_r$ are unknown to attackers. Similarly, according to Constraint 1, we also divide the attack manipulation vector as $\mathbf{m} = [\mathbf{m}_d^T, \mathbf{0}^T]^T$, where $\mathbf{m}_d^T$ denotes the inflicted damage on paths in $\mathcal{P}_m$. Then, the linear system (4.4) can be written as

$$\mathbf{y}' = \begin{bmatrix} \mathbf{y}_d \\ \mathbf{y}_r \end{bmatrix} + \mathbf{m} = \begin{bmatrix} \mathbf{R}_d \\ \mathbf{R}_r \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{m}_d \\ \mathbf{0} \end{bmatrix}. \tag{4.5}$$

Attackers have no knowledge on entries in $\mathbf{R}_r$ and $\mathbf{y}_r$, therefore we consider $\mathbf{R}_r$ and $\mathbf{y}_r$ as a random matrix and a random vector, respectively. If the partiality factor $\beta = 1$, then $\mathbf{y}_r$ and $\mathbf{R}_r$ will vanish and $\mathbf{m}_d = \mathbf{m}$, indicating attackers know and can control all paths. If $\beta = 0$, $\mathbf{m}_d$ will vanish, meaning that this network cannot be damaged by attackers, therefore in order to investigate the malicious behaviors, in this chapter, we only consider the scenario $0 < \beta \leq 1$.

4.3.4 Attack Strategy

Given the partial knowledge $\mathbf{R}_d$ and $\mathbf{y}_d$, one major goal of attackers is to find an attack manipulation vector $\mathbf{m}$, such that inferred link metrics $\hat{\mathbf{x}}$ can scapegoat victim links as the root cause. However, the unknown information in $\mathbf{R}_r$ and $\mathbf{y}_r$ renders uncertainties for attackers to find $\mathbf{m}$. Therefore, we need a deterministic rule to guide attackers to address such uncertainties.

According to Definition 2, the scapegoating purpose represents that the estimated metric $\hat{x}_i$ for link $l_i \in \mathcal{L}$ must meet certain conditions to be in normal, abnormal, or uncertain state.

This means that we can write the goal as

$$\mathbf{s}_l \preceq \hat{\mathbf{x}} \preceq \mathbf{s}_u, \tag{4.6}$$

where $\mathbf{s}_u$ and $\mathbf{s}_l$ are called the upper and lower bound vectors. By controlling $\mathbf{s}_u$ and $\mathbf{s}_l$, we can accommodate various scapegoating purposes. Inserting (4.2) and (4.4) into (4.18), we have

$$\mathbf{s}_l \preceq (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T(\mathbf{y}+\mathbf{m}) \preceq \mathbf{s}_u. \tag{4.7}$$

The normal link delay in a network is usually small (e.g., several or tens of milliseconds). By contrast, an attacker should significantly delay packets (e.g., by more than thousands of milliseconds) to cause damage to the network. Therefore, the true path measurement vector $\mathbf{y}$ should be of lesser magnitude than the attack manipulation vector $\mathbf{m}$ (i.e., $\mathbf{y}+\mathbf{m} \approx \mathbf{m}$). Then (4.7) can be approximated by

$$(\mathbf{R}_d{}^T\mathbf{R}_d + \mathbf{R}_r{}^T\mathbf{R}_r)\mathbf{s}_l \preceq \begin{bmatrix}\mathbf{R}_d^T & \mathbf{R}_r^T\end{bmatrix} \begin{bmatrix}\mathbf{m}_d \\ \mathbf{0}\end{bmatrix} \preceq (\mathbf{R}_d{}^T\mathbf{R}_d + \mathbf{R}_r{}^T\mathbf{R}_r)\mathbf{s}_u. \tag{4.8}$$

Obviously, if attackers know the routing matrix $\mathbf{R}$ completely, $\mathbf{m}_d$ can be solved from (4.8) through standard linear programming [152]. However, with the random matrix $\mathbf{R}_r$, it is infeasible to solve $\mathbf{m}_d$ directly. Therefore, we need a constraint to convert such random matrix $\mathbf{R}_r$ to a deterministic one. By considering the worst case, we have the following constraint.

Constraint 2 *Given the upper bound* $\mathbf{s}_u$ *and lower bound* $\mathbf{s}_l$ *of* $\hat{\mathbf{x}}$*, the partial routing matrix* $\mathbf{R}_d$ *and vector* $\mathbf{m}_d$ *in the attack manipulation vector* $\mathbf{m}$ *satisfies*

$$(\mathbf{R}_d{}^T\mathbf{R}_d + \mathbf{R}_r^{+T}\mathbf{R}_r^{+})\mathbf{s}_l \preceq \mathbf{R}_d^T\mathbf{m}_d \preceq (\mathbf{R}_d{}^T\mathbf{R}_d + \mathbf{R}_r^{-T}\mathbf{R}_r^{-})\mathbf{s}_u, \tag{4.9}$$

where

1. $\mathbf{R}_r^+ = \arg\sup_{\mathbf{R}_r}(\mathbf{R}_r{}^T\mathbf{R}_r\mathbf{s}_l)$, *and* $\mathbf{R}_r^+$ *does not contain two identical rows and all* 0 *row.*

2. $\mathbf{R}_r^- = \arg\inf_{\mathbf{R}_r}(\mathbf{R}_r{}^T\mathbf{R}_r\mathbf{s}_u)$, *and* $\mathbf{R}_r^-$ *does not contain two identical rows and all* 0 *row.*

Remark 5 *Constraint 2 converts the random matrix* $\mathbf{R}_r$ *into deterministic matrices* $\mathbf{R}_r^+$ *and* $\mathbf{R}_r^-$*, which maximize and minimize* $\mathbf{R}_r{}^T\mathbf{R}_r\mathbf{s}_l$*, respectively. Constraint 2 can be considered as the worst case of* (4.8) *because it shrinks the solution space of* $\mathbf{m}_d$*, i.e., it is easy to know that for any solution* $\mathbf{m}_d^*$ *satisfying* (4.9)*, it must also satisfy* (4.8)*.*

4.3.5 Formulation of Measurement Integrity Attacks

Measurement integrity attacks aim to bring damage to a network, and at the same time hide the attacker-controlled link set $\mathcal{L}_m$ but expose another set of victim links $\mathcal{L}_s$ as scapegoats to network tomography. The basic idea to implement measurement integrity attacks is that attackers cooperatively inflict damage on paths which contain victims, and do nothing on other paths. Based on this idea, attackers can choose different strategies to launch the attacks. Specifically, we consider three basic strategies: (i) chosen-victim attacks, where the victim link set $\mathcal{L}_s$ is already chosen and targeted, (ii) maximum-damage attacks, where attackers aim at finding the best victim link set $\mathcal{L}_s$ to maximize their damage, (iii) obfuscation, where attackers attempt to make network tomography show no evident performance outliers but uniform degradation over a substantial number of links.

Fig. 4.2 shows an illustrative example of how different attack strategies affect the link delay metrics obtained by network tomography. In Fig. 4.2, solid lines represent the values of end-to-end delay metrics and each dotted line denotes the envelope of the solid line under the same scapegoating strategy. We see from Fig. 4.2 that there are 10 links, and links 1 and 2 are controlled by attackers. Under chosen-victim scapegoating, the attackers choose links 5 and 6 to be scapegoats that exhibit much higher delays than other links. Under maximum-

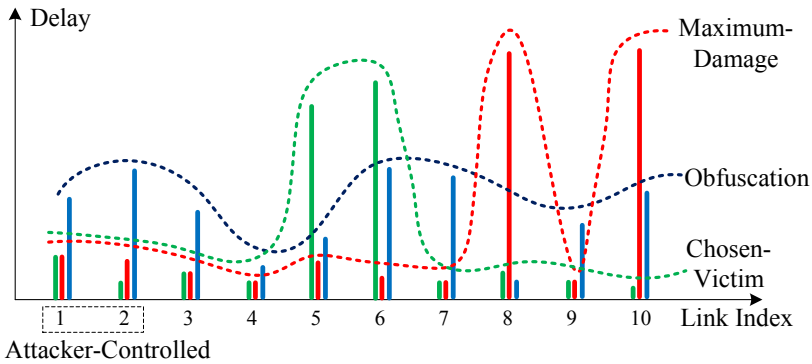


Figure 4.2. Examples of link metrics under tomography for three strategies.

damage scapegoating, the attackers found that links 8 and 10 can be the scapegoats with highest delays. Under obfuscation, the attackers can make most links exhibit similarly delays, which can confuse the network operator to find which links are truly problematic.

In the following, we mathematically formulate these attack strategies.

4.3.5.1 Chosen-Victim Scapegoating

When the victim set $\mathcal{L}_s$ is already given, this strategy can be formulated as choosing the best attack manipulation vector $\mathbf{m}$ to maximize the attack damage, at the same time satisfying the constraints for $\mathbf{m}$, $\mathcal{L}_m$, and $\mathcal{L}_s$. According to Constraint 1 and 2 and Definitions 2 and 3, we can formulate this basic scapegoating strategy as

$$\begin{aligned} \underset{\mathbf{m}}{\text{maximize}} \quad & \|\mathbf{m}\|_1 , && (4.10) \\ \text{subject to} \quad & \mathbf{m} \text{ satisfies Constraint 1 and 2,} \\ & S(l) = \mathsf{normal},\ \forall\, l \in \mathcal{L}_m, && (4.11) \\ & S(l) = \mathsf{abnormal},\ \forall\, l \in \mathcal{L}_s, && (4.12) \\ & \mathcal{L}_m \cap \mathcal{L}_s = \varnothing, && (4.13) \end{aligned}$$

where constraints (4.11) and (4.12) mean that all links associated with the attackers should appear normal, and all links in the victim set should be abnormal, respectively. These two together, combined with constraint (4.13), achieve the goal of scapegoating under network tomography.

4.3.5.2 Maximum-Damage Scapegoating

If the attackers aim to bring maximum damage to the network, they may do so by searching the best victim set in the set of all links. Therefore, maximum-damage scapegoating can be written as

$$\begin{aligned} \underset{\mathbf{m},\mathcal{L}_s \subset \mathcal{L}}{\text{maximize}} \quad & \|\mathbf{m}\|_1 , \\ \text{subject to} \quad & \mathbf{m} \text{ satisfies Constraint 1 and 2,} \\ & \text{Constraints in (4.11), (4.12), and (4.13).} \end{aligned} \tag{4.14}$$

4.3.5.3 Obfuscation

Different from the chosen-victim and maximum-damage attacks, the idea behind obfuscation is to make every link look mostly similar without evident outliers. Obfuscation does not necessarily lead to a unique strategy. As long as a strategy makes a substantial number of link metrics look approximately similar, and at the same time incurs damage to the network, it should be considered as a successful obfuscation one. We leverage the state of

uncertain in Definition 2 to define obfuscation as follows.

$$\underset{\mathbf{m},\mathcal{L}_s\subset\mathcal{L}}{\text{maximize}} \quad \|\mathbf{m}\|_1\,, \tag{4.15}$$

$$\text{subject to} \quad \mathbf{m} \text{ satisfies Constraint 1 and 2,}$$

$$S(l) = \mathsf{uncertain},\ \forall\ l \in \mathcal{L}_o = \mathcal{L}_s \cup \mathcal{L}_m, \tag{4.16}$$

$$\mathcal{L}_s \neq \varnothing, |\mathcal{L}_o| > \gamma|\mathcal{L}| \tag{4.17}$$

where $\mathcal{L}_s$ is the set of victim links that attackers want to find such that any link $l \in \mathcal{L}_o$ is manipulated under network tomography to be in the uncertain state defined in (4.16). γ is a predefined threshold ratio indicating the lower bound of the number of infected links. As we have mentioned, the uncertain state represents an intermediate state, in which a link cannot be clearly classified into either normal or abnormal. Hence, a substantial number of links (i.e., more than $\gamma|\mathcal{L}|$) in the uncertain state result in obfuscation.

Given these formally defined basic strategies, attackers are able to launch scapegoating attacks against network tomography to maximize the damage, make scapegoats, or obfuscate the network operator. In addition, attackers may also develop more sophisticated strategies based upon these three ones.

4.4 Feasibility and Detectability

After we formulate measurement integrity attack strategies, two questions naturally follow: (i) Whether these attacks are indeed feasible (i.e., whether feasible solutions exist in the optimization-based strategies)? (ii) Can we detect or locate such an attack if it is successfully launched? In this section, we answer these two questions by first analyzing the feasibility of the attack, then describing how to detect it. The method to locate attacks is discussed in Section 4.5.

4.4.1 Feasibility Analysis

Whether an attack is feasible depends on the network connectivity, selections of measurement paths, and where attackers are. Consider a simple example in Fig. 4.3(a): Attackers A_1 and A_2 aim to manipulate the end-to-end measurements to scapegoat the link between nodes C and D. They should be able to succeed if they are on all the measurement paths that go through the link between C and D. We say it is a *perfect cut* case, in which for any measurement path $P \in \mathcal{P}$ containing a victim link, there always exists a malicious node $v \in \mathcal{V}_m$ present on that path P. Fig. 4.3(b) illustrates an *imperfect cut* case, in which the path $M_1 \rightarrow B \rightarrow C \rightarrow D \rightarrow M_4$ contains neither A_1 nor A_2.

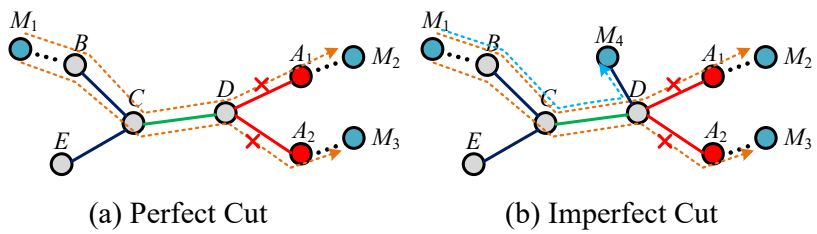


Figure 4.3. Perfect and imperfect cuts by two attackers.

4.4.1.1 Perfect Cut

We show in the following that a perfect cut always leads to a successful attack in any strategy.

Theorem 1 *A measurement integrity attack is always feasible if the set of malicious nodes $\mathcal{V}_m$ can perfectly cut the set of victim links $\mathcal{L}_s$ from all measurements paths.*

Proof: We do not need to prove the feasibility of all three strategies because the maximum-damage scapegoating (4.14) must be feasible if chosen-victim one (4.10) is feasible. Then, we write (4.10) and (4.14) into a generic form, which is (4.18), i.e., $\mathbf{s}_l \preceq \hat{\mathbf{x}} \preceq \mathbf{s}_u$,. By adjusting

s_u and s_l, we can accommodate either chosen-victim scapegoating or obfuscation because constraints (4.11), (4.12) and (4.16) indicate the estimated $\hat{\mathbf{x}}$ must meet certain conditions.

Then we need to show that for a given manipulated metric vector $\hat{\mathbf{x}}^*$ satisfying (4.18), there exists a resultant vector $\mathbf{m}^*$ that meets Constraints 1 and 2. Because Constraint 2 is derived from (4.18) in Section 4.3.4. According to (4.7) and (4.8), it is clear that $\mathbf{m}^*$ satisfies Constraint 2. Thus we only need to show the proof under Constraint 1.

We have developed three strategies, but do not need to prove the feasibility individually. It is easy to know that if the chosen-victim scapegoating (4.10) is feasible, the maximum-damage one (4.14) is also feasible. Then, we write the chosen-victim scapegoating (4.10) and obfuscation (4.15) into a generic form, and show that a feasible solution exists in the generic form when the set of malicious nodes $\mathcal{V}_m$ can perfectly cut the set of victim links $\mathcal{L}_s$ from all measurements paths.

According to Definition 2, the constraints in (4.12), (4.13) and (4.16) represent that the estimated metric $\hat{x}_i$ for link $l_i \in \mathcal{L}$ must meet certain conditions to be in normal, abnormal, or uncertain state. This means that we can write these constraints as

$$\mathbf{s}_u \succeq \hat{\mathbf{x}} \succeq \mathbf{s}_l, \tag{4.18}$$

where $\hat{\mathbf{x}}$ is link metric vector estimated by network tomography, $\mathbf{s}_u$ and $\mathbf{s}_l$ are called the upper and lower bound vectors. By controlling $\mathbf{s}_u$ and $\mathbf{s}_l$, we can accommodate either chosen-victim scapegoating or obfuscation.

Therefore, we only need to show that for a given manipulated metric vector $\hat{\mathbf{x}}^*$ satisfying (4.18), there exists a resultant vector $\mathbf{m}^*$ that meets Constraint 1 for successful scapegoating. To this end, we can first write the measurement model (5.1) as

$$\mathbf{y}' = \mathbf{y}^* + \mathbf{m}^* = \mathbf{R}\hat{\mathbf{x}}^*, \tag{4.19}$$

where $\mathbf{y}'$ is the observed measurement vector under scapegoating, and $\mathbf{y}^*$ is the true measurement vector if there is no scapegoating and satisfies

$$\mathbf{y}^* = \mathbf{R}\mathbf{x}^*, \tag{4.20}$$

with $\mathbf{x}^*$ being the true link metric vector.

It follows from (4.19) and (4.20) that

$$\mathbf{m}^* = \mathbf{R}\Delta\hat{\mathbf{x}}^*, \tag{4.21}$$

where $\Delta\hat{\mathbf{x}}^* = \hat{\mathbf{x}}^* - \mathbf{x}^*$. For the i-th entry m_i^* in $\mathbf{m}^*$, we have

$$m_i^* = \sum_j R_{i,j}\Delta\hat{x}_j^*, \tag{4.22}$$

where $R_{i,j}$ is the (i,j)-th entry in routing matrix $\mathbf{R}$ and $\Delta\hat{x}_j^*$ is the j-th entry of $\Delta\hat{\mathbf{x}}^*$.

Because $\mathcal{V}_m$ is a perfect cut, if there is no attacker on path $P_i \in \mathcal{P}$, there will be no victim link on path P_i as well. This indicates that $R_{i,j} = 0$ if link $l_j \in \mathcal{L}_m \cup \mathcal{L}_s$. In addition, if link $l_j \notin \mathcal{L}_m \cup \mathcal{L}_s$, $\Delta\hat{x}_j^* = 0$ as the attackers do not manipulate the metric of link l_j. Combining the two observations, we obtain $m_i^* = 0$ if there is no attack on path P_i, which satisfies Constraint 1 thus completes the proof. □

4.4.1.2 Imperfect Cut

If attackers only form an imperfect cut of the victim links, the formulation of an attack strategy may not always yield a feasible solution, which depends on specific network settings. We are interested in understanding the attack success probability under generic random assumptions (i.e., we do not use specific distribution models such as power-law network connectivity, but only assume that network connectivity, placement of monitors, and selection

of measurement paths are random in the network). We show that it increases with the increasing of the number of measurement paths that include at least one victim link and at least one attacker.

Theorem 2 *The success probability of a measurement integrity attack is defined as the probability that an attack strategy yields a feasible solution. Under generic random assumptions, the success probability is an increasing function of the number of measurement paths that include at least one victim link and at least one attacker.*

Proof: Let $L = |\mathcal{P}|$ be the total number of measurement paths in network tomography. Assume that attackers $\mathcal{V}_m$ are present on $k < L$ measurement paths. Define a vector space $\mathcal{M}_k^L = \{\mathbf{v} \,|\, \mathbf{v} \in \mathbb{R}^{L\times 1}$, and $L-k$ entries in $\mathbf{v}$ are always zeros$\}$, where $\mathbb{R}^{L\times 1}$ denotes the L-dimensional vector space. It is clear that for any $s \geq k$, it always holds that

$$\mathcal{M}_k^L \subset \mathcal{M}_s^L. \tag{4.23}$$

It is clear that the attack manipulation vector $\mathbf{m}_k \in \mathcal{M}_k^L$. The success probability p can be denoted as

$$p(\mathbf{m}_k) \;\; = \;\; \mathbb{P}\left(\mathcal{E}(\mathbf{m}_k)\right), \tag{4.24}$$

where $\mathcal{E}(\mathbf{m}_k)$ denotes a set satisfying

$$\mathcal{E}(\mathbf{m}_k) = \left\{\mathbf{m} | \mathbf{m} \in \mathcal{M}_k^L, \text{and } \mathbf{m} \text{ is a feasible solution}\right\}. \tag{4.25}$$

Now, consider the scenario that the total number of paths used in tomography is still fixed to L, but increase the total number of infected end-to-end measurement paths. Specifically, there are $s > k$ paths that include the victim links and at least one attack link, i.e., less

entries in the new manipulation vector $\mathbf{m}_s$ are always zeros. The success probability becomes

$$p(\mathbf{m}_s) \;=\; \mathbb{P}\left(\mathcal{E}(\mathbf{m}_s)\right), \tag{4.26}$$

where

$$\mathcal{E}(\mathbf{m}_s) = \left\{\mathbf{m} | \mathbf{m} \in \mathcal{M}_s^L, \text{and } \mathbf{m} \text{ is a feasible solution}\right\}. \tag{4.27}$$

Therefore, it suffices to prove $p(\mathbf{m}_k) < p(\mathbf{m}_s)$, or equivalently to prove

$$\mathbf{a} \in \mathcal{E}(\mathbf{m}_s), \forall\, \mathbf{a} \in \mathcal{E}(\mathbf{m}_k), \tag{4.28}$$

which immediately follows from (4.23), (4.25), and (4.27). □

4.4.2 Detecting Measurement Integrity Attacks

We have analyzed the feasibility of measurement integrity attacks. If an attack is successfully launched, we should never trust the result obtained by network tomography. It is necessary to know how to detect such an attack in a network. Our insight is that attackers have to manipulate packet delivery in certain directions to make scapegoating possible in the network. This means that if we verify the estimated link metric vector $\hat{\mathbf{x}}$, which can be obtained by (4.2), with observed measurement vector $\mathbf{y}'$ in all entries, it is likely to observe the inconsistency under the measurement model (5.1) in the presence of a measurement integrity attack. In other words, verifying $\hat{\mathbf{x}}$ and $\mathbf{y}'$ according to (5.1) results in our detection method

$$\text{scapegoating} \begin{cases} \text{exists}, & \text{if } \mathbf{R}\hat{\mathbf{x}} \neq \mathbf{y}', \\ \text{does not exist}, & \text{if } \mathbf{R}\hat{\mathbf{x}} = \mathbf{y}'. \end{cases} \tag{4.29}$$

with the following detectability.

