Data analytics methods for attack detection and localization in wireless networks

Yi Ling

DATA ANALYTICS METHODS FOR ATTACK DETECTION AND LOCALIZATION IN WIRELESS NETWORKS

by

YI LING

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2017

Approved by

ABSTRACT

Wireless ad hoc network operates without any fixed infrastructure and centralized administration. It is a group of wirelessly connected nodes having the capability to work as host and router. Due to its features of open communication medium, dynamic changing topology, and cooperative algorithm, security is the primary concern when designing wireless networks. Compared to the traditional wired network, a clean division of layers may be sacrificed for performance in wireless ad hoc networks. As a result, they are vulnerable to various types of attacks at different layers of the protocol stack. In this paper, I present real-time series data analysis solutions to detect various attacks including in- band wormholes attack in the network layer, various MAC layer misbehaviors, and jamming attack in the physical layer. And, I also investigate the problem of node localization in wireless and sensor networks, where a total of n anchor nodes are used to determine the locations of other nodes based on the received signal strengths. A range-based machine learning algorithm is developed to tackle the challenges.

ACKNOWLEDGMENTS

First I would express my greatest gratitude to my advisor Dr. Maggie Cheng for her meticulous guidance, patient advice and continuous encouragement throughout my entire PhD career. Her expertise, sincere and valuable guidance and encouragement always enlightens my path to success in PhD studies. Without her help, I could never reach the accomplishment I have so far.

I am also grateful to all the professors in Missouri University of Science and Technology, especially my committee members, Dr. Dan Lin, Dr. Fikret Ercal, Dr. Simone Silvestri and Dr. Mariesa L.Crow. It is their help and support that led me into the field of Computer Science.

My sincere thankfulness also goes to my colleague in Dr. Maggie Cheng's research team, Junwei Su. He is my best friend in my PhD life, my comrade in arms who accompanied me in the last 4 years of study.

Last but not least, a deep sense of gratitude goes to my family for their endless support for my endeavors on the road to pursue my dreams.

TABLE OF CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS iv

LIST OF ILLUSTRATIONS viii

LIST OF TABLES x

SECTION

1. INTRODUCTION 1
   - 1.1. PROBLEM STATEMENT 1
   - 1.2. DOS ATTACKS AT DIFFERENT LAYERS 3
2. IN-BAND WORMHOLE DETECTION 8
   - 2.1. BACKGROUND 8
   - 2.2. RELATED WORK 11
   - 2.3. WORMHOLE DETECTION SCHEME 13
   - 2.4. THE SEQUENTIAL CHANGE POINT DETECTION ALGORITHM 14
   - 2.5. SIMULATION 18
     - 2.5.1. Wormhole Detection in a Stationary Network 19
     - 2.5.2. Wormhole Detection in a Mobile Network 21
     - 2.5.3. SW-CLT and NP-CUSUM Detection Results 22

3. IEEE 802.11 MAC MISBEHAVIOR DETECTION 25
3.1. BACKGROUND 25
3.2. RELATED WORK 27
3.3. TIME SERIES ANALYSIS METHODS 29
3.3.1. Problem Statement 31
3.3.2. SPRT vs. Change Point Detection on Time Series 31
3.4. CHANGE POINT DETECTION ALGORITHMS 35
3.5. MISBEHAVIOR DETECTION IN IEEE 802.11 MAC 36
3.5.1. Type I Attack: Shorter DIFS 38
3.5.2. Type II Attack: Shorter DIFS and Smaller Back-off Value 38
3.5.3. Type III Attack: RTS Dropping 39
3.6. SIMULATION 40
3.6.1. Case 1: Shorter DIFS Attack 40
3.6.2. Case 3: RTS-dropping Attack 43
3.6.3. Detection Results Summary 44

4. JAMMING DETECTION 52
4.1. BACKGROUND 52
4.2. RELATED WORK 54
4.3. STATISTICAL JAMMING ATTACK DETECTION 56
4.3.1. Bayesian Estimation 56
4.3.2. Simple Threshold-Based Detector 56
4.4. TIME SERIES ANALYSIS APPROACH 57
4.4.1. Characterization of Jamming Attacks 57
4.4.2. Change Point Detection on Time Series 58

4.5. JAMMING ATTACK SIMULATION 61

4.5.1. Simulation Setup 61

4.5.2. Impact of Jamming Signal Duration 61

4.5.3. Impact of Traffic Dynamics 63

5. LOCALIZATION 69

5.1. BACKGROUND 69

5.2. RELATED WORK 71

5.3. LOCALIZATION PROBLEM 73

5.4. ALGORITHM I: LOCALIZATION WITH KNOWN TARGET TRANSMISSION POWER 74

5.5. ALGORITHM II: LOCALIZATION WITH UNKNOWN TARGET TRANSMISSION POWER 80

5.6. EXPERIMENTAL RESULTS FOR ALGORITHM I: WITH KNOWN TARGET TRANSMISSION POWER 83

5.6.1. Simulation Set-up 83

5.6.2. Impact of the Regularity of Anchors 84

5.6.3. Impact of the Sparsity of Anchors 84

5.6.4. Impact of Signal Noise 84

5.7. EXPERIMENTAL RESULTS FOR ALGORITHM II: WITH UNKNOWN TARGET TRANSMISSION POWER 85

6. CONCLUSIONS 90

BIBLIOGRAPHY 91

VITA 98

LIST OF ILLUSTRATIONS

Figure Page

2.1 Out-band wormhole. 9

2.2 In-band wormhole. 10

2.3 A 40-node network with in-band wormhole between nodes 1—2. 19

2.4 The end-to-end delay of two flows when they route through the wormhole tunnel. 21

2.5 The end-to-end delay of three flows that go through the wormhole tunnel while the background traffic steers on the original routes. 23

2.6 The end-to-end delay of two flows when they route through the wormhole tunnel. 24

3.1 SPRT algorithm terminates prematurely accepting H_0 when the misbehavior appears after the stopping point. 34

3.2 The 802.11 network under selfish attack and Node 2 (red) is the attacker in all cases. 41

3.3 The delay and throughput as well as their mean and variance for one flow (14-1). 46

3.4 The delay and throughput of all other flows show similar performance degradation. 47

3.5 The delay and throughput of four flows in the combined shorter DIFS and smaller back-off value attack. 48

3.6 The delay and throughput of flow 6-1 in the combined shorter DIFS and smaller back-off value attack. 49

3.7 The delay and inter-packet interval of a victim flow (20-2) under RTS dropping attack. 50

3.8 The transmitter data rate and inter-packet interval of a victim flow (20-2) under RTS dropping attack. 50

3.9 The cumulative means of the transmitter data rate and inter-packet interval of a victim flow (20-2) under RTS dropping attack. 51

4.1 One flow with a jammer. (node 4) 63

4.2 Top row: jamming packet duration 0.0005s, inter-packet interval 1s; Middle row: jamming packet duration 0.8s, inter-packet interval 5ms; bottom row: jamming packet duration 1.5s, inter-packet interval 5ms. 64

4.3 Jamming packet duration 0.0005s (top row), 0.8s (middle row), and 1.5s (bottom row). 65

4.4 Left column: attack case in Figure. 4.5(a); Right column: benign case in Figure. 4.5(b). 67

4.5 (a)Three flows with a jammer (node 6). (b) No jammer. 68

5.1 Location estimation error vs. noise in input. 81

5.2 With anchor nodes at grid positions, (a) estimated positions by the proposed algorithm MFL, and (b) estimated positions by the SVM algorithm. 87

5.3 With anchor nodes at grid positions, (a) estimated positions by the proposed algorithm MFL, and (b) estimated positions by the SVM algorithm. 88

5.4 (a) The impact of anchor nodes sparsity on the estimation error, (b) the impact of data noise on the estimation error. 89

LIST OF TABLES

Table Page

2.1 Comparison of SW-CLT and NP-CUSUM on change point detection. 22

3.1 Cumulative means of delay and throughput of all five flows in case 1 and case 2. 43

4.1 Delay and packet delivery ratio (PDR) of the three flows. 66

5.1 The distance estimation error. 86

5.2 The percentage of estimations with error $>$ 1 unit. 86

1. INTRODUCTION

Wireless ad hoc networks have unique characteristics like dynamic changing topology, wireless shareable communication medium, limited resources and lack of centralized administration, etc. As a result, there are a wide variety of attacks that target these weakness. In this type of network, security is not a single layer issue but a multilayer one. We have focused on physical layer, network layer and MAC layer where the possible attack are most vulnerable.

There are four major security vulnerabilities in wireless ad hoc network that need to be discussed in order to understand how to maintain a reliable and secure networks environment.

1.1. PROBLEM STATEMENT

Vulnerability of the open medium: Unlike physical hosts are connected by wires in the wired networks, the channel in wireless ad-hoc network is exposed in the air. The packets are easy to be eavesdropped and tampered with. The corrupted packets can be replayed anywhere and anytime in the network to disturb the network order.

Vulnerability of the open medium: Unlike physical hosts are connected by wires in the wired networks, the channel in wireless ad-hoc network is exposed in the air. The packets are easy to be eavesdropped and tampered with. The corrupted packets can be replayed anywhere and anytime in the network to disturb the network order.

Vulnerability of no fixed infrastructure: Ad-hoc networks nodes supposed to work independently of any pre-existing infrastructure. That means decision-making among nodes is usually decentralized. This makes the classic security methods, such as certification authorities and on-line servers hard to implement. The security and privacy have to rely on distributed cooperation among nodes. The attacker can exploit this vulnerability by breaking the cooperative algorithms. For example, the MAC protocols for wireless ad-hoc networks is vulnerable as known when it designed. Even though there are difference between different MAC protocols, the basic idea of most MAC protocols is similar. They are contention-based. By strictly following pre-defined protocols, each node competes to access the channel to avoid collisions or any other bad situation. However, if a node behaves maliciously, the MAC protocol can be broken down easily and a denial-of-service attack would be formed.

Vulnerability of dynamically changing topology: Also, the routing in ad-hoc network presents another vulnerability. Most of routing protocols are cooperative in nature. Unlike the wired network, there is no gateway or router nodes installed additional protection. If the node was compromised by attacker, it could disseminate false routing information to disturb the network, or even spread virus to compromised legitimate nodes.

Considering those vulnerabilities, several security requirements need to be met in wireless ad-hoc network. First, the service provided by the network should be always available. The depletion of some specific nodes in network core could cause unimaginable damage. Second, any malicious alteration to the information exchanged in the network should be detected and prevented. The integrity of information is essentially important in military and hospital whose use wireless ad hoc network widely. Third, confidentiality and privacy must be considered properly and comprehensively in design of any algorithm in wireless ad-hoc networks. There are differences between confidentiality and privacy. The former concerns hiding data from outer entities. The latter concerns avoiding the networks access sensitive data intrusively.

1.2. DOS ATTACKS AT DIFFERENT LAYERS

The past research work has put a lot works on the data integrity, confidentiality, privacy, and authentication for wireless ad-hoc security. However, another essential security requirement, availability has not been considered sufficiently. Nowadays, Denial of Service (DoS) attacks can reduce the availability of resource and result in massive damage.

Generally, a DoS attack is an attempt to make a host or any network resource unavailable to its intended users. Strictly speaking, a DoS attack not only referring to an adversary's attempt to disrupt, subvert, or destroy a network, but also is any event that diminishes or eliminates a network's capability to perform its function. Any hardware failure, software bugs, environment condition change or any other complicated factor can cause a DoS[60].

Although wireless ad-hoc network is considered to be used in harsh environment. It should function with the presence of error. This robustness of principle may prevent some classes of DoS attacks. For example, more efficient battery system or protocols which control the data rate prevent the depletion of nodes, or self-defense program can mitigate some classes of DoS attacks. However, the defender still need to consider the threat from intelligent, determined and well-prepared attacker. They could primarily attack the networks from protocols or design level instead just software bugs.

A layered network architecture can improve robustness. However, a clean division of layers may be sacrificed for performance in wireless ad-hoc networks [60]. Different attacks can target at any layer of ad-hoc networks. Known attacks perform on the physical, the MAC, the network layers. I will analyze existing DoS attacks layer by layer.

Physical layer: in the wireless ad hoc network, attacker can target the communication channel, which usually refer as jamming attack. It is a type of attack that uses radio interference to impede normal communications [67]. A jammer does not follow any layer protocol as a normal wireless transmitter does. For instance, if TDMA is used by the normal nodes, a jammer can send signals often at high power to interfere the transmission during normal nodes' allocated time slots; if CSMA is used by normal nodes, a jammer can defeat the CSMA protocol by occupying the channel at arbitrary time making other nodes wait for a long time. Jamming attacks will directly cause packet drops, high packet error rates, reduced throughput, and long delay. It can easily disrupt a network, regardless of higher level security mechanisms like encryption. The detection of jamming attacks requires not only to detect the radio interference that caused network performance degradation, but also to distinguish the observed degradation caused by network congestion or weak signals [62]. Many jamming detectors use test statistics derived from network measurements such as packet delivery ratio, packet error rate, and received signal strength, etc. Their detection logic is often simplistic as only summary information is used and the detection threshold is often preset at the initial setup phase. The limitation of existing work is not being able to maintain detection performance when the network dynamics increases as users join and leave the network. Dynamics in the normal operation of a network as well as the evasive techniques employed by the attacker makes fast and reliable jamming detection very challenging. Despite the large body of literature in jamming attacks, there is still unaddressed issues: mainly how to improve detection delay and accuracy. In section 4, we try to solves these challenges and advance research in the jamming attack detection.

MAC layer: attacks can also be launched in MAC layer. the MAC protocols is designed under the assumption that all participating nodes are well behaved [9]. Legitimate nodes will strictly follow the protocols, while the misbehavior nodes may deviate from the standard and cause the unfairness problems. The attacker could degrade the network service by generating a great a mount of traffic to keep the channel busy or sends a small packet as soon as he hears the start of transmission in order to corrupt the entire message.

These are similar to jamming attack in physical layer. Or the malicious nodes with simple changes of several protocol parameters can have devastating effects on the overall network performance. This is MAC misbehaviors which could also lead to DoS. Misbehaving nodes in wireless ad-hoc network intentionally alter the MAC protocols to improve their own performance at the expense of other users. The possible attacks due to MAC misbehavior have been explained below [4][17][43]:

1) Shorter IFS Attack: The IEEE 802.11 family of standards describe the DCF protocol, which controls access to the physical medium. A station must sense the status of the wireless medium before transmitting. If it finds that the medium is continuously idle for DCF Interframe Space (DIFS) duration, it is then permitted to transmit a frame. If the channel is found busy during the DIFS interval, the station should defer its transmission. The DIFS plays an important role in contention-based medium access procedure. While performing shorter DIFS attack, the compromised node waits for amount of time shorter than DIFS which can help the attacker get more chance to access the channel. Meanwhile, all other nodes in the network have to set up back-off timer after sensing the busy channel which has already been taken by attacker. In another word, the attack gets higher priority than other legitimate nodes.

2) Contention Window Manipulation Attack: According to 802.11 MAC protocols, after frame transmission has completed and the DIFS has elapsed, all the nodes trying to transmitting need to select a random back-off time from a fixed contention window (for 802.11b/g, $CW_{min} = 31$ and $CW_{max} = 1023$ while the case in 802.11a, $CW_{min} = 15$ and $CW_{max} = 1023$) and wait for that amount of time before transmitting. Each time the collisions happened(retry counter increases), the contention window moves to the next greatest power of two. It is the way preventing more collisions happening. However, if the attacker doesn't follow the back-off principles and choose a small back-off time each time when collisions happened, the attacker nodes will be able to access the channel more often than the legitimate nodes and get smaller delay and more throughput. Back-off manipulation attack could cause a lot of damage in a serious congested network. The attacker could always get the channel and let the legitimate have zero chance to transmit. 3) RTS Flooding Attack: The IEEE 802.11 allows stations to use Request to Send (RTS) and Clear to Send(CTS) signals to solve collisions resulting from hidden nodes in wireless ad-hoc networks. The node enabled RTS/CTS procedure sends a RTS at the beginning of transmitting and waits for the CTS from the destination. It can send the frames after it receives CTS. This brings out another safety issue. What if the attacker disseminates continuous RTS packets to legitimate sink node? The attacker can occupy the channel and keep the sink node busy for a long time. The sink node would probably âĂIJdieâĂİ at last, and other legitimate nodes would suffer a larger congestion. There is not much research put on this particular attack yet. In section 3, We have addressed these issues.

4) RTS Dropping Attack: In the RTS dropping attack, the receiver may not response with a CTS upon receiving a RTS. It selectively drops RTS packets, which causes the transmitter to retransmit several times, wasting time and channel resources. If the sender does not receive a CTS in the specified time, it will time out and go into the back off state. The attacker thus effectively deters the sender from channel access. In some case, the sender may perceive the absence of CTS as a result of congestion and tell the upper layer to change route in order to avoid the congested area. The ultimate benefit for the attacker is less requests from neighbors to access the medium. We have addressed the detection of such attacks by examining the performance time series from multiple nodes in this dissertation.

Network layer: attackers can also disrupt the network layer by spoofing, altering, or replaying routing information. One of severe network layer attacks is wormhole attack. The adversary could control two nodes, called wormhole end points, and a tunnel between them. It records a packet at one end point and replays it at the other end. The controlled nodes make false route advertisements to the neighbors. It will give the illusion of having a shorter path between each other, and attract large amount of traffic going through tunnel.

The adversaries can either launch an out-of band wormhole attack or an in-band wormhole attack[17]. The former usually use an external high capacity wire to link two nodes. There is no harm if attacker do nothing. But after the wormhole tunnel is established, other attacks usually are lunched afterwards. For example, the tunnel end nodes can drop packets, alter information, or even spread virus to compromised other legitimate nodes. The in-band wormhole, on the other hand, doesn't use additional infrastructure[53]. It redirects the traffic to an existing multi-hop route in the network in order to establish an inner-wormhole tunnel. Except the follow-up attacks after establishment, another serious problem is the congestion in the inner-wormhole tunnel. It attracts most of surrounding traffic to the tunnel and form a bad bottleneck in the network.

Another difference between them is in-band wormhole doesn't need extra hardware, it is easier to achieve than the out-of band attack[49]. As the wormhole attack is usually the foreplay of other more sophisticated attacks, it is important that the detection is timely and the alarm is sent as soon as possible when the legitimate nodes are aware of the existence of the wormhole[11].

Motivated by the respective advantages and limitations of the methods in the literature, we proposed a novel change point detection algorithms to detect these DoS attacks

2. IN-BAND WORMHOLE DETECTION

2.1. BACKGROUND

In wireless ad hoc networks, the routing procedure consists of nodes exchanging local information and relaying to others, and thus collectively determining a route towards a destination by following a certain principle, such as the shortest path principle. The routing process can be taken advantage of by an adversary to launch a wormhole attack, who wishes to monitor and control the routes. Wormhole attacks are severe threats as they are easy to be used by an adversary. Once a wormhole attack has succeeded, it can lead to other attacks.

In a wormhole attack, the adversary controls two nodes, called wormhole end points, and a tunnel between them. It records a packet at one end point and replays it at the other end. The controlled nodes make false route advertisements, give the illusion of a having a shorter path to the destination, and attract large amount of traffic to go through the controlled end points. In wireless networks, a category of protocols use dynamic routing and shortest path routing principle. Hearing a shorter path will trigger the route update in these networks and make the network vulnerable to a series of wormhole induced attacks.

The malicious nodes can either launch an out-band wormhole attack Figure. 2.1 or an in-band wormhole attack Figure.2.2. The out-band wormhole attack utilizes an external link between the two control points, such as a wired link or a long-range directional link. It is usually faster than going through a multi-hop path in a wireless network. If the wormhole nodes do not do anything else and simply add a link to the network to speed up the packet transportation, it would be a great service to the network as it adds channel capacity to the network. However, the real threat to the network is when other attacks are launched after a wormhole tunnel is established.

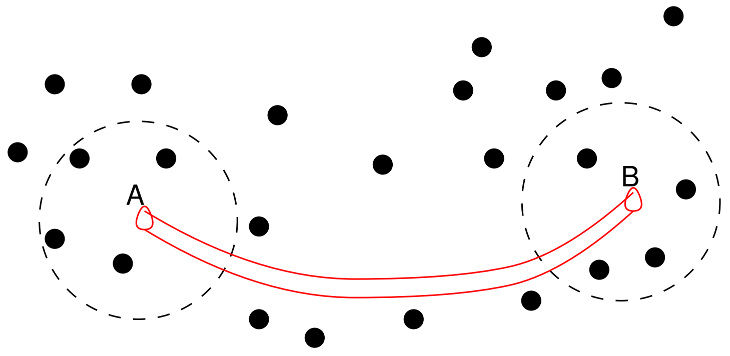


Figure 2.1. Out-band wormhole.

The in-band wormhole, on the other hand, does not use additional channel or additional hardware at the end points [27, 52]. It redirect the traffic to a multi-hop tunnel over existing wireless medium, and it consumes the network channel resource. Dropping packets and other attacks follow after the in-band wormhole tunnel is established. Even without other attacks, simply detour a lot of traffic to go through a longer path is enough damage, as it increase the hop counts on these paths. In-band wormhole attack does not need additional hardware, therefore it is easier to achieve than the out-band attack.

In this paper, we address in-band wormhole attacks. To defend agains the wormhole attack, three mechanisms are needed: a detection mechanism to sound the alarm, a localization mechanism to pinpoint the location of the malicious nodes, and a defense mechanism to mitigate the attack or to remove the wormhole tunnel. The literature has a collection of worm hole detection and mitigation methods, and some of them have detection and localization mechanisms combined into one scheme [25, 26].