Theorem 3 *Under the detection mechanism* (4.29)*, a measurement integrity attack is undetectable if attackers* $\mathcal{V}_m$ *can perfectly cut victim links* $\mathcal{L}_s$ *from measurement paths or* $\mathbf{R}$ *is a square matrix; and is detectable otherwise.*

Proof: The proof is partly based on the proof for Theorem 1.

First, if $\mathbf{R}$ is a square matrix, it is easy to verify that $\mathbf{R}\hat{\mathbf{x}} = \mathbf{y}'$ always holds. Therefore, it is not possible to detect scapegoating under the linear model (5.1).

Then, we consider that $\mathbf{R}$ is not a square matrix. If attackers $\mathcal{V}_m$ can perfectly cut victims $\mathcal{L}_s$, the attackers can always choose an attack manipulation vector $\mathbf{m}^*$ such that $\mathbf{R}\hat{\mathbf{x}} = \mathbf{y}'$ as shown in (4.19). Therefore, no inconsistency can be found in the detection method (4.29).

If attackers $\mathcal{V}_m$ do not perfectly cut victims $\mathcal{L}_s$, there always exists at least one path on which there is no attacker but at least a victim link with metric manipulated. This means that the observed measurement on this path in the sum of all true link metrics because there is no attacker on the path. However, because the metric of the victim link is manipulated by attackers, the observed measurement will be inconsistent with the sum of the manipulated link metrics. Consequently, $\mathbf{R}\hat{\mathbf{x}} \neq \mathbf{y}'$, meaning the existence of scapegoating. □

Remark 6 *Theorem 3 shows that if attackers* $\mathcal{V}_m$ *can perfectly cut victim links from measurement paths, there is no way to detect them based on the inconsistency check. This is intuitively true. For example, in Fig. 4.3(a), attackers* A_1 *and* A_2 *cut the victim link between nodes* C *and* D *completely from the measurement paths* $M_1 \rightarrow M_2$ *and* $M_1 \rightarrow M_3$*. Any information about the victim link is from these two paths whose measurements can be surely manipulated by the attackers to evade the detection.*

Remark 7 *In practice, even when there is no attack,* $\mathbf{R}\hat{\mathbf{x}}$ *may not exactly equal to* $\mathbf{y}'$ *in* (4.29) *due to randomness in packet delivery and measurement error. Therefore, the attack detection can be slightly modified to test* $\|\mathbf{R}\hat{\mathbf{x}} - \mathbf{y}'\|_1 > \alpha$*, where* α *is a given threshold that can be empirically determined.*

4.5 Locatability Analysis

According to the detection mechanism (4.29), if an attack is successfully launched and it is also detected, the inferred link performance $\hat{\mathbf{x}}$ obtained by network tomography becomes untrusted. Therefore, it is necessary to know which nodes or links are the attackers. In this section, we discuss how to locate the real attackers in the network. We first present the design motivation, then present the method of locating attacks.

4.5.1 Motivation and Example

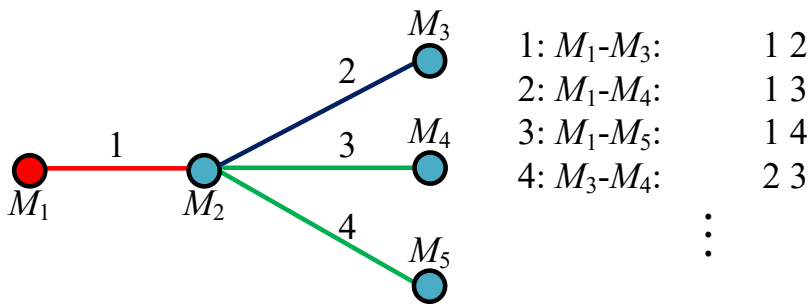


Figure 4.4. A simple network consisting of 5 nodes and 4 links.

The key idea of the measurement integrity attack is that attackers only damage the paths that contain victim links and do nothing to other paths. Therefore, a malicious link used by attackers to cause damages should be present on multiple paths, i.e., some of them contain victim links and others do not. However, if the link controlled by a attacker is the only shared link in a network, then the only explanation for the inconsistency in (4.29) is that this shared link is malicious, because it is the only link that can really inflict the traffic differentiation among different paths.

For example, in a simple network consisting of 5 nodes and 4 links (shown in Fig. 4.4), we consider three paths (i.e., path 1: $M_1 \rightarrow M_3$, path 2: $M_1 \rightarrow M_4$, and path 3: $M_1 \rightarrow M_5$), the only shared link among them is link 1. Assume link 1 is malicious and controlled by attacker

M_1, aiming to scapegoat links 3 and 4. Suppose if the presence of an attacker is detected among these three paths, we can pinpoint that link 1 is malicious. However, according to Theorem 3, only using paths 1-3 is insufficient to detect the attacker link 1 since link 1 can perfectly cut both links 3 and 4.

In order to detect the attacker, knowledge of extra paths is needed. Our design is based on the concept of path pairs in network neutrality inference [40]. A path pair $\{P_i, P_j\}$ is a single path which includes all links traveled by at least one of path i or path j. For example, path pair $\{P_2, P_3\}$ is formed by links 1, 3 and 4. In the real world, this path pair is measurable if M_2 is a monitor such that we can measure link 4 and path 2 in a same time period and then combine them together. Note that we are able to detect the attack by using other normal paths, such as path 4: $M_3 \rightarrow M_2 \rightarrow M_4$, however, if so, we cannot put the blames only on the link 1.

Now we use the path pair $\{P_2, P_3\}$, combined with paths 1-3 to locate link 1. Considering the existence of the attacker, the measurement model (5.1) can be expressed as

$$\begin{aligned} \text{Path } 1 : y'_1 &= x_1 + x_2 \\ \text{Path } 2 : y'_2 &= x_1 + x_3 \\ \text{Path } 3 : y'_3 &= x_1 + x_4 \\ \text{Path Pair } \{P_2, P_3\} : y'_4 &= x_1 + x_3 + x_4. \end{aligned} \tag{4.30}$$

Without loss of generality, and to clearly show the inconsistency, we assume the network, shown in Fig. 4.4, is congestion free, thus delays only come from the attacker. To scapegoat links 3 and 4, link 1 introduces extra delays (e.g., 1000ms) on paths 2 and 3. Then the measurements of these four paths are $\mathbf{y}' = [0, 1000, 1000, 1000]^T$. It is easy to observe that a contradiction happens among these measurements: y'_1 indicates that $x_1 = 0$, whereas y'_2, y'_3 and y'_4 indicate $x_1 = 1000$ and $x_3 = x_4 = 0$. If we apply the detection mechanism (4.29) into

the system (4.30), this contradiction will definitely lead to the inconsistency. Then link 1 can be located because it is the only reason for the inconsistency under network neutrality inference.

From this example, we have three observations to locate measurement integrity attacks leveraging network neutrality inference:

1. To locate the malicious link, we should be able to find a path set, in which the malicious link is the only shared link. For example, link 1 is the only shared link among paths 1-3.

2. Path pairs formed by the paths in the path set should be measurable. For example, the path pair $\{P_2, P_3\}$ is measurable and it is necessary because it guarantees the unique solution to the last three equations.

3. If the shared link is malicious, to observe the contradiction, at least two paths should contain different victim links, and at lease one path should not contain a victim link. For example, path 1 does not contain victim links, and paths 2 and 3 contain links 3 and 4 respectively. Then, we can use path 2, path 3 and their path pair $\{P_2, P_3\}$ to conclude link 1 is malicious, and use path 1 to contradict that link 1 is congestion free.

The basic idea in the previous example is to use inconsistency to locate a malicious link. However, inconsistency may occur regardless of the presence of an attacker. There are two types of inconsistency: (i) inconsistency resulted by measurement noise; and (ii) inconsistency incurred by attackers. For the first type, even when there is no attacker in a network, inconsistency is still possible to occur since random measurement noise is inevitable while probing the network. However, this type of inconsistency is slight, so it can be solved by setting a threshold of measurement according to [40].

For the second type, if the inconsistency is larger than the threshold, we consider that the inconsistency is resulted by attackers. According to Theorem 3, once inconsistency occurs, we can detect attacks happened in the network. But not all inconsistencies can be used

to locate malicious links. Therefore, in the following, we elaborate how to locate malicious links when attackers exist. Note that the locating process is triggered only when an attack is detected by (4.29).

4.5.2 Locating Measurement Integrity Attacks

From the example in Fig. 4.4, we iteratively inspect the maliciousness of each link throughout the entire network to locate attackers. For any link $l_\alpha \in \mathcal{L}$ in a network $\mathcal{G}$, to judge whether link l_α is malicious, we first create a path set $\mathcal{P}_\alpha \subseteq \mathcal{P}$ based on the link l_α. Then, we use $\mathcal{P}_\alpha$ to form a new system, such as the example shown in (4.30), and apply the detection mechanism (4.29) to this new system, to check if link l_α is malicious. Therefore, our attack-locating mechanism consists of two parts: (i) path set generation, and (ii) maliciousness discrimination. In the following, we elaborate each part in detail.

4.5.2.1 Path Set Generation

Directly applying the detection mechanism (4.29) to the overall path set $\mathcal{P}$ will only yield a binary result to show whether attacks exist or not, and cannot locate which links or nodes are really malicious. The purpose of creating a new path set is that we can attribute the inconsistency in (4.29) to a certain link. Specifically, to locate link l_α, the path set $\mathcal{P}_\alpha$ can be formed as follows:

1. Find all path pairs $\{P_i, P_j\}$ where $P_i, P_j \in \mathcal{P}$, such that the shared link between P_i and P_j is l_α.

2. Add P_i and P_j and their corresponding path pair $\{P_i, P_j\}$ to the path set $\mathcal{P}_\alpha$, i.e.,

$$\mathcal{P}_\alpha = \mathcal{P}_\alpha \cup \{P_i, P_j, \{P_i, P_j\}\}.$$

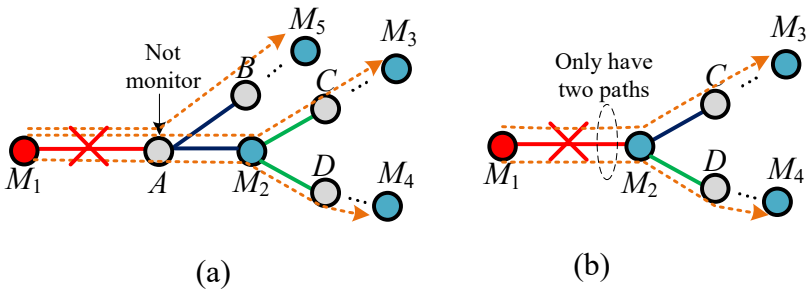


Figure 4.5. Examples of unlocalizable scenarios.

4.5.2.2 Maliciousness Discrimination

After obtaining the path set $\mathcal{P}_\alpha$, we can only focus on inspecting the measurements of paths in $\mathcal{P}_\alpha$. Then we write the measurement model as

$$\mathbf{y}' = \mathbf{R}_\alpha \mathbf{x}, \tag{4.31}$$

where $\mathbf{R}_\alpha$ is the routing matrix formed with respect to the path set $\mathcal{P}_\alpha$.

Theorem 4 *Given a link l_α and its measurement model* (4.31)*, the link l_α is malicious if scapegoating exists in* (4.31) *based on the detection mechanism* (4.29)*.*

Proof: If scapegoating is detected by the detection mechanism (4.29) in (4.31), it is clear that $\mathbf{R}_\alpha \hat{\mathbf{x}} \neq \mathbf{y}'$. This indicates that directly solving (4.31) as a linear equation system will yield no solution. Then, according to Lemma 2 in [40], link l_α must have different link rates for different paths. Therefore, link l_α is malicious. □

4.5.3 Locatability

In the following, we present the network condition to indicate when an attack link can be located in the network.

Theorem 5 *For an attack link l_α, and its path set $\mathcal{P}_\alpha$, if the following conditions hold:*

1. *for any path $P_i \in \mathcal{P}_\alpha$, nodes between link l_α and other links must be monitors;*

2. *the path set $\mathcal{P}_\alpha$ should contain at least three individual paths and two path pairs, in which both paths in one path pair contain victim links and the other path pair must contain at least one path which does not travel any victim links;*

then l_α is localizable.

Proof: First, if condition 1) holds, it is easy to verify that all path pairs in the path set $\mathcal{P}_\alpha$ are measurable. Then in the following, we prove this theorem by showing that if condition 2) holds, inconsistency exists in the measurement model (4.31). Considering an arbitrary path pair $\{P_i, P_j\} \in \mathcal{P}_\alpha$, the measurable model can be written as

$$\begin{aligned} \text{Path } P_i &: y'_i = x_\alpha + x^*_{i\alpha} \\ \text{Path } P_j &: y'_j = x_\alpha + x^*_{j\alpha} \\ \text{Path Pair } \{P_i, P_j\} &: y'_{ij} = x_\alpha + x^*_{i\alpha} + x^*_{j\alpha}, \end{aligned} \tag{4.32}$$

where $x^*_{n\alpha}$ is the adjacent link of x_α on the path P_n, which is resulted from deleting the link l_α from path P_n, and $n \in \{i, j\}$. It is easy to know that (4.32) has the unique solution

$$x_\alpha = y'_i + y'_j - y'_{ij}. \tag{4.33}$$

Then we consider three scenarios: (i) both P_i and P_j contain victim links; (ii) only one path (e.g., P_i) contains victim links; and (iii) neither P_i nor P_j includes victim links.

The ground truth metric $\tilde{x}_\alpha$ of link l_α can always be the solution to scenarios (ii) and (iii). It is easy to verify for scenario (iii) since there is no extra damage on all paths. For scenario (ii), $\tilde{x}_\alpha$ still can be the solution because the extra damage is inflicted to the adjacent link $x^*_{i\alpha}$. The proof of Lemma 3 in [40] shows that the solution to scneario (i) is exactly

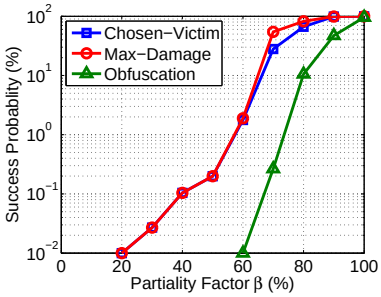


Figure 4.6. The success probabilities versus partiality factor β for three types attackers in wireline network.

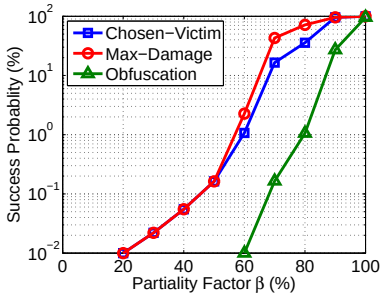


Figure 4.7. The success probabilities versus partiality factor β for three types attackers in wireless network.

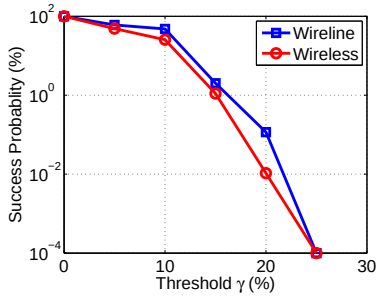


Figure 4.8. The success probabilities versus the obfuscation threshold γ in both wireline and wireless networks.

distinct from the ground truth metric $\tilde{x}_\alpha$. Therefore, if condition 2) holds, we get different solutions from different paths, thus the inconsistency exists and l_α can be located. □

Remark 8 *Theorem 5 shows two sufficient conditions to locate a malicious link. The violation of any one or both of them will lead to the failure of locating. Fig. 4.5 shows two typical examples in which the malicious link is not locatable. In Fig. 4.5, nodes* $M_1 - M_5$ *are monitors, where* M_1 *is the attacker aiming to scapegoat links between* M_2 *and* D*,* M_2 *and* C *(only exists in Fig. 4.5(a)). In Fig. 4.5(a), the malicious link does not satisfy condition 1 in Theorem 5 because node* A *is not a monitor and we cannot add the path pair between path* $M_1 \to M_3$ *and path* $M_1 \to M_5$ *to the path set. One question may raise if node* A *is a monitor: Can we directly locate the malicious link by using* M_1 *and* A *without following the previous attack-locating mechanism? The answer is no because the path between* M_1 *and* A *does not contain any intended victim by the attacker. If we directly send probe packets on this path, the obtained result will not be affected as the attacker does not manipulate the packets traversing. In Fig. 4.5(b), there are only two paths, thus the path set formed by this network does not satisfy condition 2), which needs at least two different path pairs.*

4.6 Experimental Evaluation

In this section, we use simulation experiments to evaluate the feasibility of measurement integrity attacks and effectiveness of attack detection and locating methods based on real-world and simulated network topologies.

4.6.1 Experimental Setups

4.6.1.1 Network Topology

We consider two types of network scenarios.

- Wireline networks. We use the Rocketfuel datasets [153] as the topologies for wireline networks. Rocketfuel models the topologies of autonomous systems of Internet Service Providers (ISPs), such as AT&T and Ebone. In the following, we only show the results from the AS1221 system, consisting of 108 nodes and 152 links, due to similar experimental results.

- Wireless networks. We use the random geometric graph to generate wireless network topologies because it has been widely used to model multi-hop wireless networks (e.g, [154, 155]). We adopt the extended network generation mode, and randomly distribute 100 nodes on region $[0, \sqrt{100/\lambda}]^2$ according to node density $\lambda = 5$ such that each node has 5 neighbors on average.

4.6.1.2 Parameter Setting

In experiments, we use delay as the performance metric. There is a routine traffic on each link with random delay performance from 1ms to 20ms. We consider a link normal if its delay is less than 100ms, and abnormal if the delay is greater than 800ms.

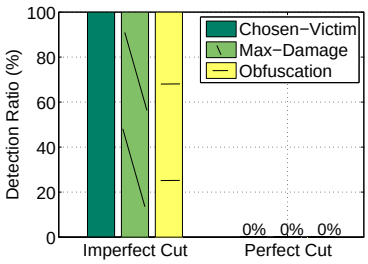


Figure 4.9. The detection ratios of chosen-victim, maximum-damage and obfuscation attackers with perfect and imperfect cuts.

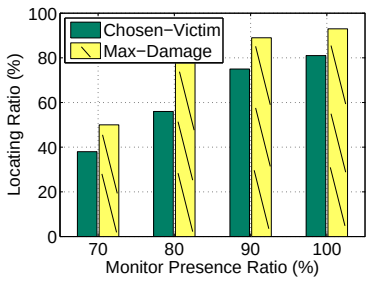


Figure 4.10. The localization ratios versus the monitor presence ratio for chosen-victim attackers, maximum-damage attackers.

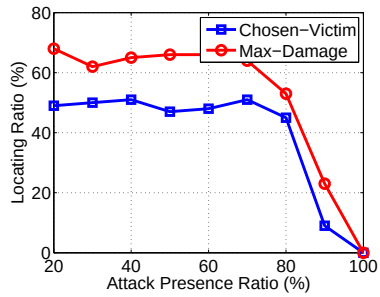


Figure 4.11. The locating ratios versus the attack presence ratio for chosen-victim attackers, maximum-damage attackers.

The objective of malicious nodes is to delay packets going through them as much as possible, and at the same time make network tomography yield a misleading result. For practical considerations, we also impose a limit on attackers that they should not delay the delivery of a packet on a measurement path for more than 2000ms.

We choose monitors and measurement paths according to a random selection algorithm based on the minimum monitor placement rule in [47]. We also randomly select nodes to be malicious in a network.

For the obfuscation attack, we set the default threshold $\gamma = 10\%$, i.e., the obfuscation attack is successful if more than 10% of total number of links are in the uncertain state.

4.6.2 Feasibility

In our experiments, in order to show the impact of measurement integrity attacks, we define the attack success probability as the ratio between the number of successful attacks and the total number of runs for a network topology. Then, in the following, we measure the feasibility of chosen-victim scapegoating, maximum-damage scapegoating and obfuscation by changing the partiality factor β and the obfuscation threshold γ in both wireline and wireless networks.

4.6.2.1 Varying the Partiality Factor

A straightforward way to show the feasibility is to measure the success probability as a function of the partiality factor β in the network. This is because, as shown in Theorems 1 and 2, an essential condition for scapegoating is the number of paths in $|\mathcal{P}_m|$, which is closely related to the partiality factor β. It is obvious that the $\beta = 100\%$ if attackers present on all measurement paths, which can perfect cut any victim link.

Fig. 4.6 depicts the success probabilities of three attack strategies in wireline network. It is easy to find that the success probability increases as the partiality factor β increases. For example, when β goes from 70% to 80%, the success probability increases accordingly from 28% to 67% for the chosen-victim scapegoating as shown in Fig. 4.6. When $\beta = 1$, the success probability becomes 100% for all attack strategies since attackers can perfectly cut all victim links. We also notice that obfuscation is less likely to succeed, compared with chosen-victim and maximum-damage scapegoating as it has to manipulate a number of victim links. In addition, maximum-damage attacks are always more likely than chosen-victim attacks. This is because the attacker does not specifically target a given victim; as long as it can find such a victim among all the nodes, it will be successful.

Fig. 4.7 shows the success probabilities under the wireless network topology. We can see from Fig. 4.7 that when β is less than a threshold, scapegoating is unlikely to succeed, because attackers know too little information about the network to launch the attacks. For example, the obfuscation always has 0 success probability when $\beta \leq 60\%$.

4.6.2.2 Varying the Threshold of Obfuscation

For obfuscation, the success probability is also related to the threshold γ since attackers must make at least $\gamma|\mathcal{L}|$ victim links exhibit the uncertain status to be considered successful. Therefore, we also evaluate how γ can affect the success probability.

Fig. 4.8 depicts the success probabilities of obfuscation in both wireline and wireless topologies when $\beta = 90$. We can see from both types of networks, the success probability decreases as γ increases. This is because a larger γ means that attackers need to affect more links at the same time, which is less likely to succeed. In addition, we notice that obfuscation is less successful in the wireless topology. For example, when $\gamma = 10\%$, the success probabilities are 48% and 60% in wireless and wireline networks, respectively. This is because the wireless topology is sparser and our monitor placement algorithm results in shorter measurement paths, which are more difficult to be affected by attackers from our observations in experiments.