In this work, we focus on the first phase of the counter-measurement, and develop a high fidelity detection mechanism. As the wormhole attack is usually the foreplay of other more sophisticated attacks, it is important that the detection is timely and the alarm is sent as

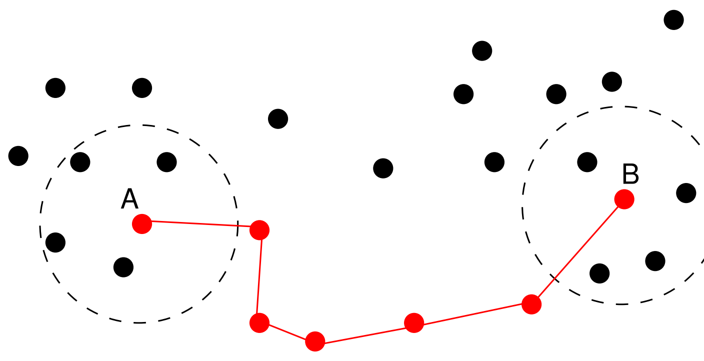


Figure 2.2. In-band wormhole.

soon as possible. When the legitimate nodes are aware of the existence of the wormhole in the network, the nodes can stop switching to a new route and effectively mitigate the effect of the wormhole. Therefore it is important that the detection rate is high and the same time the false alarm rate is low. This approach is light-weight compared to the combined approach that tries to locate the wormhole nodes at the time of detection, since to accurately locate the wormhole requires more information, and collecting the information regularly creates a large overhead for the network. In the separate approach, such heavy-weight localization mechanism is used only when the detection phase reports positive results.

With the observation that the end-to-end delay of a detoured flow will under go an abrupt change when the in-band wormhole attack is launched, we propose to study the end-to-end delay as a time series and use a sequential change point detection method to report abrupt changes. This detection scheme requires neither location devices such as GPS to provide location information of the nodes, nor tightly synchronized clocks on the nodes. It only requires some timing device to provide a timestamp in the packet when the source sends it. The clocks will not need to be tightly synchronized among the nodes in the network, as each time series is between a pair of source and destination, and the change is relative to its own previous history. We are not interested in the absolute delay between a pair of source and destination, instead, we are only interested in the relative change in the end-to-end delay. Since the clock skew between the two nodes is a constant that is added to each data point in the time series, the performance of the detection algorithm will not be impacted by clock skews. Compared to the previous work that rely on exact location and clock synchronization, the algorithm is light-weight in terms of the amount of information it needs to collect, store and process, and it also has very low communication overhead. The feature allows for a broad deployment of the detection algorithm on ordinary nodes that are not equipped with additional processing power.

The rest of the paper is organized as follows. In Section 5.2, we summarize the previous work in wormhole detection and mitigation. In Section 2.3, we describe the detection scheme, and in Section 3.4, we propose the new change point detection algorithm as the core component of the detection scheme. We present the detection performance and comparison with the previous work in Section 4.5.

2.2. RELATED WORK

A large number of wormhole detection methods have been proposed in the last decade. Based on the algorithmic features used, we divide them into statistical analysis methods and non-statistical analysis methods.

In statistical analysis, a common approach is to define a test statistics, and compute the statistics of certain measurement and compare it with a threshold value. Detection is positive when the test statistics exceeds the threshold value. The most related work to ours is [66], in which a comparison of two sequential change point detection methods are compared for in-band wormholes detection: the non-parametric cumulative sum (NP-CUSUM) and the repeated sequential probability ratio test (R-SPRT). In this study, each node in the network collects three-hop transmission delay data by periodically sending out *ping* packets to all the nodes that are three hops away.

In addition to the sequential change point detection methods, there are other statistical analysis approach such as [8**?**], in which the test statistics are taken from the information during route discovery: the relative frequency of each distinctive link appears in all routes, and the difference between the most frequently appeared link and the second most frequently appeared link. If the sample values are higher than the values in the normal state, it is determined the network is under wormhole attack. This approach requires that large number of routes in the network switch to use the wormhole tunnel in order to observe a change in the statistics. In [8], two detection methods are provided. The Neighbor Number Test (NNT) method detects the increase in the number of the neighbors of the sensors, and the All Distances Test (ADT) method detects the decrease of the lengths of the shortest paths between all pairs of sensors. It is expected that both the number of neighbors and the length of the shortest paths will be changed due to the new links created by the wormhole.The base station computes the expected histogram and the real histogram of neighbor numbers (or path lengths), and compare them by using a χ^2 test. If the computed χ^2 number is larger than a preset threshold that corresponds to a given significance level, then a wormhole is indicated.

In addition to statistical analysis approach, there are topological [14] or graph theoretical approaches [3, 36]. They first characterize the topological or graph theoretical features of normal states and attack states, and then develop a detection logic based on the characterization. The blacklist approach has also been proposed to address wormhole detection. In [25, 26], a malicious counter $Counter(i, j)$ is maintained at each guard node i for each neighbor node j, and the counter is incremented for j if any malicious activity of j is detected by guard node i. Local detection is positive if the counter cross a preset threshold value. Packet leash [23, 24] is another major approach for wormhole detection, in which a notion of leash is proposed based on the time or distance the packet is allowed to travel in the network. Similarly, in [59], the mechanism uses geographic information to detect anomalies in neighbor relations and node movements. The limitation for these approaches is that they need additional hardware such as GPS to get location information, or require tightly synchronized clocks among all nodes.

2.3. WORMHOLE DETECTION SCHEME

Wormhole detection is based on the following logic: when a route change (to the shorter one) is coupled with an abrupt increase in end-to-end delay, it is very likely the packets are detoured to go through an in-band wormhole tunnel. In a non-attack scenario, a shorter route is the result of either 1) adding in new relay nodes that create a short cut of the previous long multi-hop route, or 2) node movements that change the link connectivity so a shorter route is available. In these scenarios, shorter routes will result in shorter end-to-end delay since it takes fewer number of transmissions. It reduces not only the queuing delay at the relay nodes and propagation delay on the links, but also the number of transmissions, and hence effectively reduces consumption of channel resources. Delay and congestion have a non-linear relationship, and therefore a less congested network results in smaller delay. However in an attack scenario, the perceived shorter route is actually longer and takes more hops, therefore it increases the delay for the reasons above.

The detection scheme requires that each source add a timestamp when the packet is sent, and the destination record the end-to-end delay. For all packets it received from the same source, it creates a time series of delays. The algorithm for change point detection on a time series is applied at each destination. If changes are detected, alarms are sent to the designated node in the network. If it is a sensor network, such designated node could be the base station; if it is a wireless ad hoc network with no base station, a clustered structure is sufficient. Then all alarms will be sent to the cluster heads, and cluster heads can make decision whether a true attack has happened depends on the alarms it has received. Each destination will continuously monitor the delay but will only need to send the detection result to the designated node. Unlike the previous algorithms that use additional (often periodic) probing messages to detect changes, this algorithm does not create a large communication overhead. When a designated node receives alarms, accurate localization of the wormholes is the next step, which can be done by using a more complex procedure, such as sending probes and additional control messages. The localization procedure is an extension to detection and will be addressed in our future work.

2.4. THE SEQUENTIAL CHANGE POINT DETECTION ALGORITHM

There are a couple of change point detection algorithms in the literature. Zheng et al. provided a comparison study between the two change point detection algorithms [66]: The non-parametric cumulative sum (NP-CUSUM) and repeated sequential probability ratio test (R-SPRT). However, as will be pointed out in the next section, both algorithms involve manually-set thresholds or parameters that limit the application of the algorithms on the real-world dynamic traffic.

To detect a change point in a time series $\{x_1, x_2, \ldots, x_t, \ldots,\}$, it is assumed that before the change point, μ, the time series follows one distribution, and at time μ, a change occurs, and then the time series starts to follow another distribution. Suppose the pre-change and post-change density functions are $f(\cdot)$ and $g(\cdot)$, respectively. The hypotheses are then formulated as: If no change has occurred, then $\mathcal{H}_0$ is true; if a change has occurred, then $\mathcal{H}_\mu$ is true. The detection algorithm then decide which hypothesis is true.There are a number of algorithms to decide which hypothesis is true. The parametric version CUSUM algorithm

[32, 39] is the best when the pre- and post-change distributions are known. It uses the likelihood ratio to compute the cumulative sum of the log likelihood ratio as follows:

$$L_n = \frac{g}{f} W_n = (W_{n-1} + \log\ L_n)^+, \forall\ n \geq 1 \tag{2.1}$$

where $W_0 = 0$, and $x^+ = max(0, x)$. Then the cumulative sum is compared to a preset threshold h. The procedure declares a change as soon as the detection statistics W_n exceeds a preset threshold h:

$$T(h) = arg \min_{n:n\geq 1}\{W_n \geq h\} \tag{2.2}$$

Although mathematically sound, the reality is that the network traffic is very dynamic, and the end-to-end delay does not follow any known distribution. Without knowing the exact $f(\cdot)$ and $g(\cdot)$, the non-parametric version CUSUM is proposed to overcome the problem. The non-parametric version cumulative sum algorithm (NP-CUSUM) eliminates the need for the density functions, and use a heuristic approach to compute the detection statistics.

$$W_n = (W_{n-1} + x_n - c)^+, \forall\ n \geq 1 \tag{2.3}$$

where c is a preset constant, and the rest follows the parametric version CUSUM. However, if the traffic is highly dynamic, to set the detection threshold h and the constant c is a challenging task, and historical data will be of little use since the current data may not follow the pattern of historical data at all.

The repeated sequential probability ratio test (R-SPRT) works in a way similar to the parametric version CUSUM, but instead with two detection thresholds: an upper bound to claim $\mathcal{H}_\mu$ is true and a lower bound to claim $\mathcal{H}_0$ is true.

When the detection statistics is between the two bounds, decision is deferred and the algorithm continues.

R-SPRT is used when the distributions are known or can be reliably estimated. When a large volume of sample data is available, estimating the distribution functions g and f are possible. However, when it requires a large volume of data, it implies it requires a long time series with relatively stable traffic profile.

We develop a new sequential change point detection algorithm that requires no knowledge about the data characteristics. The algorithm uses a sliding window to calculate its detection statistics, and the detection threshold is decided based on the Central Limit Theory. The algorithm is named SW-CLT for its theoretical root.

The SW-CLT algorithm works as follows. Let m be the window size. Let t be an integer with $0 \le t \le n - 2m$. As the algorithm moves from $t = 0$ to $t = n - 2m$, it takes m consecutive data points from the time series—call it window 1, and the next m data points—call it window 2. Then we compute the sum of the x's in the two windows.

$$Y_1(t) = \sum_{i=t+1}^{t+m} x_i \quad \text{and} \quad Y_2(t) = \sum_{i=t+m+1}^{t+2m} x_i D(t) =| Y_1(t) - Y_2(t) | \tag{2.4}$$

We compare the difference between the sums of the two windows with a threshold D_{Th}, which is determined by the characteristics of the data and the desired upper bound for false alarm rate.

The detection threshold D_{Th} is computed by using the following procedure:

Let $\Phi(\cdot)$ be the cumulative distribution function (CDF) for standard normal distribution, defined as:

$$\Phi(z) = \P(a \le z) \tag{2.5}$$

Then $1 - \Phi(z)$ is the probability of ¶$(a > z)$ for a random variable a.For a desired false alarm rate ϵ, we are able to solve the cutoff value z from the following equation:

$$1 - \Phi(z) = \epsilon \tag{2.6}$$

Then the solution z for the above false alarm rate equation is scaled to be used as the detection threshold:

$$D_{Th} = z\sqrt{2m}\sigma, \tag{2.7}$$

where σ is the square root of the sample variance.

As the windows slide from the low end to the high end of the time series, if the following condition is met at time t

$$D(t) \geq z\sqrt{2m}\sigma, \tag{2.8}$$

then detection is positive. The algorithm decides a change point has occurred between the boundary of two windows, and report change time $\tilde{\mu} = t + m$.

For a true positive, the detection delay is the time difference between a reported change point and the true change point:

$$\text{Latency} = \tilde{\mu} - \mu \tag{2.9}$$

The algorithm is based on the Central Limit Theory and it works regardless of the underlying distribution of the end-to-end delay $\{x_1, \ldots, x_n\}$. For highly dynamic traffic, when the previous CUSUM and its variations fail, SW-CLT can be applied.

The use of the Central Limit Theorem eliminates the need to estimate the underlying distribution of the measurements, and relates the detection threshold with the allowable false alarm rates.

In the simulation, we will show that the changes in the time series can be big and small, however, the detection threshold is self-adjusting based on the dynamic characteristics of the measurements. When the data shows high variance, the detection threshold is also increased; and when the data is concentrated with a small variance, the detection threshold is also decreased. The advantage of the algorithm over the previous methods with a preset threshold is obvious.

2.5. SIMULATION

The simulation of wormhole attack is conducted in network simulator *ns3*. The network size is 40 nodes. We randomly generate node locations on a $500m \times 500m$ square region, and deploy them in *ns3* simulator. All nodes are equipped with 802.11b Wifi interface with data rate 11 Mbps. The network is shown in Figure. 2.3.

All nodes within 100m are connected by an edge. When node 1 and node 2 start a wormhole tunnel, there isn't a direct link to connect the two nodes. An in-band wormhole tunnel is created using the path 1-19-25-23-2. Nodes 1 and 2 as well as the nodes in the tunnel are controlled by an adversary. Node 1 will advertise there is a link from node 1 to node 2, then the routing paths are redirected to go through the tunnel if the new routes are shorter than the existing ones. Node 1 then use the in-band wormhole tunnel to send to node 2. Due to the fact that this path is longer than the previous route, it is expected that there will be noticeable increase in end-to-end delay. We try to detect the change as soon as the wormhole attack has started.

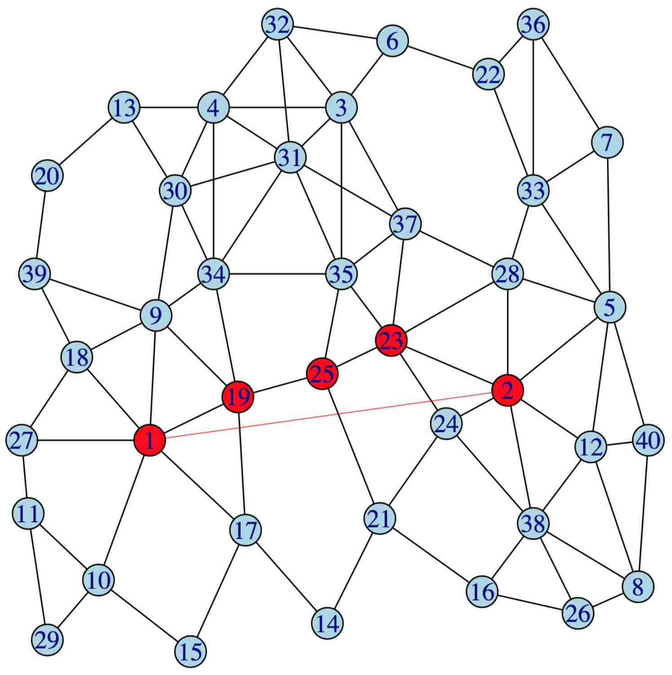


Figure 2.3. A 40-node network with in-band wormhole between nodes 1—2.

2.5.1. Wormhole Detection in a Stationary Network. In the first simulation, we set every node to be stationary. Simulation time is 100 seconds. At t=50 seconds into the simulation, packets start to be redirected to use the wormhole tunnel.It goes through the node 1, node 19, node 25, node 23, node 2.

We first observe the end-to-end delay when all traffic in the network is redirected to the wormhole tunnel. In Figure. 2.4, there are two flows 18–28 and 17–38, and both changed their paths after 50 seconds. The x-axis shows packets departure time from the source, and y-axis shows the end-to-end delay of packets.

- Flow 18–28:
  - Before 50 seconds: use path 18-9-34-35-37-28

  - After 50 seconds: use path 18-1 ... 2-28

- Flow 17–38:

  - Before 50 seconds: use path 17-14-21-16-38

  - After 50 seconds: use path 17-1...2-38

Since two flows both have minimum interference from other flows before the wormhole attack, and after the attack both switch to use the same tunnel, there is a significant and abrupt change in end-to-end delay after 50 seconds as shown in Figure. 2.4. The abrupt changes in delay are caused by both longer path and interference from each other.

Then we observe the end-to-end delay when some flows go through the wormhole tunnel while others follow their previous routes. The affected flows are those whose paths are one-hop away from either node 1 or node 2, so the source or relay nodes heard the fraudulent advertisement (of a shorter path) and decided to switch. The background flows are those whose packets stay on the original routes either because the routes won't be improved by going through the wormhole tunnel or because the source or relay nodes did not hear the fraudulent advertisement.

Due to the inter-flow interference, the end-to-end delay of the affected flows may not immediately undergo dramatic changes after 50 seconds. In Figure. 2.5, we observe the delay time series in three flows of different data rates. The three affected flows are 9–24, 18–28, and 17–38. The background traffic include flow 18–10 and flow 37-12. The packet sizes are the same for all flows, however the foreground traffic and background traffic packet transmission rates are different. For instance, in (a) the packet size is 256 bytes, foreground traffic packet interval is one packet every 0.01 second, and the background traffic packet interval is one packet every 0.025 second. As is shown in Figure. 2.5, some flow (e.g.,

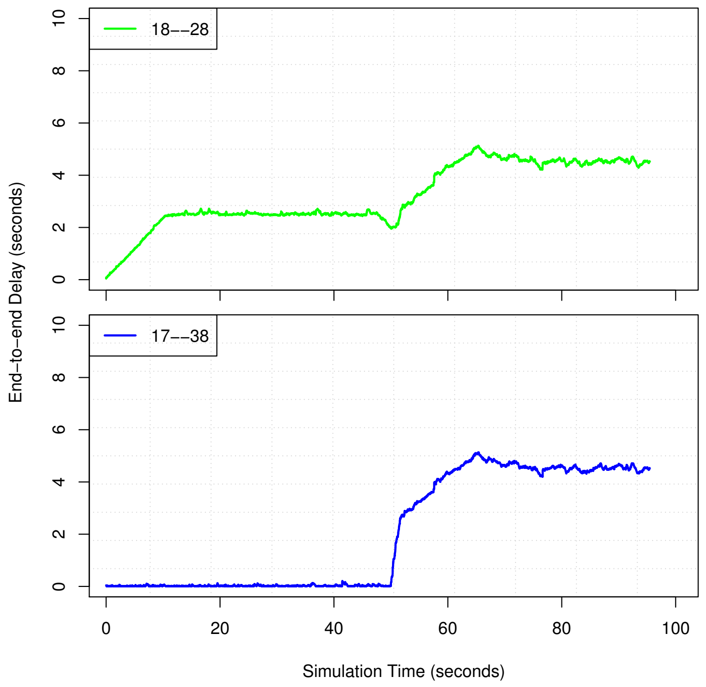


Figure 2.4. The end-to-end delay of two flows when they route through the wormhole tunnel.

17–38) experiences significant increase in delay in all cases, and some flow (e.g., 18–28) does not, and the outcome depends on the degree of congestion.

2.5.2. Wormhole Detection in a Mobile Network. In the second simulation, we use a mobile model for the network. All nodes are wandering within a small range around their original positions using the random walk 2d mobility model. The change of path in a wormhole attack scenario can be easily confused with the case when node 1 and node 2 are moving towards each other therefore the actual path length is reduced when they become closer. We will evaluate if the detection algorithm can successfully distinguish the benign case, in which route change is caused by node mobility, and the attack case, in which the route change is caused by wormhole. In the former, the actual path length is indeed decreased, while in the latter case, the actual length is increased while it is believed to be decreased.

Figure. 2.6 shows the end-to-end delay of three flows using different traffic load. As is observed, when the network is already under heavy traffic load, the net-effect of wormhole is not obvious. Although there isn't a clear or consistent pattern of change, it is detectable by SW-CLT since the detection threshold is adjusted with the data.

2.5.3. SW-CLT and NP-CUSUM Detection Results. The detection results are summarized in Table 2.1. Due to the space limit, we present the results for two extreme cases, one with dramatic change (Scenario 1), and the other with subtle changes (Scenario 2).

It is noted that in SW-CLT the only parameter that needs to be set manually is the window size m. The choice of m is not critical for the detection accuracy. We chose $m = 10$ for all scenarios. However in NP-CUSUM there are two parameters, the detection threshold h and the constant c, that are manually selected, and both are critical to the detection accuracy. For scenario 1, when the changes are big, the selection of h and c takes multiple trials to make it work, and the false alarm rate is high. For Scenario 2, when the changes are small, it is difficult to find the pair of parameters to detect the change points.

Table 2.1. Comparison of SW-CLT and NP-CUSUM on change point detection.

Algorithms	Scenarios	Detected	False Alarms	Latency (s)
SW-CLT	1 (18–28)	Yes	3	3.094
SW-CLT	1 (17–38)	Yes	7	6.995
SW-CLT	2 (9–24)	Yes	0	0.034
SW-CLT	2 (17–38)	Yes	1	0.038
SW-CLT	2 (18–12)	Yes	0	0.093
NP-CUSUM	1, all flows	Yes	>100	—
NP-CUSUM	2, all flows	No	—	—

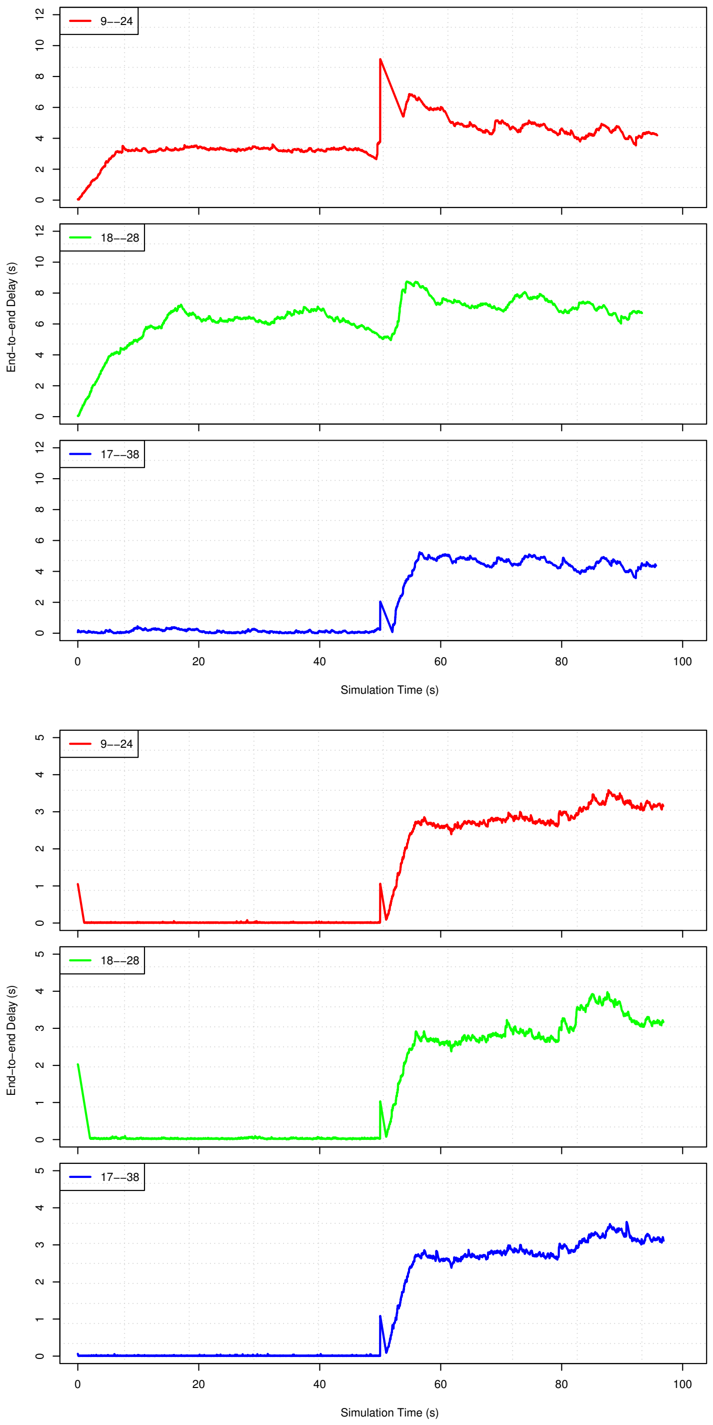


Figure 2.5. The end-to-end delay of three flows that go through the wormhole tunnel while the background traffic steers on the original routes.

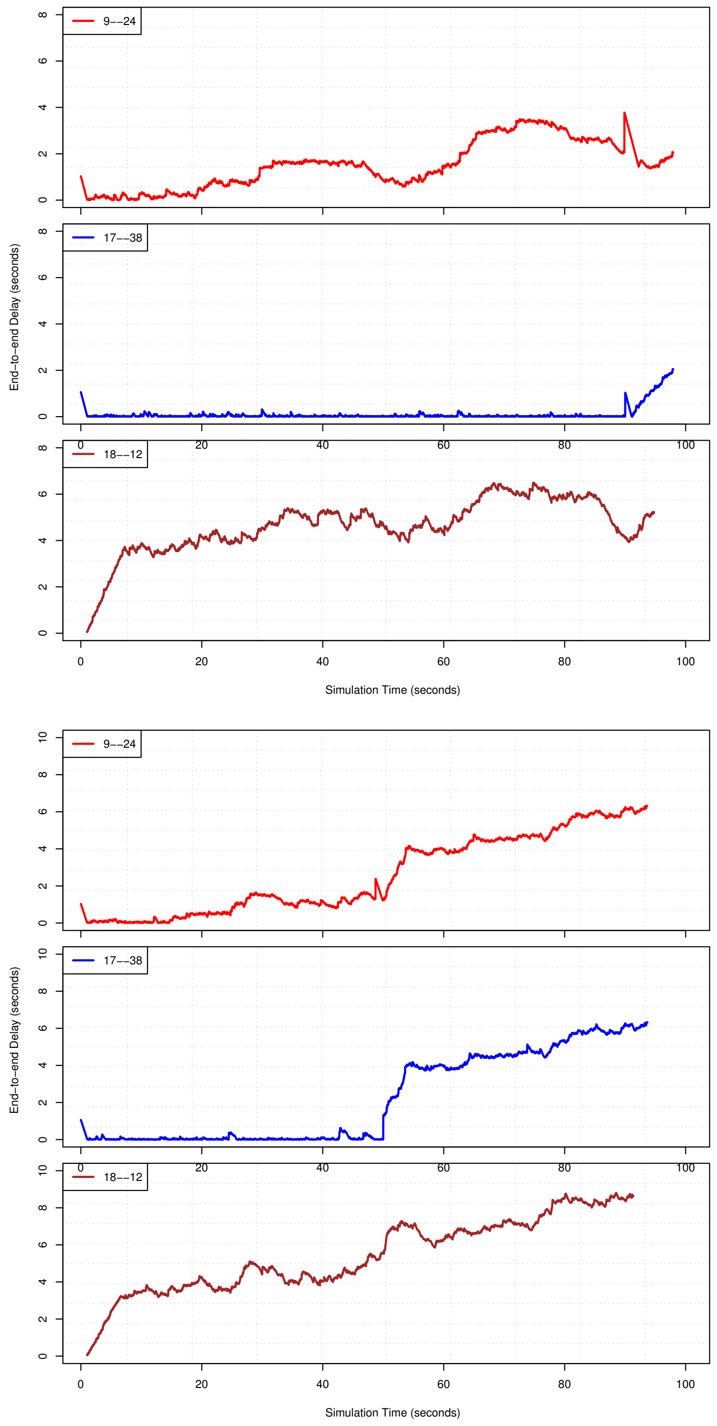


Figure 2.6. The end-to-end delay of two flows when they route through the wormhole tunnel.

3. IEEE 802.11 MAC MISBEHAVIOR DETECTION

3.1. BACKGROUND

The IEEE 802.11 MAC protocol is the *de facto* protocol for wireless local area networks (WLANs). The multiple access control mechanism features a distributed coordination function (DCF), which offers random and distributed access to participating nodes [6, 13]. One of the big success of IEEE 802.11 MAC protocol is it addresses the hidden terminal problem by using the Request to Send/ Clear to Send (RTS/CTS) message exchange before data packets are transmitted. Nodes in the vicinity of the sender and receiver will receive the RTS and CTS packets and therefore will refrain from channel access. Once a node has sensed the channel is busy, it enters binary exponential back-off to wait until the channel is clear. This is the key mechanism for collision avoidance.

In order for the protocol to be successfully executed, every node in the same WLAN is expected to conform to the protocol and follow the rules to determine the set of control parameters. These parameters include the carrier sense time Distributed Inter-Frame Space (DIFS) and Short Inter-Frame Space (SIFS), as well as the back-off value in the contention phase. If some nodes do not obey the protocol, it will create unfairness to other nodes, and even cause further damage to the network. We can categorize the attacks at the MAC layer into two categories: selfish attack, in which the offender committed mainly for its own benefit although they may also cause performance degradation of other nodes, for instance, monopolizing the channel so it can gain exclusive access to channel resource; malicious attack, in which the offender committed mainly for the purpose of creating damage for other nodes in the network. Selfish attack are also called greedy attacks or misbehaviors. Examples of such selfish misbehaviors include sender using a smaller back-off value, using shorter carrier sense time in order to have advantage in channel access, and receiver selectively dropping RTS packets in order to have more chance to transmit its own data. In this paper, we focus on the detection procedure and its application on three types of attacks: 1) manipulation on carrier sense time, 2) manipulation on back-off value, and 3) RTS dropping attack.

Although there are security mechanisms employed in IEEE 802.11 MAC, the cryptography based protection provided by WEP and WPA cannot deter such misbehaviors. The aforementioned misbehaviors are attacks on the channel resource, not on the data packets transmitted in the network. As long as the nodes are considered legitimate nodes in the network and are given secure keys for authentication, the misbehaviors cannot be stopped. The challenge in defending against such misbehaviors is that they all appear to be conforming to the protocol in the sense they all follow the carrier sense and then the sequence of RTS/CTS/DATA/ACK message exchange. In case of the RTS dropping attack by the receiver, the sender can be easily confused with collision on the RTS packet and retransmit one. To detect the misbehaviors, examining one sequence of packets exchanged is not enough, instead, a more sophisticated detection mechanism is needed that involves comparison of network wide observations as well as performance measurements in the past.

In this paper, we use a time-series analysis approach for the detection of MAC layer misbehaviors: we model the network performance measurements taken over time as time series and then apply a change point detection algorithm on time series to detect the abrupt changes caused by the misbehaviors. The objective is to detect misbehaviors as soon as they have started so that there is plenty of time to find the root cause and take effective counter-measures accordingly. In the context of network security, we take a two-step procedure to protect the network: detection and diagnosis. The detection procedure is active all the time. It is set out to hunt for anything that appears to be suspicious. Upon receiving alarms from the detection procedure, the diagnosis procedure will be activated to determine the root cause of the suspicious behavior. Therefore the requirement for the detection algorithm is to detect any suspicious behavior as soon as it starts. We argue that a real-time detection method that uses current data to perform attack detection is the most suitable for the application, as the data to process are network performance measures taken over time. Using archived data will require large storage and high processing power, and yet the historical data provides little insight about the current state of the network.

The rest of the paper is organized as follows. In Section 5.2, we survey the previous work in MAC layer attacks and detection methods. In Section 4.3, we provide the background for misbehavior detection using time series analysis approach. In Section 3.4, we present the core algorithmic component of the detection method. In Section 4.4, we show how the algorithm is used for the detection of each type of misbehavior. In Section 4.5, we present the detection performance.

3.2. RELATED WORK

Attacks on the lower layer protocols of wireless networks have been extensively studied in recent years, for instance, attacks on the network layer to disrupt routing [12], attacks on the MAC layer to gain more channel access [41, 44], and attacks on the physical layer to jam the channel [34]. Studies on MAC layer attacks as well as prevention and mitigation techniques constituted the main body of the literature. We further distinguish two types of attacks at the MAC layer: malicious attacks and selfish attacks.

In a malicious attack, an attacker intentionally attacks on a service node by keeping the channel busy near the service node [18], causes denial of service (DoS) to other nodes by injecting enormous amount of traffic into the network[69], or masquerades as a legitimate node by MAC address spoofing in order to disrupt network services [47]. In a selfish attack, a selfish node violates the rules of the protocol in order to gain more channel access to itself at the cost of performance degradation of others [16, 19, 43].

IEEE 802.11 MAC offers distributed and random access to nodes, but also leaves a lot of room for a selfish nodes to exploit. The selfish attacks are mainly targeted at the Distributed Coordinated Function (DCF) of IEEE 802.11 MAC [6, 13]. In the literature, the most addressed attack is cheating on the backoff rule [1, 9, 15, 16, 20, 28, 30, 44]. The well-behaved nodes would follow the binary exponential back-off rule to decide its waiting time during the backoff stage, whereas a selfish node can just choose a small back-off value and increase its channel access time.

To detect the misbehavior of cheating on the back-off rule, many studies used the comparison of a selfish node and the well-behaved nodes on their performance measures. Without knowing whether there is a selfish node and who is the selfish node, the access point would periodically collect data from all nodes to get a center line (or average). A general principle of these detection methods is that if a node's data deviate from the center line too much, it is considered selfish. Performance measures used as detection statistics include throughput [15, 16, 43], the mean time between received packets [15], RTS and data packets retransmission rates [55], and node's channel access time [19], etc. This approach involves determining the average from all nodes, and setting a detection threshold. The detection threshold is either a constant shift from the center line, or a constant fraction of the center line, where the constant shift or constant coefficient are manually set or calculated from historical data.

Another category of detection methods involve more sophisticated statistical methods, such as the sequential probability ration test (SPRT) method [44], or the statistical process control (SPC) method [1]. The SPRT method in [44] determines if a selfish node exists in a network by observing a sequence of n data points representing packet inter-arrival time. The distribution of packet inter-arrival time is estimated, and then SPRT is performed on each new data point using the estimated probability density functions under both hypotheses. It stops when either of the stopping criteria is met, i.e., either accepting the null hypothesis that there is no selfish behavior, or accepting the alternative hypothesis that there is selfish behavior. Although it is a sequential method, it is not designed to detect the selfish behavior at the moment it starts; it is rather used to determine whether the n data points are generated by a selfish node or a well-behaved node. It is assumed that the node behavior has been consistent in the sequence of n data points, therefore it is not a change point detection method. Section 4.3 provides more detailed description of the method.

The SPC method in [1] uses the data collected from normal cases without misbehavior to set the upper and lower control limits and the center line, and monitors the performance metrics (i.e.,throughput and packet inter-arrival time) using the control chart, and decides that there is selfish behavior when the metrics deviate from the chart significantly. Detection is in real-time if the new data points fall out of the control limits. However, the control chart is determined using historical data.

The change point detection approach in this paper differs from all previous work in the sense that it is not assumed that the misbehavior exists from the beginning of the time series. The detection method can detect the misbehavior as soon as it starts. It also does not use archived data to set the detection threshold. The detection threshold is calculated from the current data, and therefore it significantly improves the detection rate and false alarm rate for highly dynamic data.

3.3. TIME SERIES ANALYSIS METHODS

Network Measurements: MAC layer misbehaviors cause performance degradation of the victims. For instance, if one node becomes selfish and starts to use shorter carrier sense time to give itself higher priority in accessing the channel, other flows will have less channel resource to use. This will result in longer delay and less throughput than in a normal situation. If a node decides to switch to selfish behavior at some point in time, such behavior should be observed in 1) the degraded performance measurements of other flows, and 2) the improved performance of its own. The performance degradation in an attack scenario is distinguishable from the degradation caused by network congestion on a busy network or from channel condition deterioration, since in a benign scenario, the performance degradation is from all flows.

The detection procedure requires that we sample round-trip time from successfully received packets starting from sending RTS until receiving ACK, and also monitor the data packets received over time. Attack detection is running on each active node. A sender node should sample the round-trip time over time, and a receiver node should record the received data packets from a particular sender over time. The number of packets received over time is bursty even in the non-attack scenario due to the random nature of IEEE 802.11 MAC. If we use this time series for attack detection, it will generate too many false alarms. Therefore, instead of using the instantaneous throughput, we use the cumulative throughput over time as the time series to perform attack detection. Detection algorithm is performed at each node independently. Each node run a sequential change point detection algorithm on the time series it generates, some may requires using derived data instead of raw data to perform. If a node has detected a change point, alarms are sent to the designated node in the network. If it is a sensor network, such designated node could be the base station; if it is a wireless ad hoc network with no base station, a clustered structure is sufficient. Then all alarms will be sent to the cluster heads, and cluster heads can make decision whether a true attack has happened depends on the alarms it has received.

3.3.1. Problem Statement. In mathematical abstraction, detecting abrupt changes caused by the misbehaving nodes is casted as a change point detection problem in time series. A time series is a sequence of data points $\{x_1, x_2, \ldots, x_t, \ldots,\}$ representing some measurements/observations taken from a stochastic process over a continuous time interval, and a change point in the time series is a time instance starting from which the probability distribution of the stochastic process has changed. It is assumed that before the change point, the time series follows one distribution, and after the change point, and the time series starts to follow another distribution. Suppose the pre-change and post-change density functions are $f(\cdot)$ and $g(\cdot)$, respectively, and the change point is μ. The alternative hypothesis is then formulated as:

If no change has occurred, then $\mathcal{H}_0$ is true; if a change has occurred, then $\mathcal{H}_\mu$ is true. The detection algorithm is tasked with deciding which hypothesis is true, and if the alternative hypothesis is true, the algorithm reports the change point. It is desired that the algorithm have high detection rate and low false alarm rate, and a low detection latency, which is the time interval from the true value of μ to the time a change is detected.

3.3.2. SPRT vs. Change Point Detection on Time Series. The repeated sequential probability ratio test (SPRT) [58] also works on a sequence of data and processes each data point sequentially. In SPRT, two hypotheses are made:

The algorithm uses two probability density functions, $\P(x \mid H_1)$ and $\P(x \mid H_0)$. Given a time series $\{x_1, \ldots, x_n\}$, the probability ratio is calculated using the following equation:

$$r_n = r_{n-1} \frac{P(x_n \mid H_1)}{P(x_n \mid H_0)}, \quad \forall n \geq 2 \tag{3.1}$$

$$r_1 = \frac{P(x_1 \mid H_1)}{P(x_1 \mid H_0)} \tag{3.2}$$

Intuitively, when r_n is large, it means that the sequence $\{x_1, \ldots, x_n\}$ is more likely generated from the distribution under H_1 than from the distribution under H_0, therefore H_1 should be accepted. The algorithm proceeds until the detection statistics r_n hits a threshold value.

Instead of using one detection threshold, it uses two detection thresholds: an upper bound A to claim the alternative hypothesis is true and a lower bound B to claim null hypothesis is true. The two detection thresholds A and B are set by using the given type I and type II error rates. The algorithm starts from the first data point and sequentially process each new data point x_i until one of the stopping criteria is met:

- when $r_n \leq B$, H_0 is accepted;
- when $r_n \geq A$, H_1 is accepted;
- when $B < r_n < A$, defer decision and continue with the next data point.

When the detection statistics r_n is below the lower bound, the null hypothesis is accepted stating that the data follow the distribution of well-behaved nodes; when the detection statistics r_n is above the upper bound, the alternative hypothesis is accepted stating that the data follow the distribution of misbehaving nodes; when the detection statistics is between the upper and lower bounds, decision is deferred and the algorithm continues to next data point. SPRT is suitable when the distributions under both hypotheses are known or can be reliably estimated. In practice, the true distribution is unknown. When a large volume of sample data is available, estimating the probability density functions $\P(x \mid H_1)$ and $\P(x \mid H_0)$ are possible. However, this requires a long time series with relatively stable traffic profile, which is hard to obtain in a highly dynamic network.

Another significant difference between SPRT and change point detection is that SPRT uses hypothesis testing approach to find out if there is misbehavior, whereas change point detection uses hypothesis testing to find out if there is change point in the time series. In SPRT, between H_1 and H_0, only one them is always true. This means there is no change point in the time series. In case the misbehavior starts somewhere in the middle of the time series, SPRT can prematurely terminates with conclusion that H_0 is true. In Figure. 3.1, the algorithm SPRT terminates in the first segment of data accepting H_0 whereas the misbehavior starts after the stopping point.

CUSUM Algorithms: There are a number of algorithms for change point detection on time series. The parametric version CUSUM algorithm [32, 39] is the best when the pre- and post-change distributions are known. It uses the likelihood ratio to compute the cumulative sum of the log likelihood ratio as follows:

$$L_n = \frac{g}{f} \tag{3.3}$$

$$W_n = (W_{n-1} + \log\ L_n)^+, \forall\ n \geq 1 \tag{3.4}$$

where $W_0 = 0$, and $x^+ = max(0, x)$. Then the cumulative sum is compared to a preset threshold h. The procedure declares a change as soon as the detection statistics W_n exceeds a preset threshold h:

$$T(h) = arg \min_{n:n\geq 1}\{W_n \geq h\} \tag{3.5}$$

Although mathematically sound, the reality is that the network traffic is very dynamic, and the end-to-end delay does not follow any known distribution. Without knowing the exact

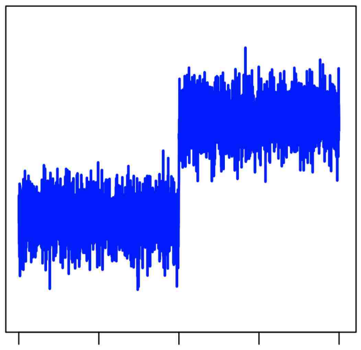


Figure 3.1. SPRT algorithm terminates prematurely accepting H_0 when the misbehavior appears after the stopping point.

$f(\cdot)$ and $g(\cdot)$, the non-parametric version CUSUM is proposed to overcome the problem. The non-parametric version cumulative sum algorithm (NP-CUSUM) eliminates the need for the density functions, and use a heuristic approach to compute the detection statistics.

$$W_n = (W_{n-1} + x_n - c)^+, \forall\, n \geq 1 \tag{3.6}$$

where c is a preset constant, and the rest follows the parametric version CUSUM. However, if the traffic is highly dynamic, to set the detection threshold h and the constant c is a challenging task, and historical data will be of little use since the current data may not follow the pattern of historical data at all.

Despite the popularity of the CUSUM family algorithms, which pretty much is due to the lack of other types of algorithms, these algorithms involve manually-set thresholds or parameters that limit the application of the algorithms on the real-world dynamic traffic. Network traffic is highly dynamic and random, to use a manually set parameters based on historical data will not work.

3.4. CHANGE POINT DETECTION ALGORITHMS

Let m be the window size. Let t be an integer with $0 \leq t \leq n-2m$. As the algorithm moves from $t = 0$ to $t = n-2m$, it takes m consecutive data points from the time series—call it window 1, and the next m data points— call it window 2. Then we compute the sum of the x's.

$$Y_1(t) = \sum_{i=t+1}^{t+m} x_i \quad \text{and} \quad Y_2(t) = \sum_{i=t+m+1}^{t+2m} x_i \tag{3.7}$$

$$D(t) = \mid Y_1(t) - Y_2(t) \mid \tag{3.8}$$

We compare the difference between the sums of the two windows with a threshold D_{Th}, which is determined by the characteristics of the data and the desired upper bound for false alarm rate. The detection threshold D_{Th} is computed by using the following procedure:

Let $\Phi(\cdot)$ be the cumulative distribution function (CDF) for standard normal distribution, defined as:

$$\Phi(z) = \P(a \leq z) \tag{3.9}$$

Then $1 - \Phi(z)$ is the probability of $\P(a > z)$ for a random variable a that follows the standard normal distribution.