4.6.3 Detection

We then use the detection method proposed in Section 4.4.2 to detect measurement integrity attacks. According to Theorem 3, there is no way for the method to detect an attack if attackers perfectly cut a victim. We separate experiments into the perfect cut and imperfect cut cases. We set the threshold $\alpha = 200$ms in all experiments.

Fig. 4.9 shows the detection ratios over all three attack strategies in the perfect cut and imperfect cut cases, respectively. From Fig. 4.9, the detection ratio in the presence of all three attacks is 100% when attackers can perfectly cut victim links, and 0% otherwise, which verifies the theoretical predictions in Theorem 3. We also find that the detection method yields no false alarm in all attack detection experiments.

4.6.4 Locating Attacks

We locate malicious links by leveraging the method in Section 4.5.2. According to the condition 1 in Theorem 5, a malicious link is not locatable if both end nodes of the link are not monitors. Therefore, locatability of a link in a network is directly related to the monitor presence ratio. We conduct our experiments by showing the relationship between

the monitor presence ratio and the locating ratio, which is defined as the probability that a malicious link can be located.

Fig. 4.10 shows the locating ratio under different attack strategies, where 5% nodes are malicious. In Fig. 4.10, we see that the locating ratio increases when we place more monitors, since more paths can meet the condition 1 of Theorem 5. However, even when all nodes are monitors, we still cannot guarantee to locate every malicious link since locating attack links is also related to the network topology. For example, the case shown in Fig 4.5(b) is not locatable even if all nodes are monitors. In addition, the locating ratio of the maximum-damage scapegoating is larger than the chosen-victim attacks because the size of the victim link set $\mathcal{L}_s$ of the maximum-damage scapegoating is usually larger than the size of $\mathcal{L}_s$ for the chosen-victim attacks, thus providing more chances to satisfy the condition 2. Note that we do not evaluate the locating ratio for the obfuscation strategy because the success probability of the obfuscation attack is very low with 5% attackers. In other words, it is very unlikely to launch a feasible obfuscation attack with such a limited number of attackers, thus we do not need to locate it.

Fig. 4.11 shows the relationship between the locating ratio and the attack presence ratio (defined as the ratio of the number of measurement paths including at least one victim and at least one attacker over the number of total measurement paths including any victim) where 75% nodes are monitors. We can see a slight fluctuation of locating ratios when the attack presence ratio increases from 20% to 80%. But when the attack presence ratio reaches 90%, the locating ratio decreases sharply, since almost all links are controlled by attackers, and the numbers of normal links and victim links decrease dramatically, which are necessary to locate attack links.

4.7 Discussions and Future Work

In this section, we discuss our results associated with the feasibility and defense of measurement integrity attacks, as well as the potential impacts on other related work. Then we provide our future work.

To launch measurement integrity attacks, the attackers must have the information about the measurement paths, which the network operator can definitely attempt to hide. For example, the operator can avoid publishing such information or avoid using some protocols containing path information, such as AODV routing for wireless networks, to prevent attackers from inferring such information from probe packets in the network. This can constitute the first line of defense. Nevertheless, from a security point of view, it should not be assumed that attackers can never get such information. Moreover, measurement integrity attacks do pose a threat to affect the trustiness of the measurement results. Follow-up actions, such as fault recovery, do rely on such results. Our results indicate that instead of simply assuming *seeing-is-believing*, we should always be cautious of malicious manipulation in network measurement.

Our future work includes both monitor placement and hiding routing information to combat measurement integrity attacks against network tomography.

- For the monitor placement, existing monitor placement methods mainly focus on minimizing the number of monitors or enhancing the robustness. The theoretical results in Theorem 3 and experimental results in Figs. 4.6 and 4.7 reveal that a measurement integrity attack becomes more likely as the number of attackers increases. Hence, we aim to design a new monitor placement algorithm to first ensure identifiability under network tomography, then minimize each node's presence ratio on measurement paths such that when it is compromised, its impact to measurement manipulation is minimized.

- For hiding routing information, routing information, existing in the routing matrix, is necessary for launching the attacks. Therefore, defenders can leverage anonymous routing protocols to hide the routing information, to prevent attackers from inferring the link metrics. We aim to understand how to hide routing information to minimize the impact of measurement integrity attacks against network tomography.

4.8 Related Work

4.8.1 Network Tomography

Network tomography is a generic way to compute network component (usually network link) metrics from measurements on end-to-end paths in a network. In essence, network tomography can be considered as an algorithmic process to transfer end-to-end measurements into link metric estimates. Existing work mainly focused on algorithm design and applications (e.g., [33, 34, 35, 32, 36, 37, 38]); and some recent papers also considered the problem of placement of monitors and identifiability of link metrics (e.g., [44, 45, 46, 47]). Network tomography has been proposed for measurement, fault diagnosis and localization in both wireline networks (e.g., [32, 36, 37, 38]) and wireless networks (e.g., [33, 34, 35]).

In general, these papers implicitly assume that individual link metrics can be inversely derived from the path measurements that indeed reflect the real link performance aggregate. In fact, it is not guaranteed that there exists no anomaly or malicious behavior in today's large-scale networks. However, potential security vulnerabilities in network tomography have not yet been investigated in the literature.

4.8.2 Packet Dropping and Delaying Attacks

There are various malicious attacks against a network, such as passive eavesdropping, active interfering, leakage of secret information, data tampering, impersonation, message dis-

tortion and denial-of-service attacks (e.g., [156, 157, 158, 159, 160]). Measurement integrity attacks drop or delay packets to damage a network, which is related to packet dropping attacks, such as black hole attacks that attract and drop all packets routed to malicious nodes and grey hole attacks (also called selective forwarding attacks) that only drop certain selected packets [161].

However, such traditional attacks can be discovered by finding out the links which always suffer long delay or high loss under network tomography [160]. In contrast, our measurement integrity attack strategy can not only hide the real identities of attackers in network tomography, but also make some legitimate nodes or links the scapegoats. Therefore, the proposed attack strategy is a new one that is able to deteriorate the network performance, while misleading network tomography based diagnostics.

4.8.3 Attack Detection and Defense

Existing network defense approaches are usually deployed in individual host systems (e.g., end nodes or edge routers). These mechanisms can directly detect anomalies on some particular victims. For example, the process of tracing back the forged IP packets to their true sources rather than the spoofed IP addresses that was used in the attack is called traceback. There are various IP traceback mechanisms that have been proposed to date (e.g., [162, 163]). Packet marking and filtering mechanism aims to mark legitimate packets at each router along their path to the destination so that victims' edge routers can filter the attack traffic (e.g., [164, 165]).

There are a few studies to detect network neutrality violations [40, 166, 167, 168, 169]. For example, the strategy in [166] relies on detecting whether traffic on specific ports is blocked. Authors in [167, 168] proposed a system to detect neutrality violations by inferring whether an ISP discriminates traffic based on performance data obtained passively.

There are also studies related to monitoring and analyzing network traffic to protect a system from network-based threats. For instance, route-based packet filtering system uses routing information to distinguish if a traffic flow at a router is valid and ensure that resources are made available only for legitimate use (e.g., [170, 171]). The work in [172] designed a strategy to detect misbehaving routers that absorb, discard or misroute packets. Such mechanism usually requires explicit communication among routers. The work in [173] presented a heuristic data structure to monitor traffic characteristics of network devices like routers to detect and eliminate attacks. In addition, traffic monitoring can also be leveraged for detecting anomalous packet forwarding [174].

Network tomography is performed by the network operator to obtain the global picture of the healthiness of a network. Therefore, the detection proposed in this chapter is a network-wide approach that should follow immediately the network tomography process to detect whether such a process is manipulated or exploited by malicious behavior. Our network-wide attack detection approach to protect network tomography can be regarded as complementary to defense strategies deployed in individual host systems (e.g., end nodes and routers).

4.9 Summary

In this chapter, we have provided theoretical and experimental results to analyze the feasibility of measurement integrity attacks against network tomography. We consider three basic strategies: chosen-victim, maximum-damage and obfuscation attacks, and show that malicious nodes can substantially damage a network and at the same time manipulate end-to-end measurements to make legitimate nodes scapegoats. We also present the conditions to detect and locate these attacks. The results in this chapter indicate that the current *seeing-is-believing* assumption in network tomography renders a security vulnerability. Instead of

simply trusting measurements, we should be always aware of measurement integrity attacks and carefully revisit existing designs for security in various applications.

Chapter 5: Anonymity Analysis of Randomized Routing Protocols

5.1 Abstract

Preventing the source-destination network flow information from being disclosed is pivotal for anonymous wireless network applications. However, the advance of network inference, which is able to obtain the flow information without directly measuring it, poses severe challenges towards this goal. Randomized routing is capable of hiding the flow information by injecting substantial errors to the network inference process.

In this chapter, we systematically study the behavior of randomized routing protocols, and categorize them into three templates, k-random-relay, k-random-neighbor and k-random-path based on their routing behaviors. We propose technical models to characterize these templates in terms of their induced inference errors and their delay costs. We also use simulations to validate the theoretical results. Our work provides the first systematic study on understanding both the benefit and the cost of using randomized routing to hide the flow information in wireless networks.

5.2 Preliminary and Problem Statement

In this section, we first present the network model and the preliminary of network inference. Then, we state our research problems. The notations of this chapter are summarized as follows. (i) $f(n) = O(g(n))$ denotes there exists a constant c such that $f(n) \leq cg(n)$; $f(n) = \Omega(g(n))$ means $g(n) = O(f(n))$; $f(n) = \Theta(g(n))$ means $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. (ii) Given a $m \times n$ matrix $\mathbf{A}$ with entry x_{ij}, then $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2}$,

and $\mathbf{tr}\{\mathbf{A}\}$ is the trace of $\mathbf{A}$. (iii) For a vector $\mathbf{v} = [v_1, \cdots, v_n]$, $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$; $|\cdot|$ denotes the cardinality operator.

5.2.1 Network Model

We model the wireless network by using a random geometric graph (RGG) [175] denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the node set and $\mathcal{L}$ is the undirected link set. Let $N = |\mathcal{V}|$ and $L = |\mathcal{L}|$. In this network, N nodes are randomly placed in a region $\Omega = [0, \sqrt{N/\lambda}]^2$, where λ denotes the node density. We assume that λ is sufficiently large such that the network is connected asymptotically almost surely [175]. Denote by r the transmission range of each node.

In the network, packet exchange occurs in node pairs, yielding multiple end-to-end data flows. We denote by $\mathcal{F}$ the end-to-end flow set consisting of the potential flow for each node pair. Obviously, $|\mathcal{F}| = N(N-1)/2$, which is the number of node pairs. Each flow $f_i \in \mathcal{F}$ is associated with a metric x_i denoting the data rate on flow f_i. Denoted by a column vector $\mathbf{x} = [x_i]_{i \in [1, |\mathcal{F}|]}$ the flow rate vector for the network. We consider $x_i = 0$ if flow f_i does not exist (i.e., there is no communication).

The flow rate vector $\mathbf{x}$ contains two types of information:

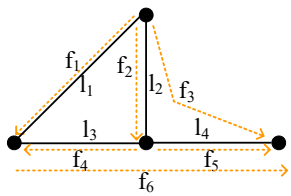
1. who is communicating with whom in the network;

2. how much data rate a pair of communicating parties has.

The disclosure of such information is undesirable or even prohibited in many network security scenarios [176, 177, 178].

5.2.2 Attack Model and Network Inference

What kind of methods an adversary can use to obtain $\mathbf{x}$? Note that $\mathbf{x}$ is not generally available to the adversary because flow information is indicated at network or higher layers



(a) Network topology.

	f_1	f_2	f_3	f_4	f_5	f_6
l_1	1	0	0	0	0	0
l_2	0	1	1	0	0	0
l_3	0	0	0	1	0	1
l_4	0	0	1	0	1	1

(b) A routing matrix **A**.

Figure 5.1. Example network topology (a) and the routing matrix (b).

[11], whose data is usually encrypted at the physical or link layer. Therefore, the adversary has to infer such information based on physical/link-layer activities, which is called network inference [58]. In this chapter, we consider an omniscient adversary who (i) knows the network topology and (ii) knows the routing protocol used in the network, and (iii) is capable of monitoring each link $l_i \in \mathcal{L}$ in the network. This strong attack model enables us to clearly compare the benefits of different randomized routing protocols under the worst-case standard.

By letting column vector $\mathbf{y} = [y_1, y_2, \cdots, y_L]^T$ (where y_i denotes the rate of link l_i and $\cdot^T$ is the matrix transpose operator), the goal of the adversary is written as obtaining the estimated value of $\mathbf{x}$, denoted by $\hat{\mathbf{x}}$, from the measured link rate vector $\mathbf{y}$. Network inference is an approach to achieve the adversary's goal. In particular, the relationship between flow rate vector $\mathbf{x}$ and the link rate vector $\mathbf{y}$ can be captured by the following linear system.

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{5.1}$$

where $\mathbf{A}$ is the routing matrix with size $L \times |\mathcal{F}|$, whose entry

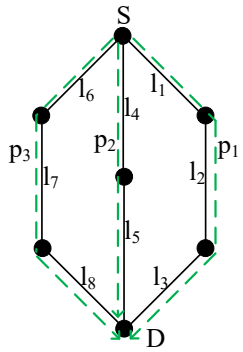
$$a_{ij} = \begin{cases} 1, & \text{if flow } f_j \text{ goes through link } l_i; \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

The routing matrix $\mathbf{A}$ demonstrates how a flow is concatenated by links, which is the function of routing protocols. Therefore, given a network, the routing matrix can be usually determined by the routing protocol. Fig. 5.1 shows an illustrative example of a network topology and its corresponding routing matrix $\mathbf{A}$. In this network, we have the link set $\mathcal{L} = \{l_1, \cdots, l_4\}$ and flow set $\mathcal{F} = \{f_1, \cdots, f_6\}$, where each flow $f_i \in \mathcal{F}$ is determined by the shortest path routing protocol. For example, flow f_1 only goes through link l_1 because this path, with length 1 (the length is measured by the number of hops), is the shortest one; and the entry corresponding to f_1 and l_1 is therefore 1 in the routing matrix $\mathbf{A}$ as shown in Fig. 5.1(b).

In order to estimate $\mathbf{x}$, the adversary must know the routing matrix $\mathbf{A}$ in (5.1), which is usually an under-determined system. If the adversary indeed knows such information, it can use standard network inference methods (e.g., $\mathcal{L}_1$-norm minimization [63]) to obtain $\hat{\mathbf{x}}$, the estimated version of $\mathbf{x}$. The inference error can be denoted as

$$I_R = \|\hat{\mathbf{x}} - \mathbf{x}\|_2. \tag{5.3}$$

For example, in Fig. 5.1, only flow f_3 has data steam with rate 10bps, i.e., $\mathbf{x} = [0, 0, 10, 0, 0, 0]$. Then the attacker can obtain the link rate vector $\mathbf{y} = [0, 10, 0, 10]$, indicating that links l_2 and l_4 have data transmission with rate 10bps. If the routing matrix $\mathbf{A}$ is also available, the attacker can obtain the estimated flow rate vector $\hat{\mathbf{x}}$. Note that we do not confine the adversary to any particular inference method. If we know what method the adversary uses, we may provide a method-specific defense. In this chapter, we assume the worst case that the network inference approach used by the adversary is unknown to us.



(a) Network topology.

R_1: Shortest Path Routing

$f_{SD}^{R_1}$: $[0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^T$ (path p_2)

R_2: Randomized Routing

$f_{SD}^{R_2}$: $[1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$ (path p_1)

or

$[0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^T$ (path p_2)

or

$[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1]^T$ (path p_3)

(b) One column in **A** under protocols.

Figure 5.2. The relationship between link and flow rates.

5.2.3 Randomized Routing Protocols: Benefit and Cost

In general, the routing matrix $\mathbf{A}$ is not immediately available to the adversary. However, as the adversary is assumed to know what routing protocol is used, it can build the routing matrix by itself given the network topology. If a deterministic routing protocol (e.g., shortest path) is used, it can be very likely that the adversary builds the routing matrix $\hat{\mathbf{A}}$ in close value to the true routing matrix $\mathbf{A}$, then obtains an accurate estimate of $\mathbf{x}$.

Randomized routing protocols make the behavior of routing packets unpredictable and accordingly incurs more inference error than deterministic ones to the adversary. To illustrate how a randomized routing protocol may work, we consider a simple network topology in Fig. 5.2(a). The network consists of 8 links (i.e., links $l_1 - l_8$) and one traffic flow f_{SD} between nodes S and D, which contains three potential routing paths (i.e., paths p_1, p_2 and p_3). Because there is only one flow, the routing matrix $\mathbf{A}$ only has one column representing the flow f_{SD}. We define a path set $\mathcal{P}_{SD} = \{p_1, p_2, p_3\}$ for flow f_{SD}, and two routing protocols R_1 and R_2 for the shortest path routing protocol and a randomized routing protocol, respectively.

Under protocol R_1, flow f_{SD} deterministically uses path p_2 to forward packets because it is the shortest path, leading to no randomness to choose a path. Aware of the shortest-path

routing, the adversary can immediately reconstruct a routing matrix $\hat{\mathbf{A}}$ which is exactly the same as the ground-truth matrix $\mathbf{A}$. Then, the adversary can use it to accurately recover $\mathbf{x}$.

Under protocol R_2, a path is selected uniformly at random from the path set $\mathcal{P}_{SD}$. Then, the adversary's estimation $\hat{\mathbf{A}}$ equals to $\mathbf{A}$ with probability 1/3, as shown in Fig. 5.2(b). This extra error due to the routing matrix mismatch will be introduced to the network inference, offering higher protection of the network flow information. Note that for this illustrative example in Fig. 5.2, the adversary may further determine which path is chosen by observing which links are active. However, simply observing which links are active cannot be applied to a multi-hop wireless network, where a large number of node pairs exchange packets simultaneously in the network.

On one hand, protocol R_2 causes more inference error to the adversary. On the other hand, however, R_2 incurs more delay during the data transmission because packets are not always be forwarded along the shortest path. Therefore, the cost of R_2 is the increase of delay. To provide formal models, we define the benefit (the inference error) and the cost (the delay) of randomized routing as follows.

Definition 5 *Within a network $\mathcal{G}$ with N nodes under a routing protocol R, the inference error for the adversary is measured by $I_R = \mathbb{E}\|\hat{\mathbf{x}} - \mathbf{x}\|_2$, where $\hat{\mathbf{x}}$ and $\mathbf{x}$ are the adversary's estimated flow rate vector and true flow rate vector, respectively. The delay between a node pair is the average number of hops for data delivery between the pair. Given a routing protocol R, the average delay h_R, is obtained by averaging the delays over all node pairs.*

The adversary cannot directly observe the end-to-end flow information, and the flow information can be obtained only through inference. This is why we use the inference error as the benefit of different randomized routing strategies. The benefit does not confine to protecting the flow information. For example, it can also be applied to balance the traffic load or mitigate the wireless link failures. However, the goal of this chapter is how to provide

defense against network inference, and we find that randomized routing can achieve this goal. Therefore, we explore the randomized routing strategy from the security perspective, and characterize the benefit as the inference error, which is directly related to the defense performance.

In addition, we use the number of hops as the cost since it is directly related to the inference error. The number of hops is easily available and widely used information serving as the cost for routing discovery in many wireless network protocols, such as AODV [179] in MANET. Therefore, in our chapter, we also use the number of hops as the cost metric to measure the performance of each template, and focus on analyzing protocols using the number of hops as the performance cost.

5.2.4 Templates and Problem Statement

To formally characterize randomized routing protocols, we propose three templates to analyze the benefit and cost of randomized routing in wireless networks.

1. T_1: k-random-relay: for each packet, the routing path is formed by selecting k nodes (called relays) uniformly at random from $\mathcal{V}$ excluding the source and destination nodes. Then, the packet is transmitted through these relays. The shortest path routing is used between two consecutive relays. Onion routing used in the Tor [64] or Crowds [180] networks are popular real-world applications of this template although they are currently used in wireline networks.

2. T_2: k-random-neighbor: for each packet, at each hop, one neighbor is selected with equal probability from k neighbors with shortest distances to the destination. The random grid routing [181] and greedy forwarding routing [182] are two simplistic versions of this template.

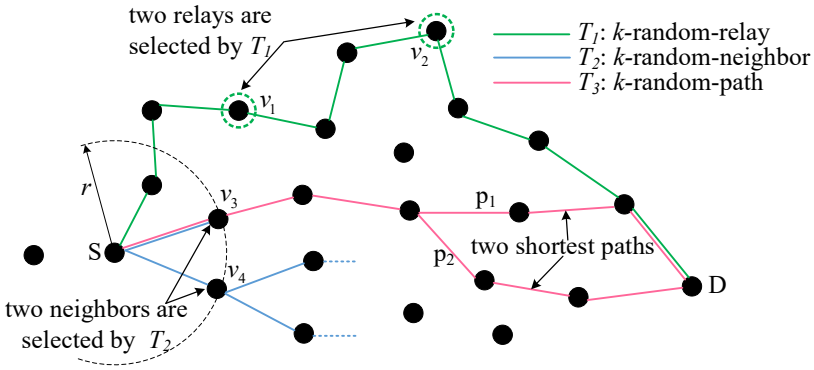


Figure 5.3. Examples of three routing templates for $k = 2$.

3. T_3: k-random-path: each packet is transmitted through one path selected uniformly at random from k shortest paths. This template is available when the overall path information is available. The template can be considered as a randomized version of the k-shortest-path routing protocol [62].

The examples for the three templates is shown in Fig. 5.3, which has one flow f_{SD} between nodes S and D with $k = 2$. Under k-random-relay, two relays $v_1, v_2 \in \mathcal{V}$ are randomly selected, then a packet can be transmitted through these two relays (i.e., the green line), where the paths between the source and the first relay, between the two relays, and between the second relay to the destination are all the shortest paths. Under k-shortest-neighbor, the source S randomly picks one neighbor v_4 from two neighbors v_3 and v_4 which have the shortest distance towards the destination D. Then, v_4 adopts the same rule for the second hop, and so on. The path for the k-random-path template is randomly selected from 2 shortest paths p_1 and p_2.

In these template, k is an independent variable of N and is determined by practical applications or the real-world scenarios. For example, in the Tor network, k is fixed to 3 no matter how many nodes in the network. In this chapter, to provide randomness, we assume $k > 1$ is a constant independent of N. The proposed templates cover a wide range of randomized routing behaviors. However, which template is more suitable for a network and

how to choose the routing template are based on several factors such as node density and communication rage of each node. The template selection for a particular network scenario is orthogonal to the research in this chapter that focuses on performance and cost analysis. Our objective in this chapter is to model and analyze the inference error and delay of each template to offer a fundamental basis or guideline for designing practical randomized routing protocols.

5.3 Theoretical Modeling and Strategy

In this section, we provide a theoretical metric to measure the inference error induced by randomized routing. Then, we discuss our strategy to model routing templates.

5.3.1 Genie Bound

It is expected that randomized routing increases the inference error I_R, which is related to a particular inference method used by the adversary to derive $\hat{\mathbf{x}}$ in (5.1). To remove such a dependency, we use the genie bound [183] to measure I_R as it is a generic metric regardless of the inference method. Specifically, for the under-determined linear system (5.1), the genie bound, serving as a lower error bound of all possible inference methods, can be obtained in three steps:

1. form a new deterministic linear system with the assistance of a genie, which is expressed as

$$\mathbf{y} = \mathbf{A}_g \mathbf{x}_g, \tag{5.4}$$

where $\mathbf{x}_g$ denotes the flow rate vector for node pairs that indeed have real traffic flows, obtained by removing zero entries (e.g., non-existing flows) from $\mathbf{x}$. $\mathbf{A}_g$ is the routing matrix presenting the relationship between real flows and links;

2. derive the least square estimation of $\mathbf{x}_g$ as

$$\hat{\mathbf{x}}_g = (\mathbf{A}_g^T \mathbf{A}_g)^{-1} \mathbf{A}_g^T \mathbf{y}; \tag{5.5}$$

3. obtain the genie bound by deriving the minimum mean square error between $\hat{\mathbf{x}}_g$ and $\mathbf{x}_g$, i.e.,

$$G(\mathbf{x}_g) = \mathbb{E}\left(\|\hat{\mathbf{x}}_g - \mathbf{x}_g\|_2^2\right). \tag{5.6}$$

Note that in step 1, with the help of the genie, the under-determined linear system (5.1) is converted into a determined system (5.4). This removes the effect of the choice of the inference method used by the attacker. The genie bound is a general and method-independent bound, and is widely used in solving under-determined system to provide a lower error bound for any inference method. Formally, we define the genie bound as a generic metric to quantify the inference error I_R as follows.

Definition 6 *Given the link rate vector* $\mathbf{y}$*, and routing matrix* $\mathbf{A}$ *determined by the routing template* T*, the inference error* I_T *can be rewritten as*

$$I_T = \mathbb{E}\left(\|\hat{\mathbf{x}}_g - \mathbf{x}_g\|_2^2|\right), \tag{5.7}$$

where $\hat{\mathbf{x}}_g$ *and* $\mathbf{x}_g$ *are derived in the three-step genie bound procedure.*

5.3.2 Routing Template Modeling

The inference error is due to the randomness in selecting an end-to-end path for a flow. In a real network, it is very likely that there are multiple paths for a flow, and how to select a path is based on the routing protocol. For example, in Fig. 5.2, the shortest path protocol

simply selects path p_2 whereas random routing selects one randomly from paths p_1, p_2 and p_3.

Fundamentally, these three templates provide different behaviors to select a path from the total path set of a flow. For example, in Fig. 5.2, under the template T_3, flow f_{SD} randomly selects a path from $\mathcal{P}_{SD}$ if it is available. However, if only the neighbor information is available and the template T_2 is used, T_2 will randomly choose a link from the link set $\{l_1, l_4, l_6\}$ for the next hop to forward packets, leading to a different manner to choose a path from f_{SD}. All these random factors eventually lead to a random path selection for a flow. Therefore, our strategy is to use a general path-selection model to capture all behaviors of different randomized routing templates.

Before modeling the difference among different routing templates, we first define $\mathcal{P}_i$ as the path set consisting of all potential paths for the flow $f_i \in \mathcal{F}_g$. Denoted by $\mathcal{P}_\Pi = \{\mathcal{P}_i\}_{i\in[1,F]}$ the path set for all source-destination pairs in the network. To build the relationship between a path and its corresponding vector in the routing matrix, we define a function mapping J which maps paths to their corresponding columns in the routing matrix. For example, in Fig. 5.2, $J(p_1) = [1, 1, 1, 0, 0, 0, 0, 0]^T$.

Based on the previous definitions, we introduce three matrices as follows.

1. The overall routing matrix $\mathbf{M} = [\mathbf{m}_1, \cdots, \mathbf{m}_F]$ consisting of all potential paths. It means that each entry $\mathbf{m}_i = J(\mathcal{P}_i)$ for $1 \leq i \leq F$, and $\mathbf{M} = J(\mathcal{P}_\Pi)$.

2. The template matrix $\mathbf{T} = \text{diag}(\mathbf{t}_1, \cdots, \mathbf{t}_F) \in \mathbb{R}^{|\mathcal{P}_\Pi| \times \sum g_i}$ is a rectangular diagonal matrix. Each entry $\mathbf{t}_i$ is a $|\mathcal{P}_i| \times g_i$ matrix, where $1 \leq g_i \leq |\mathcal{P}_i|$ denotes the number of paths selected by the routing template for the flow f_i. Each column in $\mathbf{t}_i$ has only one entry with value 1 denoting which path is selected and the remaining entries are all zeros.

3. The selection matrix $\mathbf{C} = \text{diag}(\mathbf{c}_1, \cdots, \mathbf{c}_F) \in \mathbb{R}^{\sum g_i \times F}$, where each $\mathbf{c}_i$ is a $g_i \times 1$ all zero column vector except one 1 indicating which path is selected for the actual packet transmission.

In the following, by leveraging the above three matrices, we propose a technical model to separate different factors in performance modeling for randomized routing.

Model 1 *Given a network topology $\mathcal{G}$, and the routing template T, the routing matrix $\mathbf{A}_g$ can be modeled as $\mathbf{A}_g = \mathbf{MTC}$, where matrices $\mathbf{M}$, $\mathbf{T}$, and $\mathbf{C}$ are defined as above.*

We use the network topology in Fig. 5.2(a) as an example. Assume the routing protocol is T_3: k-random-path with $k = 2$, and there is only one flow f_{SD} in the network. Then, the routing matrix $\mathbf{A}_g \in \mathbb{R}^{8\times 1}$ can be expressed as

$$\mathbf{A}_g = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T}_{\mathbf{M}} \times \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{T}} \times \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\mathbf{C}}. \tag{5.8}$$

Three columns in $\mathbf{M}$ denote all possible paths between nodes S and D. Thus, $\mathbf{M}$ only relies on the network topology and is independent of routing behavior. There are two columns in $\mathbf{T}$, meaning two possible path selections (i.e., p_1 and p_2) in k-random-path with $k = 2$. The last matrix $\mathbf{C}$ denotes the path p_1 is finally used. Hence, the routing behavior decides $\mathbf{T}$ and $\mathbf{C}$.

Remark 9 *We decompose the routing matrix $\mathbf{A}_g$ into $\mathbf{MTC}$ such that each matrix represents one factor to affect the values in $\mathbf{A}_g$. The overall routing matrix $\mathbf{M}$ only depends on the network topology. The template matrix $\mathbf{T}$ relies on a randomized routing template, and which template to be used is based on many factors such as the link capacity constraints or*

wireless interference. The selection matrix $\mathbf{C}$ *indicates which path is used to send packets. Each of its column vector has only one entry with value 1, meaning that only one path is selected for a source-destination pair. Note that this model does not consider the scenario that one packet is sent over multiple paths, which is not typical in wireless networks due to the broadcast nature of the wireless channel.*

5.4 Theoretical Results

In this section, we first present the theoretical results of the inference error and delay of each randomized routing template. Then, we provide the proofs of the results.

5.4.1 Main Results

5.4.1.1 Inference Error

We consider a network $\mathcal{G}$ with N nodes and F flows. For the network, without loss of generality, we assume the flow rate $x_i \in \mathbf{x}_g$ of each flow is a random variable with mean μ and variance σ^2. Then, we have the following results on the inference error for each template.

Theorem 6 *The inference errors (in terms of the genie bound in Defition 6) of templates* $T_1 - T_3$ *satisfy*

1. *for the template* T_1*: k-random-relay,*

$$\Theta\left(\frac{F^2\mu^2}{\sqrt{N}/(k+1)+F}\right) \leq I_{T_1} \leq \Theta\left(2F^2(\mu^2+\sigma^2)\right); \tag{5.9}$$

2. *for the template* T_2*: k-random-neighbor,*

$$\Theta\left(\frac{F^2\mu^2}{N/\varphi(N,k)+F}\right) \leq I_{T_2} \leq \Theta\left(2F^2(\mu^2+\sigma^2)\right); \tag{5.10}$$

3. for the template T_3: k-random-path,

$$\Theta\left(\frac{F^2\mu^2(k-1)^2}{k^2(\sqrt{N}+F)}\right) \leq I_{T_3} \leq \Theta\left(\frac{2F^2(\mu^2+\sigma^2)}{k/(k-1)}\right), \tag{5.11}$$

where $\varphi(N,k) = \max\{N^{\log(k-1)}\log(k-1)\ln(N), \sqrt{N}\}$

Remark 10 *Theorem 6 shows the impact regions of three routing templates. We note that only the upper bound of k-random-path is positively related to k, indicating that when k is small, k-random-relay and k-random-neighbor templates are possible to induce a higher inference error than k-shortest-path. In addition, we find that the bound is an approximately linear (quadratic) function of the number of flows F in the network. This interestingly reveals that increasing the number of flows in the network (i.e., making the communication scenario more complex) in fact incurs more errors than trying to actively creating randomness in routing behavior. A reasonably small k should be chosen in practice due to the sub-linear relationship between the error and k.*

5.4.1.2 Delay

We have the following theorem to analyze the costs of delay for three randomized routing templates.

Theorem 7 *The delays of templates $T_1 - T_3$ satisfy*

1. for the template T_1: k-random-relay,

$$h_{T_1} = \Theta((k+1)\sqrt{N}); \tag{5.12}$$

2. for the template T_2: k-random-neighbor,

$$\Theta(\varphi(N,k)) \leq h_{T_2} \leq \Theta(N\ln(N)); \tag{5.13}$$

3. for the template T_3: k-random-path,

$$h_{T_3} = \Theta(\sqrt{N}), \tag{5.14}$$

where $\varphi(N,k) = \max\{N^{\log(k-1)}\log(k-1)\ln(N), \sqrt{N}\}$.

Remark 11 *Theorem 7 shows the delay impacts of the three randomized routing templates. It is noted that the delay of k-random-relay increases linearly with k. When k is small, the lower bound of k-random-neighbor is the same as k-random-path (i.e.,* $\sqrt{N}$*); thus they are generally better than k-random-relay in terms of the delay cost. When k is larger, the delay cost of k-random-neighbor becomes substantially large. This is because when k increases to the average degree of the network, all neighbors can be selected randomly, leading to a random walk behavior of packet forwarding with the* $N \log N$ *delay cost.*

Remark 12 *Theorems 6 and 7 show that there is no uniformly best template among the templates in terms of both security and cost. However we notice that k-random-neighbor with a large k is not a good choice for a practical randomized routing design in that (i) it has the highest delay cost compared with others, (ii) the inference error is still on the same order with others; and (iii) for a packet with a limited lifespan (e.g., it can be controlled by Time-to-Live parameter), it cannot guarantee to deliver packets to the destination.*

Our results also indicate that k-random-path achieves the inference error with the same order of the other two templates while maintaining the lowest delay cost. But it requires knowing the global path information of the network. For k-random-relay, both the inference error and the delay is larger than k-random-path by a constant order of magnitude.

5.4.2 Proofs of Results

Now we prove the theoretical results. With Model 1, we first provide a lemma regarding the generic result of the inference error which depends on the delay overhead.

Lemma 1 *For a routing template T satisfying Model 1, in which the average number of columns of its template matrix is denoted by g_T and the delay is denoted by h_T, the inference error I_T induced by T satisfies*

$$\Theta\left(\frac{F^2\mu^2(g_T-1)^2}{g_T^2(N/h_T+F)}\right) \leq I_T \leq \Theta\left(\frac{2F^2(g_T-1)}{g_T/(\mu^2+\sigma^2)}\right). \tag{5.15}$$

Proof: See Appendix. □

Remark 13 *Lemma 1 provides a generic impact region of the inference error of any routing template T. We notice that the inference error $I_T = 0$ when $g_T = 1$, meaning that the deterministic routing protocol has no inference error.*

Next, we prove Theorem 7 to obtain the delay costs for three templates, then prove Theorem 6 which requires some results in the proof of Theorem 7.

Proof of Theorem 7: We first prove 3): h_{T_3} and 1): h_{T_1}, then we prove 2): h_{T_2}.

3) In the network $\mathcal{G}$, we assume the distance of flow f_i is d_i (the number of hops on the shortest path). For k-random-path template, the average delay for flow f_i is given by $h_{f_i} = d_i + \frac{1}{k}\sum_{j=1}^{k} c_{ij}$, where c_{ij} is a positive constant denoting the extra number of hops for jth path compared with the shortest one for flow f_i. Then the average is given by

$$h_{T_3} = \frac{1}{F}\sum_{i=1}^{F} d_i + \frac{1}{Fk}\sum_{i=1}^{F}\sum_{j=1}^{k} c_{ij}. \tag{5.16}$$

According to Lemma 6, we have $\frac{1}{F}\sum_{i=1}^{F} d_i = \Theta(\sqrt{N})$. In addition, it is easy to know $\frac{1}{Fk}\sum_{i=1}^{F}\sum_{j=1}^{k} c_{ij} = \Theta(1)$, then we can obtain $h_{T_3} = \Theta(\sqrt{N})$.

1. For k-random-relay, we randomly select k relays to forward each packet and each path between two consecutive relays adopts the shortest path. Then the average delay can be directly written as $h_{T_3} = \Theta((k+1)\sqrt{N})$.

2. The routing path of k-random-neighbor is discovered hop by hop, which is similar with the random walk (or Markov chain) model [184, 185, 186, 187]. In the following, we first briefly introduce the difference between the random walk model and k-random-neighbor, then we state how to use the random walk to model the k-random-neighbor template.

In a network $\mathcal{G}$, for an arbitrary node pair $v_i, v_j \in \mathcal{V}$, denote $h_{i,j}$ as the delay for the flow between v_i and v_j. Denote by $\mathcal{N}_i$ the node set containing all neighbors of node v_i. Then the delay cost of the flow between v_i and v_j satisfies the following recursive function

$$h_{i,j} = \begin{cases} 1 + \sum_{w \in \mathcal{N}_i} p_{iw} h_{w,j}, & \text{if } i \neq j \\ 0, & \text{otherwise,} \end{cases} \tag{5.17}$$

where p_{iw} is the transfer probability that the link between v_i and v_w is selected. In the random walk model, every neighbor in $\mathcal{N}_i$ has the same probability to be selected, i.e,

$$p_{iw} = \frac{1}{|\mathcal{N}_i|}, \text{ for } \forall v_w \in \mathcal{N}_i. \tag{5.18}$$

For k-random-neighbor with source and destination nodes v_i and v_j, we define a new neighbor set $\mathcal{K}_i \subseteq \mathcal{N}_i$ for node v_i containing k neighbors which have the shortest distances to the destination v_j. Then, the transfer probability p_{iw} can be given by

$$p_{iw} = \frac{1}{k}, \text{ for } \forall v_w \in \mathcal{K}_i. \tag{5.19}$$

Intuitively, if we can remove all neighbors that belong to $\mathcal{N}_i$ but not to $\mathcal{K}_i$, then $\mathcal{K}_i = \mathcal{N}_i$, and k-random-neighbor becomes the same as the random walk model. To do so, for the original network $\mathcal{G}$, given a source-destination pair v_i and v_j, we generate a new network $\mathcal{G}_{ij}^k = \{\mathcal{V}_{ij}^k, \mathcal{L}_{ij}^k\}$, where $\mathcal{V}_{ij}^k$ is obtained by removing nodes that the source node v_i will never go through under k-random-neighbor for the node pair v_i and v_j. Denote by $\mathcal{L}_{ij}^k$ the corresponding remaining link set for the node set $\mathcal{V}_{ij}^k$. Then, k-random-neighbor can be modeled as a random walk for the network $\mathcal{G}_{ij}^k$.

Considering the new generated network $\mathcal{G}_{ij}^k$, according to (5) and (6) from [188], the expected delay (5.17) over all node pairs is given by

$$\mathbb{E}(h_{i,j}) = \Theta\left(\mathbb{E}\left(N_{ij}^k \ln\left(\sqrt{N_{ij}^k}\right)\right)\right), \tag{5.20}$$

where $N_{ij}^k = |\mathcal{V}_{ij}^k|$. For the upper bound, we have $\mathbb{E}(N_{ij}^k) \leq N$. For the lower bound, k-random-path has the smallest delay overhead because it is obtained by averaging k shortest paths, then according to (5.12), we have that $\mathbb{E}(h_{i,j}) \geq \Theta(\sqrt{N})$. Furthermore, according to Lemma 9, we have

$$\mathbb{E}(h_{i,j}) \geq \begin{cases} \Theta\left(N^{\log_{\delta-1}^{k-1}} \ln\left(N^{\log_{\delta-1}^{k-1}}\right)/2\right), & \text{if } k > 2 \\ \Theta\left(\log(N) \ln\left(\sqrt{\log(N)}\right)\right), & \text{if } k = 2. \end{cases} \tag{5.21}$$

Combining them together, we obtain

$$\mathbb{E}(h_{i,j}) \geq \max\{N^{\log(k-1)} \log(k-1) \ln(N), \sqrt{N}\}, \tag{5.22}$$

which completes the proof. □

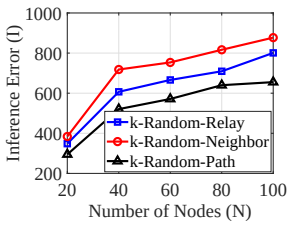


Figure 5.4. Inference error for three routing templates when $k = 3$.

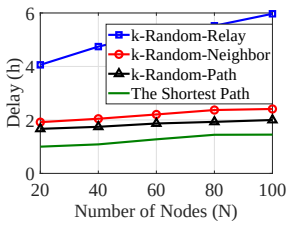


Figure 5.5. Delay overhead for three routing templates when $k = 3$.

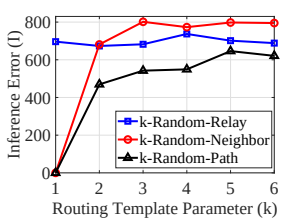


Figure 5.6. Inference error for three routing templates when $N = 50$.

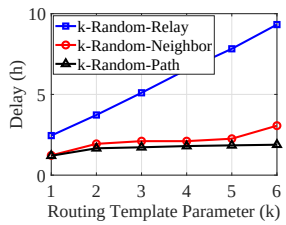


Figure 5.7. Delay overhead for three routing templates when $N = 50$.

Proof of Theorem 6: From the generic result (5.15) in Lemma 1, h_T and $(g_T - 1)/g_T$ depend on routing templates. Note that h_T is available in Theorem 7. Therefore, in the following, we derive $(g_T - 1)/g_T$ under different routing templates.

For k-shortest-relay, we randomly select k relays from the remaining $N - 2$ nodes (excluding source and destination nodes). When the node density λ is sufficiently large such that any node pair is connected, we have $g_{T_1} = \Theta(N^k)$, yielding $(g_{T_1} - 1)/g_{T_1} = \Theta(1)$.

For k-shortest-neighbor, at each hop, one neighbor is selected uniformly at random from k possible neighbors. Assume the average distance for a node pair is d, which can be expressed as a function $d = f(N)$. Then, the total number of paths g_{T_2} can be approximated by $k^{f(N)}$. Since $f(N)$ is an increasing function of N, g_{T_2} increases exponentially in N. Therefore, we have $(g_{T_2} - 1)/g_{T_2} = \Theta(1)$.

For k-shortest-path, it is obvious that $g_{T_3} = \Theta(k)$. Then, inserting it into (5.15) completes the proof. □

5.5 Experimental Evaluation

In this section, we use simulations to show the inference errors and delay performance under routing templates.

5.5.1 Experimental Setups

5.5.1.1 Network Configuration

In experiments, we use RGG to simulate wireless network topologies, where $N \in [20, 100]$ nodes are randomly placed on region $[0, \sqrt{N/\lambda}]$. The node density and transmission range are $\lambda = 5$ and $r = 2$, respectively. In this network, we randomly select $F = \sqrt{N}$ node pairs to have real flow, in which the flow rate of each node pair x_i subjects to the normal distribution with mean $\mu = 10$ and variance $\sigma^2 = 2$.

5.5.1.2 Performance Metrics

Under the worst case that the adversary's inference is unknown, we use the genie bound to measure the inference errors of different routing templates. The method to derive the genie bound is described in Definition 6. The delay is measured by averaging the number of hops for all flows in the network.

5.5.1.3 Routing Template Scenarios

The randomness of each routing template depends on the variable k. Therefore, in our experiments, with respect to k, we consider two different scenarios: (i) fixed k and (ii) dynamic k.

5.5.2 Fixed k Scenario

We first consider the scenario that three templates adopt the fixed $k = 3$. Specifically, For each packet, template T_1: k-random-relay randomly selects 3 relays to forward the packet. For template T_2: k-random-neighbor, each packet is forwarded to one of the 3 neighbors which have the shortest distance to the destination. Template T_3: k-shortest-path directly

selects one path from 3 shortest paths for each flow. We apply $F = \sqrt{N}$ into Theorem 6, and obtain that under any routing template, the inference error satisfies $\Theta(\sqrt{N}) \leq I \leq \Theta(N)$.

Fig. 5.4 depicts the inference error under three randomized routing templates as we change the number of nodes N from 20 to 100. First of all, we see that for every routing template, the inference error increases when the number of nodes increases, which verifies Theorem 6. For instance, the interference error is 718.1 for the k-random-neighbor template when there are 40 nodes. Since each flow on average has the rate 10, we know the attacker has the wrong estimation on at least $718.1/10^2 \approx 7.18$ flows, and this number increases to 8.77 flows when the number of nodes becomes to 100. In addition, we also notice that the difference between templates does not change as we increase N, implying that the difference of induced errors among different templates is on a constant order.