For a desired false alarm rate ϵ, we can solve the cutoff value z from the following equation:

$$1 - \Phi(z) = \epsilon \tag{3.10}$$

Then the solution z for the above false alarm rate equation is scaled to be used as the detection threshold:

$$D_{Th} = z\sqrt{2m}\sigma, \tag{3.11}$$

where σ is the square root of the sample variance.As the windows slide from the low end to the high end of the time series, if the following condition is met at time t

$$D(t) \geq z\sqrt{2m}\sigma, \tag{3.12}$$

then detection is positive. The algorithm decides a change point has occurred between the boundary of two windows, and report change time $\tilde{\mu} = t + m$. For a true positive, the detection delay is the time difference between a reported change point and the true change point:

$$\text{Latency} = \tilde{\mu} - \mu \tag{3.13}$$

The algorithm is based on the Central Limit Theory and it works regardless of the underlying distribution of the end-to-end delay $\{x_1, \ldots, x_n\}$. For highly dynamic traffic, when the previous CUSUM and its variations fail, CLT-based algorithm can be applied. The use of the Central Limit Theorem eliminates the need to estimate the underlying distribution of the measurements, and relates the detection threshold with the allowable false alarm rates.

3.5. MISBEHAVIOR DETECTION IN IEEE 802.11 MAC

In IEEE 802.11 MAC, ordinary asynchronous traffic all uses the distributed coordination function for medium access. Before the sender transmits a frame, it performs carrier sense. The carrier sense duration is dependent upon the Inter-Frame Space (IFS) value at the present time. If the medium is idle for IFS amount of time, it transmits a frame. Each atomic unit for frame transmission is a four-frame sequence: RTS-CTS-DATA-ACK. If the medium is busy or becomes busy before the IFS amount of time counting down to zero, it waits until the current transmission ends and waits for IFS amount of time again before going to contention. During the contention period, each node performs binary exponential back-off to select a waiting time. The node that selects a smaller back-off time among the contenders will be the winner and will transmit a frame. For asynchronous traffic, there are two different IFS values: Distributed coordination function IFS (DIFS) and Short IFS (SIFS). DIFS is used before sending RTS, while SIFS is used in the later part of the sequence before CTS, DATA, and ACK.

In this paper, we address three types of attacks targeted at IEEE 802.11 MAC protocol. A selfish node can exploit the vulnerability of the protocol to give itself more channel access time and degrade the throughput performance of other nodes. Type I Attack: shorter DIFS,Type II Attack: shorter DIFS combined with smaller back-off value,Type III Attack: RTS dropping.The back-off value manipulation has been extensively studied in the past and we do not intend to repeat in this study. It is also observed that only when the network traffic load is high and nodes frequently enter the back-off stage, the impact of this attack is high. If there is no contention among nodes, there is no chance for a selfish node to use this attack. Furthermore, nodes still need to wait for DIFS amount of time before counting down on the back-off counter. Therefore we think it is necessary to consider the joint effect of shorter DIFS and smaller back-value, and compare it with the first attack that uses shorter DIFS only.

When the attacker launches an attack, the other flows experience performance degradation notable from the time series of performance measures. At the moment of detection, we cannot distinguish what type of attack it is. Further investigation is needed to distinguish whether the change is caused by an attacker or by a new flow that just joined the network. The focus of this paper is to first detect such changes in real-time as soon as new data points are collected. The identification of attacks will be the next step after detection.

3.5.1. Type I Attack: Shorter DIFS. Based on the carrier sense protocol, a node refrain from transmission if the medium is busy. If a node uses a shorter DIFS value than its peers, it gains a lot of advantage. It is essentially a priority to use the transmission medium. If two nodes both have a frame to transmit and they start to perform carrier sense at the same time, the node with shorter DIFS will start to transmit ahead of the other one. Carrier sense prevents the other node from interrupting the ongoing transmission so the other node will have to wait until the medium is cleared. If a node always uses shorter DIFS values, it will have a larger share of the channel resource than other nodes, so the throughput and delay performance will be improved, meanwhile the throughput and delay performance of other flows will be degraded.

To detect the selfish behavior, we cannot rely on the selfish node to perform attack detection on its own data and report itself. We can only detect the misbehavior from the performance degradation of other flows. The delay time series in an attack scenario won't be a constant shift from its normal state. As delay increases, the jitter or variance of delay also increases. If we put the delay data in a time series, it is expected that the distribution of the cumulative means of delay and variance of delay have a change point at the moment of attack. Throughput time series will show similar increase in both cumulative means and variance. Applying the change point detection algorithm on any one of the time series will be able to detect the attack.

3.5.2. Type II Attack: Shorter DIFS and Smaller Back-off Value. In the carrier sense period, if a sender sensed that the medium is busy, it will refrain from channel access.

After the medium becomes free for DIFS or SIFS amount of time, if all senders start to transmit a frame right away, there is potential collision among the senders in the back-off stage. Therefore it is necessary to make each sender wait for a random number of slots before transmitting a frame. This period is called contention. The random number is chosen between zero and a power of 2 (not inclusive) based on the binary exponential back-off algorithm. In the non-attack case, every node chooses its random number independently, the one that chooses the smallest number will start to transmit first and deter others from transmission. In the attack case, the attacker either chooses a random number from a smaller window or uses a fixed back-off value, γ, instead of following the exponential back-off algorithm during the contention period. The attacker will have higher probability to gain channel access than the rest of nodes.

It is observed that when the attacker uses a smaller γ, it gains more channel access than using a larger γ. Whether the combined use of shorter DIFS and smaller back-off value further create more detrimental effect on other flows, this is interesting to find out. Intuitively, a more aggressive attacker will cause more damage to the network. However the data from simulation show that this is not the case.

3.5.3. Type III Attack: RTS Dropping. In the RTS dropping attack, the receiver may not response with a CTS upon receiving a RTS. It selectively drops RTS packets, which causes the transmitter to retransmit several times, wasting time and channel resources. If the sender does not receive a CTS in the specified time, it will time out and go into the back-off state. The attacker thus effectively deters the sender from channel access. In some case, the sender may perceive the absence of CTS as a result of congestion and tell the upper layer to change route in order to avoid the congested area. The ultimate benefit for the attacker is less requests from neighbors to access the medium. The attacker thus *clears* the channel for itself so that it can have more channel access time.

Absence of CTS is often regarded as a result of collision, or poor wireless channel condition. It may not be immediately taken as a sign of attack. The distinction between congestion and selfish misbehavior is possible when more flows are observed. In the congestion case, all flows show performance degradation along with high retransmission rates of RTS packets, whereas in the attack case, only the flow that has the attacker as a relay node or receiver will suffer, and the attacker as a sender actually has improved throughput due to increased channel access time.

3.6. SIMULATION

To simulate the misbehaviors, we hacked into IEEE 802.11 WiFi MAC in ns-3 to create three attacker classes different from well-behaved nodes. The well-behaved nodes will follow the IEEE 802.11 MAC protocol and use default parameters, while the attacker classes can deviate from the rules of the protocol and use different parameters.

We use ns-3 to generate network traffic, and then retrieve performance measures such as delay, throughput and inter-packet interval from the trace file. To collect delay data, the sender needs to time-stamp the packet at the MAC layer when RTS is sent. receiver can recollect end-to-end delay, throughput, and inter-packet interval from the received packets.Figure. 3.2 shows the network used for simulation. Nodes are deployed in a $150m \times 150m$ terrain area. Every sender is one hop away from its receiver. Node 2 is the selfish node in all cases.

3.6.1. Case 1: Shorter DIFS Attack. Simulation Setup: The IEEE 802.11 family of standards describe the DCF protocol, which controls access to the physical medium. A station must sense the status of the wireless medium before transmitting. If it finds that the medium is continuously idle for DIFS amount of time, it is then permitted to transmit a frame. If the channel is found busy during the DIFS interval, the station should defer

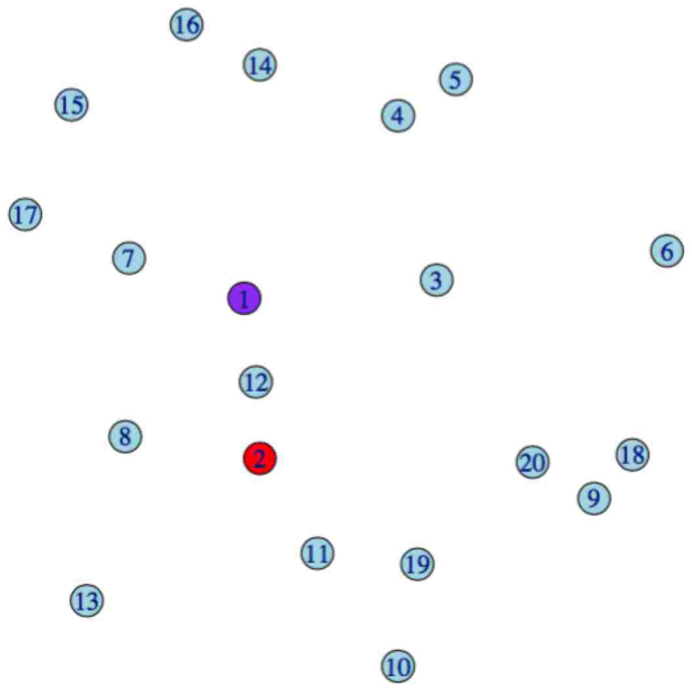


Figure 3.2. The 802.11 network under selfish attack and Node 2 (red) is the attacker in all cases.

its transmission. The DIFS plays an important role in contention-based medium access process. If the attacker can modify the value of DIFS and get a smaller one, it can get more chance to access the channel and let other legitimate nodes suffer various delay. In IEEE 802.11b, the default value of DIFS is SIFS + 2 $\times$ slot-time, with SIFS=10μs and a slot-time=20μs. In this simulation, the selfish node uses DIFS=SIFS.

There are five flows from the well-behaved nodes: 19 - 1, 14 - 1, 12 - 1, 10 - 1, 6 - 1. The attacker starts to behave selfishly at simulation time 50 second. The attacker is near the sink node, and for some flows it is also near the sender. Due to the shorter DIFS, the attacker has better chance to send out RTS and reserve the channel, thus deter the other flows from transmission. We expect that all other flows will experience a performance degradation.

Results: With the attacker using a shorter DIFS value, other flows are given less channel access time. Figure. 3.3 shows the delay and throughput as well as the mean and variance of them for the flow from node 14 to node 1. Although the throughput shows oscillation, the cumulative mean of throughput is decreasing therefore the throughput is still decreased over time. Other flows are shown in Figure. 3.4. As we can see, the delay of a normal flow increases and the throughput drops. All victim flows show similar pattern.

In 802.11 MAC, the contention window size is measured in slot-time, with each slot-time=20μs. A well-behaved node follows the binary exponential back-off rule to select a random number γ between [0, CW-1], where CW is the contention window size. The minimum contention window size is 32 and the maximum contention window size is 1024. A node would double its contention window size each time it enters the contention state without a successful transmission until it reaches the maximum value. Upon successful transmission, the contention window size is reset to the minimum value 32.

The selfish node can cheat on the back-off rule by using a smaller back-off value. In this simulation, we choose a fixed $\gamma = 2$. In the contention period, when the channel is sensed idle for DIFS amount of time, the back-off counter starts to count down. When the counter reaches zero, a node transmits a RTS frame. Since the selfish node uses DIFS=SIFS and a fixed back-off value $\gamma = 2$, whereas the normal nodes have DIFS=SIFS+2 $\times$ slot-time, it is equivalent to say that the selfish node waits no time after a normal DIFS amount of time to transmit a new RTS packet, while the normal nodes have to wait an additional random number of time-slot.

The network setup is the same as in case 1. The results show that all five flows experience significant throughput loss, companioned by increased delay. Figure. 3.5 shows the delay and throughput of four flows. The fifth flow from node 6 to node 1 is shown with more detail in Figure. 3.6. Applying the change point detection algorithm on the cumulative means time series can easily detect the performance change caused by the selfish node at near 50 seconds.

A counter-intuitive observation is that the performance degradation of normal flows are not as severe as in the first case where the attacker only uses shorter DIFS but still follows the same rule for contention. Table 3.1 shows the cumulative means of delay and throughput of each victim flow. This is due to the fact that the selfish node no longer participates in contention when the well-behaved nodes start to contend, and therefore the number of nodes in the contention period is reduced. As a result, all other flows experience lighter traffic load in the contention period.

Table 3.1. Cumulative means of delay and throughput of all five flows in case 1 and case 2.

	19-1	14-1	12-1	10-1	6-1
Case 1 Delay	0.0283	0.0541	0.0064	0.0360	0.0521
Case 2 Delay	0.0042	0.0044	0.0035	0.0038	0.0048
Case 1 Throughput	818.95	812.22	819.12	817.44	816.26
Case 2 Throughput	818.95	819.20	819.20	819.20	819.03

3.6.2. Case 3: RTS-dropping Attack. Simulation Setup: In the RTS-dropping attack, the selfish receiver may not response with a CTS by selectively dropping a number of RTS packets, which will cause the transmitter node to retransmit serval times, deterring other nodes from transmission while the selfish node can ignore it and go ahead with its own transmission. Since the transmitter does not receive CTS in the specified time, it will time out on CTS and use binary exponential time to back off. And it definitely make the network latency higher.

We use the same network as shown in Figure. 3.2 to simulate the attack. There are five flows in the network: 20-2, 11 - 2. 8 - 2. 7 - 2, 5 - 2. Node 2 is the selfish node that selectively drops RTS. In the simulation, the ratio of CTS to RTS is 1/20, i.e., it drops the first 20 RTS packets and responds to the 21^{st} with CTS, and repeats.

Results: Figure. 3.7 shows the delay and throughput for one flow. Other flows show similar patterns. It is interesting to observe that the delay does not show significant increase. This is due to the fact that the RTS retransmission limit is set to 7. With a high drop rate of $20/21$, the RTS packets including the retransmitted RTS are mostly dropped, and only a small portion can get through. The ones that did get through did not experience much delay since the network traffic load is lighter. This is clearly observed on the delay chart where the received packets are sparser after 50 seconds. Note that the x-axis shows the receiving time of the packets. Figure. 3.7(b) shows that the inter-packet interval experiences significant increase after the attack starts.

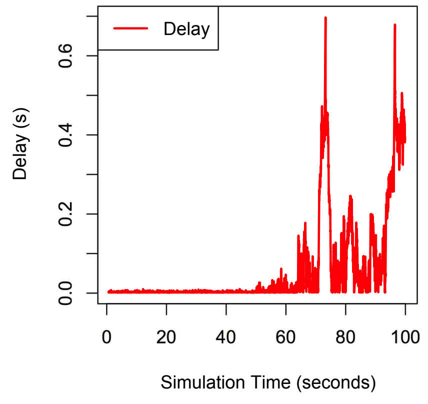
In this case, instead of using the delay time series, the detection is performed on the inter-packet interval and throughput time series. Both data are available at the receiver, however, we cannot rely on the receiver to perform attack detection since the receiver is the attacker. The sender has to keep track of successfully transmitted data packets and record the packet interval based on the sending time and the total number of bytes transmitted. Figure. 3.8 show the data rate and the inter-packet interval for the successfully transmitted packets.To directly apply the change point detection algorithm on the transmitter data rate and packet interval time series will generate many false alarms, however when we use the cumulative average of the data rate and packet interval, the time series are smoothed out, and the "change point" corresponding to the attack time is obvious, as shown in Figure. 3.9.

3.6.3. Detection Results Summary. Comparison between SW-CLT and non-parametric CUSUM: SW-CLT has smaller false alarm rate and higher detection rate. The false alarm rate of SW-CLT is bounded from above by the given ϵ. SW-CLT also has a bounded detection latency. In this simulation, we have been using window size $m = 10$, therefore for the successful detections, the detection latency is bounded by 10 data points after the attack time. Nonparametric CUSUM does not have a bounded detection latency.Detection results for three types of attacks:

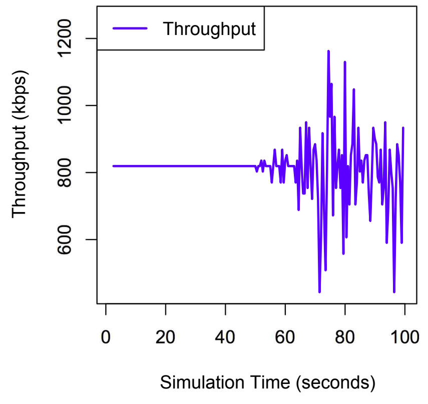
- Shorter DIFS attack: applying the SW-CLT change point detection algorithm on delay, throughput, and cumulative means of throughput series can detect the attack at 50 seconds. Applying it on other time series will have larger detection latency. The cumulative means of throughput provides the overall best detection result— the smallest false alarm rate and the lowest detection latency.

- Shorter DIFS and smaller back-off value attack: applying SW-CLT on the cumulative means of delay and throughput can detect the attack at 50 seconds, with delay series having higher false alarm rate.

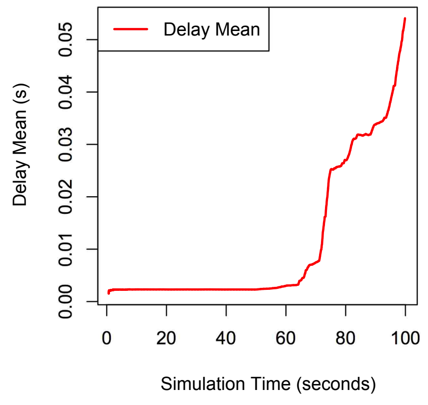
- RTS Dropping attack: although the change point is visible on the transmitter data rate and transmitted packet interval time series, applying SW-CLT on these two time series generates high false alarms. Applying SW-CLT on the cumulative means of data rate and packet interval generates fewer false alarms.



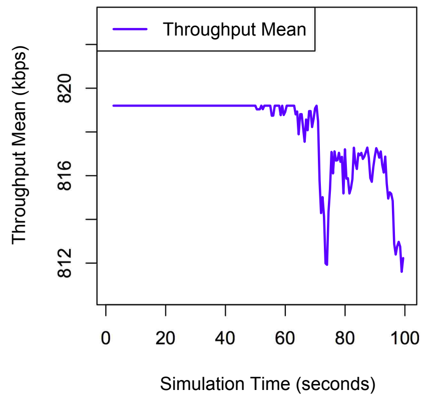
(a) Delay



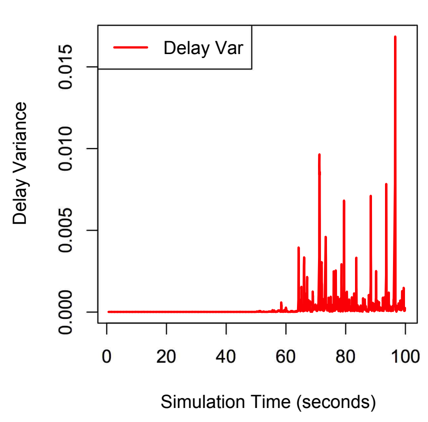
(b) Throughput



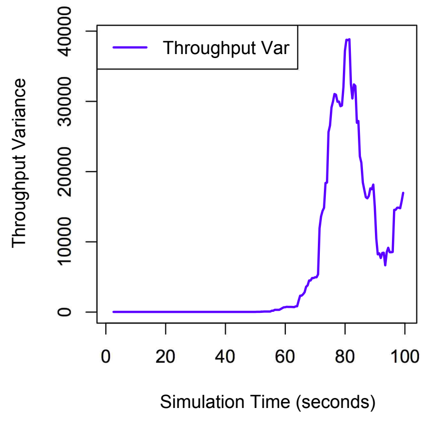
(c) Delay Mean



(d) Throughput Mean

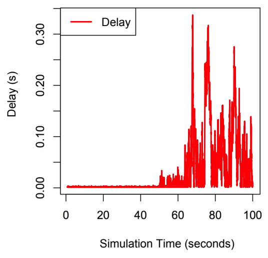


(e) Delay Variance

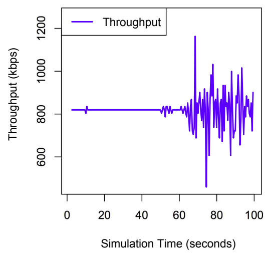


(f) Throughput Variance

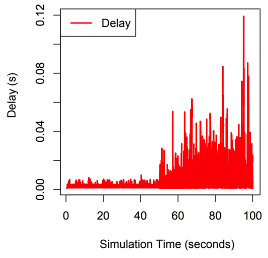
Figure 3.3. The delay and throughput as well as their mean and variance for one flow (14-1).



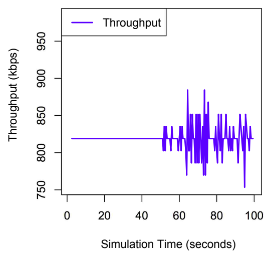
(a) Delay of flow 19-1



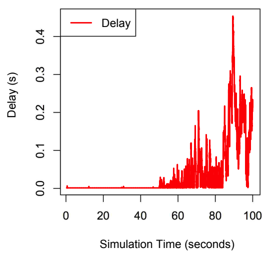
(b) Throughput of flow 19-1



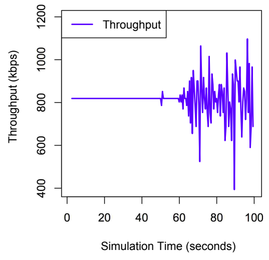
(c) Delay of flow 12-1



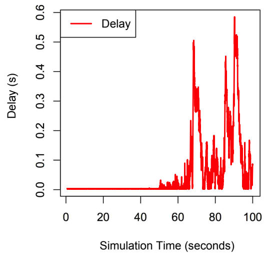
(d) Throughput of flow 12-1



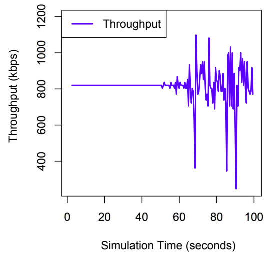
(e) Delay of flow 10-1



(f) Throughput of flow 10-1

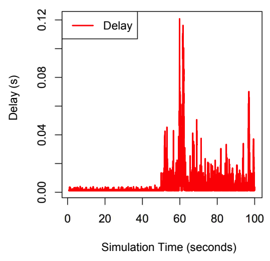


(g) Delay of flow 6-1

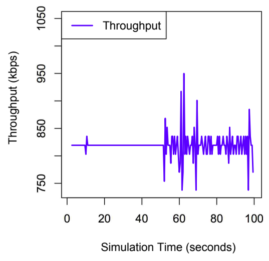


(h) Throughput of flow 6-1

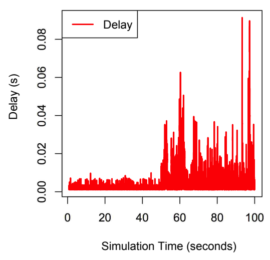
Figure 3.4. The delay and throughput of all other flows show similar performance degradation.



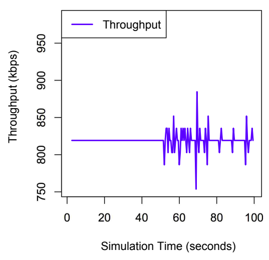
(a) Delay of flow 19-1



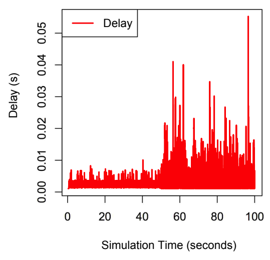
(b) Throughput of flow 19-1



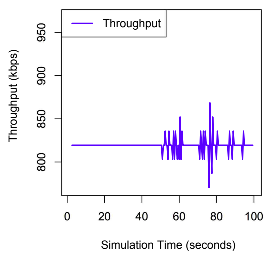
(c) Delay of flow 14-1



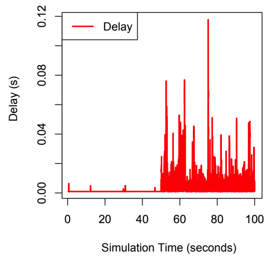
(d) Throughput of flow 14-1



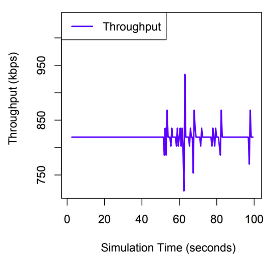
(e) Delay of flow 12-1



(f) Throughput of flow 12-1

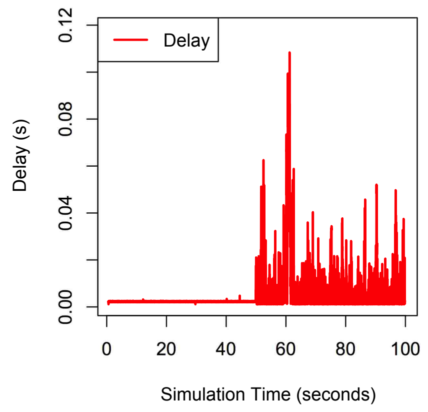


(g) Delay of flow 10-1

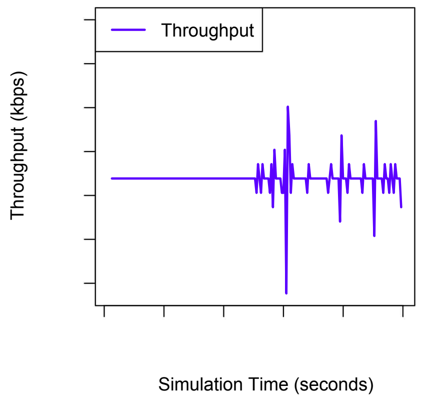


(h) Throughput of flow 10-1

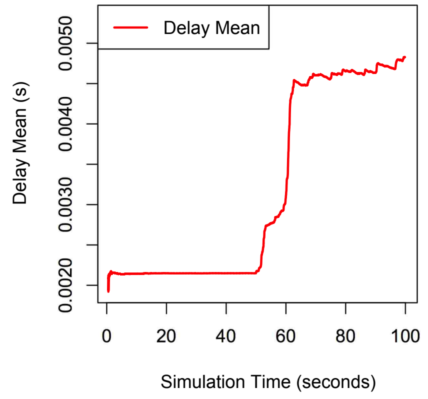
Figure 3.5. The delay and throughput of four flows in the combined shorter DIFS and smaller back-off value attack.



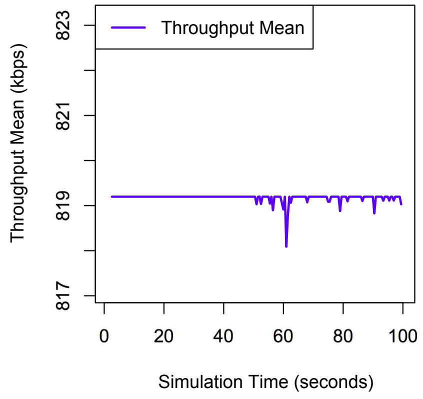
(a) Delay of flow 6-1



(b) Throughput of flow 6-1

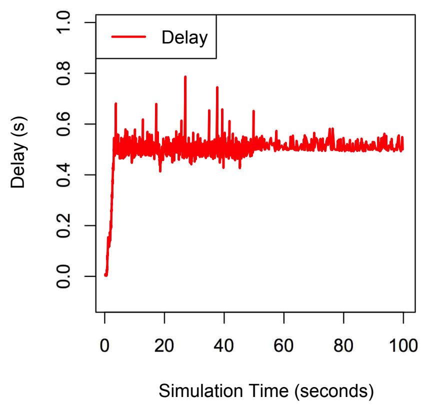


(c) Mean Delay of flow 6-1

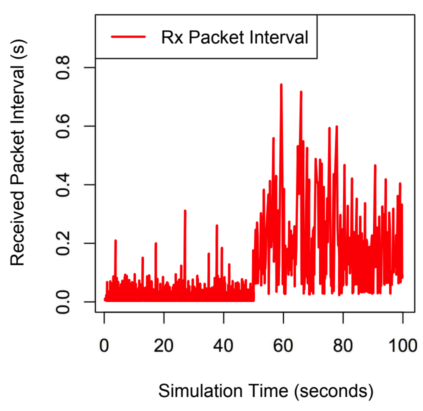


(d) Mean Throughput of flow 6-1

Figure 3.6. The delay and throughput of flow 6-1 in the combined shorter DIFS and smaller back-off value attack.

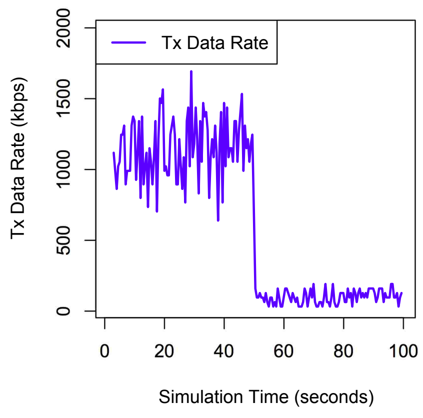


(a) Delay

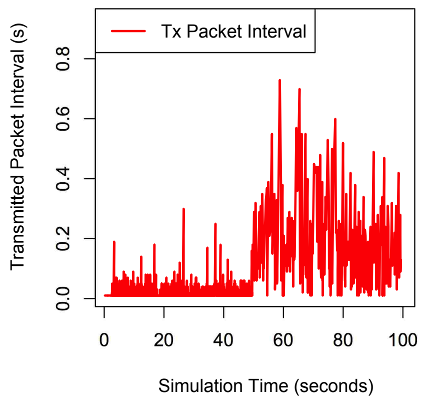


(b) Rx Inter-packet Interval

Figure 3.7. The delay and inter-packet interval of a victim flow (20-2) under RTS dropping attack.

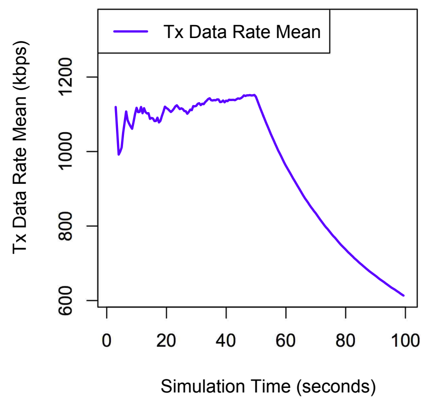


(a) Tx Data Rate

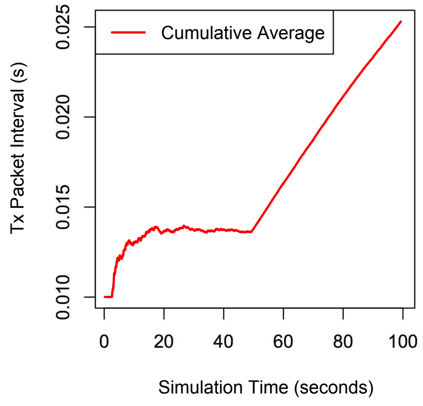


(b) Tx Inter-packet Interval

Figure 3.8. The transmitter data rate and inter-packet interval of a victim flow (20-2) under RTS dropping attack.



(a) Tx Data Rate



(b) Tx Inter-packet Interval

Figure 3.9. The cumulative means of the transmitter data rate and inter-packet interval of a victim flow (20-2) under RTS dropping attack.

4. JAMMING DETECTION

4.1. BACKGROUND

Wireless transmissions are exposed to shared media. This is a nice feature that makes them pervasively available but also makes them vulnerable to physical layer attacks. Jamming attack is a type of security attack that uses radio interference to impede normal communications [68]. A jammer does not follow the MAC layer protocol as a normal wireless transmitter does. For instance, if TDMA is used by the normal nodes, a jammer can send signals— often at high power— to interfere the transmission during normal nodes' allocated time slots; if CSMA is used by normal nodes, a jammer can defeat the CSMA protocol by occupying the channel at arbitrary time making other nodes wait for a long time. Jamming attacks will directly cause packet drops, high packet error rates, reduced throughput, and long delay. To effectively protect wireless networks from jamming attacks, it is important that the jamming attack is timely detected. In this paper, we focuses on detection in real-time and leaves mitigation in future work.

The detection of jamming attacks requires not only to detect the radio interference that caused network performance degradation, but also to to distinguish the observed degradation caused by a jammer from those caused by network congestion or weak signals. Many jamming detectors use test statistics derived from network measurements such as packet delivery ratio, packet error rate, and received signal strength, etc. for jamming attack detection, however the detection logic is often simplistic as only summary information is used and the detection threshold is often preset at the initial setup phase. The limitation of existing work is not being able to maintain detection performance when the network dynamics increases as users join and leave the network. Dynamics in the normal operation of a network as well as the evasive techniques employed by the attacker makes fast and reliable jamming detection very challenging. Despite the large body of literature in jamming attacks, there is still unaddressed issues: mainly how to improve detection delay and accuracy.

In this paper, we aim to address these challenges and advance the research in jamming attack detection. The proposed methodology involves profiling and detecting a change of state in time series, which improves over the existing method in time series change point detection for data with high variance, and the application of the method can be very broad and not limited to a specific type of jamming attack. The features that distinguish this work from previous work in jamming detection are: 1) new detection methodology— advanced time series analysis techniques are used to detect the change of network state, which improves over the previous methods that compare the extremes (i.e., maxima) with a preset threshold; 2) short detection delay— detection happens as soon as the attack starts, which makes it suitable for real-time detection. This is a significant difference from retrospective analysis, which has to wait until the attack has lasted for a long period of time to conclude that a jamming attack has happened; 3) improved detection effectiveness— false positive rate and false negative rate are both low. In particular, it distinguishes a jammer's interference from a normal node's interference, and it distinguishes the packet drops due to jamming from those due to weak signals.

The rest of the manuscript is organized as follows. In Section 5.2, we briefly review the representative work in jamming attack detection; in Section 4.3, we present the statistical methods for jamming attack detection; in Section 4.4, we present the proposed time series based detection method and its application in wireless networks for jamming attack detection; in Section 4.5 we show the performance of the detection method in simulated wireless networks. Section **??** concludes the paper and points out future research directions.

4.2. RELATED WORK

Due to the shared nature of wireless channels, wireless networks are susceptible to channel jamming attacks. A variety of detection methods have been investigated to detect different types of jamming attacks. We survey and compare jamming detection research in two aspects: detection methodology and detection statistics. Many jamming detection methods involve sampling at attack-free time to decide initial network parameters and detection thresholds [10], and jamming detection during normal operation is simply done by comparing the current measurements with the preset thresholds. Unlike previous work, the proposed method in this paper does not use a preset threshold and the detection threshold is dynamically updated as new measurements are collected.

Detection statistics used in the literature range from packet-level network measurements such as Packet Delivery Ratio and Bad Packet Ratio (BPR), to network-wide statistics such as Energy Consumption Amount (ECA) and channel utilization [37], etc. In some work, physical layer statistics such as signal collision ratio [29], received signal strength (RSS) have also been used for jamming detection. In [51], an anti-jamming scheme is developed to protect the reactive alarm systems. It monitors the received signal strength during the reception of bits in sensor networks, and is able to differentiate packet errors due to jamming attacks from errors caused by weak signals. In [50], chip error rate-based metric is used instead of packet-level metrics. It involves extracting statistics from the jamming-free symbols of the DSSS synchronizer to discern jammed packets from those lost due to bad channel conditions. The technique has been successfully applied to the most sophisticated reactive jammers. In [63], authors presented four jammer attack models that can be used against a wireless network and proposed two detection methods. The two detection methods both use the notion of consistency checking, which is simply comparing the maximum value with a preset threshold value. Packet Delivery Ratio (PDR), signal strength, and carrier sensing time are used as detection statistics, and consistency checks on signal strength and location are performed in order to further distinguish attack scenarios and benign scenarios. The proposed detection method in this paper is also a type of consistency checking in the sense that we check the changes in network measurements before and after the attack, however it is change point detection on a time series and therefore is more advanced and effective than simple comparison of summary statistics.

[64] is a improvement over [63] by including not only PDR but also Packet Send Ratio (PSR) at the transmitter side in order to distinguish the types of the jamming attack. Packet Delivery Ratio-based detection method also includes [48], in which a collaborative detection method is proposed to evaluate the Packet Delivery Ratio in an given area instead of a pair of nodes. In [54] a decentralized jamming detection method is presented for ad hoc networks. Cross layer metrics from MAC and network layers are used to differentiate congestion and jamming.

Jamming detection for sensor networks has attracted a lot of research interests in recent years mainly due to the low power of sensor nodes and relatively high power of jamming nodes. In [31], a distributed detection scheme in wireless sensor networks is presented. It uses a fixed set of sensors for collecting collision results and sending them to a center node for final decision making. Bayesian decision framework is used at the central node to make decision with the objective of minimizing the error probability. In [61], not only detection of jamming attacks but also mapping of jammed areas has been studied. In [61], jamming detection is performed by monitoring channel utilization. If the channel utility is below a preset threshold, it is determined that it is jammed. A jammed node will defeat CSMA limitation and send messages to neighbors, then neighbor nodes collectively build a map of the jammed region.In defense of wireless networks against jamming attacks, the first step is detection and the next step is mitigation. Mitigation can be done by avoiding or actively competing with the jammer. In [62], two mitigation methods are presented for sensor network. One approach is to simply retreat from the jammer by using either spectral evasion (channel surfing) or spatial evasion (spatial retreats). The second approach aims to compete more actively with the interferer by adjusting resources, such as increasing power levels or using coding to achieve communication in the presence of the jammer.

4.3. STATISTICAL JAMMING ATTACK DETECTION

In the Bayesian estimation approach [31], it is assumed that each node has a known probability to transmit in each slot, based on which, the collision probability is derived if no jamming has occurred. It is also assumed that the jammer has a fixed probability to jam the network.

4.3.1. Bayesian Estimation. Bayesian decision framework is used to determine whether jamming has occurred based on the number of observed collisions during a period of time. Each node monitors whether there is a collision independently and sends the observation to a fusion center. The fusion center will make the final decision based on the collective observations. The fusion center uses the *Maximum A Posterior (MAP)* estimator to decide whether a jamming attack has occurred.

For the simple Bayesian decision framework to work, the network has to follow the rigid assumptions, for instance, each node has a fixed probability to transmit in a slot. If different nodes have different transmission probabilities, or the probabilities are unknown or not constant, the method would be unsuitable. It is also hard to apply it in a random access network due to the MAC layer randomness.

4.3.2. Simple Threshold-Based Detector. The majority of statistical detection methods falls in this category, called *statistical anomaly detection* [10]. In this approach, the system profile in the normal state is created, and then anomalies are detected by comparing with the normal state profile. Based on the historic data, a threshold is determined and the parameters are compared with a preset threshold. For instance, the $6 - \sigma$ method is a widely used method, in which the upper and lower limits of the normal range is set by $\mu + 6\sigma$ and $\mu - 6\sigma$, where μ, σ are the mean and standard deviation of the normal samples from historical data. Values that fall out of the upper and lower limits of the "band" is considered abnormal. Although the detection logic sounds correct, the difficulty in the real world is that network traffic is very dynamic, and fixed thresholds cannot be applied to a constantly changing network.

The proposed method in this paper is also a statistical anomaly detection method in the sense we also treat attacks as anomalies. However, different from the previous work, we do not use fixed thresholds for detection. The detection threshold is computed in real time based on a sequential time-series analysis method.

4.4. TIME SERIES ANALYSIS APPROACH

The proposed method differs from previous work in both detection statistics and the algorithmic idea.

4.4.1. Characterization of Jamming Attacks. In previous work [10], the parameters used to describe the system profile include Packet Delivery ratio(PDR), Bad Packet Ratio(BPR), and Energy Consumption Amount(ECA). The assumption is that jammer does not follow MAC layer protocol, and therefore the jamming signal will interfere with the ongoing traffic, which causes the rise of BPR and ECA and the drop of PDR. However, if CSMA is used, the legitimate node will refrain from transmission when the jamming signal is active. The PDR or BPR won't be severely impacted. They will still be worse than in the no-attack case, but won't be significantly worse. For any type of network, regardless of the MAC layer protocol used, the damage is ultimately on the network performance.

Throughput will be decreased and delay will be increased due to the reduced channel access time. If the two measurements are not impacted, then the jamming attack has not caused serious damage to the network. In this paper, we characterize jamming attacks by the performance degradation in throughput and delay of normal nodes. The jamming indicator is the synchronized performance degradation of all affected nodes coupled with the occurrence of longer signal blocks and higher signal power than normal nodes.

4.4.2. Change Point Detection on Time Series. Having observed the limitation of previous work in threshold-based detection, we propose a real time detection method suitable for dynamic data set. In this method, we avoid creating a static normal state profile as network traffic is forever changing. The detection threshold won't be a preset value based on archived data. We model the delay and throughput measurements taken over time as time series, and try to find a change point from the time series, which is likely an indicator of the network state change. At the time of detection on a single time series, we only conclude that an abrupt change has been observed but the cause of the change is unknown. We use separate procedures for detection and further determination of the exact type of jamming attacks. The detection procedure is always active, and it is set out to hunt for any anomaly in real time. Only after an anomaly is detected, the detailed analysis to decide the cause of it is activated.

There are a handful of algorithms in the literature for change point detection on time series. The most widely used is the CUSUM family [32, 39]. However, the original parametric version CUSUM algorithm requires the knowledge of the distributions before and after the change point. In practice, these distributions are unknown. The non-parametric version CUSUM and its variations all require manual selection of the detection thresholds, which again fall in the drawback of the aforementioned statistical anomaly detection methods for having preset thresholds.

In this paper, we apply a different method for time series change point detection. Let $\{x_1, x_2, \ldots, x_n\}$ be the time series of interest. The change point detection problem is the problem of deciding which of the following hypotheses is true, where $f()$ and $g()$ are two different distribution functions:

If the null hypothesis H_0 is true, then there is no change point; if the alternative hypothesis H_A is true, then a change point has been detected.

Instead of using preset threshold values as those used in the non-parametric version CUSUM algorithms, we calculate the threshold on-the-go and update the detection threshold as soon new measurements are sampled.

The algorithm uses the upper bound for false alarm rate to decide the detection threshold.

Let $\Phi(\cdot)$ be the cumulative distribution function (CDF) for standard normal distribution, defined as:

$$\Phi(z) = \P(a \leq z) \tag{4.1}$$

Then $1 - \Phi(z)$ is the probability of $\P(a > z)$ for a random variable a that follows the standard normal distribution. Suppose a desired false alarm rate ϵ is given as input, we can solve the cutoff value z from the following equation:

$$\Phi(z) = 1 - \epsilon \tag{4.2}$$

The solution value of z will be used in the detection procedure to compute the detection threshold. The algorithm works as follows:

1. Decide a window size m. Usually for an N data point time series, $m = N^{1/3}$ is used.

2. Read in two windows worth of data $\{x_{i+1}, \ldots, x_{i+2m}\}$ each containing m data points. As the algorithm proceeds with the next data point, the two windows each move one data point.

3. Compute the sum of m data points in each window, and then the absolute difference between the sums of the two windows

$$D(i) = \left| \sum_{t=1}^{m} x_t^{(i)} - \sum_{t=m+1}^{2m} x_t^{(i)} \right|^2 \tag{4.3}$$

4. Update the sample variance σ^2

5. Compute the detection threshold $Th(i)$ by using the following procedure:

$$Th(i) = z\sigma\sqrt{2m} \tag{4.4}$$

6. As soon as the the detection statistics $D(i)$ exceeds the detection threshold $Th(i)$, the algorithm returns with positive result (i.e., H_A is true), and T is the detection time:

$$T = arg \min_i D(i) \geq z\sigma\sqrt{2m} \tag{4.5}$$

   If no change point has been detected, update $i = i + 1$, read in a new data point x_{i+2m} and advance each window by one position, and then repeat the procedure [3]-[6].

Remark: An important parameter is ϵ. A small ϵ means small false alarm rate. The detection threshold will increase as ϵ decreases, and the algorithm will be less sensitive to minor changes.

4.5. JAMMING ATTACK SIMULATION

4.5.1. Simulation Setup.

4.5.1. Simulation Setup. The jamming attacks are simulated in network simulator *ns-3* (https://www.nsnam.org). We redeveloped the jamming modules in *ns-3.21* based on a legacy jamming module from Network Security Lab (University of Washington). The legacy module was developed for *ns-3.11*.

A wireless network is simulated where a node can reach all other nodes in one-hop. Normal nodes send frames with a frame header. Jammers are implemented as transmitters sending giant packets of arbitrary length. Jamming signals do not have frame headers; they are a continuous block of signals. Normal nodes and jammers all use a fixed transmission power 0.04 W.

In the second simulated network, a total of 6 normal nodes are deployed in a square region (See Figure. 4.5(a)). Node 6 is the jammer. We observe three flows $0 \rightarrow 1$, $2 \rightarrow 3$, and $4 \rightarrow 5$. The simulation time is 100 seconds, and the jammer starts attack at 50.8 seconds.

In the third simulated network, a total of 12 nodes are deployed in the same square region as in the previous case (see Figure. 4.5(b)). There is no jammer in this case. Three flows $0 \rightarrow 1$, $2 \rightarrow 3$, and $4 \rightarrow 5$ are existing flows and we observe that as new transmitters join the network later how the new transmissions impact the existing flows. Flow $6 \rightarrow 7$ starts at 30.5 seconds, flow $8 \rightarrow 9$ starts at 31 seconds, and flow $10 \rightarrow 11$ starts at 31.5 seconds. All three new flows are sending at a packet rate of 10 packets per second.

4.5.2. Impact of Jamming Signal Duration. The first objective of the simulation study is to observe how the jamming signals impact the transmitting and receiving of normal packets. A jammer does not follow the MAC protocol for channel access. It is expected that jamming signals will not only collide with transmitted packets, but also refrain the normal nodes from sending due to the channel capturing effect. In the simulated network, a total of 4 normal nodes are deployed in a square region (See Figure. 4.1). Node 4 is the jammer. We observe the flow from node 0 to node 3. Node 0 transmits to node 3 at a packet rate of one packet per second, and the packet size is 200 bytes. The simulation time is 100 seconds, and the jammer starts attack at 30.8 seconds.

Several measurements are taken at the receiver side when packets are received: duration of the received packet, number of bytes in the received packets, end-to-end delay of the packet, and signal-to-interference-plus-noise ratio (SINR) together with the time of reception. Other information can be derived from these measurements, such as throughput, inter-packet time interval (gap), etc. When the jamming attack starts, it is expected that normal nodes will experience increased delay and reduced throughput. Jammers usually use either higher transmission power in order to create strong radio interference, or larger blocks of continuous transmission in order to capture the channel, or the combination of both. We observe the SINR values and the duration of the received signals.

In order to study the impact of jamming packet duration, we used three different durations for the network in Figure. 4.1: 0.0005s, 0.8s and 1.5s. The impact of the jamming packet duration can be clearly observed on the network measurements, especially the end-to-end delay. When the jamming packet duration is short, the normal packets get transmitted more often during the jamming packet intervals, therefore the total number of packets being transmitted is more, and the throughput is higher compared to the scenario with longer jamming packets. The delay shows oscillation in Figure. 4.2(b). This is due to the effect of queue management— packets staying in the queue for more than 0.5s are dropped so new packets will have shorter delay. The delay in Figure. 4.2(c) also has oscillation, but with a longer cycle. This is due to the long jamming duration— even if the packets staying in the queue for more than 0.5s get dropped, the new packets may still not have a chance to transmit due to channel being jammed. The SINR data further confirmed that the received

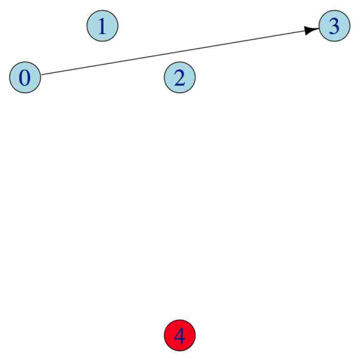


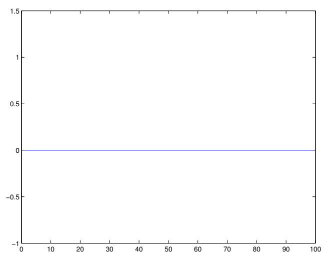
Figure 4.1. One flow with a jammer. (node 4)

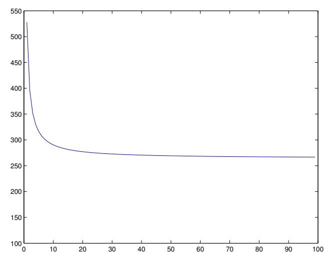
packets (blue lines) are more sparsely interspaced in (f) than in (e) after the jamming attack, and throughout data show that the throughput drop is more significant in (i) than in (h).

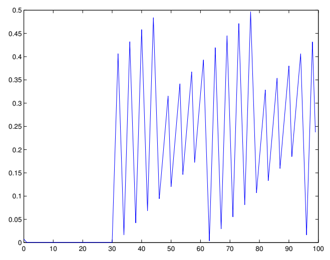
We apply the detection algorithm on the delay and throughput time series in Figure. 4.2 The results are that abrupt changes are detected at simulation time 31s for middle and bottom rows on both delay and throughput data, but not detected from the top row due to the large inter-packet interval that makes it easy for normal nodes' packets to get through even with the presence of a jammer.

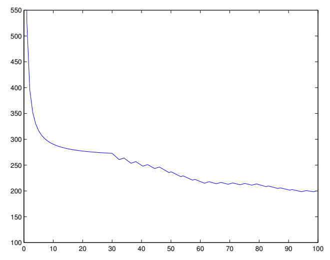
We also apply the detection algorithm on the other two time series derived from the SINR measurements as shown in Figure. 4.3. Left column shows the signal duration and right column shows the packet inter-frame space. The detection results are: all abrupt changes are detected from the left column (signal duration) even for the top row, which is not obvious to observe from delay and throughput series in Figure. 4.2, also not obvious from the received packet IFS in Figure. 4.3(b).

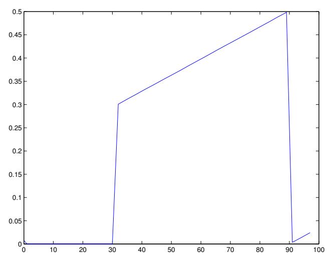
4.5.3. Impact of Traffic Dynamics. The second objective of the simulation study is to evaluate the detection algorithm in a multiuser environment and examine how the detection performance will be impacted by the dynamics of the network traffic. We compare


(a) Delay


(b) Throughput


(c) Delay


(d) Throughput


(e) Delay

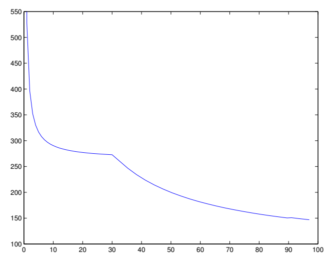
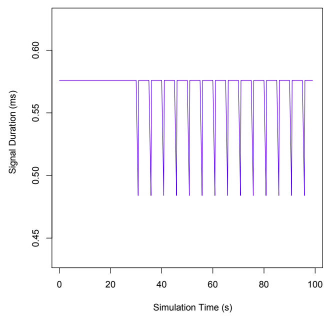
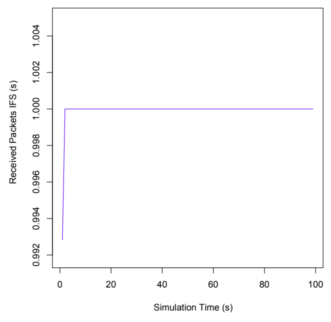

(f) Throughput

Figure 4.2. Top row: jamming packet duration 0.0005s, inter-packet interval 1s; Middle row: jamming packet duration 0.8s, inter-packet interval 5ms; bottom row: jamming packet duration 1.5s, inter-packet interval 5ms.

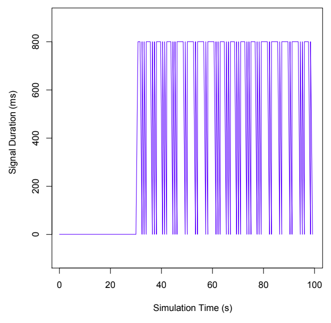
an attack scenario and a non-attack scenario. For the attack scenario in Figure. 4.5(a), we used jamming packet duration 0.8s, and observe the effect of jamming on three flows, in



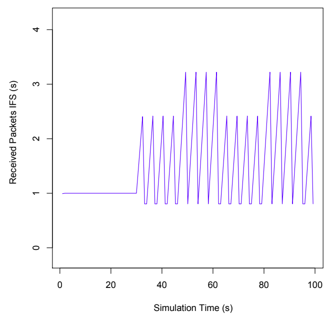
(a) Signal Duration



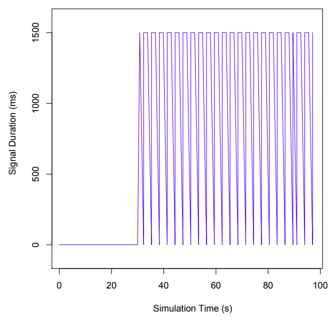
(b) Received Packets IFS



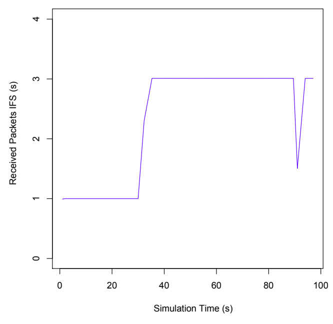
(c) Signal Duration



(d) Received Packets IFS



(e) Signal Duration



(f) Received Packets IFS

Figure 4.3. Jamming packet duration 0.0005s (top row), 0.8s (middle row), and 1.5s (bottom row).

which either the transmitter, the receiver, or neither is close to the jammer. The network measurements all look similar to the series in Figure. 4.2. However the three flows are affected to different degrees as their distances to the jammer varies. Table 4.1 shows the delay and packet delivery ratio (PDR) of the three flows. Due to the CSMA effect, flows $0 \rightarrow 1$ and $2 \rightarrow 3$ are more severely impacted than flow $4 \rightarrow 5$.

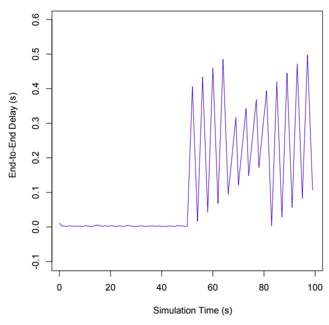
Table 4.1. Delay and packet delivery ratio (PDR) of the three flows.

	$0 \rightarrow 1$	$2 \rightarrow 3$	$4 \rightarrow 5$
Delay$Mean (ms)	99.35	99.26	0.41
Delay$Jitter (ms)	100.28	100.41	0.028
PDR	84%	84%	100%

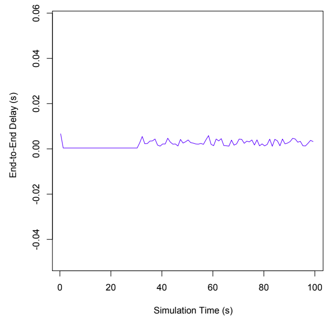
For the non-attack scenario in Figure. 4.5(b), there are more flows joining the network but no jammers. In a multiple flow scenario, it is possible that adding another flow will degrade the performance of other flows because of the contention of channel. It is important that the detection algorithm can tell the difference between the performance drop due to congestion in non-attack scenarios from that due to jamming attacks.In order to distinguish the attack case in Figure. 4.5(a) from the benign case in Figure. 4.5(b), we perform change point detection on the end-to-end delay time series, SINR duration of the received signal, and intervals of the received good packets. The three time series are aligned in time (or synchronized). If a change point is detected on all three time series at times close to each other, then detection is positive.

For Figure. 4.5(a) and (b), the detection algorithm is applied to the three time series shown in Figure. 4.4.

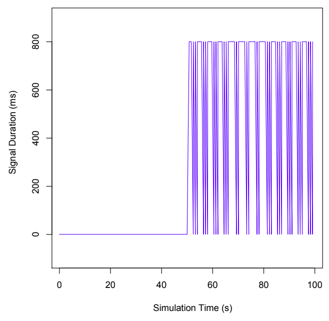
Detection result is summarized as follows: All but Figure. 4.4(d) reported positive from the change point detection algorithm. For the positive ones, changes are detected at simulation time 51s. In the attack case, detection of abrupt changes in delay is coupled with abrupt changes in signal duration and packet IFS; in the benign case, detection is positive



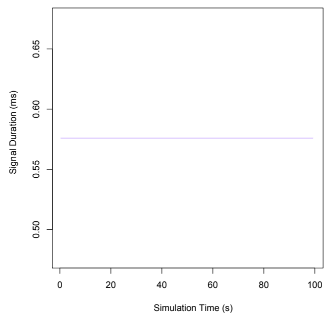
(a) End-to-End Delay



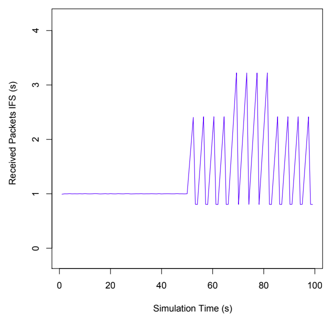
(b) End-to-End Delay



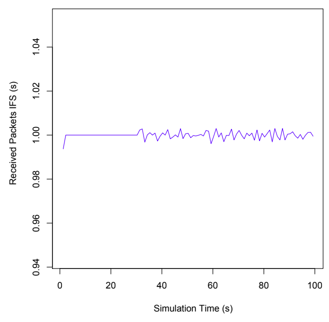
(c) Signal Duration



(d) Signal Duration



(e) Received Packets IFS



(f) Received Packets IFS

Figure 4.4. Left column: attack case in Figure. 4.5(a); Right column: benign case in Figure. 4.5(b).

for the delay and packet IFS but negative for the signal duration; in addition, the anomaly score measured by the difference between the detection statistics and detection threshold is not significant. The implication of this study is that if a jammer uses the packet size and interval the same as a normal node, we will be able to detect the slight changes brought by it, but won't be able to conclude it is a jammer since it behaves like a normal node.

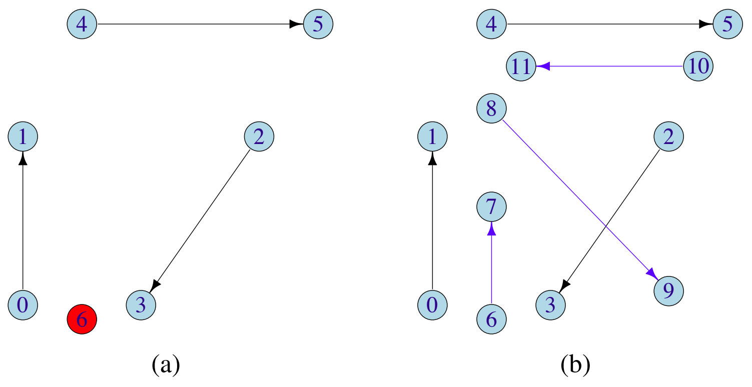


Figure 4.5. (a)Three flows with a jammer (node 6). (b) No jammer.

5. LOCALIZATION

5.1. BACKGROUND

In sensor networks and wireless networks, sometimes we need to locate the source of transmission based on the received signal strength. In sensor networks, sometimes due to the cost of precision positioning, for large-scale sensor networks, it is common to have only a small number of nodes accurately positioned, and the rest of nodes approximately deployed. The small number of nodes, whose exact locations are known, will serve as anchors to help determine the locations of other nodes. This process is referred to as *localization*. Sensor node localization is important because the scheduling and routing functions of a sensor network will benefit from knowing the locations of all sensor nodes. Sensor networks can also be used to track an object based on the received signals from the object. A similar localization problem arises in wireless networks when there is a jamming attack and all the nodes that received the signal from the jammer can collectively determine the position of the jammer. Once the jammer is precisely located, corresponding mitigation and defense strategies can be developed. In this paper, we address the common algorithm aspect in sensor/target/jammer localization problems. We refer to the node whose location is to be determined as the unknown node. The unknown node could be another sensor node, an object being monitored, or a jammer to be located.

We present a new machine learning-based methodology for the general localization problem in wireless networks. Unlike the previous work that uses a signal propagation model and received signal strengths as inputs to determine the target location, the proposed algorithm does not rely on any pre-assumed signal propagation model, and hence is named *Model-Free Localization (MFL)*. We first address the localization problem when the transmission power of the unknown node is known, then we address the localization problem when the transmission power of the unknown node is not known.

The proposed machine-learning algorithm MFL learns from the instances that are correctly labeled. In the target localization problem, the labeled instances would be the anchor nodes whose exact positions are known. Therefore, the localization performance improves as the number of anchors increases. However, even when there are very few number of anchor nodes (it can be as few as only three anchors), MFL works significantly better than the Support Vector Machine (SVM)-based algorithms [38, 56]. MFL is well-suited in a large sensor network where there are only a few anchor nodes available. MFL is also highly efficient for a network where there are a large number of targets whose positions need to be determined. This is because the machine learning algorithm has an estimation step and a prediction step, and estimation has higher complexity than prediction, but estimation does not need to be repeated for each target. Once a relationship has been estimated, it can be used for all targets, and only the prediction step is repeated for each target.

The article is organized as follows. In section 5.2, we survey major methodologies in localization and representative works. In section 5.3, we provide a formal statement of the problem. In section 5.4, we present the details of the proposed algorithm MFL. We first address the localization problem with the target node's transmission power known in section 5.4, and then we address the more challenging problem of localizing a node with unknown transmission power in section 5.5. In sections 5.6 and 5.7, we use extensive simulation study to evaluate the performance of the algorithms.

5.2. RELATED WORK

There are a variety of localization schemes in the literature. Some use graph connectivity information, such as hop counts [2, 56], and most use the received signals [5]. These schemes can be broadly divided into range-based and range-free schemes [22].

The range-based schemes first estimate pairwise distances between the target and the anchors, by using information derived from the received packets or signals, then the estimated distances are used to recover the location of the target in the second step. Broadly speaking, the range-based schemes also include the algorithms that use time of arrival ([42]), time difference of arrival ([45]), and angle of arrival ([40]), etc. to derive the distance information. In this paper, we focus on the algorithms that use the received signal strengths for ranging purpose.

The ranging step typically uses a pre-assumed signal propagation model, and uses sample data for the calibration of parameters in the model. However, difficulty arises in choosing the signal propagation model. The relationship between the received signal strength and the distance between the sender and the receiver is very noisy across the network. This is due to the impairment of signals caused by multi-path propagation, environment white noise, as well as device variability, etc. [7, 38]. If the assumed model structure is wrong, calibration of model parameters can not correct it. Since it is a two-step procedure, errors introduced in the first step will be carried into the second step. Therefore the methods based on existing ranging schemes inherit these inaccuracies.

The range-free schemes avoid the difficulty in signal modeling by bypassing the direct estimation of distance. Instead, they define a set of overlapping regions in the area, and use signal information to estimate which regions the source of the signal resides in. It involves solving a classification problem for each predefined region. Each classification problem is to decide whether the source resides in a particular region. The location of unknown target is then computed as the mean of the centroids of the regions that contain the source [38]. This approach can result in very large localization error.

Other range-free algorithms that use regression analysis for direct estimation of the target location from signal strengths have also occurred in the literature [57]. However, they attempted to build one regression model for the entire network based on the signal strength matrix. These approaches suffer from low accuracy due to the nonuniform impairment of signals across the network, therefore model fitting of any model is intrinsically inaccurate.

The classification method based on support vector machines can work well only if there are enough number of anchor nodes to provide enough number of support vectors [46]. It shows excellent performance if the anchor nodes are densely and uniformly deployed on a grid structure. Challenges arise when anchors are sparse and their positions are not at grid points. The range-based methods that use pre-assumed signal models work well only if the signal models are accurate. Performance degradation is observed when the relationship between signal strength and distance is noisy. Despite so many algorithms in the literature, these challenges have not been sufficiently addressed. In this paper, we will develop a new machine learning approach to address these challenges.

In addition to the absolute positioning techniques mentioned above, there are also relative localization schemes. For instance, in [21], raw satellite measurements from a network of receivers are used to derive relative location from each other. For indoor localization, a big challenge is the multi-path effect. While most work considers the multi-path effect a “curse” in signal modeling, there is also innovative work that explores the power of multi-path propagation to achieve good accuracy for indoor positioning [65]. There is also recent work in indoor localization that takes advantage of crowd sourcing, which traces user walking trajectories and uses a large number of user contributed trajectories for indoor localization [33]. While we recognize these recent trends that explore new network configuration and new information to advance localization technology, we focus on the algorithmic aspect of the fundamental methodology. Among the many contributions in localization, the most related work is the support vector-based learning algorithm in [38], which has the same problem definition and uses the same input, and therefore serves as a valid comparison study.

5.3. LOCALIZATION PROBLEM

In the context of localization, the nodes whose coordinates are known are called *anchor nodes*, and the node whose coordinates are to be determined is called *target nodes*. The target node could be another sensor node, a jammer, or an object whose radio signals are received by the anchor nodes. The anchor nodes collectively determine the location of the target node based on the signals they receive from the target node. In case there are multiple target nodes, the anchor nodes need to be able to distinguish whose signal it is in order to locate the target nodes.

The localization process requires each anchor node collect the received signals from all other anchor nodes. This can be done by having all anchor nodes beacon a signal periodically so that others can record it. Once the target node starts to transmit, anchor nodes record the signal and use this signal to determine the location of the unknown transmitter. We can generalize the problem to have multiple target nodes as follows:

For a sensor or wireless network with n ($n \geq 3$) anchor nodes and m target nodes, given the coordinates of the n anchor nodes, and the received signal strength vectors $\{S_j | j = 1, \ldots, n\}$ with each vector including the received signals from all other anchors and the target nodes, determine the coordinates of the m target nodes. The signal strength vector S_j of an anchor node j contains at least one sample from each of the other nodes in the network. If the signal variability is high or the noise level is high, multiple samples from each node, especially from each of the anchor nodes, are necessary to get accurate results. As shown next, the machine learning algorithms have an estimation procedure and a prediction procedure. The samples that an anchor node receives from other anchor nodes will be used as training data to train the machine learning algorithm, and therefore more training data will yield more accurate estimation; the samples that an anchor node receives from a target node will be used to determine the location of the target node. If the signal variability is high, the mean of the multiple samples from a target node will be used for prediction.

5.4. ALGORITHM I: LOCALIZATION WITH KNOWN TARGET TRANSMISSION POWER

We first address the localization problem when the target node's transmission power is known. In Part II, we will address the more challenging version with unknown transmission power.In this section, we present a two-step localization algorithm. The first step takes the received signal strength vector as input and estimates the distances between the target node and the anchor nodes, and the second step uses the resulting distances from the first step as input and estimate the accurate coordinates of the target node. The first step is called *Ranging*, and hence the proposed algorithm belongs to the range-based category.

The Algorithm. For each pair of transmitter and receiver, the relationship between the received signal strength and the distance between the transmitter and receiver is represented as a signal model. There are several signal models developed so far, varying in complexity and accuracy. For instance, the simplified path loss model, which is deterministic and takes the form of a polynomial kernel:

$$P_r(d) = P_t K \left(\frac{d}{d_0}\right)^{-\gamma}. \tag{5.1}$$

where the constant γ is the path-loss exponent. The typical values of γ is between 2 and 6, varying with the antenna orientation and environment factors (indoor vs.outdoor, obstructed vs. open space etc.). γ is often determined empirically. d is the actual distance between the sender and the receiver, and d_0 is a reference distance. It is assumed when $d \leq d_0$, there is no loss in signal power.The most widely used signal propagation model is the log-normal shadowing model, usually expressed in dB form:

$$\frac{P_r}{P_t}(dB) = 10\log_{10} K - 10\gamma \log_{10}\frac{d}{d_0} + \mathcal{N}(0, \sigma_{N_{dB}}). \tag{5.2}$$

where $\mathcal{N}(0, \sigma_{N_{dB}})$ is a Gaussian random variable with zero mean and $\sigma^2_{N_{dB}}$ variance that models the random variation of the received signal strength.

The literature has also seen the use of a different fading channel model, in which the signal energy decays exponentially with respect to distance square [38]. This model takes the form of a Gaussian kernel with additive random noise:

$$P_r(d) = P_t e^{-\frac{d^2}{\sigma_f}} + \mathcal{N}(0, \tau), \tag{5.3}$$

where $\mathcal{N}(0, \tau)$ denotes an independently generated normal random variable with zero mean and standard deviation τ.

Had we known the exact signal propagation model, the distance can be conveniently computed from the inverse function of the signal model. However, the signal propagation model is only an approximate model, and can be highly inaccurate sometimes when the environment and devices vary. First, the exact signal propagation model is usually unknown, which means not only we do not know the parameters in the models, we may not even know which model is suitable for describing the signals. Second, unless the wireless channel condition is exactly the same for the entire network, we cannot have a universal signal model applicable to the entire network. In reality, the channel condition can be different at different locations within a network due to impairments from multi-path effects, obstruction, and ambient noise interference, etc. Third, transmitters and receivers can add device-level variability and make any pre-assumed model inaccurate. In the previous range-based localization algorithms, usually a signal model is selected from one of the known models, and signal samples are used only to calibrate the parameters. The problem is that if the type of function is not selected correctly, tuning the constant parameters of the model cannot fix it. It is pointed out that unless the problems with the signal modeling have been fixed, localization methods based on ranging inherit these inaccuracies[38].

In this paper, we will avoid model selection by human. The algorithm is named *Model Free Localization* mainly because the ranging step does not use any pre-assumed signal model. We show that localization methods based on ranging can achieve fairly good performance provided that the issues with signal modeling are sufficiently addressed.

In particular, we address the first issue, i.e., the unknown model issue, by using a machine learning method, which results in the suitable type of model and the corresponding parameters of the model. To address the second and the third issues, i.e., the channel and device variability issues, we use a different strategy from previous work— instead of trying to build a global signal model, we try to infer a signal model for each anchor node.

There are n anchor nodes in the network, so there will be a total of n estimation problems to solve. For each anchor node, we use the received signal strength at this node from other anchors and the distance from this node to other anchors as training data. The predictor variables are functions of the received signal strength, and the response variable or outcome variable is a function of the distance.

$$Y(\log d) \sim X(\log RSS, \log(-\log RSS)). \tag{5.4}$$

In (5.4), Y is the outcome variable, and X is the collection of predictor variables. X can have several components. RSS is the received signal strength. $RSS \in (0, 1]$ is normalized by the transmitter power. The i^{th} entry of the training data contains data for the i^{th} anchor node. The general form in (5.4) will allow each anchor node to have a different set of regression coefficients that account for different signal models in reality.

Once the coefficients have been decided, the distance from a new target node to anchor node i can be calculated as follows:

$$\log d_i = \beta_{i,0} + \beta_{i,1} \log RSS + \beta_{i,2} \log(-\log RSS). \tag{5.5}$$

The coefficient for the $\log RSS$ term is associated with the polynomial path-loss model, and the coefficient for the $\log(-\log RSS)$ term is associated with the exponential model. If the real signal model follows the polynomial path-loss model, then the $\log RSS$ term will be dominant, and vice versa. If the real signal model follows neither one but decays faster than a polynomial model and slower than an exponential model, then the weights of both terms will be comparable.Using this model, we can predict the distance without assuming a specific radio signal model, therefore the problems associated with signal modeling will not have an impact on the accuracy of distance prediction. Moreover, since the regression coefficients are estimated per anchor node, it allows transmitter and receiver as well as environment variability without assuming a generic model for the entire network.

It is common that the sensitivity of the receivers across different radio chips is different. If one anchor node has high fluctuation in received signal strength even when all other parameters are kept constant, the highly noisy data generated in the training set will only affect the distance prediction to this anchor node alone and has no impact on others. However, if the transmitters also have high variability, which means if a transmitter is configured to transmit at power level P_t, it will transmit at a power level very close to P_t but not necessarily exactly equal to P_t [35]. Higher variability at transmitter will pollute the training data of all receivers. In this case, the only solution to this problem is for each receiver to record multiple samples from a single transmitter.

Examples.The ranging algorithm based on (5.4) - (5.5) results in fairly good range estimation. We ran the algorithm over three sets of data: (1) data generated by the log-normal shadowing model, (2) data generated by the exponential decay model, and (3) mixed data from multiple models. Nodes are deployed on a square area of 10×10 units.

The algorithm performs over all three sets of data without knowing what model is used to generate the data, and the estimation errors are shown in Table 5.1. The three columns represent the mean training error, the mean test error, and the overall error, which is the mean estimation error by including both training errors and test errors.

Another view of the estimation error is presented in Table 5.2, which shows the percentage of estimations that have estimation error larger than 1 unit. It is noticeable that the performance does not degrade with the mixing of data, i.e., when signals for different nodes follow different propagation models, The algorithm can still accurately estimate the distance based on the received signal strength while being oblivious to the signal propagation models.

The performance degradation is mainly from large noise in data itself, and the degradation is significant even when all nodes follow the same signal propagation model. As the variance in noise increases, the estimation error reasonably increases.

For instance, for the data generated by the exponential model, the standard deviation $\tau = 0.02$ is quite large for a random variable between 0 and e^{-1} with most values concentrated near 0, and the test error can be as large as 1.66 on a square of 10×10 units. The performance improves as the number of anchor nodes is increased. For instance, if we use

9 anchor nodes instead of 7, the test error decreases from 1.66 to 0.95 for the same noise variance.If the distance data were accurate with no errors, three anchor nodes are enough to get the unique and exact coordinates of the target node.

However the distance data is only an estimate with non-zero errors. We will instead use the nonlinear Least Squares Estimation (LSE) to find the target node coordinates.The nonlinear Least Squares Estimation is based on the following model:where $X = \{x_1, \ldots, x_n\}$ is the input vector and $Y = \{y_1, \ldots, y_n\}$ is the observation vector, θ is the unknown $p \times 1$ vector parameter, $g(x_i, \theta)$ is a known continuous nonlinear function.

$$y_i = g(x_i, \theta) + \epsilon_i, \qquad i = 1, \ldots, n, \tag{5.6}$$

Noise ϵ is assumed to be i.i.d. with zero mean and unknown variance $\sigma_\epsilon^2 > 0$.The LSE computes a value $\hat{\theta}$ that minimizes the sum of squared errors:

$$\hat{\theta} = arg \min_{\theta} \left(\sum_{i=1}^{n} (y_i - g(x_i, \theta))^2 \right) \tag{5.7}$$

In the sensor node localization problem, x_i is the coordinates of the i^{th} anchor node and has dimension 2×1, y_i is the observed distance between the target node and the i^{th} anchor node, i.e., $y_i = d_i$, θ is the coordinates of the target node and has dimension 2×1. The function $g(x_i, \theta)$ is the distance function given by

$$g(x_i, \theta) = \sqrt{(x_{i,1} - \theta_1)^2 + (x_{i,2} - \theta_2)^2} \tag{5.8}$$

A target node's coordinates are computed by solving a nonlinear Least Square Error problem:

$$\sum_{i=1}^{n} \left(d_i - \sqrt{(x_{i,1} - \theta_1)^2 + (x_{i,2} - \theta_2)^2} \right)^2 \tag{5.9}$$

We compare our LSE based estimation with a straightforward regression based estimation for computing the target node coordinates from given distance values. The regression method uses a node's location as the response variable and uses the distances from anchor nodes to this node as predictor variables:

$$Y(\mathcal{P}) \sim X(d_1, d_2, \ldots, d_n) \tag{5.10}$$

Since there are two coordinates, it takes two regression analyses to get the two coordinates separately, while the LSE based method only needs to solve the optimization problem once. We assume the distance data are corrupted by a random noise. The regression based method has overall error 1.6973, training error 0, and test error 2.0574 on a 10×10 square region, regardless of the noise level. LSE based estimation improves significantly as the variance in noise decreases, as shown in Figure 5.1.

5.5. ALGORITHM II: LOCALIZATION WITH UNKNOWN TARGET TRANSMISSION POWER

When the target node's transmission power is unknown, using the ranging procedure in Section 5.4 won't be able to estimate the distances from the target to the anchor nodes since the regression model uses RSS as predictor variable, which is the ratio of the received signal strength to the transmission power. Although we can estimate the function (5.4) from the anchor nodes, we won't be able to use target node's RSS in (5.5) to determine the distance. We will develop a new regression model that estimates the relationship between distance d and the transmission power and received power. A new regression model is built

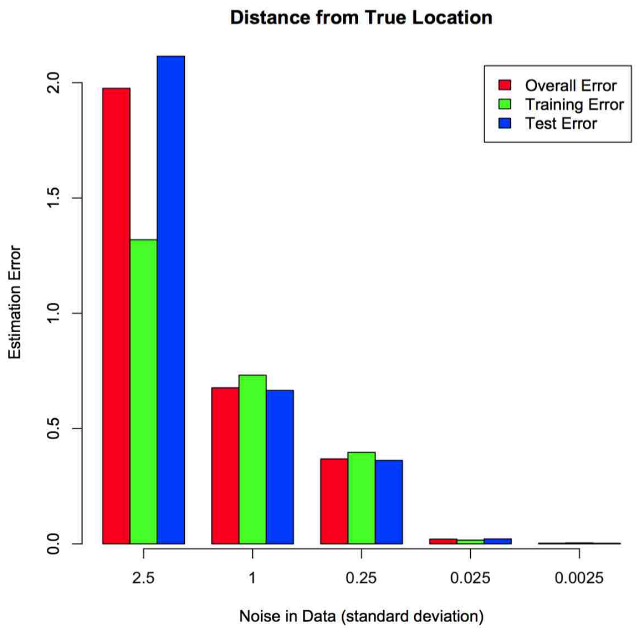


Figure 5.1. Location estimation error vs. noise in input.

for each anchor node as receiver:

$$Y(\log d) \sim X(\log \frac{Pr}{Pt}, \log(-\log \frac{Pr}{Pt})), \tag{5.11}$$

Using the anchor nodes's signal vectors and the locations, we can estimate the function coefficients $\beta_{i,0}$, $\beta_{i,1}$ and $\beta_{i,2}$ in (5.12):

$$\log d_i = \beta_{i,0} + \beta_{i,1} \log \frac{Pr}{Pt} + \beta_{i,2} \log(-\log \frac{Pr}{Pt}). \tag{5.12}$$

However,we will not directly use (5.12) to estimate the distances from the target node to the anchor nodes since the transmission power P_t of the target node is unknown. Instead, we will use the transmission power P_t as a variable and use the nonlinear LSE to find the target node's transmission power and coordinates. First, the distance d_i to anchor node i is expressed as a function of transmission power P_t.

$$d_i = e^{\beta_{i,0}+\beta_{i,1}(\log P_r - \log P_t)+\beta_{i,2}\log(\log P_t - \log P_r)}. \tag{5.13}$$

Then nonlinear LSE estimation is used to estimate P_t and the coordinates (θ_1, θ_2) of the target node as follows:

$$\sum_{i=1}^{n} \left(d_i(P_t) - \sqrt{(x_{i,1} - \theta_1)^2 + (x_{i,2} - \theta_2)^2} \right)^2 \tag{5.14}$$

where $d_i(P_t)$ is a function of P_t as indicated in (5.13).If there is only one anchor node, with known target transmission power, using (**??**) can estimate the distance from this anchor node to the target node. However with unknown target transmission power, using (5.12) can yield infinite number of pairs of (P_t, d_i). In other words, we have to know one of them to decide the other. One equation won't be able to solve two variables.

How do we find the true values of $(P_t, \theta_1, \theta_2)$? The problem is addressed within the procedure of LSE by using multiple anchor nodes ($i = 1, \ldots, n$) and by minimizing the squared error. We know the transmission power is at least as large as the maximum value of the received signal power. Therefore we set the initial value of P_t to be the maximum received signal power: $P_t = \max_i P_{r,i}$, and set the initial position of the target node to be the center of the network. After the first iteration, we use the estimated $(P_t, \theta_1, \theta_2)$ to be the initial values of the next iteration and iteratively improve the results until they converge. The stopping criterion is when the differences between the values from two consecutive iterations are within a preset threshold ϵ:

$$\mid P_t^{(k)} - P_t^{(k-1)} \mid \leq \epsilon, \text{ and} \tag{5.15}$$

$$| \theta_1^{(k)} - \theta_1^{(k-1)} | \leq \epsilon, \text{ and} \tag{5.16}$$

$$| \theta_2^{(k)} - \theta_2^{(k-1)} | \leq \epsilon \tag{5.17}$$

In the experiment, we observed that with $\epsilon = 10E-2$, convergence is achieved within $2 \sim 3$ iterations.

5.6. EXPERIMENTAL RESULTS FOR ALGORITHM I: WITH KNOWN TARGET TRANSMISSION POWER

Comparison with the Kernel Method : We compare our algorithm with kernel based algorithms. Kernel based algorithms utilize kernel functions to decide the location of the target node. There are plenty of choices for kernel functions. According to [38], the second-tier linear kernel model performs much better than the first-tier model. Therefore in this study we compare our algorithm with the support vector machine method with a second-tier linear kernel.

5.6.1. Simulation Set-up. We compare the two algorithms as the noise level, the number of anchor nodes, and the regularity of anchor nodes change. Due to the various configurations required in this study and the limitation of the real-world data, we use synthetic data for this simulation.

Target nodes and anchor nodes are deployed on a 10×10 square region. The signal data are generated from a variety of path loss models plus random noise. However, neither of the two algorithms is provided with the knowledge of the models or the procedure that generated the signals.

5.6.2. Impact of the Regularity of Anchors. Figures 5.2 - 5.3 show the location estimation results from the proposed algorithm and the SVM algorithm. Lines connect the estimate positions (in blue dots) to their true positions (in black unfilled circles). Anchor nodes are shown in triangles. Figure 5.2 shows the results when the anchor nodes are at grid positions, and Figure 5.3 shows the results when the anchor nodes are randomly deployed. We can see that SVM performs better when the anchor nodes are uniformly deployed than randomly deployed. The average estimation error for SVM is 1.502 and 1.904 for Figure 5.2 and 5.3 respectively. However the proposed range-based algorithm MFL does not show much difference whether the anchor nodes are uniformly deployed or not. The average estimation error is only 0.0934 for Figure 5.2 and 0.0972 for 5.3. Results are for noise level $\sigma = 0.05$, and 9 anchor nodes.

5.6.3. Impact of the Sparsity of Anchors. Next, we show the effects of the number of anchor nodes and the noise level on location estimation.The proposed algorithm is labeled as "MFL". Since the SVM algorithm performs better when anchor nodes are on grid positions, we will deploy anchor nodes on grid positions for the following simulations. The number of anchor nodes ranges from $2 \times 2 = 4$ to $12 \times 12 = 144$.

In SVM algorithm, $K \times K$ regions are used for classification, so there will be K^2 classification problems. In this simulation, two levels of granularity are used: $K = 5$ and $K = 20$. As K increases, the estimation accuracy of SVM algorithm statistically improves, although only marginally, at the cost of longer running time.

The mean estimation errors are plotted in Figure 5.4(a). The SVM algorithm performs poorly when the anchors are sparse, but MFL can do much better than SVM and perform fairly well even when only four anchor nodes are used.

5.6.4. Impact of Signal Noise. We compared the two algorithms when the noise level changes. The range for σ goes from 0.005 to 0.05. It was believed that noisy data are bad for range-based algorithms, and that kernel-based algorithms were designed to overcome the issue of a noisy relationship between signal strength and distance. However, SVM classification is still based on signal strength and it only turns a continuous distance estimation into a discrete classification problem. When data are noisy, the inaccuracy is inherited into location classification. Figure 5.4(b) shows the location estimation error. $4 \times 4 = 16$ anchor nodes are used in this simulation, and anchor nodes are all at grid positions. The SVM algorithm used two levels of granularity, with $K = 5$ and $K = 10$.

In [38] it is reported that the level of accuracy with the kernel-based approach is on the order of one third of the distance between the sensor nodes. The simulation results in this paper show that the mean of the distances in our deployment is 4.86, and the mean estimation error is 1.65. The level of accuracy observed here is consistent with what is reported in [38]. This is when the anchor nodes are densely and uniformly distributed at grid positions. The performance degrades when anchor nodes are fewer and more irregular. However, the proposed algorithm MFL performs well for both dense and sparse scenarios, and for both uniform and random deployment. The average error reported by MFL is only 0.058 on a 10×10 square region.

5.7. EXPERIMENTAL RESULTS FOR ALGORITHM II: WITH UNKNOWN TARGET TRANSMISSION POWER

In this experiment, we use jammer localization to test the algorithm since usually we do not know the transmission power of the jammer. Location error is measured as distance to the true location, power level error is measured as percentage of the true power level.

The experiment is done by deploying a total of 19 anchor nodes and one jammer node in the square region of $1000m \times 1000m$. The jammer node varies its position and transmission power: position #1= (-150,50), and position #2= (-130,30). Transmission power P_t = 16.0206, 20 and 25 dB. There are a total of 6 experiments. In all 6 experiments, we set the initial position to be the center of the region. Does the initial position have a large impact on the converged result? -Not necessarily. We put the initial position to be the true position, and the converged results are the same as those with initial position=(0,0).

Table 5.1. The distance estimation error.

Data	Overall	Training Error	Test Error
lognormal, $\sigma = 0.05$	0.0686	0.0486	0.1153
exponential, $\tau = 0.005$	0.5845	0.2265	1.4199
mixed	0.2167	0.1512	0.3696
lognormal, $\sigma = 0.5$	0.5783	0.4291	0.9264
exponential, $\tau = 0.02$	0.7437	0.3501	1.6620
mixed	0.6370	0.5380	0.8681

Table 5.2. The percentage of estimations with error > 1 unit.

Data	Overall	Training Data	Test Data
lognormal, $\sigma = 0.05$	0	0	0
exponential, $\tau = 0.005$	17.1%	1.4%	15.7%
mixed	5.7%	1.4%	4.3%
lognormal, $\sigma = 0.5$	20.0%	7.1%	12.9%
exponential, $\tau = 0.02$	24.3%	8.6%	15.7%
mixed	21.4%	10.0%	11.4%

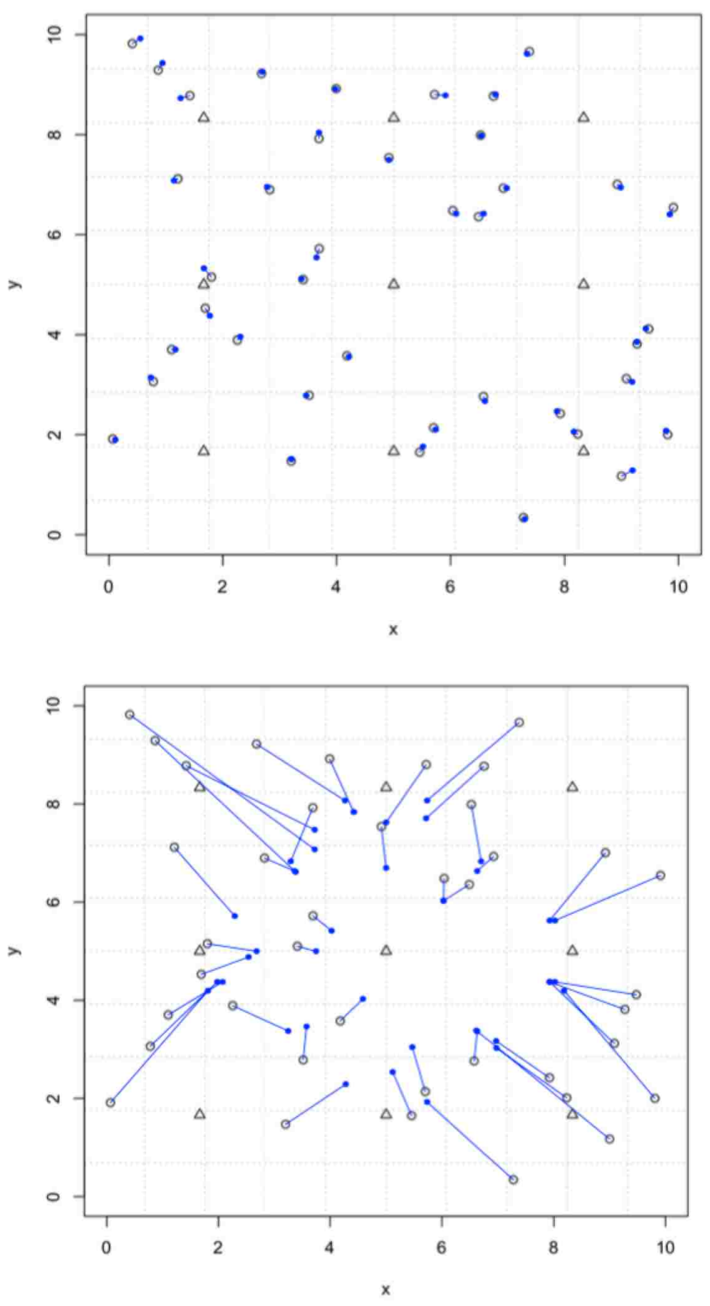


Figure 5.2. With anchor nodes at grid positions, (a) estimated positions by the proposed algorithm MFL, and (b) estimated positions by the SVM algorithm.

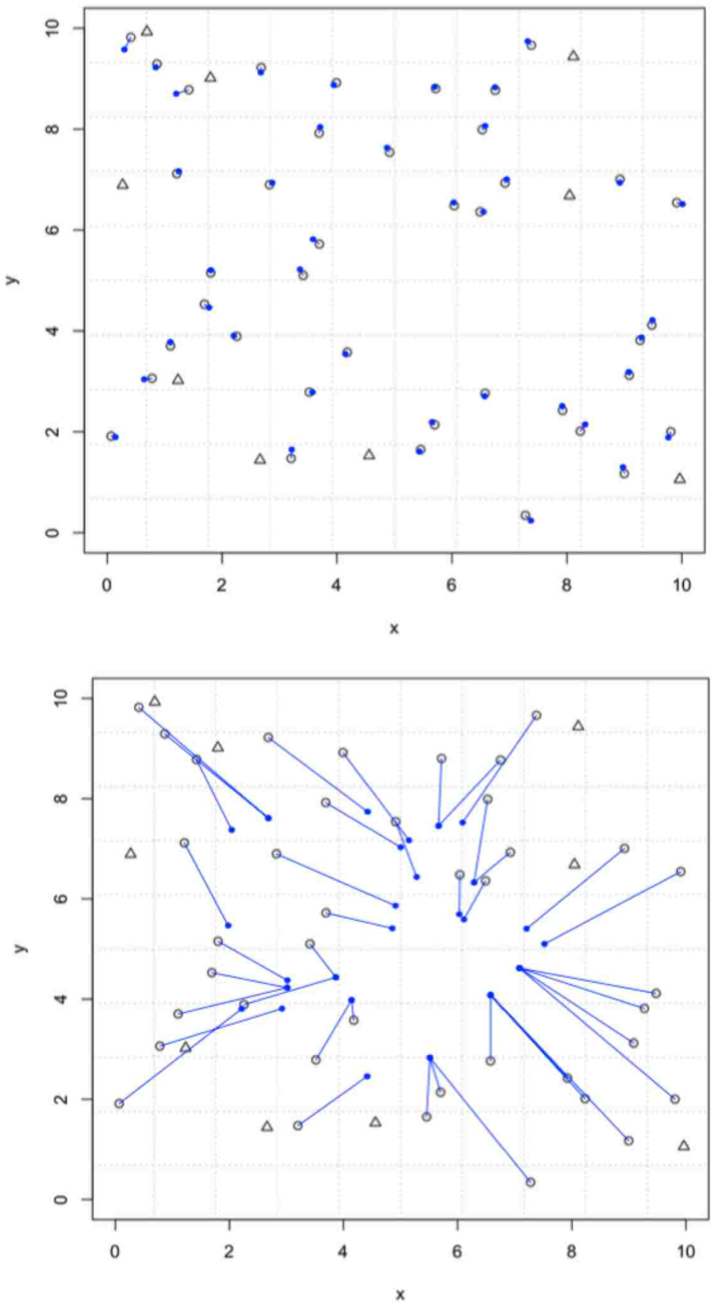
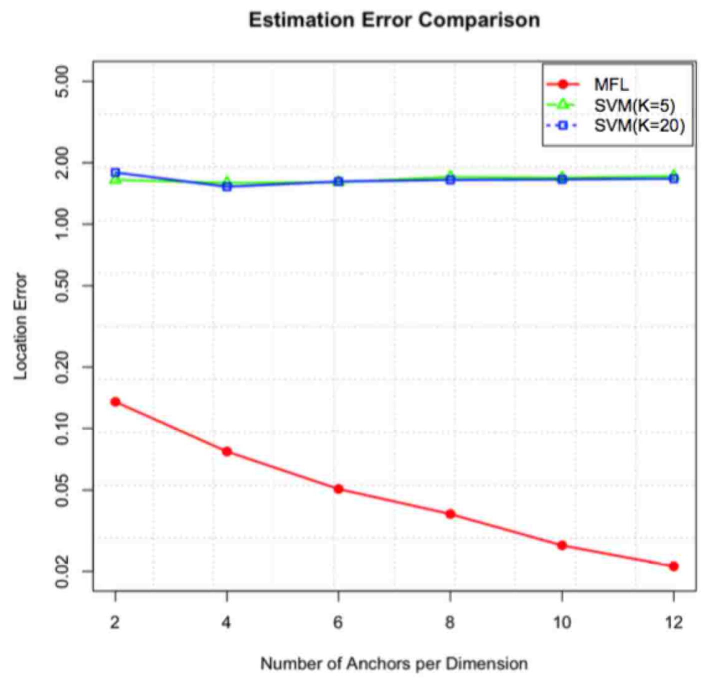


Figure 5.3. With anchor nodes at grid positions, (a) estimated positions by the proposed algorithm MFL, and (b) estimated positions by the SVM algorithm.



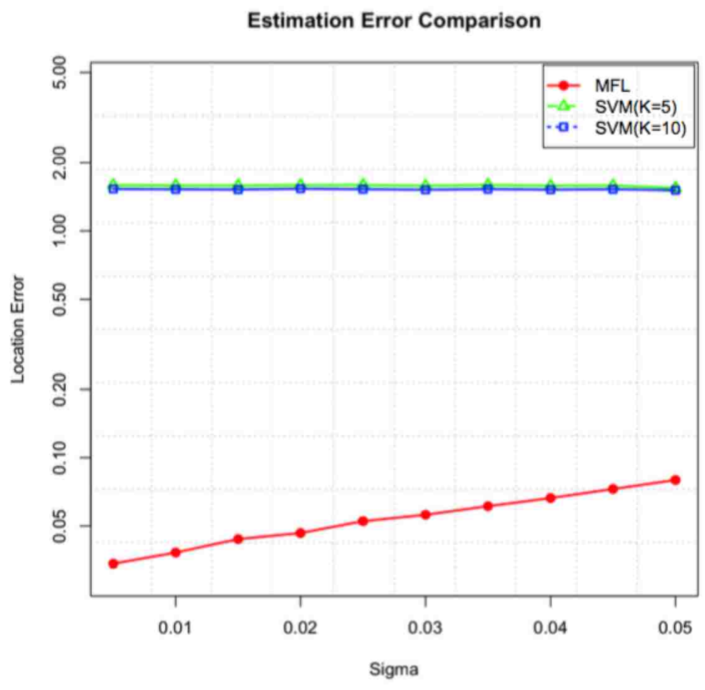


Figure 5.4. (a) The impact of anchor nodes sparsity on the estimation error, (b) the impact of data noise on the estimation error.

6. CONCLUSIONS

This paper addresses the in-band wormhole detection problem, the MAC layer misbehavior detection problem and physical layer jamming detection in wireless ad hoc network. I formulate the detection problem as a change point detection problem in a time series, and propose a new sequential change point detection algorithm, called SW-CLT. The SW-CLT algorithm outperforms the widely used CUSUM algorithm in the sense it adjust its detection threshold according to the variance in the data instead of using a fixed preset threshold, and is more suitable to detect changes in dynamic data where the underlying distribution is unknown and unpredictable.

I Also proposed a machine learning algorithm for localization problems in wireless sensor networks. The algorithm does not assume any signal model; instead it uses sample data to estimate the function and the parameters of the model all at the same time. Different from previous works that use regression tools to tune the parameters of a pre-assumed signal model, our algorithm is model-free, and therefore does not inherit the inaccuracies associated with a particular signal model when the relationship between signal strength and distance deviates from the model. The proposed algorithm can perform fairly well under extreme cases when the anchor nodes are sparsely and irregularly deployed, which often present difficulty for other methods.

BIBLIOGRAPHY

[1] A. Aaroud, M.-A. El Houssaini, A. El Hore, and J. Ben-Othman. Real-time detection of mac layer misbehavior in mobile ad hoc networks. *Applied Computing and Informatics*, 2015.

[2] S. Afzal and H. Beigy. A localization algorithm for large scale mobile wireless sensor networks: a learning approach. *The Journal of Supercomputing*, 69(1):98–120, 2014.

[3] X. Ban, R. Sarkar, and J. Gao. Local connectivity tests to identify wormholes in wireless networks. In *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, page 13. ACM, 2011.

[4] D. Bansal, S. Sofat, and P. Kumar. Distributed cross layer approach for detecting multilayer attacks in wireless multi-hop networks. In *Computers & Informatics (ISCI), 2011 IEEE Symposium on*, pages 692–698. IEEE, 2011.

[5] X. Bao and J. Li. Mobile wireless localization through cooperation. *CoRR*, abs/1002.3602, 2010.

[6] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on selected areas in communications*, 18(3):535–547, 2000.

[7] N. BULUSU, J. HEIDEMANN, and D. ESTRIN. GPS-less low cost outdoor localization for very small devices. Technical Report Tech. Rep. 00-729, Computer Science Department, University of Southern California, 2000.

[8] L. Buttyï£¡n and I. Vajda. Statistical wormhole detection in sensor networks. In *In ESAS*, pages 128–141. Springer, 2005.

[9] A. A. Cardenas, S. Radosavac, and J. S. Baras. Detection and prevention of mac layer misbehavior in ad hoc networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 17–22. ACM, 2004.

[10] M. Çakiroğlu and A. T. Özcerit. Jamming detection mechanisms for wireless sensor networks. In *Proceedings of the 3rd International Conference on Scalable Information Systems*, InfoScale '08, pages 4:1–4:8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] M. X. Cheng, Y. Ling, and W. B. Wu. In-band wormhole detection in wireless ad hoc networks using change point detection method. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[12] M. X. Cheng, Y. Ling, and W. B. Wu. In-band wormhole detection in wireless *ad hoc* networks using change point detection method. In *IEEE ICC 2016*, pages 1–6, 2016.

[13] I. C. S. L. M. S. Committee et al. Wireless lan medium access control (mac) and physical layer (phy) specifications, 1997.

[14] D. Dong, M. Li, Y. Liu, X.-Y. Li, and X. Liao. Topological detection on wormholes in wireless ad hoc and sensor networks. *Networking, IEEE/ACM Transactions on*, 19(6):1787–1796, Dec 2011.

[15] M.-A. El Houssaini, A. Aaroud, A. El Hore, and J. Ben-Othman. Analysis and simulation of mac layer misbehavior in mobile ad-hoc networks. In *Codes, Cryptography and Communication Systems (WCCCS), 2014 5th Workshop on*, pages 50–54. IEEE, 2014.

[16] V. R. Giri and N. Jaggi. Mac layer misbehavior effectiveness and collective aggressive reaction approach. In *Sarnoff Symposium, 2010 IEEE*, pages 1–5. IEEE, 2010.

[17] L. Guang and C. Assi. On the resiliency of mobile ad hoc networks to mac layer misbehavior. In *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 160–167. ACM, 2005.

[18] V. Gupta, S. Krishnamurthy, and M. Faloutsos. Denial of service attacks at the mac layer in wireless ad hoc networks. In *MILCOM 2002. Proceedings*, volume 2, pages 1118–1123. IEEE, 2002.

[19] A. Hamieh, J. Ben-Othman, A. Gueroui, and F. Naït-Abdesselam. Detecting greedy behaviors by linear regression in wireless ad hoc networks. In *2009 IEEE International Conference on Communications*, pages 1–6. IEEE, 2009.

[20] T. Hayajneh, G. Almashaqbeh, and S. Ullah. A green approach for selfish misbehavior detection in 802.11-based wireless networks. *Mobile Networks and Applications*, 20(5):623–635, Oct. 2015.

[21] W. Hedgecock, M. Maroti, A. Ledeczi, P. Volgyesi, and R. Banalagay. Accurate real-time relative localization using single-frequency GPS. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pages 206–220, New York, NY, USA, 2014. ACM.

[22] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[23] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM*, volume 3, pages 1976–1986. IEEE, 2003.

[24] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *Journal on Selected Areas in Communications*, 24(2):370–380, 2006.

[25] I. Khalil, S. Bagchi, and N. B. Shroff. Liteworp: a lightweight countermeasure for the wormhole attack in multihop wireless networks. In *Dependable Systems and Networks (DSN)*, pages 612–621. IEEE, 2005.

[26] I. Khalil, S. Bagchi, and N. B. Shroff. Mobiworp: Mitigation of the wormhole attack in mobile multihop wireless networks. *Ad Hoc Networks*, 6(3):344–362, 2008.

[27] P. Kruus, D. Sterne, R. Gopaul, M. Heyman, B. Rivera, P. Budulas, B. Luu, T. Johnson, N. Ivanic, and G. Lawler. In-band wormholes and countermeasures in olsr networks. In *Securecomm and Workshops, 2006*, pages 1–11, Aug 2006.

[28] P. Kyasanur and N. H. Vaidya. Selfish mac layer misbehavior in wireless networks. *IEEE transactions on mobile computing*, 4(5):502–516, 2005.

[29] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1307–1315. IEEE, 2007.

[30] M. Li, S. Salinas, P. Li, J. Sun, and X. Huang. Mac-layer selfish misbehavior in ieee 802.11 ad hoc networks: Detection and defense. *IEEE Transactions on Mobile Computing*, 14(6):1203–1217, June 2015.

[31] Y. Lin and M. Li. Distributed detection of jamming and defense in wireless sensor networks. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pages 829–834, March 2009.

[32] G. Lorden. Procedures for reacting to a change in distribution. *Ann. Math. Statist.*, 42(6):1897–1908, 1971.

[33] C. Luo, H. Hong, and M. C. Chan. Piloc: A self-calibrating participatory indoor localization system. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, pages 143–154, Piscataway, NJ, USA, 2014. IEEE Press.

[34] N. Lyamin, A. Vinel, M. Jonsson, and J. Loo. Real-time detection of denial-of-service attacks in ieee 802.11p vehicular networks. *IEEE Communications Letters*, 18(1):110–113, January 2014.

[35] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An empirical characterization of radio signal strength variability in 3-d ieee 802.15.4 networks using monopole antennas. In *Third European Workshop in Wireless Sensor Networks (EWSN 2006)*, volume 3868, pages 326–341, 2006.

[36] R. Maheshwari, J. Gao, and S. R. Das. Detecting wormhole attacks in wireless networks using connectivity information. In *INFOCOM*, pages 107–115. IEEE, 2007.

[37] J. Ng, Z. Cai, and M. Yu. A new model-based method to detect radio jamming attack to wireless networks. In *2015 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2015.

[38] X. Nguyen, M. I. Jordan, and B. Sinopoli. A kernel-based learning approach to ad hoc sensor network localization. *ACM Trans. Sen. Netw.*, 1(1):134–152, Aug 2005.

[39] E. S. Page. Continous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[40] R. Peng and M. Sichitiu. Angle of arrival localization for wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, volume 1, pages 374–382, Sept 2006.

[41] S. Radosavac, A. A. Cárdenas, J. S. Baras, and G. V. Moustakides. Detecting ieee 802.11 mac layer misbehavior in ad hoc networks: Robust strategies against individual and colluding attackers. *Journal of Computer Security: Special Issue on Security of Ad Hoc and Sensor Networks*, 15(1):103–128, Jan. 2007.

[42] S. Ravindra and S. N. Jagadeesha. Time of arrival based localization in wireless sensor networks : A linear approach. *CoRR*, abs/1403.6697, 2014.

[43] M. Raya, I. Aad, J.-P. Hubaux, and A. El Fawal. Domino: Detecting mac layer greedy behavior in ieee 802.11 hotspots. *IEEE Transactions on Mobile Computing*, 5(12):1691–1705, 2006.

[44] Y. Rong, S. K. Lee, and H.-A. Choi. Detecting stations cheating on backoff rules in 802.11 networks using sequential analysis. In *INFOCOM*. Citeseer, 2006.

[45] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, pages 166–179, New York, NY, USA, 2001. ACM.

[46] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, MIT Press, 2001.

[47] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell. Detecting 802.11 mac layer spoofing using received signal strength. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[48] K. Siddhabathula, Q. Dong, D. Liu, and M. Wright. Fast jamming detection in sensor networks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 934–938. IEEE, 2012.

[49] N. Song, L. Qian, and X. Li. Wormhole attacks detection in wireless ad hoc networks: A statistical analysis approach. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8–pp. IEEE, 2005.

[50] M. Spuhler, D. Giustiniano, V. Lenders, M. Wilhelm, and J. B. Schmitt. Detection of reactive jamming in dsss-based wireless communications. *Wireless Communications, IEEE Transactions on*, 13(3):1593–1603, 2014.

[51] M. Strasser, B. Danev, and S. Čapkun. Detection of reactive jamming in sensor networks. *ACM Trans. Sen. Netw.*, 7(2):16:1–16:29, Sept. 2010.

[52] X. Su and R. Boppana. On mitigating in-band wormhole attacks in mobile ad hoc networks. In *IEEE International Conference on Communications (ICC)*, pages 1136–1141, June 2007.

[53] X. Su and R. V. Boppana. On mitigating in-band wormhole attacks in mobile ad hoc networks. In *2007 IEEE International Conference on Communications*, pages 1136–1141. IEEE, 2007.

[54] G. Thamilarasu, S. Mishra, and R. Sridhar. A cross-layer approach to detect jamming attacks in wireless ad hoc networks. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–7. IEEE, 2006.

[55] O. N. Tiwary. Detection of misbehaviour at mac layer in wireless networks. *proceedings of International Journal of Scientific and Engineering Research*, 3(5), 2012.

[56] D. Tran and T. Nguyen. Localization in wireless sensor networks based on support vector machines. *Parallel and Distributed Systems, IEEE Transactions on*, 19(7):981–994, July 2008.

[57] F. Vanheel, J. Verhaevert, E. Laermans, I. Moerman, and P. Demeester. A linear regression based cost function for wsn localization. In *19th Intl Conf on Software, Telecom. and Computer Networks (SoftCOM)*, pages 1–5, 2011.

[58] A. Wald. *Sequential Analysis*. J. Wiley & Sons, New York, 1947.

[59] W. Wang, B. Bhargava, Y. Lu, and X. Wu. Defending against wormhole attacks in mobile ad hoc networks. *WIRELESS COMMUNICATIONS AND MOBILE COMPUTING*, 6:483–503, 2006.

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

[61] A. D. Wood, J. A. Stankovic, and S. H. Son. Jam: A jammed-area mapping service for sensor networks. In *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pages 286–297. IEEE, 2003.

[62] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *IEEE network*, 20(3):41–47, 2006.

[63] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '05, pages 46–57, New York, NY, USA, 2005. ACM.

[64] B. Yu and L.-y. Zhang. An improved detection method for different types of jamming attacks in wireless networks. In *Systems and Informatics (ICSAI), 2014 2nd International Conference on*, pages 553–558. IEEE, 2014.

[65] C. Zhang, F. Li, J. Luo, and Y. He. ilocscan: Harnessing multipath for simultaneous indoor source localization and space scanning. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pages 91–104, New York, NY, USA, 2014. ACM.

[66] S. Zheng, T. Jiang, and J. Baras. Performance comparison of two sequential change detection algorithms on detection of in-band wormholes. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pages 270–275, March 2009.

[67] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher. Rid: Radio interference detection in wireless sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 891–901. IEEE, 2005.

[68] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher. Rid: radio interference detection in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 891–901. IEEE, 2005.

[69] Y. Zhou, D. Wu, and S. M. Nettles. On mac-layer denial of service attacks in ieee 802.11 ad hoc networks: Analysis and counter measures. *Int. J. Wire. Mob. Comput.*, 1(3/4):268–275, 2006.

VITA

Yi Ling was born in Shang Hai, China, he received his B.S.in computer science in Beijing Forestry University. Then, he joined the Computer Science Department of Missouri University of Science and Technology (formerly the University of Missouri-Rolla) as a Ph.D. student in August, 2013. He received Ph.D degree in computer science in May, 2017.