Fig. 5.5 shows the delay overhead of different routing templates. From Fig. 5.5, we observe that the slope of the curve of the k-random-relay template is around 4 times that of the shortest path protocol (e.g., when $N = 100$, the delay of k-random-relay and the shortest path is 5.97 and 1.45 respectively). Note that, under the scenario $k = 3$, $N^{\log k} \ln N < (k+1)\sqrt{N}$, therefore, the delay performance of k-random-neighbor is less than k-random-relay.

5.5.3 Dynamic k Scenario

For the dynamic k scenario, we fix the number of nodes $N = 50$. The total number of paths increases exponentially for k-random-neighbor as k increases. Fig. 5.6 depicts the relationship between the inference error and k. We notice that the inference error increases as k increases for all templates. However, as k keeps increasing, the error starts to increase slightly and remains approximate the same value, which verifies Theorem 6 that reveals the inference error increases sub-linearly in k and a large k will not increase the error significantly.

Fig. 5.7 shows that delay cost of three routing templates with different k. From Fig. 5.7, we observe that the delay increases linearly for k-random-relay because increasing k results in adding extra paths and linear increase of the delay according to Theorem 7. It is also observed that k-random-neighbor and k-random-path have similar delays, because their delays are on the same order $\Theta(\sqrt{N})$ when $\log k$ is small.

5.6 Related Work

5.6.1 Random Routing

Due to the dynamics in the wireless environment, network nodes are usually subject to sleep modes [184], channel fluctuation [188], mobility [189, 190], interference [191] from neighbors. Therefore, to account for such stochastic and asymmetric natures, random routing strategies and their performance have been widely studied in wireless networks [181, 186, 184, 188, 187, 185]. For example, [184, 188, 187, 185, 186] leveraged the random walk to model randomized routing strategies and evaluate the delay and packet delivery ratio. Recently, the work in [61] presents the initial results that being randomized can help security and cause substantial errors in malicious network inference. However, there is still a lack of study regarding how exactly a randomized protocol should be designed to protect network flow infomration from being disclosed. In this chapter, we categorize randomized routing into three templates k-random-relay, k-random-neighbor and k-random-path, and carefully analyze the inference error and delay cost of each template.

5.6.2 Security of Network Inference

Network flow based attack and defense strategies have been explored in a line of works [64, 180, 192, 193, 194, 176, 177, 178, 195, 196, 197]. However, in the wireless network, the PHY or MAC layer activities are public, therefore the flow information is still achievable

through the network inference, which is a family of network monitoring techniques that build the network characteristics from indirect measurements [57, 39, 40, 41, 42]. Existing studies on the network inference mainly focused on optimizing inference methods to obtain more information from given measurements. For example, authors in [41, 58] proposed new ways to detect anomaly links. In [196], authors proposed a faster and practical interference method by combining the tomography and gravity; and a more accurate interference method is investigated in [197] when the measurement frequency is changing. The improved inference methods in fact open up an opportunity for the adversary to obtain the flow information. For example, the work in [43] proposed an attack strategy to retrieve the traffic pattern by using interatrial based traffic analysis. Therefore, in this chapter, we investigate the fundamental reason of such vulnerability and find the randomized routing protocol can lead to more inference errors, so that protect the real flow information against leakage.

5.7 Theoretical Results Analysis

In this section, we first prove Lemma 1, then we introduce and prove other necessary lemmas.

5.7.1 Proof of Lemma 1

Proof: From Model 1, considering the worst case that both $\mathbf{M}$ and $\mathbf{T}$ are available to the adversary, for real flows, we have the linear system

$$\mathbf{y} = \mathbf{RCx}_g, \tag{5.23}$$

where $\mathbf{R} = \mathbf{MT}$ is the routing matrix under the template represented by the matrix $\mathbf{T}$. As aforementioned, the selection matrix $\mathbf{C}$ is unknown by the adversary, then we let $\mathbf{D} = \{\mathbf{d}_i\} \in \mathbb{R}^{\sum g_i \times F}$ be the selection matrix of the adversary, that randomly selects a path for

any flow. Considering the worst case that the ground-truth matrix $\mathbf{C}$ and $\mathbf{D}$ have the same dimension, then from the adversary perspective, the linear system (5.23) can be written as

$$\mathbf{y} = \mathbf{RD}\mathbf{x}_g. \tag{5.24}$$

Taking the least square estimation of $\mathbf{x}_g$, we obtain $\hat{\mathbf{x}}_g = [(\mathbf{RD})^T\mathbf{RD}]^{-1}(\mathbf{RD})^T\mathbf{y}$, then putting it into the genie bound metric (5.7), we have that

$$\begin{aligned} G(\mathbf{x}_g) &= \mathbb{E}\left(\|\hat{\mathbf{x}}_g - \mathbf{x}_g\|_2^2\right) \\ &= \mathbb{E}\left(\|[(\mathbf{RD})^T\mathbf{RD}]^{-1}(\mathbf{RD})^T(\mathbf{RC}\mathbf{x}_g) - \mathbf{x}_g\|_2^2\right) \\ &= \mathbb{E}\left(\|[(\mathbf{RD})^T\mathbf{RD}]^{-1}(\mathbf{RD})^T(\mathbf{RC}) - I]\mathbf{x}_g\|_2^2\right) \\ &= \mathbb{E}\left(\|\mathbf{U}[\mathbf{RC} - \mathbf{RD}]\mathbf{x}_g\|_2^2\right), \end{aligned} \tag{5.25}$$

where $\mathbf{U} = [(\mathbf{RD})^T\mathbf{RD}]^{-1}(\mathbf{RD})^T$. Then by leveraging Lemma 3, the following inequality holds with high probability.

$$\begin{aligned} &\lambda_{\min}(\mathbf{U}^T\mathbf{U})\|\Delta\mathbf{x}_g\|_2^2 \\ \leq &\|\mathbf{U}[\mathbf{RC} - \mathbf{RD}]\mathbf{x}_g\|_2^2 \leq \lambda_{\max}(\mathbf{U}^T\mathbf{U})\|\Delta\mathbf{x}_g\|_2^2. \end{aligned} \tag{5.26}$$

From Lemma 4, $\lambda_{\min}(\mathbf{U}^T\mathbf{U})$ and $\lambda_{\max}(\mathbf{U}^T\mathbf{U})$ can be replaced by $\lambda_{\max}^{-1}(\mathbf{RD}(\mathbf{RD})^T)$ and $\lambda_{\min}^{-1}(\mathbf{RD}(\mathbf{RD})^T)$ respectively, then (5.25) can be expressed as the following inequality with high probability

$$\begin{aligned} &\mathbb{E}\left(\frac{\|\Delta\mathbf{x}_g\|_2^2}{\lambda_{\max}(\mathbf{RD}(\mathbf{RD})^T)}\right) \\ \leq &G(\mathbf{x}_g) \leq \mathbb{E}\left(\frac{\|\Delta\mathbf{x}_g\|_2^2}{\lambda_{\min}(\mathbf{RD}(\mathbf{RD})^T)}\right), \end{aligned} \tag{5.27}$$

where $\Delta = \mathbf{RC} - \mathbf{RD}$. Then leveraging Lemma 5, we can derive the following asymptotically solution

$$\frac{\mathbb{E}\|\Delta \mathbf{x}_g\|_2^2}{\Theta\left(h(N) + \frac{Fh(N)^2}{N}\right)} \leq I_T \leq \frac{\mathbb{E}\|\Delta \mathbf{x}_g\|_2^2}{\Theta(h(N))}. \tag{5.28}$$

Now we derive the numerator. Denote entries in Δ as $\{\delta_{ij}\}_{i\in[1,L],j\in[1,F]}$, where each entry $\delta_{ij} \in \{1, -1, 0\}$. Because the adversary randomly select one path for each flow, we have that

$$\begin{aligned}
\Pr\{\delta_{ij} = 1\} =& \Pr\{\delta_{ij} = 1 | \mathbf{c}_j \neq \mathbf{d}_j\} \Pr\{\mathbf{c}_j \neq \mathbf{d}_j\} \\
=& \frac{g_j - 1}{g_j} \Pr\{\delta_{ij} = 1 | \mathbf{c}_j \neq \mathbf{d}_j\} \\
=& \frac{g_j - 1}{g_j} \Theta\left(\frac{h(N)}{N}\right)\left(1 - \Theta\left(\frac{h(N)}{N}\right)\right).
\end{aligned} \tag{5.29}$$

Furthermore, it is easy to know $\Pr\{\delta_{ij} = -1\} = \Pr\{\delta_{ij} = 1\}$. Since all nodes are uniformly distributed in a region Ω. Then for routing template T, $\mathbb{E}(g_i) = g_T$ for $1 \leq i \leq F$, and we have the following two equations:

$$\begin{aligned}
\mathbb{E}\{\delta_{ij}\} &= \Pr\{\delta_{ij} = 1\} - \Pr\{\delta_{ij} = -1\} \\
&= \frac{g_T - 1}{g_T} \Theta\left(\frac{h(N)}{N}\right),
\end{aligned} \tag{5.30}$$

and

$$\begin{aligned}
\mathbb{E}\{\delta_{ij}^2\} &= \Pr\{\delta_{ij} = 1\} + \Pr\{\delta_{ij} = -1\} \\
&= \frac{2(g_T - 1)}{g_T} \Theta\left(\frac{h(N)}{N}\right).
\end{aligned} \tag{5.31}$$

Then, we have the lower bound as

$$\begin{aligned}\mathbb{E}\|\Delta\mathbf{x}_g\|_2^2 &= \mathbb{E}\left(\sum_{i=1}^{L}\left(\sum_{j=1}^{F}\delta_{ij}x_j\right)^2\right)\\ &= \Theta(N)\mathbb{E}\left(\left(\sum_{j=1}^{F}\delta_{ij}x_j\right)^2\right)\\ &\geq \Theta(N)\left(\mathbb{E}\left(\sum_{j=1}^{F}\delta_{ij}x_j\right)\right)^2\\ &= \Theta\left(\frac{[F\mu(g_T-1)h(N)]^2}{g_T^2N}\right).\end{aligned} \tag{5.32}$$

On the other hand, by using Cauchy-Schwarz inequality, the upper bound is

$$\begin{aligned}\mathbb{E}\|\Delta\mathbf{x}_g\|_2^2 &= \Theta(N)\mathbb{E}\left(\left(\sum_{j=1}^{F}\delta_{ij}x_j\right)^2\right)\\ &\leq \Theta(N)\mathbb{E}\left(\sum_{j=1}^{F}\delta_{ij}^2\right)\mathbb{E}\left(\sum_{j=1}^{F}x_j^2\right)\\ &= \Theta\left(\frac{2F^2(g_T-1)(\mu^2+\sigma^2)h(N)}{g_T}\right).\end{aligned} \tag{5.33}$$

Replacing (5.32), (5.33) into (5.28), we complete the proof. □

5.7.2 Proof of Necessary Lemmas

Lemma 2 *The probability that each element in the matrix* $\mathbf{R}$ *is*

$$\Pr\{r_{ij}=1\} = \Theta\left(\frac{h(N)}{N}\right). \tag{5.34}$$

Proof: From Model 1, we have that

$$\Pr\{r_{ij}=1\} = \Pr\{\text{path } p_j \text{ traverses link } l_i\}. \tag{5.35}$$

We assume there are u links in the network and path p_j contains u_j links, then (5.35) can be rewritten as

$$\Pr\{\text{flow } f_j \text{ traverses link } l_i | u, u_j\} = u_j/u. \tag{5.36}$$

Take the expectation to u and u_j, we have that

$$\Pr\{r_{ij} = 1\} = \mathbb{E}_{u_j}\left(\mathbb{E}_u\left(\frac{u_j}{u}|u_j\right)\right). \tag{5.37}$$

From Taylor series at $\mathbb{E}(u|u_j)$, we have

$$\mathbb{E}_u\left(\frac{u_j}{u}|u_j\right) = \frac{u_j}{\mathbb{E}_u(u|u_j)} + f\left(O\left(\frac{\mathrm{Var}_u(u|u_j)}{\mathbb{E}^3(u|u_j)}\right)\right). \tag{5.38}$$

By leveraging Lemma 7, (5.37) can be derived as

$$\Pr\{r_{ij} = 1\} = \mathbb{E}\left(\frac{u_j}{N(\lambda\pi r^2 - 1)/2}\right) = \frac{\mathbb{E}(u_j)}{\Theta(N)}. \tag{5.39}$$

According to Definition 5, $\mathbb{E}(u_j) = h(N)$, then we complete the proof. □

Lemma 3 *For a matrix* $\mathbf{A}$*, it satisfies*

$$\lambda_{\min}(\mathbf{A})\|\alpha\|_2^2 \leq \|\mathbf{A}\alpha\|_2^2 \leq \lambda_{\max}(\mathbf{A})\|\alpha\|_2^2, \tag{5.40}$$

where $\lambda_{\min}(\mathbf{A})$ *and* $\lambda_{\max}(\mathbf{A})$ *is the minimum and maximum eigenvalues of matrix* $\mathbf{A}$*.*

Proof: Clearly, we know

$$\|\mathbf{A}\alpha\|_2^2 = \|\alpha^T\mathbf{A}^T\mathbf{A}\alpha\| = \frac{\|\alpha^T\mathbf{A}^T\mathbf{A}\alpha\|}{\alpha^T\alpha}\|\alpha\|_2^2. \tag{5.41}$$

According to [198], $\frac{\|\alpha^T\mathbf{A}^T\mathbf{A}\alpha\|}{\alpha^T\alpha}$ has the minimum value $\lambda_{\min}(\mathbf{A}^T\mathbf{A})$ and the maximum value $\lambda_{\max}(\mathbf{A}^T\mathbf{A})$, then we can complete the proof. □

Lemma 4 *For a matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *where* $m > n$, *let* $\mathbf{G} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, *then* $\lambda_{\max}(\mathbf{G}^T\mathbf{G}) = \lambda_{\min}^{-1}(\mathbf{A}^T\mathbf{A})$ *and* $\lambda_{\min}(\mathbf{G}^T\mathbf{G}) = \lambda_{\max}^{-1}(\mathbf{A}^T\mathbf{A})$.

Proof: Based on the singular value decomposition, for the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, there exists a couple of unitary matrices $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n,n}$, such that $\mathbf{A} = \mathbf{U}\Lambda\mathbf{V}^T$ where $\Lambda = \text{diag}(s_1, \cdots, s_n) \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix. The numbers $s_i = \lambda_i(\sqrt{\mathbf{A}^T\mathbf{A}})$ for $i = 1, \cdots, n$ are singular values of $\mathbf{A}$. If one sees the diagonal matrix $D \coloneqq \text{diag}(s_1^2, \cdots, s_n^2) \in \mathbb{R}^{n\times n}$, we have

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}D\mathbf{V}^T. \tag{5.42}$$

Then $(\mathbf{A}^T\mathbf{A})^{-1} = \mathbf{V}D^{-1}\mathbf{V}^T$, and $\mathbf{G}$ can be derived as

$$\mathbf{G} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T = \mathbf{V}D^{-1}\mathbf{V}^T\mathbf{V}\Lambda\mathbf{U}^T = \mathbf{V}\Lambda^{-1}\mathbf{U}^T, \tag{5.43}$$

where $\Lambda^{-1} = \text{diag}(s_1^{-1}, \cdots, s_n^{-1}) \in \mathbb{R}^{n\times m}$. Similarly,

$$\mathbf{G}^T\mathbf{G} = \mathbf{U}(\Lambda^{-1})^2\mathbf{U}^T = \mathbf{U}D^{-1}\mathbf{U}^T. \tag{5.44}$$

Comparing (5.42) and (5.44) we know that $\lambda(\mathbf{G}^T\mathbf{G}) = \lambda^{-1}(\mathbf{A}^T\mathbf{A})$, which completes the proof. □

Lemma 5 *For a random binary matrix* $\mathbf{A}$ *with size* $L \times F$, *where the expectation of each entry of* $\mathbf{A}$ *is* $\mathbb{E}(a_{ij}) = \Theta(h(N)/N)$ *for* $h(N) = O(N)$ *and* $L = \Theta(N)$. *Then if* $F \to \infty$ *with* $\lim_{L\to\infty} F/L < \infty$, *then the following two statements are satisfied with high probability,*

1. *for the minimum eigenvalue,* $\lambda_{\min}(\mathbf{A}^T\mathbf{A}) = \Theta(h(N))$
2. *for the maximum eigenvalue,*

$$\lambda_{\max}(\mathbf{A}^T\mathbf{A}) \leq \Theta\left(h(N) + \frac{Fh^2(N)}{N}\right)$$

Proof: For the statement (1), from Lemma 8, we have a function f such that the $\mathrm{Var}(f\mathbf{A}) = 1$. Then statement (1) can be rewritten as

$$\lambda_{\min}(\mathbf{A}^T\mathbf{A}) = \frac{L}{f^2}\lambda_{\min}(L^{-1}(c\mathbf{A})^T(c\mathbf{A})). \tag{5.45}$$

From [198], we have $\lambda_{\min}(L^{-1}(c\mathbf{A})^T(c\mathbf{A})) = \Theta(1)$ with high probability. Then (5.45) can be further derived as

$$\lambda_{\min}(\mathbf{A}^T\mathbf{A}) = \frac{L}{f^2}\Theta(1) = \frac{\Theta(N)}{\Theta(h(N)/N)} = \Theta(h(N)). \tag{5.46}$$

For the statement (2), we define two matrix $\mathbf{U}$ and $\mathbf{V}$ where $\mathbf{V}$ is an all-one matrix and each entry in $u_i \in \mathbf{U}$ satisfies $\mathbb{E}(u_i) = 0$. Then we have $\mathbf{A} = \mathbf{U} + \mathbf{V}h(N)/N$. Applying to $\mathbf{A}^T\mathbf{A}$, we have that

$$\begin{aligned}\lambda_{\max}(\mathbf{A}^T\mathbf{A}) &= \lambda_{\max}((\mathbf{U}^T + \mathbf{V}^T h(N)/N)(\mathbf{U} + \mathbf{V}h(N)/N)) \\ &\leq \lambda_{\max}(\mathbf{U}^T\mathbf{U}) + 2h(N)/N\lambda_{\max}(\mathbf{U}^T\mathbf{V}) \\ &\quad + (h(N)/N)^2\lambda_{\max}(\mathbf{V}^T\mathbf{V}).\end{aligned} \tag{5.47}$$

For the second term in (5.47), since $\mathbf{V}$ is an all-one matrix with rank 1, then we have that

$$\lambda_{\max}(\mathbf{U}^T\mathbf{V}) = \mathrm{tr}\{\mathbf{U}^T\mathbf{V}\} = \sum_{u_{ij}\in\mathbf{U}} u_{ij}. \tag{5.48}$$

According to the large number law, $\lambda_{\max}(\mathbf{U}^T\mathbf{V}) = o(NF)^{1/p}$ for $1 < p < 2$. Then similarly,

$$\lambda_{\max}(\mathbf{V}^T\mathbf{V}) = \sum_{v_{ij}\in\mathbf{V}} 1 = \Theta(FN) \tag{5.49}$$

From [198], the first term satisfy $\lambda_{\max}(\mathbf{U}^T\mathbf{U}) = \Theta(h(N))$. Replacing $\lambda_{\max}(\mathbf{U}^T\mathbf{V})$, $\lambda_{\max}(\mathbf{V}^T\mathbf{V})$, $\lambda_{\max}(\mathbf{U}^T\mathbf{U})$ into (5.47), we can complete the proof. □

Lemma 6 *For network $\mathcal{G}$ with N nodes, the average delay over all flows by leveraging the shortest path protocol is $d = \Theta(\sqrt{N})$.*

Proof: The delay is positively related to the distance between the source and destination nodes. Let e_i be the Euclidean distance for flow f_i. Since each hop covers a distance of $\Theta(r)$, the delay of flow by using the shortest path is f_i is $\Theta(e_i/r)$. The average delay over all flows is $\Theta(\frac{1}{N}\sum_{i=1}^{N} e_i/r)$. Since all nodes are randomly distributed in a region $\Omega = [0, \sqrt{N/\lambda}]^2$, for a large N, we have $\frac{1}{N}\sum_{i=1}^{N} e_i = \Theta\left(\sqrt{N/\lambda}\right)$. Therefore the average delay can be derived as $d = \Theta\left(\frac{\sqrt{N/\lambda}}{r}\right) = \Theta(\sqrt{N})$, which completes the proof. □

Lemma 7 *In a RGG with N nodes, L links, and density λ and transmission range r, then L is on the order of N with high probability, i.e.,* $\Pr\{L = \Theta(N)\} = 1 - \Theta(e^{-\Theta(1)N})$.

Proof: In RGG, the degree of an arbitrary node equals to the number of nodes dropping into the transmission range of this node, then the degree of an arbitrary subjects to Poisson distribution with parameter $\lambda\pi r^2 - 1$, and thus the distribution of the total number of link is also Poisson distribution with parameter $N(\lambda\pi r^2 - 1)/2$. Then based on the Chernoff bound, for a small constant $c_1 < (\lambda - 1)\pi r^2$ and a large constant $c_2 > (\lambda - 1)\pi r^2$, we have the following upper and lower bounds

$$\begin{aligned}\Pr\{L \geq c_1 N\} &\leq 1 - \frac{e^{\frac{-N(\lambda\pi r^2-1)}{2}}\left(\frac{eN\lambda(\pi r^2-1)}{2}\right)^{c_1 N}}{(c_1 N)^{c_1 N}} \\ &= 1 - e^{\frac{-N(\lambda\pi r^2-1)}{2}} e^{c_1 N \log\left(\frac{e\lambda(\pi r^2-1)}{2c_1}\right)} \\ &= 1 - \Theta(e^{-\Theta(1)N})\end{aligned} \tag{5.50}$$

and

$$\begin{aligned}\Pr\{L \le c_2 N\} &\ge 1 - \frac{e^{\frac{-N(\lambda\pi r^2-1)}{2}} \left(\frac{eN\lambda(\pi r^2-1)}{2}\right)^{c_2 N}}{(c_2 N)^{c_2 N}} \\ &= 1 - \Theta(e^{-\Theta(1)N})\end{aligned} \tag{5.51}$$

where $\xi = \Theta(e^{-\Theta(1)N})$. Then we complete the proof. □

Lemma 8 *For the routing matrix* $\mathbf{R}$ *in Lemma 2, there exists a function* $f = \sqrt{N/h(N)}$ *such that each entry in* $\mathbf{R}$ *has finite mean and invariance 1.*

Proof: Denote each entry in $f\mathbf{R}$ as e_{ij}, then the mean of e_{ij} can be derived as

$$\mathbb{E}(e_{ij}) = \mathbb{E}(f r_{ij}) = f \Pr\{r_{ij} = 1\} = f\Theta\left(\sqrt{\frac{h(N)}{N}}\right) = \Theta(1), \tag{5.52}$$

and the variance is

$$\begin{aligned}\mathrm{Var}(e_{ij}) &= \mathbb{E}(e_{ij}^2) - \mathbb{E}^2(e_{ij}) = f^2\mathbb{E}(r_{ij}^2) - \Theta(1) \\ &= f^2\Theta\left(\frac{h(N)}{N}\right) - \Theta(1) = \Theta(1).\end{aligned} \tag{5.53}$$

Then we can complete the proof. □

Lemma 9 *For a network* $\mathcal{G}$ *with* N *nodes and its corresponding generated network* $\mathcal{G}_{ij}^k$ *with* N_{ij}^k *nodes in the proof of Theorem 7. We have the lower bound* $\mathbb{E}(N_{ij}^k) > \Theta(N^{\log_{\delta-1}^{k-1}})$ *if* $k > 2$ *and* $\mathbb{E}(N_{ij}^k) > \Theta(\log(N))$ *if* $k = 2$.

Proof: For source node v_i, we define node set $\mathbb{M}_i^d$ containing nodes which have distance d with the source node v_i, and $\mathbb{M}_i^0 = \{v_i\}$. Similarly, $\mathcal{K}_i^d$ be the corresponding node set after removing nodes from $\mathbb{M}_i^d$ based on the rule in the proof of Theorem 7. Let $|\mathbb{M}_i^d| = M_i^d$ and

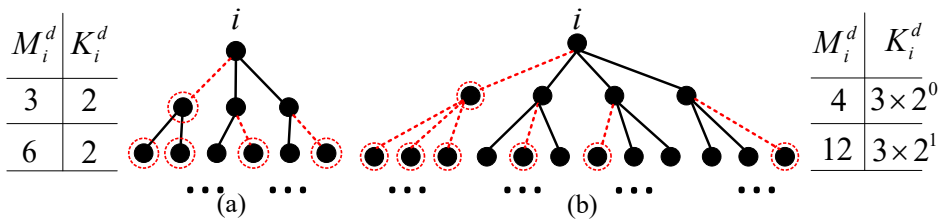


Figure 5.8. Examples of two network topologys, where (a) $\delta = 3$, and $k = 2$, and (b) $\delta = 4$, and $k = 3$, and dotted lines indicate removed links and dotted circles denote removed nodes.

$|\mathcal{K}_i^d| = K_i^d$. Since the network is connected, then we have that

$$\sum_{d=0}^{l} M_i^d = N, \quad \sum_{d=0}^{l} K_i^d = N_{ij}^k, \tag{5.54}$$

where l is the maximum distance.

To derive the low bound of $\mathbb{E}(N_{ij}^k)$, we should remove the most nodes from N nodes. To do so, we consider an extreme case satisfies two requirements:

1. the destination node is located at the maximum distance l away from the source node v_i,

2. the network is a tree structure, i.e., except for the source node, for any node $v_d \in \mathbb{M}_i^d$, it connects with only one node $v_{d-1} \in \mathbb{M}_i^{d-1}$ (as shown in Fig. 5.8).

Under this case, once a node $v_d \in \mathbb{M}_i^d$ is removed, all nodes that are connected with this node in $\mathbb{M}_i^{d+j}$ for $j \geq 1$ will be removed. Then the expected number of nodes with distance d can be given by

$$\begin{cases} \mathbb{E}(M_i^d) = \delta(\delta-1)^{(d-1)}, & \text{for the network } \mathcal{G} \\ \mathbb{E}(K_i^d) \geq k(k-1)^{(d-1)}, & \text{for the network } \mathcal{G}_{ij}^k. \end{cases} \tag{5.55}$$

Then we sum all distances together. For networks $\mathcal{G}$ and $\mathcal{G}_{ij}^{k}$, according to (5.55), we obtain two equations

$$\sum_{d=1}^{l} \mathbb{E}(M_i^d) = N - 1 = \sum_{d=1}^{l} \delta(\delta - 1)^{(d-1)}, \tag{5.56}$$

and

$$\sum_{d=1}^{l} \mathbb{E}(K_i^d) = \mathbb{E}(N_{ij}^k) - 1 \geq \sum_{d=1}^{l} k(k-1)^{(d-1)}. \tag{5.57}$$

Combining (5.56) and (5.57) together, we can derive the lower bound of $\mathbb{E}(N_{ij}^k)$. Note that according to (5.57), $k = 2$ is a special scenario, therefore we split our results into two scenarios, i.e., $k = 2$ and $k > 2$. Fig. 5.8 shows an illustrative example of $k = 2$ and $k = 3$. Then after jointly manipulating (5.56) and (5.57), we complete the proof. □

5.8 Summary

In this chapter, we revisited the network inference and found that the inference accuracy depends on routing protocols. We categorized randomized routing protocols into three templates, k-random-relay, k-random-neighbor and k-random-path, and theoretically analyzed the inference errors and incurred delay costs of them. We conducted simulations to confirm that randomized routing templates can inflict different inference errors and delays, yielding different capabilities to prevent the flow information leakage.

Chapter 6: Conclusion

In this dissertation, we proposed four data-oriented approaches that renovate or calibrate current mobile network and security designs.

For the mobile network design, we found that due to the broadcast nature of wireless channel, the intensive traffic environment can always jeopardize the performance within a mobile network. To cope with this issue, we revisited the fundamental part within wireless mobile networks, and deal with two main aspects to improve mobile network system performance: packet corruption and packet collision. The first one is due to the unpredictable wireless fading, and the second one is resulted from concurrent transmissions. We discovered that the inter-packet data and RTS data can be leveraged to further improve mobile network efficiency and reliability.

For the network security design, we revisited network tomography from a data-driven perspective, and found that there are two vulnerabilities. The first one is of the measurement integrity. By taking advantage of this vulnerability, we developed an attack strategy, called measurement integrity attack, which not only destroys the measuring system, but it can even mislead the system to scapegoat other innocent users. The second vulnerability is of the measurement confidentiality. In particular, To prevent disclosing the flow information by network inference, we found that random routing strategies are capable of hiding the flow information. Then, by leveraging the measurement data, we systematically studied the behavior of different randomized routing protocols, and explore the fundamental reason why randomized routing strategies can prevent information leakage against network inference.

References

[1] Jan-Jaap Van de Beek, Magnus Sandell, and Per Ola Borjesson. ML estimation of time and frequency offset in OFDM systems. *IEEE Trans. Signal Process.*, 45:1800–1805, 1997.

[2] Paul H Moose. A technique for orthogonal frequency division multiplexing frequency offset correction. *IEEE Trans. Commun.*, 42:2908–2914, 1994.

[3] Linda M Davis, Iain B Collings, and Peter Hoeher. Joint MAP equalization and channel estimation for frequency-selective and frequency-flat fast-fading channels. *IEEE Trans. Commun.*, 49:2106–2114, 2001.

[4] Timothy M Schmidl and Donald C Cox. Robust frequency and timing synchronization for OFDM. *IEEE trans. commun.*, 45:1613–1621, 1997.

[5] Mihai-A Badiu, Gunvor E Kirkelund, Carles Navarro Manchón, Erwin Riegler, and Bernard H Fleury. Message-passing algorithms for channel estimation and decoding using approximate inference. In *Prof. of IEEE ISIT*, 2012.

[6] Georgios Angelopoulos, Muriel Medard, and Anantha Chandrakasan. Harnessing partial packets in wireless networks: Throughput and energy benefits. *IEEE Trans. Wireless Commun.*, 2016.

[7] Peter Larsson, Lars Kildehøj Rasmussen, and Mikael Skoglund. Throughput analysis of hybrid-AQR–a matrix exponential distribution approach. *IEEE Trans. Commun.*, 64:416–428, 2016.

[8] Yaoyu Wang, Wenzhong Li, and Sanglu Lu. The capability of error correction for burst-noise channels using error estimating code. In *Prof. of IEEE SECON*, 2016.

[9] Dai Jia, Zesong Fei, Jinhong Yuan, Shuang Tian, and Jingming Kuang. A hybrid EF/DF protocol with rateless coded network code for two-way relay channels. *IEEE Trans.Commun.*, 64:3133–3147, 2016.

[10] Mohammad Sadegh Mohammadi, Qi Zhang, and Eryk Dutkiewicz. Reading damaged scripts: partial packet recovery based on compressive sensing for efficient random linear coded transmission. *IEEE Trans. Commun.*, 64:3296–3310, 2016.

[11] Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11*, 2013.

[12] Low-rate wireless personal area networks (LR-WPANs). *IEEE Std 802.15.4*, 2011.

[13] Mahanth Gowda, Souvik Sen, Romit Roy Choudhury, and Sung-Ju Lee. Cooperative packet recovery in enterprise WLANs. In *Prof. of IEEE INFOCOM*, 2013.

[14] Bo Han, Aaron Schulman, Francesco Gringoli, Neil Spring, Bobby Bhattacharjee, Lorenzo Nava, Lusheng Ji, Seungjoon Lee, and Robert R Miller. Maranello: Practical partial packet recovery for 802.11. In *Prof. of NSDI*, 2010.

[15] Mei-Hsuan Lu, Peter Steenkiste, and Tsuhan Chen. Design, implementation and evaluation of an efficient opportunistic retransmission protocol. In *Prof. of ACM MobiCom*, 2009.

[16] Shyamnath Gollakota and Dina Katabi. Zigzag decoding: combating hidden terminals in wireless networks. In *Prof. of ACM SIGCOMM*, 2008.

[17] Advanced radio settings. https://www.cisco.com/assets/sol/sb/isa500_emulator/help/guide/ae1269129.html, 2019.

[18] Tplink user guide. https://static.tp-link.com/2017/201712/20171212/1910012191_TL-WR902AC\%203.0_UG.pdf, 2017.

[19] Arris router setup. http://www.cktv.ru/files/modem/ARRIS_Router_Setup_Web_GUI_UG.pdf, 2012.

[20] Dlink user manual. http://files.dlink.com.au/Products/DSL-2740M/Manuals/DSL-2740M_A1_Manual_v1.00(DI).pdf, 2017.

[21] Linksys user guide. https://content.etilize.com/User-Manual/1026969914.pdf, 2014.

[22] Zhijun Li and Tian He. Webee: Physical-layer cross-technology communication via emulation. In *Proc. of ACM MobiCom*, 2017.

[23] Xinyu Zhang and Kang G Shin. Cooperative carrier signaling: Harmonizing coexisting wpan and wlan devices. *IEEE/ACM Transactions on Networking (TON)*, 21:426–439, 2013.

[24] Song Min Kim and Tian He. Freebee: Cross-technology communication via free side-channel. In *Proc. of ACM MobiCom*, 2015.

[25] Renato DC Monteiro and Ilan Adler. Interior path following primal-dual algorithms. part i: Linear programming. *Mathematical programming*, 44(1-3):27–41, 1989.

[26] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on Information Theory*, 51, 2005.

[27] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[28] Liang Ma, Ting He, Ananthram Swami, Don Towsley, Kin K. Leung, and Jessica Lowe. Node failure localization via network tomography. In *Proc. of ACM IMC*, 2014.

[29] Ting He, Chang Liu, Ananthram Swami, Don Towsley, Theodoros Salonidis, Andrei Iu. Bejan, and Paul Yu. Fisher information-based experiment design for network tomography. In *Proc. of IEEE SIGMETRICS*, 2015.

[30] Hongyi Yao, S. Jaggi, and Minghua Chen. Network coding tomography for network failures. In *Proc. of IEEE INFOCOM*, 2010.

[31] Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network tomography: Recent developments. *Statistical Science*, 19:499–517, 2004.

[32] Joseph D Horton and Alejandro Lopez-Ortiz. On the number of distributed measurement points for network tomography. In *Proc. of ACM IMC*, 2003.

[33] Chung-Kai Yu, Kwang-Cheng Chen, and Shin-Ming Cheng. Cognitive radio network tomography. *IEEE Trans. Veh. Technol.*, 59, 2010.

[34] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Sensor network tomography: Monitoring wireless sensor networks. *ACM SIGCOMM Computer Communication Review*, 32, 2002.

[35] Yongjun Li, Wandong Cai, Guangli Tian, and Wei Wang. Loss tomography in wireless sensor network using Gibbs sampling. In *Proc. of EWSN*, 2007.

[36] Tian Bu, Nick Duffield, Francesco Lo Presti, and Don Towsley. Network tomography on general topologies. In *Proc. of ACM SIGMETRICS*, 2002.

[37] Mohammad Hamed Firooz and Sumit Roy. Link delay estimation via expander graphs. *IEEE Trans. Commun.*, 62:170–180, 2014.

[38] Michael Rabbat, Robert Nowak, and Mark Coates. Multiple source, multiple destination network tomography. In *Proc. of IEEE INFOCOM*, 2004.

[39] Augustin Soule, Anukool Lakhina, Nina Taft, Konstantina Papagiannaki, Kave Salamatian, Antonio Nucci, Mark Crovella, and Christophe Diot. Traffic matrices: balancing measurements, inference and modeling. In *ACM SIGMETRICS*, 2005.

[40] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network neutrality inference. In *Proc. of ACM SIGCOMM*, 2014.

[41] Ting He. Distributed link anomaly detection via partial network tomography. 2018.

[42] Satyajeet Ahuja, Srinivasan Ramasubramanian, and Marwan Krunz. SRLG failure localization in all-optical networks using monitoring cycles and paths. In *IEEE INFOCOM*, 2008.

[43] Yunzhong Liu, Rui Zhang, Jing Shi, and Yanchao Zhang. Traffic inference in anonymous manets. In *IEEE SECON*, 2010.

[44] Aiyou Chen, Jin Cao, and T. Bu. Network tomography: Identifiability and fourier domain estimation. *IEEE Trans. Signal Process.*, 58:6029–6039, 2010.

[45] Ting He, Liang Ma, Athanasios Gkelias, Kin K. Leung, Ananthram Swami, and Don Towsley. Robust monitor placement for network tomography in dynamic networks. In *Proc. of IEEE INFOCOM*, 2016.

[46] Liang Ma, Ting He, Kin K. Leung, Ananthram Swami, and Don Towsley. Monitor placement for maximal identifiability in network tomography. In *Proc. of IEEE INFOCOM*, 2014.

[47] Liang Ma, Ting He, Kin K. Leung, Ananthram Swami, and Don Towsley. Identifiability of link metrics based on end-to-end path measurements. In *Proc. of ACM IMC*, 2013.

[48] Craig A Shue, Andrew J Kalafut, and Minaxi Gupta. Abnormally malicious autonomous systems and their Internet connectivity. *IEEE/ACM Trans. Netw.*, 20:220–230, 2012.

[49] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: routing attacks on privacy in Tor. In *Proc. of USENIX Security*, 2015.

[50] Lucian Constantin. Attackers slip rogue, backdoored firmware onto Cisco routers. *PC World - Security*, 2015.

[51] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Proc. of IEEE S&P*, 2003.

[52] Patrick Tague and Radha Poovendran. Modeling node capture attacks in wireless sensor networks. In *Proc. of Allerton Conference on Communication, Control, and Computing*, 2008.

[53] Manjesh K Hanawal, Diep N Nguyen, and Marwan Krunz. Jamming attack on in-band full-duplex communications: Detection and countermeasures. In *IEEE INFOCOM*, 2016.

[54] Zhenghao Zhang and Avishek Mukherjee. Friendly channel-oblivious jamming with error amplification for wireless networks. In *IEEE INFOCOM*, 2016.

[55] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38, 2002.

[56] Chai K Toh. *Ad hoc mobile wireless networks: protocols and systems*. Pearson Education, 2001.

[57] AHeroIII Coates, Alfred O Hero III, Robert Nowak, and Bin Yu. Internet tomography. *IEEE Signal Process. Mag.*, 19, 2002.

[58] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, 2004.

[59] Morteza Mardani and Georgios B Giannakis. Estimating traffic and anomaly maps via network tomography. *IEEE/ACM Trans. Netw.*, 24, 2016.

[60] Hiroyuki Kasai, Wolfgang Kellerer, and Martin Kleinsteuber. Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking. *IEEE Trans. Netw. Service Manag.*, 13, 2016.

[61] Zhuo Lu and Cliff Wang. Network anti-inference: A fundamental perspective on proactive strategies to counter flow inference. In *IEEE INFOCOM*, 2015.

[62] Jochen W Guck, Amaury Van Bemten, Martin Reisslein, and Wolfgang Kellerer. Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation. *Commun. Surveys Tuts.*, 20, 2018.

[63] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. Inf. Theory*, 55, 2009.

[64] Paul Syverson, R Dingledine, and N Mathewson. Tor: The secondgeneration onion router. In *USENIX Security*, 2004.

[65] Song Noh, Michael D Zoltowski, Youngchul Sung, and David J Love. Pilot beam pattern design for channel estimation in massive MIMO systems. *IEEE J. Sel. Topics Signal Process.*, 8:787–801, 2014.

[66] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[67] David Tse and Pramod Viswanath. *Fundamentals of wireless communication*. Cambridge university press, 2005.

[68] Xiufeng Xie, Eugene Chai, Xinyu Zhang, Karthikeyan Sundaresan, Amir Khojastepour, and Sampath Rangarajan. Hekaton: Efficient and practical large-scale MIMO. In *Prof. of ACM MobiCom*, 2015.

[69] Yang Chen, Xiaofang Liu, Yangyi Cui, Jiming Zou, and Shiyan Yang. A multiwinding transformer cell-to-cell active equalization method for lithium-ion batteries with reduced number of driving circuits. *IEEE Trans. Power Electron.*, 31:4916–4929, 2016.

[70] Jun Huang, Yu Wang, and Guoliang Xing. LEAD: leveraging protocol signatures for improving wireless link performance. In *Prof. of ACM MobiSys*, 2013.

[71] Peter Larsson, Lars K Rasmussen, and Mikael Skoglund. Throughput analysis of ARQ schemes in Gaussian block fading channels. *IEEE Trans. Commun.*, 62:2569–2588, 2014.

[72] Ali Chelli, Emna Zedini, Mohamed-Slim Alouini, John R Barry, and Matthias Pätzold. Performance and delay analysis of hybrid ARQ with incremental redundancy over double Rayleigh fading channels. *IEEE Trans. Wireless Commun.*, 13:6245–6258, 2014.

[73] Mihai-Alin Badiu, Carles Navarro Manchón, and Bernard Henri Fleury. Message-passing receiver architecture with reduced-complexity channel estimation. *IEEE Commun. Lett.*, 17:1404–1407, 2013.

[74] Maya Rodrig, Charles Reis, Ratul Mahajan, David Wetherall, John Zahorjan, and Ed Lazowska. CRAWDAD dataset uw/sigcomm2004 (v. 2006-10-17). Downloaded from http://crawdad.org/uw/sigcomm2004/20061017, October 2006.

[75] Aaron Schulman, Dave Levin, and Neil Spring. CRAWDAD dataset umd/sigcomm2008 (v. 2009-03-02). Downloaded from http://crawdad.org/umd/sigcomm2008/20090302, March 2009.

[76] Olivier Besson and Petre Stoica. On parameter estimation of MIMO flat-fading channels with frequency offsets. *IEEE Trans. Signal Process.*, 51:602–613, 2003.

[77] Qiang Li, Kah Chan Teh, and Kwok Hung Li. Low-complexity channel estimation and turbo equalisation for high frequency channels. *IET Commun.*, 7:980–987, 2013.

[78] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[79] Monitoring 802.11n and 802.11ac networks. http://www.tamos.com/htmlhelp/commwifi/monitoring_802_11n_networks.htm.

[80] Matt Ettus. USRP user's and developer's guide. *Ettus Research LLC*, 2005.

[81] Siavash M Alamouti. A simple transmit diversity technique for wireless communications. *IEEE J. Sel. Areas Commun.*, 16:1451–1458, 1998.

[82] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. CSMA/CN: carrier sense multiple access with collision notification. *IEEE/ACM Trans. Netw.*, 20:544–556, 2012.

[83] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. No time to countdown: Migrating backoff to the frequency domain. In *Prof. of ACM MobiCom*, 2011.

[84] Songwei Fu and Yan Zhang. CRAWDAD dataset due/packet-delivery (v. 2015-04-01). Downloaded from http://crawdad.org/due/packet-delivery/20150401/packet-metadata, April 2015. traceset: packet-metadata.

[85] Adnan Iqbal, Khurram Shahzad, Syed Ali Khayam, and Yongju Cho. CRAWDAD dataset niit/bit_errors (v. 2008-07-08). Downloaded from http://crawdad.org/niit/bit_errors/20080708/802.15.4, July 2008. traceset: 802.15.4.

[86] Guanghui Liu, Liaoyuan Zeng, Hongliang Li, Linfeng Xu, and Zhengning Wang. Adaptive interpolation for pilot-aided channel estimator in OFDM system. *IEEE Trans. Broadcast.*, 60:486–498, 2014.

[87] Alireza Movahedian and Michael McGuire. On the capacity of iteratively estimated channels using LMMSE estimators. *IEEE Trans. Veh. Technol.*, 64:97–107, 2015.

[88] Jin Xie, Wei Hu, and Zhenghao Zhang. Revisiting partial packet recovery in 802.11 wireless LANs. In *Prof. of ACM MobiSys*, 2011.

[89] Yaxiong Xie, Zhenjiang Li, Mo Li, and Kyle Jamieson. Augmenting wide-band 802.11 transmissions via unequal packet bit protection. In *Prof. of IEEE INFOCOM*, 2016.

[90] Jun Huang, Guoliang Xing, Jianwei Niu, and Shan Lin. CodeRepair: PHY-layer partial packet recovery without the pain. In *Prof. of IEEE INFOCOM*, 2015.

[91] Linghe Kong and Xue Liu. mZig: Enabling multi-packet reception in ZigBee. In *Prof. of ACM MobiCom*, 2015.

[92] Xiufeng Xie, Xinyu Zhang, Swarun Kumar, and Li Erran Li. piStream: Physical layer informed adaptive video streaming over LTE. In *Prof. of ACM MobiCom*, 2015.

[93] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. Passive Wi-Fi: Bringing low power to Wi-Fi transmissions. In *Prof. of NSDI*, 2016.

[94] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Prof. of ACM SIGCOMM*, 2010.

[95] Bo Chen, Vivek Yenamandra, and Kannan Srinivasan. Flexradio: Fully flexible radios and networks. In *Prof. of NSDI*, 2015.

[96] Adriana B Flores, Sadia Quadri, and Edward W Knightly. A scalable multi-user uplink for Wi-Fi. In *Prof. of NSDI*, 2016.

[97] Clayton Shepard, Abeer Javed, and Lin Zhong. Control channel design for many-antenna MU-MIMO. In *Prof. of ACM MobiCom*, 2015.

[98] Ezzeldin Hamed, Hariharan Rahul, Mohammed A Abdelghany, and Dina Katabi. Real-time distributed MIMO systems. In *Proc. of ACM SIGCOMM*, pages 412–425, 2016.

[99] Jim Geier. Wi-Fi: Define minimum SNR values for signal coverage. *Enterprise Networking Planet: Standards & Protocols*, 2008.

[100] Sergio Verdu et al. *Multiuser detection*. Cambridge university press, 1998.

[101] Andrew J Viterbi and Andrew J Viterbi. *CDMA: principles of spread spectrum communication*, volume 122. Addison-Wesley Reading, MA, 1995.

[102] Linksys official support. https://www.linksys.com/us/support-product?pid=01t80000003ouh0AAA, 2019.

[103] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52:1289–1306, 2006.

[104] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[105] Piotr Indyk. Explicit constructions for compressed sensing of sparse signals. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 30–33. Society for Industrial and Applied Mathematics, 2008.

[106] Emmanuel Candes and Justin Romberg. l1-magic: Recovery of sparse signals via convex programming. 4:14, 2005.

[107] Allen Y Yang, Zihan Zhou, Arvind Ganesh Balasubramanian, S Shankar Sastry, and Yi Ma. Fast l1-minimization algorithms for robust face recognition. *IEEE Transactions on Image Processing*, 22:3234–3246, 2013.

[108] David L Donoho, Michael Elad, and Vladimir N Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52:6–18, 2006.

[109] Introduction to fft and ofdm. http://shodhganga.inflibnet.ac.in/bitstream/10603/42180/10/10_chapter\%202.pdf, 2019.

[110] The viterbi algorithm. http://www.cim.mcgill.ca/~latorres/Viterbi/va_alg.htm, 2019.

[111] k largest(or smallest) elements in an array. https://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/, 2012.

[112] How expensive is an operation on a cpu. https://streamhpc.com/blog/2012-07-16/how-expensive-is-an-operation-on-a-cpu/, 2012.

[113] Qualcomm. Qualcomm ipq4018 soc. https://www.qualcomm.com/products/ipq4018, 2019.

[114] ZBT apg222. https://www.alibaba.com/product-detail/newest-atheros-chipset-IPQ4018-802-11ac_60541163350.html, 2019.

[115] GL.iNET B1300 IPQ4018 dual band wifi router. https://www.alibaba.com/product-detail/GL-iNET-B1300-ipq4018-dual-band_60779146003.html, 2018.

[116] George Turin. An introduction to matched filters. *IRE transactions on Information Theory*, 6:311–329, 1960.

[117] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. Successive interference cancellation: A back-of-the-envelope perspective. In *Proc. of ACM SIGCOMM*, 2010.

[118] Daniel Halperin, Thomas Anderson, and David Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *Proc. of ACM MobiCom*, 2008.

[119] Jeffrey G Andrews. Interference cancellation for cellular systems: a contemporary overview. *IEEE Wireless Communications*, 12:19–29, 2005.

[120] Aditya Gudipati, Stephanie Pereira, and Sachin Katti. Automac: Rateless wireless concurrent medium access. In *Proc. of ACM MobiCom*, 2012.

[121] Wenjie Zhou, Tanmoy Das, Lu Chen, Kannan Srinivasan, and Prasun Sinha. Basic: backbone-assisted successive interference cancellation. In *Proc. of ACM MobiCom*, 2016.

[122] Shyamnath Gollakota, Samuel David Perli, and Dina Katabi. Interference alignment and cancellation. In *Proc. of ACM SIGCOMM*, 2009.

[123] Sachin Katti, Shyamnath Gollakota, and Dina Katabi. Embracing wireless interference: Analog network coding. *Proc. of ACM SIGCOMM*, 2007.

[124] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. Xors in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, 2006.

[125] Junmei Yao, Tao Xiong, and Wei Lou. Beyond the limit: A fast tag identification protocol for rfid systems. *Pervasive and Mobile Computing*, 21:1–18, 2015.

[126] Lu Chen, Fei Wu, Jiaqi Xu, Kannan Srinivasan, and Ness Shroff. Bipass: Enabling end-to-end full duplex. In *Proc. of ACM MobiCom*, 2017.

[127] Karthikeyan Sundaresan, Mohammad Khojastepour, Eugene Chai, and Sampath Rangarajan. Full-duplex without strings: Enabling full-duplex with half-duplex clients. In *Proc. of ACM MobiCom*, 2014.

[128] Bo Chen, Yue Qiao, Ouyang Zhang, and Kannan Srinivasan. Airexpress: Enabling seamless in-band wireless multi-hop transmission. In *Proc. of ACM MobiCom*, 2015.

[129] Jung Il Choi, Mayank Jain, Kannan Srinivasan, Phil Levis, and Sachin Katti. Achieving single channel, full duplex wireless communication. In *Proc. of ACM MobiCom*, 2010.

[130] Anwar Hithnawi, Su Li, Hossein Shafagh, James Gross, and Simon Duquennoy. Crosszig: combating cross-technology interference in low-power wireless networks. In *Proc. of IEEE IPSN*, 2016.

[131] Gonglong Chen, Wei Dong, Zhiwei Zhao, and Tao Gu. Towards accurate corruption estimation in zigbee under cross-technology interference. In *Proc. of IEEE ICDCS*, 2017.

[132] Kyle Miller, Atresh Sanne, Kannan Srinivasan, and Sriram Vishwanath. Enabling real-time interference alignment: Promises and challenges. In *Proc. of ACM MobiHoc*, 2012.

[133] Swarun Kumar, Diego Cifuentes, Shyamnath Gollakota, and Dina Katabi. Bringing cross-layer mimo to today's wireless lans. In *Proc. of ACM SIGCOMM*, 2013.

[134] Tianji Li, Mi Kyung Han, Apurv Bhartia, Lili Qiu, Eric Rozner, Yin Zhang, and Brad Zarikoff. Crma: Collision-resistant multiple access. In *Proc. of ACM MobiCom*, 2011.

[135] Manjunath Doddavenkatappa, Mun Choon Chan, Ben Leong, et al. Splash: Fast data dissemination with constructive interference in wireless sensor networks. In *Proc. of NSDI*, 2013.

[136] Yin Wang, Yunhao Liu, Yuan He, Xiang-Yang Li, and Dapeng Cheng. Disco: Improving packet delivery via deliberate synchronized constructive interference. *IEEE Transactions on Parallel & Distributed Systems*, pages 713–723, 2015.

[137] Sudipta Saha and Mun Choon Chan. Design and application of a many-to-one communication protocol. In *Proc. of IEEE INFOCOM*, 2017.

[138] Narendra Anand, Jeongkeun Lee, Sung-Ju Lee, and Edward W Knightly. Mode and user selection for multi-user mimo wlans without csi. In *Proc. of IEEE INFOCOM*, 2015.

[139] Sanjib Sur, Ioannis Pefkianakis, Xinyu Zhang, and Kyu-Han Kim. Practical mu-mimo user selection on 802.11 ac commodity networks. In *Proc. of ACM MobiCom*, 2016.

[140] Peshal Nayak, Michele Garetto, and Edward W Knightly. Multi-user downlink with single-user uplink can starve tcp. In *Proc. of IEEE INFOCOM*, 2017.

[141] Yasaman Ghasempour and Edward W Knightly. Decoupling beam steering and user selection for scaling multi-user 60 ghz wlans. In *Proc. of ACM MobiHoc*, 2017.

[142] Oscar Bejarano, Roger Pierre Fabris Hoefel, and Edward W Knightly. Resilient multi-user beamforming wlans: Mobility, interference, and imperfect csi. In *Proc. of IEEE INFOCOM*, 2016.

[143] Aditya Gudipati and Sachin Katti. Strider: Automatic rate adaptation and collision handling. In *Proc. of ACM SIGCOMM*, 2011.

[144] Jonathan Perry, Peter A Iannucci, Kermin E Fleming, Hari Balakrishnan, and Devavrat Shah. Spinal codes. In *Proc. of ACM SIGCOMM*, 2012.

[145] Zhenyu Song, Longfei Shangguan, and Kyle Jamieson. Wi-fi goes to town: Rapid picocell switching for wireless transit networks. In *Proc. of ACM SIGCOMM*, 2017.

[146] Anfu Zhou, Teng Wei, Xinyu Zhang, Min Liu, and Zhongcheng Li. Signpost: Scalable mu-mimo signaling with zero csi feedback. In *Proc. of ACM MobiHoc*, 2015.

[147] Clayton Shepard, Hang Yu, Narendra Anand, Erran Li, Thomas Marzetta, Richard Yang, and Lin Zhong. Argos: Practical many-antenna base stations. In *Proc. of ACM MobiCom*, 2012.

[148] Qi Zhao, Zihui Ge, Jia Wang, and Jun Xu. Robust traffic matrix estimation with imperfect information: Making use of multiple data sources. In *Proc. of ACM SIGMETRICS*, 2006.

[149] Abishek Gopalan and Srinivasan Ramasubramanian. On identifying additive link metrics using linearly independent cycles and paths. *IEEE/ACM Trans. Netw.*, 20:906–916, 2012.

[150] Liang Ma, Ting He, Kin K Leung, Ananthram Swami, and Don Towsley. Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement. *IEEE/ACM Trans. Netw.*, 22:1351–1368, 2014.

[151] Lei Yang and Fengjun Li. mTor: a multipath Tor routing beyond bandwidth throttling. In *Proc. of IEEE CNS*, 2015.

[152] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2:575–601, 1992.

[153] Rocketfuel: An ISP topology mapping engine. *University of Washington*, 2002. [Online].
http://www.cs.washington.edu/research/networking/rocketfuel/.

[154] Mathew Penrose. *Random Geometric Graphs*. Oxford Univ. Press, 2003.

[155] Martin Haenggi, Jeffrey G Andrews, Francois Baccelli, Olivier Dousse, and Massimo Franceschetti. Stochastic geometry and random graphs for the analysis and design of wireless networks. *IEEE J. Sel. Areas Commun.*, 27:1029–1046, 2009.

[156] Dzevdan Kapetanovic, Gan Zheng, and Fredrik Rusek. Physical layer security for massive MIMO: An overview on passive eavesdropping and active attacks. *IEEE Commun. Mag.*, 53:21–27, 2015.

[157] Jaydip Sen, Sripad Koilakonda, and Arijit Ukil. A mechanism for detection of cooperative black hole attack in mobile Ad Hoc networks. In *Proc. of IEEE ISMS*, 2011.

[158] Tom Chothia, Yusuke Kawamoto, Chris Novakovic, and David Parker. Probabilistic point-to-point information leakage. In *Proc. of IEEE CSF*, 2013.

[159] Lu Yu, Juan Deng, Richard R Brooks, and Seok Bae Yun. Automobile ECU design to avoid data tampering. In *Proc. of ACM CISR*, 2015.

[160] Bobby Sharma. A distributed cooperative approach to detect gray hole attack in MANETs. In *Proc. of ACM WCI*, 2015.

[161] Yanli Yu, Keqiu Li, Wanlei Zhou, and Ping Li. Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures. *J. Netw. and Comput. Appl.*, 35:867–880, 2012.

[162] Long Cheng, Dinil Mon Divakaran, Aloysius Wooi Kiak Ang, Wee Yong Lim, and Vrizlynn LL Thing. FACT: A framework for authentication in cloud-based IP traceback. *IEEE Trans. Inf. Forensics Security*, 2016.

[163] Guang Yao, Jun Bi, and Athanasios V Vasilakos. Passive IP traceback: Disclosing the locations of IP spoofers from path backscatter. *IEEE Trans. Inf. Forensics Security*, 10:471–484, 2015.

[164] Abraham Yaar, Adrian Perrig, and Dawn Song. Pi: A path identification mechanism to defend against DDoS attacks. In *Proc. of IEEE S&P*, 2003.

[165] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Protection from distributed denial of service attacks using history-based IP filtering. In *Proc. of IEEE ICC*, 2003.

[166] Robert Beverly, Steven Bauer, and Arthur Berger. The internet is not a big truck: toward quantifying network neutrality. In *Proc. of Springer PAM*, 2007.

[167] Mukarram Bin Tariq, Murtaza Motiwala, and Nick Feamster. Nano: Network access neutrality observatory. Technical report, 2008.

[168] Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, and Mostafa Ammar. Detecting network neutrality violations with causal inference. In *Proc. of ACM CoNEXT*, 2009.

[169] Udi Weinsberg, Augustin Soule, and Laurent Massoulie. Inferring traffic shaping and policy parameters using end host measurements. In *Proc. of IEEE INFOCOM*, 2011.

[170] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proc. of ACM SIGCOMM*, 2001.

[171] Roshan Thomas, Brian Mark, Tommy Johnson, and James Croall. Netbouncer: client-legitimacy-based high-performance DDoS filtering. In *Proc. of IEEE DISCEX*, 2003.

[172] John R Hughes, Tuomas Aura, and Matt Bishop. Using conservation of flow as a security mechanism in network protocols. In *Proc. of IEEE S&P*, 2000.

[173] Thomer M Gil and Massimiliano Poletto. MULTOPS: A data-structure for bandwidth attack detection. In *Proc. of USENIX Security*, 2001.

[174] Alper T Mizrak, Stefan Savage, and Keith Marzullo. Detecting compromised routers via packet forwarding behavior. *IEEE Netw.*, 22:34–39, 2008.

[175] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 2006.

[176] Paul Barford and David Plonka. Characteristics of network traffic flow anomalies. In *ACM SIGCOMM*, 2001.

[177] Xinyuan Wang, Shiping Chen, and Sushil Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *IEEE S&P*, 2007.

[178] Amir Houmansadr, Negar Kiyavash, and Nikita Borisov. RAINBOW: A robust and invisible non-blind watermark for network flows. In *NDSS*, 2009.

[179] Charles E Perkins and Elizabeth M Royer. Ad-hoc on-demand distance vector routing. In *IEEE WMCSA*, 1999.

[180] Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM TISSEC*, 1, 1998.

[181] Dulanjalie C Dhanapala, Anura P Jayasumana, and Qi Han. Performance of random routing on grid-based sensor networks. In *IEEE CCNC*, 2009.

[182] Brad Karp and Hsiang-Tsung Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *ACM MobiCom*, 2000.

[183] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory*, 52, 2006.

[184] C-K Chau and Prithwish Basu. Exact analysis of latency of stateless opportunistic forwarding. In *IEEE INFOCOM*, 2009.

[185] László Lovász et al. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2, 1993.

[186] Fan Chung and S-T Yau. Discrete green's functions. *Elsevier Journal of Combinatorial Theory, Series A*, 91, 2000.

[187] Yanhua Li and Zhi-Li Zhang. Random walks on digraphs: A theoretical framework for estimating transmission costs in wireless routing. In *IEEE INFOCOM*, 2010.

[188] Issam Mabrouki, Xavier Lagrange, and Gwillerm Froc. Random walk based routing protocol for wireless sensor networks. In *ICST ValueTools*, 2007.

[189] A El Gamal, James Mammen, Balaji Prabhakar, and Devavrat Shah. Throughput-delay trade-off in wireless networks. In *IEEE INFOCOM 2004*, 2004.

[190] Matthias Grossglauser and David NC Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10, 2002.

[191] Michael J Neely and Eytan Modiano. Capacity and delay tradeoffs for ad hoc mobile networks. *IEEE Trans. Inf. Theory*, 51, 2005.

[192] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency—choose two. In *IEEE S&P*, 2018.

[193] Nethanel Gelernter and Amir Herzberg. On the limits of provable anonymity. In *ACM WPES*, 2013.

[194] Parv Venkitasubramaniam and Lang Tong. A game-theoretic approach to anonymous networking. *IEEE/ACM Trans. Netw.*, 20, 2012.

[195] Parvathinathan Venkitasubramaniam, Ting He, and Lang Tong. Anonymous networking amidst eavesdroppers. *IEEE Trans. Inf. Theory*, 54, 2008.

[196] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *ACM SIGMETRICS*, 2003.

[197] Kun Xie, Can Peng, Xin Wang, Gaogang Xie, Jigang Wen, Jiannong Cao, Dafang Zhang, and Zheng Qin. Accurate recovery of internet traffic data under variable rate measurements. *IEEE/ACM Trans. Netw.*, 26, 2018.

[198] Anirvan M Sengupta and Partha P Mitra. Distributions of singular values for some random matrices. *Physical Review E*, 60, 1999.

[199] I Richard Savage. Probability inequalities of the tchebycheff type. *Journal of Research of the National Bureau of Standards-B. Mathematics and Mathematical Physics B*, 65:211–222, 1961.

Appendix A: Supplementary Evidences of Chapter 3

In this appendix, we provide the detailed evidences and proofs of Chapter 3.

A.1 Firmware Analysis

We analyze popular WiFi drivers and AP firmware to understand these practical constraints: (i) Linux kernel drivers: ath9k (Qualcomm/Atheros 802.11n chipsets), brcmsmac (Broadcom 802.11n chipsets), and iwlwifi (Intel 802.11n/ac chipsets); (ii) 802.11ac AP firmware in Linksys EA8500, EA9500, TP-Link AC1200, C5400, and AD7200.

In particular, ath9k allows the maximum length of a data payload to be either 8,192 or 65,535 bytes (aPSDUMaxLength for 802.11n is 65,535) based on its version number (as shown in Listing A.1); brcmsmac uses the maximum duration of 5,000 μs (aPPDUMaxTime for 802.11n is 10,000 μs) (as shown in Listing A.2); iwlwifi allows the maximum duration to be 4,000 μs (aPPDUMaxTime for 802.11ac is 5,484), as the excerpted code shown in Listing A.3. In addition, Linksys EA8500, EA9500, and TP-Link AC1200, C5400 and AD7200 share the same code: the maximum length of a data payload is 65,535 bytes (aPSDUMaxLength for 802.11ac is 4,692,480)(as shown in Listing A.4).

Listing A.1. Source code in Qualcomm/Atheros ath9k (802.11n)

```
/* hw.h */
...
#define ATH_AMPDU_LIMIT_MAX     (64 * 1024 - 1)
...
/* hw.c */
    ...
    if (AR_SREV_9160_10_OR_LATER(ah) || AR_SREV_9100(ah))
       pCap->rts_aggr_limit = ATH_AMPDU_LIMIT_MAX;
    else
       pCap->rts_aggr_limit = (8 * 1024);
    ...
```

Listing A.2. Source code in Broadcom brcmsmac (802.11n)

```
/* ampdu.c */
...
/* max dur of tx ampdu (in msec) */
#define AMPDU_MAX_DUR        5
...
    ampdu->dur = AMPDU_MAX_DUR;
...
```

Listing A.3. Source code of Intel iwlwifi (802.11ac)

```
/* mvm/constants.h */
...
#define IWL_MVM_RS_AGG_TIME_LIMIT       4000
...
/* mvm/rs.c */
...
    lq_cmd->agg_time_limit =
        cpu_to_le16(IWL_MVM_RS_AGG_TIME_LIMIT);
...
```

Listing A.4. The same source code in Linksys EA8500/EA9500 and TP-Link AC1200/C5400/AD7200.

```
/* include/linux/ieee80211.h */
...
/*
Maximum length of AMPDU that the STA can receive.
Length = 2 ^ (13 + max_ampdu_length_exp)-1 (octets)
*/
enum ieee80211_max_ampdu_length_exp {
    ...
    IEEE80211_HT_MAX_AMPDU_64K = 3
};
...
```

A.2 Statistical Results of Packet Traces

Our comprehensive data collections capture WiFi packet traces under a diversity of traffic load conditions over long time periods. For each transmitter in the packet traces, we compute the NAV distribution in its RTS packets. We find that the NAV distribution is highly patterned or uneven in its value space. Figure A.1 shows the NAV distributions of top five devices that send the largest numbers of RTS packets in different measurement environments. We observe from Figure A.1 that each device's NAV values almost concentrate in the small value regions. For example, in the lab scenario, 92.1% NAV values in RTS packets from device 3 is 156.

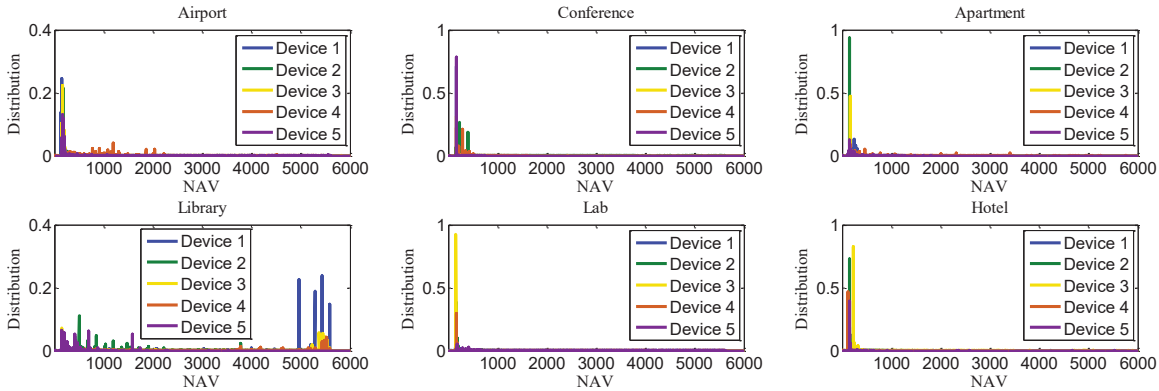


Figure A.1. NAV distributions at different locations.

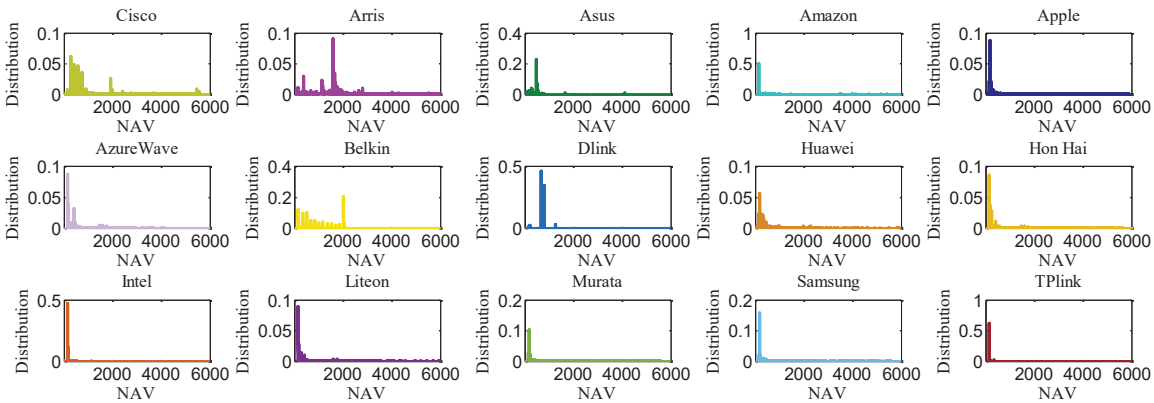


Figure A.2. NAV distributions of different vendors.

We also notice that, interestingly, WiFi devices from the same manufacturer (identified by their MAC addresses) do exhibit similar NAV distribution in their RTS packets. Figure A.2 illustrates the distributions of all NAV values in RTS packets transmitted by devices from 15 common manufacturers. It is seen from Figure A.2 that the distributions exhibit different patterns by manufacturers, mainly due to their distinct firmware designs. Similar to Figure A.1, Figure A.2 shows the NAV distributions are highly uneven and patterned with a small number of NAV values much more likely to show up in RTS packets than the others.

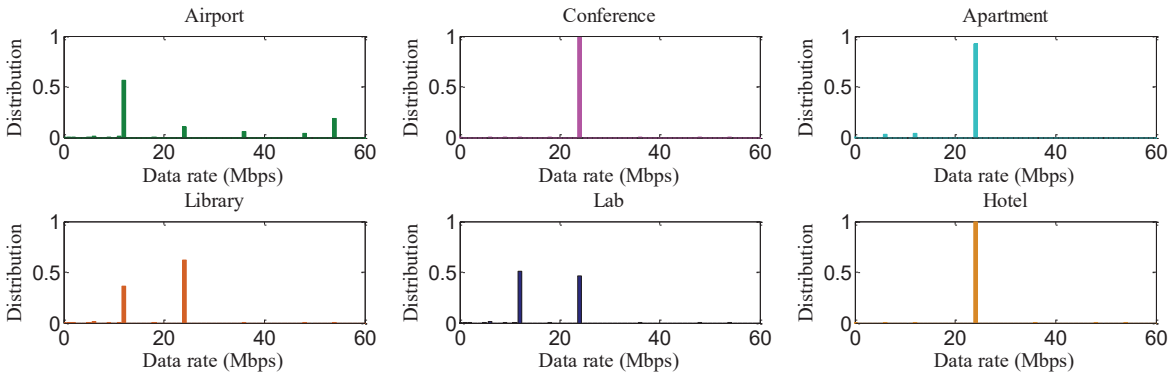


Figure A.3. RTS data rate distributions.

In addition to NAV, we also measure the data rates of RTS packets. Figure A.3 depicts the distribution of RTS dta rates in different environments. We can see that a large number of devices adopt 12 Mbps and 24 Mbps data rates to send RTS. Furthermore, in the conference and hotel scenarios, most devices even only use the 24 Mbps data rate.

Based on the packet trace analysis, if we select the NAV values that are most likely in RTS packet from each device to construct the comb matrix $\mathbf{M}$, we should be able to decrease the size of $\mathbf{M}$ at the cost of a small performance penalty.

A.3 Performance Analysis of γ-Decimation

In this part, we provide the performance analysis of γ-decimation. For the tooth vector set $\mathcal{M} = \{\mathbf{m}_i\}_{i\in[1,M]}$, where $M = |\mathcal{M}|$ and $\mathbf{m}_i \in \mathcal{C}^{L\times 1}$, γ-decimation selects M/γ tooth vectors with largest correlation values to form a new comb matrix. Denote by $\mathbb{M}'$ the set consisting of these selected M/γ tooth vectors by γ-decimation. Without loss of generality, we assume the collided signal $\mathbf{y}$ contains the first S tooth vectors, i.e., $\mathbf{m}_1, \cdots, \mathbf{m}_S$. In this section, notations are summarized as follows: (i) $o(1)$ denotes a function that converges to 0 as $L \to \infty$; (ii) $\mathbb{E}(\cdot)$ and $\mathsf{Var}(\cdot)$ denote the expectation and variance operators, respectively;

(iii) for a complex number m, m^* is the complex conjugate of m, and $|m|$ is the magnitude of m. Now we state the following theorem to show the performance of γ-decimation:

Theorem 8 *Define event A as the event that $\mathbf{m}_1$, $\mathbf{m}_2$, $\cdots$, $\mathbf{m}_s$ are all selected by γ-decimation in $\mathbb{M}'$. Then, it holds that $\mathbb{P}(A) = 1 - o(1)$ (i.e., event A happens with high probability).*

Proof: We first normalize the correlation between comb matrix $\mathbf{M}$ and the received vector $\mathbf{y}$. From (3.2), we have

$$\begin{aligned}\mathbf{z} &= \frac{1}{L}\mathbf{M}^H\mathbf{y} \\ &= \frac{1}{L}\begin{bmatrix} \mathbf{m}_1^H\mathbf{m}_1 & \mathbf{m}_1^H\mathbf{m}_2 & \cdots & \mathbf{m}_1^H\mathbf{m}_M \\ \mathbf{m}_2^H\mathbf{m}_1 & \mathbf{m}_2^H\mathbf{m}_2 & \cdots & \mathbf{m}_2^H\mathbf{m}_M \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{m}_M^H\mathbf{m}_1 & \mathbf{m}_{M-1}^H\mathbf{m}_2 & \cdots & \mathbf{m}_M^H\mathbf{m}_M \end{bmatrix}\begin{bmatrix} g1 \\ g2 \\ \vdots \\ g_M \end{bmatrix}.\end{aligned} \tag{A.1}$$

Let $\mathbf{z} = [z_1, z_2, \cdots, z_M]^H$. It holds that $\forall z_i \in \mathbf{z}$,

$$z_i = \frac{1}{L}\sum_{s=1}^{M} g_s \mathbf{m}_i^H \mathbf{m}_s = \frac{1}{L}\sum_{s=1}^{M}\sum_{k=1}^{L} g_s m_{i,k}^* m_{s,k}. \tag{A.2}$$

Because each tooth vector $\mathbf{m}_i$ has the random property by coding, for any entry $m_{j,i} \in \mathbf{m}_i$, we have $\mathbb{E}(m_{j,i}) = 0$ and let $\mathbb{E}(|m_{j,i}|^2) = \sigma^2$.

Since tooth vectors $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_S$ are the ones to be resolved, we know the first S members in $\mathbf{g}$ are not zeros. Define Y as

$$Y = \sum_{m=S+1}^{M} \mathbf{1}_{\{|z_m| > h\}}, \tag{A.3}$$

denoting the number of false alarms (i.e., noise exceeding the threshold), where h is a threshold, and $\mathbf{1}_{\{|z_m|>h\}}$ is the indicator function defined as

$$\mathbf{1}_{\{|z_m|>h\}} = \begin{cases} 1, & \text{if } |z_m| > h \\ 0, & \text{otherwise.} \end{cases} \tag{A.4}$$

To evaluate the performance of γ-decimation, we define another event

$$B = \left(\bigcap_{s=1}^{S} \{|z_s| > h\} \right) \bigcap \{Y \leq (\gamma M - S)\},$$

where the first part denotes that the correlation values $z_1, z_2, \cdots, z_S$ are all above the given threshold h and the second part indicates that there are at most $(\gamma M - S)$ other correlation values above h. If event B happens, $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_S$ will be selected by γ-decimation and thus event A must happen. This means $\mathbb{P}(A|B) = 1$ and $\mathbb{P}(A) \geq \mathbb{P}(B)$. Therefore, according to Fréchet inequalities, we obtain

$$\begin{aligned} \mathbb{P}(A) &\geq \mathbb{P}\left(\left(\bigcap_{s=1}^{S} \{|z_s| > h\} \right) \bigcap \{Y \leq (\gamma M - S)\} \right) \\ &\geq \mathbb{P}\left(\bigcap_{s=1}^{S} \{|z_s| > h\} \right) - \mathbb{P}\left(Y > (\gamma M - S)\right). \end{aligned} \tag{A.5}$$

From (A.5), we need two steps to finish the proof:

1. Prove $\mathbb{P}\left(\bigcap_{s=1}^{S} \{|z_s| > h\} \right) = 1 - o(1)$.
2. Prove $\mathbb{P}\left(Y > (\gamma M - S)\right) = o(1)$.

For step 1, by Fréchet inequalities, we have that

$$\mathbb{P}\left(\bigcap_{s=1}^{S}\{|z_s| > h\}\right) \geq \sum_{s=1}^{S}\mathbb{P}(|z_s| > h) - (S-1). \tag{A.6}$$

According to Cantelli's inequality [199], for $1 \leq s \leq S$, we have the probability

$$\mathbb{P}(|z_s| > h) \geq 1 - \frac{\mathsf{Var}(|z_s|)}{\mathsf{Var}(|z_s|) + (h - \mathbb{E}(|z_s|))^2}. \tag{A.7}$$

Next we derive $\mathbb{E}(|z_s|)$ and $\mathsf{Var}(|z_s^2|)$ respectively. Because $1 \leq s \leq S$, without loss of generality, we consider the first element z_1. From (A.2), by leveraging Lemma 10, we have

$$\mathbb{E}(|z_1|) = \mathbb{E}(\frac{1}{L}g_1\mathbf{m}_1^H\mathbf{m}_1 + \frac{1}{L}\sum_{s=2}^{S}g_s\mathbf{m}_1^H\mathbf{m}_s) = g_1\sigma^2, \tag{A.8}$$

and

$$\begin{aligned}
\mathbb{E}(|z_1|^2) &= \mathbb{E}\left(\frac{1}{L^2}\left(\mathbf{m}_1^H\sum_{s=1}^{S}g_s\mathbf{m}_s\right)^2\right) \\
&= \frac{1}{L^2}\mathbb{E}\left(\sum_{s=1}^{S}\sum_{k=1}^{L}g_s m_{1,k}^* m_{s,k}\sum_{s'=1}^{S}\sum_{k'=1}^{L}g_{s'}m_{1,k'}^* m_{s',k'}\right) \\
&= \frac{1}{L^2}\sum_{s=1}^{S}\sum_{k=1}^{L}\sum_{s'=1}^{S}\left(\sum_{k'=1,k'\neq k}^{L}g_s g_{s'}\mathbb{E}(m_{1,k}^* m_{s,k})\right. \\
&\quad \left.\times\mathbb{E}(m_{1,k'}^* m_{s',k'}) + g_s g_{s'}\mathbb{E}(m_{1,k}^* m_{s,k} m_{1,k}^* m_{s',k})\right). \\
&= g_1^2\sigma^4 + \frac{1}{L}\Xi_1,
\end{aligned} \tag{A.9}$$

where

$$\Xi_1 = \left(g_1^2(\mathbb{E}(|m_{1,k}|^4) - \sigma^4) + \mathbb{E}((m_{1,k}^*)^2)\mathbb{E}(m_{1,k}^2)\sum_{s=2}^{S}g_s^2\right).$$

From (A.8) and (A.9), we obtain the variance of z_1 as

$$\mathsf{Var}(|z_1|) = \mathbb{E}(|z_1|^2) - (\mathbb{E}(|z_1|))^2 = \frac{1}{L}\Xi_1. \tag{A.10}$$

Replacing (A.10) into (A.7), we have

$$\mathbb{P}(|z_1| > h) \geq 1 - \frac{\Xi_1}{\Xi_1 + L(h - g_s\sigma^2)^2}. \tag{A.11}$$

Then, (A.6) can be rewritten as

$$\begin{aligned} \mathbb{P}\left(\bigcap_{s=1}^{S}\{|z_s| > h\}\right) &\geq 1 - \sum_{s=1}^{S}\frac{\Xi_s}{\Xi_s + L(h - g_s\sigma^2)^2} \\ &\geq 1 - S\frac{\Xi_{\max}}{\Xi_{\max} + L(h - g_{\max}\sigma^2)^2}, \end{aligned} \tag{A.12}$$

where $\Xi_{\max} = \max\{\Xi_s\}_{s\in[1,S]}$, and $g_{\max} = \max\{g_s\}_{s\in[1,S]}$. When $L \to \infty$, the probability converges to 1.

For step 2, letting $y = \gamma M - S$, by Markov's inequality, we have

$$\mathbb{P}(Y > y) \leq \frac{1}{y}\mathbb{E}(Y), \tag{A.13}$$

then from (A.3), we have

$$\mathbb{E}(Y) = \mathbb{E}\left(\sum_{m=S+1}^{M}\mathbf{1}_{\{|z_m|>h\}}\right) = \sum_{m=S+1}^{M}\mathbb{P}(|z_m| > h). \tag{A.14}$$

According to Chebyshev's inequality, we can obtain

$$\mathbb{P}(||z_m| - \mathbb{E}(|z_m|)| > h) \leq \frac{\mathsf{Var}(|z_m|)}{h^2}. \tag{A.15}$$

Next, we derive $\mathbb{E}(|z_m|)$ and $\mathsf{Var}(|z_m|)$. Without loss of generality, we consider the last element z_M. Similarly, we have

$$\mathbb{E}(|z_M|) = \frac{1}{L}\mathbb{E}\left(\sum_{s=1}^{S}\sum_{k=1}^{L} g_s m_{M,k}^* m_{s,k}\right) = 0, \tag{A.16}$$

and

$$\begin{aligned}\mathbb{E}(|z_M|^2) &= \frac{1}{L^2}\mathbb{E}\left(\sum_{s=1}^{S}\sum_{k=1}^{L} g_s m_{M,k}^* m_{s,k}\right)^2 \\ &= \frac{1}{L^2}\sum_{s=1}^{S}\sum_{k=1}^{L}\sum_{s'=1}^{S} g_s g_{s'}\mathbb{E}(m_{M,k}^* m_{s,k} m_{M,k}^* m_{s',k}) \\ &= \frac{1}{L}\sum_{s=1}^{S} g_s^2\mathbb{E}((m_{M,k}^*)^2)\mathbb{E}(m_{s,k}^2).\end{aligned} \tag{A.17}$$

Let $\Psi_M = \sum_{s=1}^{S} g_s^2\mathbb{E}((m_{M,k}^*)^2)\mathbb{E}(m_{s,k}^2)$, then we can obtain

$$\mathsf{Var}(|z_M|) = \mathbb{E}(|z_M|^2) - (\mathbb{E}(|z_M|))^2 = \frac{1}{L}\Psi_M. \tag{A.18}$$

Replacing (A.16) and (A.18) into (A.15), we have

$$\mathbb{P}(|z_M| > h) \leq \frac{\Psi_M}{Lh^2}. \tag{A.19}$$

Thus (A.14) can be rewritten as

$$\mathbb{E}(Y) \leq \sum_{m=S+1}^{M} \frac{\Psi_M}{Lh^2} \leq (M-S-1)\frac{\Psi_{\max}}{Lh^2}, \tag{A.20}$$

where $\Psi_{\max} = \max\{\Psi_m\}_{m\in[S+1,M]}$. Finally,

$$\mathbb{P}(Y > y) \leq \frac{1}{y}\mathbb{E}(Y) \leq \frac{(M-S-1)\Psi_{\max}}{Lyh^2}. \tag{A.21}$$

When $L \to \infty$, $\mathbb{P}(Y > y)$ converges to 0, which completes the proof. □

Lemma 10 *Given tooth vectors* $\mathbf{m}_i$ *and* $\mathbf{m}_j$*, for all* $m_{k,i} \in \mathbf{m}_i$ *and* $m_{s,j} \in \mathbf{m}_j$ *satisfying* $\mathbb{E}(m_{k,i}) = \mathbb{E}(m_{s,j}) = 0$ *and* $\mathbb{E}(|m_{k,i}|^2) = \mathbb{E}(|m_{s,j}|^2) = \sigma^2$*, the following two statements are true: (i) if* $i \neq j$*,* $\mathbb{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j) = 0$ *and* $\mathbb{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j|^2) = \frac{1}{L}\sigma^4$*; (ii) if* $i = j$*,* $\mathbb{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i) = \mathbb{E}(\frac{1}{L}\mathbf{m}_j^H\mathbf{m}_j) = \sigma^2$ *and* $\mathbb{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i|^2) = \frac{1}{L}(L-1)\sigma^4 + \frac{1}{L}3\sigma^4$.

Proof: Let $z = \frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j = \frac{1}{L}\sum_{k=1}^{L} m_{i,k}^* m_{j,k}$.

For statement (i), since $\mathbb{E}(m_{i,k}) = \mathbb{E}(m_{j,k}) = 0$, we have

$$\mathbb{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j) = 0.$$

Furthermore, as we know $\mathbb{E}(|m_{i,k}|^2) = \mathbb{E}(|m_{s,k}|^2) = \sigma^2$, we can obtain

$$\begin{aligned}\mathbb{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j|^2) &= \frac{1}{L^2}\mathbb{E}\left(\sum_{k=1}^{L} m_{i,k}^* m_{j,k} \sum_{q=1}^{L} m_{i,q}^* m_{j,q}\right) \\ &= \frac{1}{L}\sigma^4\end{aligned} \tag{A.22}$$

For statement (ii), it is easy to know that

$$\mathbb{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_j) = \mathbb{E}(\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i) = \sigma^2$$

and

$$\begin{aligned}\mathbb{E}(|\frac{1}{L}\mathbf{m}_i^H\mathbf{m}_i|^2) &= \mathbb{E}\left(\frac{1}{L}\sum_{k=1}^{L} m_{i,k}^* m_{i,k}\right)^2 \\ &= \frac{1}{L^2}L(L-1)\sigma^4 + \frac{1}{L^2}\sum_{k=1}^{L}\mathbb{E}(|m_{i,k}|^4) \\ &= \frac{1}{L}(L-1)\sigma^4 + \frac{1}{L}3\sigma^4.\end{aligned} \tag{A.23}$$

Therefore, we complete the proof. □

Appendix B: Copyright Permissions

Below is permission for the reuse of materials in Chapter 2.

CCC Marketplace™

This is a License Agreement between Shangqing Zhao ("You") and ACM (Association for Computing Machinery) ("Publisher") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by ACM (Association for Computing Machinery), and the CCC terms and conditions.

All payments must be made in full to CCC.

Order Date	10-Jun-2021	Type of Use	Republish in a thesis/dissertation
Order License ID	1125172-1		
ISBN-13	978-1-4503-4916-1	Publisher	ACM, Inc.
		Portion	Chapter/article

LICENSED CONTENT

Publication Title	Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking	Country	United States of America
		Rightsholder	ACM (Association for Computing Machinery)
Date	01/01/2017	Publication Type	Conference Proceeding
Language	English		

REQUEST DETAILS

Portion Type	Chapter/article	Rights Requested	Main product
Page range(s)	2-4, 15-49	Distribution	Worldwide
Total number of pages	37	Translation	Original language of publication
Format (select all that apply)	Electronic	Copies for the disabled?	No
Who will republish the content?	Academic institution	Minor editing privileges?	No
Duration of Use	Life of current edition	Incidental promotional use?	No
Lifetime Unit Quantity	Up to 499	Currency	USD

NEW WORK DETAILS

Title	Stateful Inter-Packet Signal Processing for Wireless Networking	Institution name	University of South Florida
		Expected presentation date	2021-06-18
Instructor name	Shangqing Zhao		

ADDITIONAL DETAILS

Order reference number	N/A	The requesting person / organization to appear on the license	Shangqing Zhao

REUSE CONTENT DETAILS

Title, description or numeric reference of the portion(s)	Stateful Inter-Packet Signal Processing for Wireless Networking	Title of the article/chapter the portion is from	Data-Oriented Approaches towards Mobile, Network and Secure Systems
Editor of portion(s)	N/A	Author of portion(s)	Shangqing Zhao, Zhengping Luo, Zhuo Lu, Xiang Lu, Yao Liu
Volume of serial or monograph	N/A	Issue, if republishing an article from a serial	N/A
Page or page range of portion	2-4, 15-49	Publication date of portion	2017-10-04

Below is permission for the reuse of materials in Chapter 3.

Comb Decoding towards Collision-Free WiFi

Authors:
Shangqing Zhao, Zhe Qu, Zhengping Luo, Zhuo Lu, and Yao Liu, *University of South Florida*

Abstract:
Packet collisions happen every day in WiFi networks. RTS/CTS is a widely-used approach to reduce the cost of collisions of long data packets as well as combat the hidden terminal problem. In this paper, we present a new design called comb decoding (CombDec) to efficiently resolve RTS collisions without changing the 802.11 standard. We observe that an RTS payload, when treated as a vector in a vector space, exhibits a comb-like distribution; i.e., a limited number of vectors are much more likely to be used than the others due to RTS payload construction and firmware design. This enables us to reformulate RTS collision resolution as a sparse recovery problem. We create algorithms that carefully construct the search range for sparse recovery, making the complexity feasible for system design and implementation. Experimental results show that CombDec boosts the WiFi throughput by 33.6%–46.2% in various evaluation scenarios.

NSDI '20 Open Access Sponsored by NetApp

Below are permissions for the reuse of materials in Chapter 4.

Measurement Integrity Attacks against Network Tomography: Feasibility and Defense

Author: Shangqing Zhao
Publication: IEEE Transactions on Dependable and Secure Computing
Publisher: IEEE
Date: Dec 31, 1969

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

When Seeing Isn't Believing: On Feasibility and Detectability of Scapegoating in Network Tomography

Conference Proceedings:
2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)

Author: Shangqing Zhao

Publisher: IEEE

Date: June 2017

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Below is permission for the reuse of materials in Chapter 5.

How Can Randomized Routing Protocols Hide Flow Information in Wireless Networks?

Author: Shangqing Zhao
Publication: IEEE Transactions on Wireless Communications
Publisher: IEEE
Date: Nov. 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW