NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INVESTIGATING THE DETECTION OF MULTI-HOMED DEVICES INDEPENDENT OF OPERATING SYSTEMS

by

Javan A. Rhinehart

September 2017

Co-Advisor:	Murali Tummala
Co-Advisor:	John C. McEachen
Second Reader:	Bryan Martin

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE INVESTIGATING THE DETECTION OF MULTI-HOMED DEVICES INDEPENDENT OF OPERATING SYSTEMS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Javan A. Rhinehart			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Networks protected by firewalls and physical separation schemes are threatened by multi-homed devices. The purpose of this study is to detect multi-homed devices on a computer network. More specifically, the goal is to evaluate passive detection of multi-homed devices running various operating systems while communicating on a network. TCP timestamp data was used to estimate clock skews using linear regression and linear optimization methods. Analysis revealed that detection depends on the consistency of the estimated clock skew. Through vertical testing, it was also shown that clock skew consistency depends on the installed operating system. The linear programming and linear regression methods agree with one another when clock skews are consistent, indicating that linear regression is sufficient to identify multi-homed hosts in networks with low network delay. Further analysis showed inconsistencies of clock skew estimation on newer versions of OS X and freeBSD 12.0; the clock skews from these operating systems prevented multi-homed fingerprinting using the proposed detection scheme.

14. SUBJECT TERMS software-defined network, multi-homed host, network monitoring, fingerprinting, clock skew			**15. NUMBER OF PAGES** 83
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

INVESTIGATING THE DETECTION OF MULTI-HOMED DEVICES INDEPENDENT OF OPERATING SYSTEMS

Javan A. Rhinehart
Lieutenant Commander, United States Navy
B.S., University of Washington, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2017

Approved by: Murali Tummala
Co-Advisor

John C. McEachen
Co-Advisor

Bryan J. Martin
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Networks protected by firewalls and physical separation schemes are threatened by multi-homed devices. The purpose of this study is to detect multi-homed devices on a computer network. More specifically, the goal is to evaluate passive detection of multi-homed devices running various operating systems while communicating on a network. TCP timestamp data was used to estimate clock skews using linear regression and linear optimization methods. Analysis revealed that detection depends on the consistency of the estimated clock skew. Through vertical testing, it was also shown that clock skew consistency depends on the installed operating system. The linear programming and linear regression methods agree with one another when clock skews are consistent, indicating that linear regression is sufficient to identify multi-homed hosts in networks with low network delay. Further analysis showed inconsistencies of clock skew estimation on newer versions of OS X and freeBSD 12.0; the clock skews from these operating systems prevented multi-homed fingerprinting using the proposed detection scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

DOD	Department of Defense
ICMP	Internet control message protocol
IP	Internet protocol
ISP	Internet service provider
MAC	media access control
NAT	network address translation
NIC	network interface card
NTP	network time protocol
OS	operating system
RTO	retransmission timeout
RTT	round trip time
RTTM	round trip time measurement
SDN	software-defined network
SSH	secure shell
TCP	transmission control protocol
TSecr	timestamp echo response
TSval	timestamp value

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Thank you to my professors, advisors, and mentors, whose guidance, wisdom, and encouragement made it possible for me to complete this thesis. Most importantly, to my wife and family: your love, support, and understanding gave me the endurance to cross the finish line.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

As industry shifts from traditional network infrastructure toward software-defined networks (SDN), network security continues to be a primary concern for network managers at all levels. Likewise, as customer demand for services from networks rises, it is increasingly difficult to manage policies, handle dynamic internet loads, and respond quickly to faults and changes [1]. SDN is a new paradigm that changes the way networks are designed, managed, and secured [2]; such networks have the potential to increase network performance and security while reducing costs [3]. SDNs completely change the way in which network managers think about their network and security within the network.

A. MOTIVATION

Similar to the commercial sector, the United States military recognizes the benefits of implementing SDN and is actively exploring the best methods to use SDN on its networks. Any new technology involves many unknowns, so the military must explore all possible security threats in SDNs prior to wide-scale adoption.

Many concerns surround network security, and the segregation of classified networks is one area affecting U.S. military network operations. The Committee on the National Security System Advisory Memorandum requires that all unclassified networks be physically separated from classified networks by at least three meters [4]. This distance ensures that there is no path for data from the classified network to interfere with the unclassified network.

The physical barrier between a classified and unclassified network can be bypassed by a multi-homed device having multiple network connections: originating at the multi-homed device and terminating at the segregated networks. Multi-homing a device can also be used to bypass a firewall in a similar way. Therefore, a method to detect when a device is multi-homed needs to be developed. Preventing circumvention of firewalls and physical network segregation is essential to network separation security.

B. OBJECTIVE

A plethora of operating systems support multi-homed connections, which can be installed on virtually any modern computer. A hostile user can connect a multi-homed device to a network to bypass firewalls or physically isolated security schemes. The bypassing of security protocols presents a real threat to any sensitive network, so Martin [5] originally proposed a scheme to detect multi-homed devices on a network by determining and comparing clock skews of all devices connected to the network.

Martin [5] determined the clock skews by passively collecting Transmission Control Protocol (TCP) timestamp data, which has been shown to reliably reveal the clock skew of a Network Interface Card (NIC) [5], [6]. The ability to calculate clock skews in [5] was limited to Raspberry Pis with Raspbian as the installed operating system. Because of the limitations of Raspberry Pis, it remains to be demonstrated that the detection methods Martin employed can be expanded to other devices and operating systems capable of connecting to a computer network and potentially an SDN.

The objective of this thesis is to validate the detection method used in [5] when applied to a larger set of devices and operating systems. Specifically, to validate the multi-homed detection methods, we will determine whether a fingerprinter can passively collect TCP timestamp information and, in doing so, detect multi-homed devices with various operating systems installed while communicating on a computer network. The detection of multi-homed devices will assist in thwarting security threats to firewalls and physical separation schemes, ultimately making commercial- and military-grade networks more secure.

C. RELATED WORK

In their research, Khono et al. [6] introduced the idea of using the estimated clock skew of a host to identify a physical device on a network. They fingerprinted a given host CPU by exploiting microscopic deviations created during the manufacturing process. By using these deviations, they extrapolated a unique clock skew for each host. To determine the estimated clock skew of a CPU, they used the TCP timestamp option introduced in

RFC 1323 by determining timestamp offsets and then taking the first derivative of the offsets. They then compared the resulting clock skews with one another to show that they were unique and that the same device consistently produced the same clock skew [6].

Polcak [7] further explored clock skew identification and successfully demonstrating that various operating systems could be fingerprinted by TCP and Internet Control Message Protocol (ICMP) data. They also showed that the majority of clock skews fall between -50 to 100 parts per million (ppm), reducing the ability to fingerprint networks with a large number of hosts due to the increased number of false positives [7]. Finally, they showed that the estimated clock skew did not change with network time protocol (NTP) updates [7].

Martin [5] demonstrated that multi-homed devices could be detected using a fingerprinter by comparing the estimated clock skews of Raspberry Pis. With a large set of clock skews from multiple trials, Martin showed that the clock skews approached a Gaussian distribution, from which a confidence interval could be determined. To detect the presence of a multi-homed device, Martin compared a given clock skew to the confidence interval; if the clock skew fell between the upper and lower bounds of the calculated confidence interval, then it represents a 95% certainty that the two hosts originated from the same device.

D. THESIS ORGANIZATION

The remainder of this thesis is organized as follows. Chapter II discusses how SDNs are organized and the security threat posed by multi-homed devices. The discussion then shifts to how operating systems generate TCP timestamp information, as well as how the clock skew is generated from the TCP timestamp. The proposed scheme to detect multi-homed devices is then discussed in Chapter III, and the results from the SDN dual-homed test bed are reported in Chapter IV. Triple-homed test bed results are presented in Chapter V. Significant findings from the results, along with potential future work are discussed in Chapter VI. Appendix A and B contain the MATLAB code used to calculate the estimated clock skews and the confidence intervals, respectively.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The Department of Defense (DOD) goes to great lengths to segregate networks based on the classification of information on the network, as required by [4]. The DOD has determined that a physical separation scheme is the only acceptable method to ensure that data cannot cross between classified and unclassified networks. The ability of a user to circumvent such security measures by using a multi-homed device and connecting it to multiple networks concerns for network administrators and represents a prime opportunity by the hacking community [8]. The related goal of this thesis is to determine methods to mitigate multi-homed threats that may potentially be implemented on an SDN. This chapter focuses on relevant background information that will set up the detection scheme outlined in Chapter III. This chapter first discusses SDNs, then explains how a multi-homed host can bypass security measures. The discussion continues with TCP timestamps and how they are generated. The chapter concludes with an explanation of how a clock skew is determined and a discussion of confidence intervals.

A. SOFTWARE DEFINED NETWORKS

Although legacy internet protocol (IP) networks have been adopted worldwide, they are complex, difficult to manage, and expensive to establish [9]. SDNs have the potential to alleviate many of the difficulties and high costs of traditional IP networks by separating the control logic (the control plane) from the physical traffic (the data plane) [1], [7], [10], [11]. With the implementation of SDNs, proprietary routing protocols previously implemented in hardware can now be implemented in software. This separation allows for complex switches and routers used in legacy networks to be downgraded to cost-effective simple switches and routers that receive routing flow rules from a centralized controlling device [7].

SDNs also introduce an application plane, which consists of services provided by the network operator and perform tasks as specified [10]. A simple representation of the SDN planes is shown in Figure 1. Here, the application plane installs programs and

services onto the control plane via the northbound application programming interface (API). The control plane monitors traffic on the data plane and pushes flow rules from the control plane to the data plane via the southbound API [10]. Together, the northbound and southbound API control all services and flow rules within the SDN. Although not shown in Figure 1, this simplified network can be expanded by connecting multiple data planes to a single control plane or multiple control planes to multiple data planes, such that each data plane will have a dedicated southbound API to its assigned controller to handle flow-rule traffic.

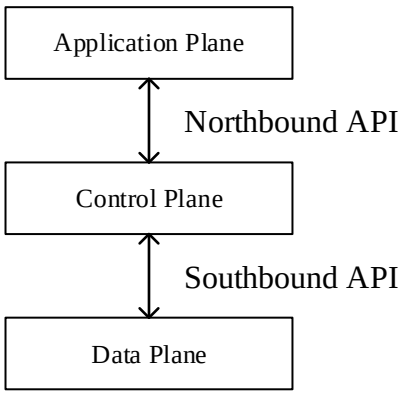

Figure 1. Functional Planes in a Software-Defined Network. Adapted from [10].

B. MULTI-HOMED HOSTS

A multi-homed host is a device that establishes multiple connections to a single network or to multiple networks by using multiple NICs. Each NIC will have a unique MAC address and IP address assigned. Generally, multi-connection hosts create redundancy in a network by connecting the host to separate network nodes [8]. For example, to increase network reliability, a multi-homed host will have multiple connections to the Internet service providers (ISP). In this example, if one of the nodes becomes unavailable, the host will recognize the break in the connection and stop transmitting via the affected NIC. The host will then direct all future traffic to the NIC with the viable connection. This redundancy produced by multi-homed connections

ensures that single point node failures do not prevent communication within the network [5], [12].

A multi-homed connection has the potential to become a security threat to a protected network if it is connected to two networks, one of which is separated by a firewall or physical air gap. Protecting a network is vitally important to many commercial industries and the DOD. Denying or limiting access to a network is normally accomplished by either employing a firewall or physically separating a network from other networks. In this instance, the multi-homed connection serves as a bridge between the two networks, and a hostile actor can take advantage of this situation by bypassing the security measures to gain access to the denied network. In Figure 2, we show how the physically separated network can be bridged by a multi-homed device and remove the physical separation of the two networks.

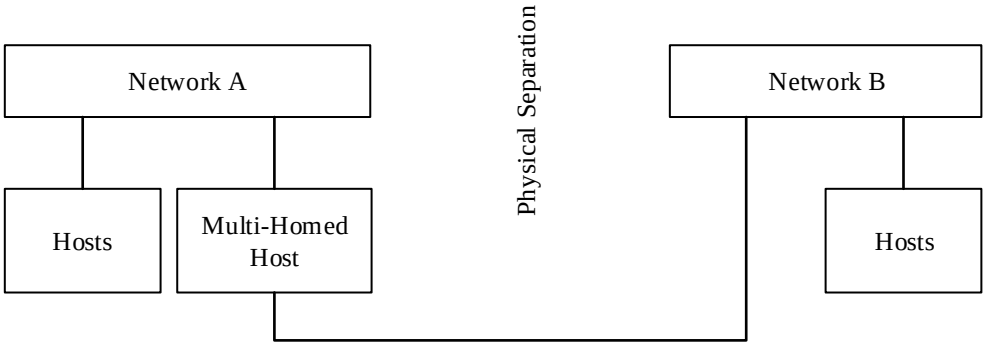

Figure 2. Physically Separated Networks Where a Multi-Homed Device Bypasses Security Measures

C. TCP TIMESTAMPS

The TCP timestamp option was introduced in RFC 1323 to meet the high throughput demand of high-speed fiber optics. RFC 1323 was implemented to protect against wrapped sequence (PAWS) numbers and to improve round trip time measurements (RTTM) [13]. RTTM is the time interval between sending a TCP package and receiving an acknowledgement from the destination; it is essential in the calculation

of the retransmission timeout (RTO) used in TCP [13]. Because of the high speeds and large bandwidth capabilities of fiber optic connections, TCP needed protection against receiving duplicate sequence numbers from the same device. PAWS used TCP timestamp data to identify instances in which duplicate sequence numbers are detected [13]. The timestamp option is 10 bytes long, as shown in Figure 3.

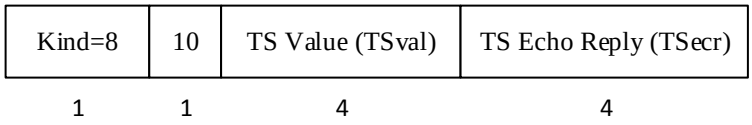

Figure 3. TCP Timestamp Option Field. Source: [13].

The timestamp value (TSval) field is 4 bytes long and is obtained from an internal clock, referred to as the "timestamp clock." According to RFC 1323, the time values must be proportional to real time. The timestamp echo reply (TSecr) is valid only if the ACK bit is set in the TCP header. When the bit is set, it will echo the timestamp value of the remote TCP connection [13].

According to RFC 1323, to generate TCP timestamps, the installed operating system employs either a virtual clock or a hardware clock. RFC 1323 only requires that the time TSval values are approximately proportional to real time to allow for RTTM. For example, within the Linux Kernel, the function *SOF_TIMESTAMPING_TX_ACK* is called when a TCP timestamp is requested by the user or an application [14]. Upon verification of send buffer acknowledgement, the function reports the hardware time via a *recvmsg()* [14]. According to kernel.org [14], the *recvmsg()* is a control message struct and is passed to the TCP timestamp option field. These timestamps are essential in the clock skew calculations used for multi-homed detection.

D. CLOCK SKEW

All network devices have an internal clock, made up of hardware and software components that synchronize operations within the device [15]. This internal clock uses a crystal oscillator, which oscillates at a design-specified resolution and precision. The

resolution is the smallest interval in which time can be incremented and precision is how close the measurement is to the real measurement. In computer clocks, precision is determined automatically and is expressed as a power of two [16]. Because of microscopic deviations resulting from the manufacturing process, each oscillator in a CPU will tick with miniscule differences [6], [15], [17]. These differences in the tick of the clock can be collected remotely to calculate a clock skew by first using the timestamps of the target clock and then measuring the offset of a local clock [3], [15], [17].

The use of clock skews to identify network devices by passively monitoring timestamps was first introduced by Kohno et al. [6]. To calculate the estimated clock skew for a target, we use the notation of a time reported by a clock C at a given time t as $R_C(t)$. For the remainder of this analysis, clock C will be referred to as the fingerprinter and clock D as the fingerprintee. The time offset between clock C and clock D is

$$O_{C,D}(t) = R_C(t) - R_D(t). \tag{1}$$

Assuming that $O_{C,D}(t)$ is a differentiable function of t, then the clock skew for a given host is [7]

$$S_{C,D}(t) = \frac{d}{dt} O_{C,D}(t). \tag{2}$$

Due to the fact that the fingerprintee clock $R_D(t)$ is not directly observable by the fingerprinter, we must use an indirect approach to determine the fingerprintee clock. The observed TCP timestamp packets from the fingerprintee are represented as offset points $(\tau_1(t), \tau_2(t))$ where $\tau_1(t)$ is the observation time of the fingerprinter and $\tau_2(t)$ is the timestamp value sent by the fingerprintee. Using $\tau_2(t)$, the fingerprintee clock $R_D(t)$ can now be determined as

$$R_D(t) = \tau_2(t) - \epsilon(t) \tag{3}$$

where $\epsilon(t)$ is the network delay to transmit a packet from the fingerprintee to the fingerprinter [7]. Substituting (3) into (1), we get

$$O_{C,D}(t) = R_C(t) - \tau_2(t) - \epsilon(t)\,. \tag{4}$$

We can now calculate $S_{C,D}(t)$ as we have all of the values in (4). The estimated clock skew and the actual clock skew for a device are parallel lines that have the same angle α and shifted in the vertical direction by a constant ϵ as shown in Figure 4 [7]. The red hashed line is the true clock skew of the target host and the solid green line is the upper bound solution of the estimated clock skew. The constant ϵ is a direct result from taking calculating $\frac{d}{dt}O_{C,D}(t)$ in (4) [7].

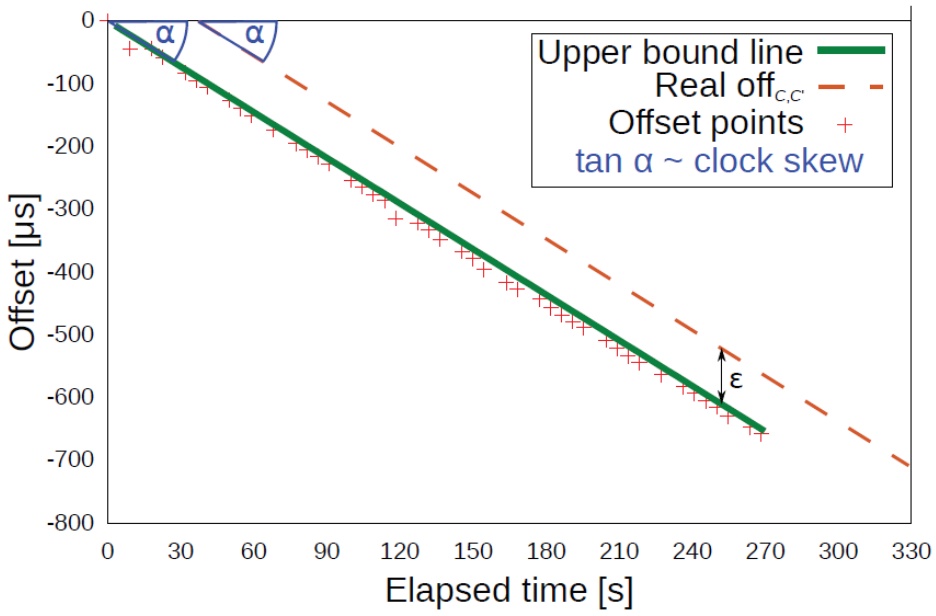

Figure 4. Estimated Clock Skew and Actual Clock Skew. Source: [7].

E. CONFIDENCE INTERVALS

Due to the uncertainties in the estimated clock skews from the randomness of collected data, the true mean value of the clock skew μ cannot be directly measured or

known [5]. The estimated clock skews are a continuous random variable x and assumed to be Gaussian with a density function $f_X(x)$ [5]. A confidence interval gives a range $[C_L, C_U]$ in which the true mean value of the clock skew falls within a specified probability of $1-\rho$ [5]. The range of said interval is defined as

$$P[C_L \le z \le C_U] = 1-\rho, \tag{5}$$

where ρ is the acceptable error for $0 < \rho < 1$ [18]. The bounds of a confidence interval for a given density function $f_X(x)$, true mean μ, and acceptable error ρ are shown in Figure 5.

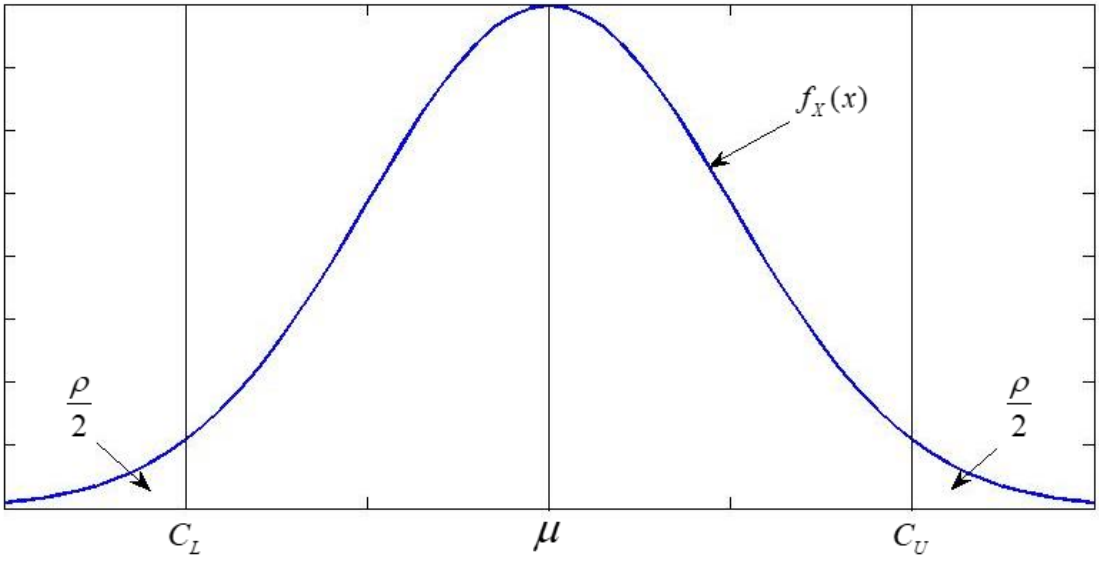

Figure 5. The Bounds C_L and C_U of the Confidence Intervals for a Given Density Function with a True Mean μ and Acceptable Error ρ. Source: [5].

The bounds C_U and C_L are determined by solving [18]

$$\frac{\rho}{2} = \int_{C_U}^{\infty} f_X(x)dx \tag{6}$$

and

$$\frac{\rho}{2} = \int_{-\infty}^{C_L} f_X(x)dx. \quad (7)$$

If a given clock skew falls within the bounds of the confidence interval from an IP address, it can then be argued that the given clock skew originated from the same device with the probability of $1-\rho$.

To summarize, in this chapter we introduced SDN and the various planes and APIs that make up the network. We then defined multi-homed hosts and examined how they can be used to bypass security separation protocols. We also discussed TCP timestamp options, their purpose, and how they are generated in a Linux Kernel. The chapter defined clock skew and described confidence intervals. The next chapter will look at the methodology to remotely detect a multi-homed device on a SDN.

III. MULTI-HOMED MULTI-OS DEVICE DETECTION USING CLOCK SKEW

The overall objective of this research is to validate multi-homed detection methods when applied to a larger set of devices and operating systems as compared to the devices used in [5]. A multi-homed host has multiple NICs installed and can be connected to one or more networks with each NIC having a unique IP address. TCP timestamps will be passively collected at the fingerprinter. Once the data is collected, it will be used to determine a clock skew for each IP address on the network. The resultant clock skews will then be compared to one another to determine which IP addresses are multi-homed.

The remainder of the chapter is organized as follows; first, we propose the general scheme for multi-homed device detection based on the estimated clock skew for a given IP address. We then discuss the clock skew estimation from data collected from both the fingerprinter and fingerprintee. Next, we describe how linear programming and linear regression methods can be used to estimate the clock skew. Finally, we examine the confidence interval for 95% confidence based on a Gaussian distribution.

A. PROPOSED SCHEME

A graphical representation of the proposed scheme is depicted in Figure 6 and is similar in approach to the one taken in Martin's work [5]. First, we will passively collect TCP timestamp data. With this data, we can determine the estimated clock skew for a given IP address as it has been shown that the clock skew is consistent and unique enough to identify physical devices remotely [6]. Then the estimated clock skew will be used to fingerprint a host by calculating confidence intervals and determining which clock skews fall within the bounds of the intervals. The hosts whose IP addresses fall within the confidence interval will be examined further to determine if the host is multi-homed. Individual details with regards to the steps of Figure 6 are described in further detail in the following sections.

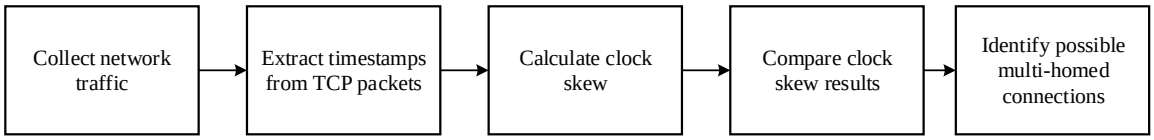

Figure 6. General Diagram of Proposed Scheme. Adapted from [5].

B. TRAFFIC COLLECTION

A general SDN is given in Figure 7. In this general network, a controller is connected to one or more switches that form the base of the data plane. The connections between the controller and the switches make up the southbound API as discussed in Chapter II. The northbound API is not shown in the diagram as the application layer lies within the controller and application commands are pushed to the controller via software running on the controller. The hosts are connected to the switches as prescribed by the testing setup. The hosts are a combination of a diverse mixture of hardware configurations with various operating systems installed based on the type of testing required. The number of dual-homed and triple-homed connections from the hosts also depends on the testing parameters that will need to be established prior to testing. The fingerprinter will passively collect traffic as shown in Figure 7.

Three general steps are required to ensure that TCP timestamp data is received at the fingerprinter. First, the operating system must have the TCP timestamp option enabled as some operating systems do not enable TCP timestamp option by default [7]. Second, the operating system must then have an available TCP port upon which the TCP connection will be established. If the TCP port is not listening for traffic, then the host device will not allow the data to be processed. Finally, the host operating system must have proper firewall rules established to ensure that TCP connections are not blocked.

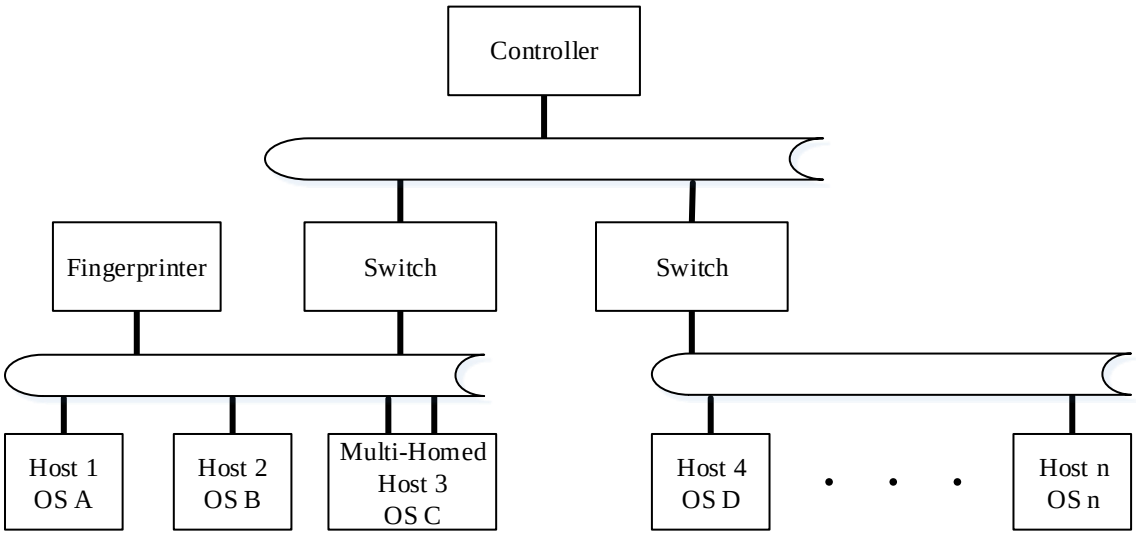

Figure 7. Generic Network Configuration. Adapted from [5].

For the purposes of this research, the term multi-homed will cover both dual and triple-homed devices. For triple-homed devices, we will also conduct vertical testing. Vertical testing is defined as maintaining the device hardware constant while changing the installed operating system to determine the clock skew of the hardware. By keeping the hardware constant and varying the operating system, we can examine any differences in estimated clock skews.

C. CLOCK SKEW DETERMINATION

Clock skew calculation is an essential step in multi-homed detection and is the third block of the proposed scheme in Figure 6. The network traffic analyzed must contain TCP timestamp information for clock skew estimation as discussed in Chapter II. After a trace Θ of TCP timestamp data has been collected from all target IP addresses on the network, clock skews can be calculated using the procedure outlined in [5], [6]. Let t_i be the time that the fingerprinter observed the i^{th} packet in Θ and let T_i be the TCP timestamp within the i^{th} packet. The observed time offset is calculated by

$$x_i = t_i - t_1, \tag{8}$$

where x_i is the observed time difference between the i^{th} packet and the observed time of the first packet in Θ. The timestamp offset w_i is given by

$$w_i = \frac{T_i - T_1}{f}, \tag{9}$$

where T_i is the timestamp of the i^{th} packet, T_1 is the first timestamp in Θ, and f is the operating frequency of the fingerprintee clock. After the observed time offset and timestamp offset are known, the observed offset y_i of the i^{th} packet is calculated by

$$y_i = w_i - x_i . \tag{10}$$

Let us now define O_Θ as a set of offsets corresponding to the trace Θ and is represented by the following notation

$$O_\Theta = \{(x_i, y_i) : i \in \{1, ..., N\}\}, \tag{11}$$

where N is the number of packets in Θ and (x_i, y_i) is determined in (8) and (10), respectively. We assume that O_Θ is differentiable in time, thus we can model the data as a line with the following form

$$\alpha x_i + \beta \geq y_i . \tag{12}$$

The estimated clock skew is the slope of the line in (12). To determine the slope, we will use linear programming and linear regression methods as described in the following sections.

1. Linear Programming

One method to calculate the estimated clock skew for a given set of offsets O_Θ is to use linear programming. The solution to linear programming minimizes the objective function [6]

$$J = \frac{1}{N} \sum_{i=1}^{N} (\alpha x_i + \beta - y_i) \ . \tag{13}$$

The solution in Equation (13) yields the slope of the clock skew α [6]. Linear programming solutions contain both an upper bound and a lower bound solution. Only

the upper-bound line solution is used since all of the timestamps are positive valued and progress forward in time [6]. The clock skew is calculated by evaluating Equation (13) for each IP address on the network.

2. Linear Regression

A second method to calculate clock skew estimations is to use linear regression where a best fit line is applied to a set of offsets O_Θ. The solution for linear regression is in the form of $\alpha x + \beta = y$. Similar to linear programming, the clock skew is the slope of the output line. Linear regression presents several advantages over linear programming: it is less computationally intensive and less complex to implement. The disadvantage of linear regression is that variable network delays may cause inaccuracies to clock skew values [6]. For linear regression method to be used, network delays must be smaller than the resolution of the host system clock to generate the timestamp clock. The clock skews from linear regression will be compared to the clock skews from the linear programming to provide a comparison between the two methods.

D. DETECTION OF MULTI-HOMED HOSTS

Multi-homed detection is the last step of the proposed scheme in Figure 6. After clock skew estimations are completed, we must compare the estimated clock skews to determine which IP addresses belong to potential multi-homed devices. As discussed in Section E, Chapter II, the clock skews are random variables, which are assumed to have a Gaussian distribution [5], [18].

Using a distribution of clock skews from multiple trials, we calculate the mean clock skew and a confidence interval for each IP address. The sample mean for the i^{th} host is determined by

$$m_i = \frac{1}{n}\sum_{i=1}^{n} \alpha_i \tag{14}$$

where α_i is the estimated clock skew for the i^{th} host and n is the number of trials [18].

We then determine if the sample mean m_i falls within j^{th} host confidence interval

$$C_{L,j} \le m_i \le C_{U,j}. \tag{15}$$

If m_i falls within the bounds of j^{th} hosts confidence interval, then it can be assumed that the i^{th} and j^{th} originated from the same device with a confidence of $1-\rho$. Conversely, if m_i does not fall within the bounds of (9), it can be assumed that i^{th} and j^{th} did not originate from the same device. The detection flow chart is summarized in Figure 8.

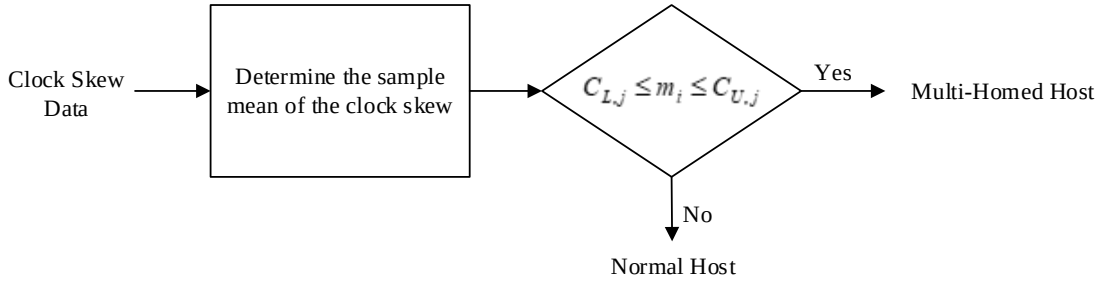

Figure 8. Detection Flow Chart. Source: [5].

To summarize, this chapter laid out a detailed description of the proposed scheme to support the objective of this thesis. The scheme requires timestamped responses within the TCP header options field. From the observed timestamps, clock skews between the fingerprinter and target host are calculated using linear programming and linear regression methods. The clock skew is a unique value for a given device for which a confidence interval can be determined. The confidence interval provides a means to detect a multi-homed device on a network. The next two chapters will discuss the specifics of the test bed and results from such testing.

IV. DUAL-HOMED TESTING AND RESULTS

The proposed scheme for remote collection and calculation of clock skews and detecting multi-homed devices from TCP timestamps was introduced in Chapter III. This chapter is an extension of those ideas, beginning with a description of the network configuration to remotely identify dual-homed devices in an SDN environment. Specific methods used to passively capture traffic are also presented in this chapter. After the TCP traffic is captured, we calculated the clock skews for all host IP addresses on the network by using linear programming and linear regression methods. Once all of the clock skews have been determined, we applied confidence interval analysis to identify multi-homed hosts. This chapter will focus on dual-homed testing while Chapter V will discuss triple-homed testing and results.

A. TEST BED

The test bed laid out in this chapter describes the actual SDN network and components used to validate the proposed scheme. These components are used for generating TCP timestamps, passively collecting said timestamps, and calculating clock skew estimations. After the clock skews are calculated for each IP address, a confidence interval is used to determine whether an IP address belongs to a given host.

1. Network Setup for Dual-Homed Testing

The components, configuration of installed operating systems, and the associated static IP address, which formed the test bed for dual-homed testing, are listed in Table 1 and shown in Figure 9. The switch was a HP 3800 used to connect all of the hosts within the network with a subnet mask of 255.255.255.0. There were seven Raspberry Pi 3 Model Bs on the network corresponding to hosts 1–7. All of the Raspberry Pis had Ubuntu MATE 16.04 LTS installed for the operating system. Hosts 1–6 were single-homed devices and added to the network to show that these devices did not interfere with

the data collection of the dual-homed devices. Host 7 was the only dual-homed Raspberry Pi on the network.

Table 1. Summary of Hosts for Dual-Homed Testing

Host	Device/Hardware	Operating System	IP Address 1	IP Address 2
1	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.1	-
2	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.2	-
3	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.3	-
4	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.4	-
5	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.5	-
6	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.6	-
7	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.10	10.10.8.11
8	Dell Latitude E6430	Windows 7	10.10.8.12	10.10.8.13
9	Dell Latitude E5420	Fedora 26	10.10.8.14	10.10.8.15
10	Dell Latitude E6540	Linux Mint 18.1	10.10.8.16	10.10.8.17
11	MacBook Pro	OSX 10.11.3 El Capitan	10.10.8.18	10.10.8.19
12	Lenovo IdeaPad U430	Windows 10	10.10.8.20	10.10.8.21
13	MacBook Pro	OSX 10.6.3 Snow Leopard	10.10.8.24	10.10.8.25

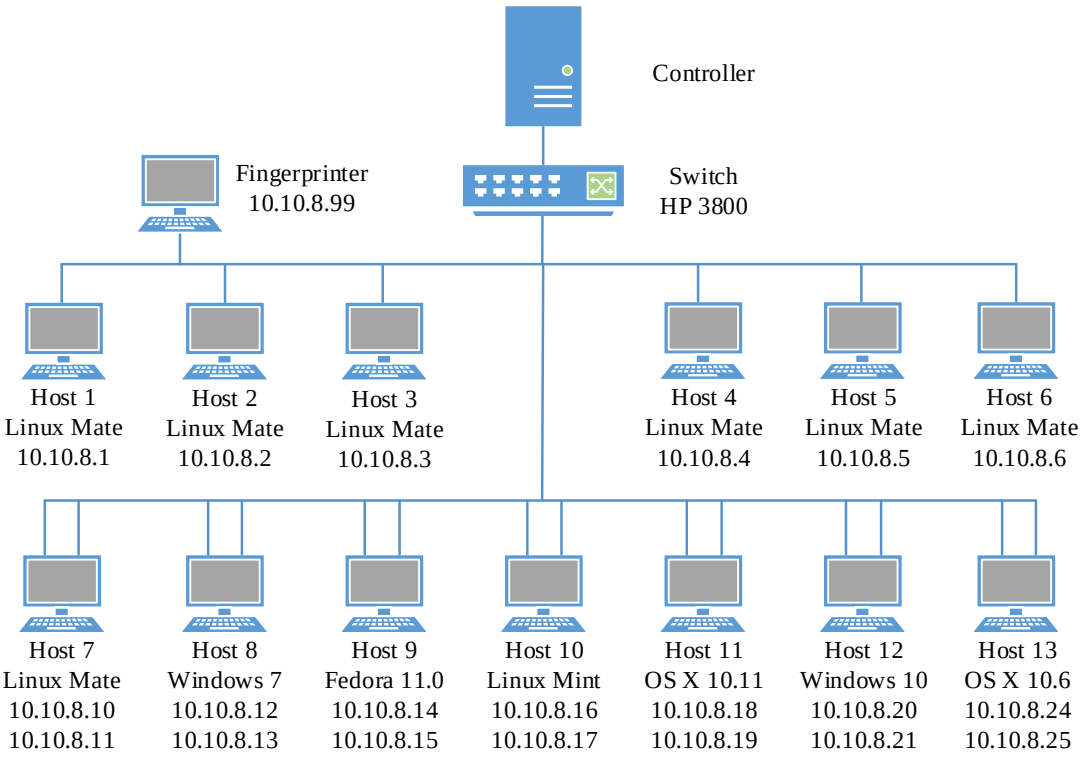

Figure 9. SDN Test Bed for Dual-Homed Testing

Hosts 8–13 were laptops with operating systems configured in accordance with Table 1 and shown in Figure 9. The operating systems selected for testing represent a wide range of the common operating systems available at the time. With the exception of OS X 10.6.3 Snow Leopard, operating systems that are no longer supported by the vendor were not selected due to lack of support. The laptops were dual-homed devices with two static IP addresses. To establish dual connections, the first connection used the devices built-in Ethernet port. The second connection was established by using a USB Ethernet adapter. The fingerprinter was a Dell T1600 running Ubuntu 14.04 LTS and directly connected to the HP 3800. The devices had to be configured after being connected to the network.

2. Host Preparation

Each host had to be independently setup and verified to ensure it would communicate properly within the SDN network. The devices required several steps to properly handle TCP timestamp requests and subsequently respond to said request. To ensure that a device was configured properly to communicate within the network, all hosts required the following steps to be taken:

1. Static assignment of IP address
2. Firewall rules to allow inbound and outbound TCP traffic
3. TCP port activation

The test bed did not have a DHCP server installed on the network, thus IP addresses were statically assigned. This served to minimize confusion in the event a new IP address was automatically assigned to a NIC. To prevent the automatic blocking of TCP timestamp requests, the rules within the firewalls were adjusted such that TCP requests were not blocked. The ports on the target device must also be active and listening for traffic. If the ports are not activated, a given device will ignore the request and no data will be transmitted.

There may be cases where an operating system does not enable RFC 1323 TCP timestamp by default [7]. In those cases, TCP timestamp reporting must be manually enabled using instructions specified by the operating systems vendors.

B. TRAFFIC GENERATION AND COLLECTION

Hping3 was used to generate traffic between the fingerprinter and fingerprintee. It is a command-line TCP/IP packet assembler and analyzer written by Salvatore Sanfilippo and distributed to Kali Linux [19]. *Hping3* establishes a TCP connection with the target host and enables the TCP timestamp option field. Traffic was established at the default rate of one request per second and allows for network traffic to be observed by Wireshark.

The responses from the hosts were captured and filtered using Wireshark. Filters in Wireshark were set to show only TCP traffic that contained TSVal and TSecr values. If a host TSVal and TSecr were blank in the TCP options segment of the TCP return, then the host network configuration needed to be set in accordance with the steps stated in Section A.2 of this chapter.

C. DUAL-HOMED CLOCK SKEW RESULTS

Once the test traffic was collected via Wireshark, the data then had to be imported into MATLAB. The traffic generator, *hping3*, sends a TCP timestamp request every second to a specified IP address and port. During an experimental trial, at least 3600 samples per IP address were collected, which is approximately 60 minutes of data collection. The 3600-sample minimum ensured that the collection had enough data points to provide for stable estimation and minimized any network jitter that could potentially affect clock skew estimations.

To determine the clock skews using linear programming for each host, the MATLAB function *linprog* was applied to the clock offset values as given in (11). Likewise, to determine clock skews with linear regression, the MATLAB function *fit* was used with a linear polynomial as the fit type object. The output of both clock skew

calculation methods yielded a solution in the form $\alpha x + \beta \leq y$, where α is the clock skew and β is the *y*-intercept. The MATLAB code for clock skew determination is in Appendix A. The built-in MATLAB functions *tic* and *toc* were used to compute the computational times for linear programming and linear regression. For the 3600-sample trial, the linear regression method was 2-3 times faster than the linear programming method.

The clock skew for each host was calculated for each 3600-sample trial. Only the upper-bound solution was considered, as all of the delays between the network hosts and the fingerprinter were positive and progressed forward in time as discussed in Section D, Chapter III. The upper-bound solution for a dual-homed Raspberry Pi 3 running Ubuntu Mate 16.04 LTS corresponding to IP address 10.10.8.11 is shown in Figure 8. The linear programming slope was 13.9614 ppm and the linear regression method yielded similar results with an estimated clock skew of 13.9712 ppm. The red dots, in Figure 8, are the individual time offsets between the host and fingerprinter, and the blue line is the clock skew solution for the slope as calculated by the fingerprinter using the linear programming method.

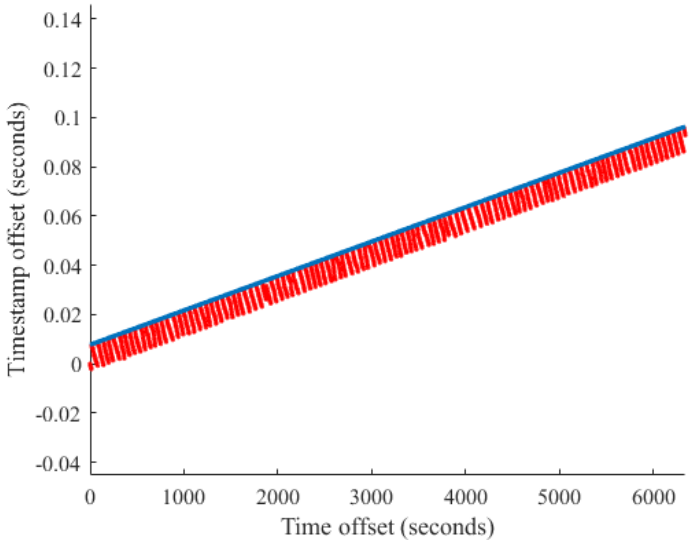

Figure 10. Upper-Bound Clock Skew Solution for Host 7 (10.10.8.11) over a Single Trial

Comparing the slopes of all hosts in a single trial shows the variation of clock skews present within the network. Each host has an independent slope that is unique to the host [6], [7]. A plot of all of the dual-homed hosts clock skews from a single trial is shown in Figure 11.

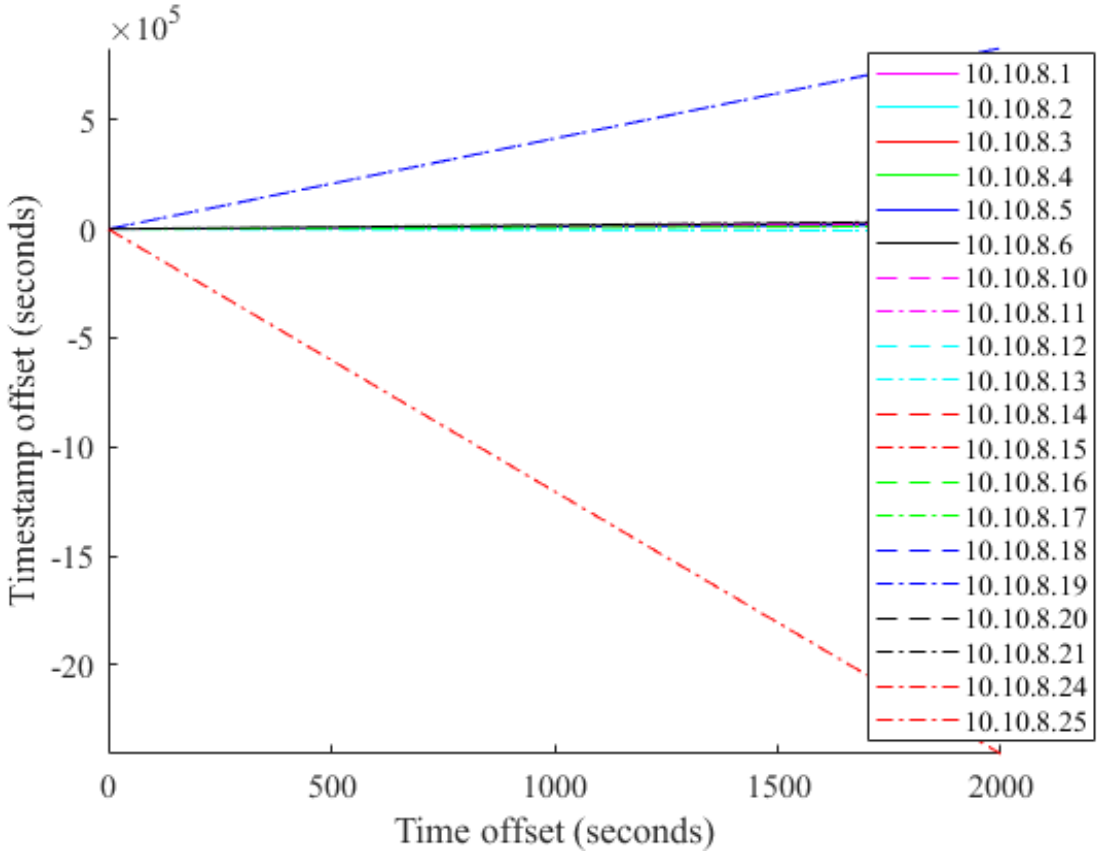

Figure 11. Clock Skews of All Hosts for a Single Trial

By zooming into Figure 11, we can observe more detail associated with the dual-homed hosts. The IP addresses that originate from the same host have the same slope but not necessarily the same *y*-intercept as shown in Figure 12. The difference in the location of the *y*-intercept is due to timestamp delay $\epsilon(t)$ as discussed in Section D, Chapter III. The solid lines in Figure 12 are single-homed and the hashed or dotted lines are those that are dual-homed. For ease of comparison, IP addresses originating from the same host are

colored with the same color. For example, the orange arrow on the graph points to two parallel lines that correspond to host 9 (IP addresses 10.10.8.14 and 10.10.8.15). The two lines share similar clock skews (slopes) at 7.1764 and 7.1761 ppm, respectively. They are only shifted in the *y*-direction due to differences in timestamp delays $\epsilon(t)$.

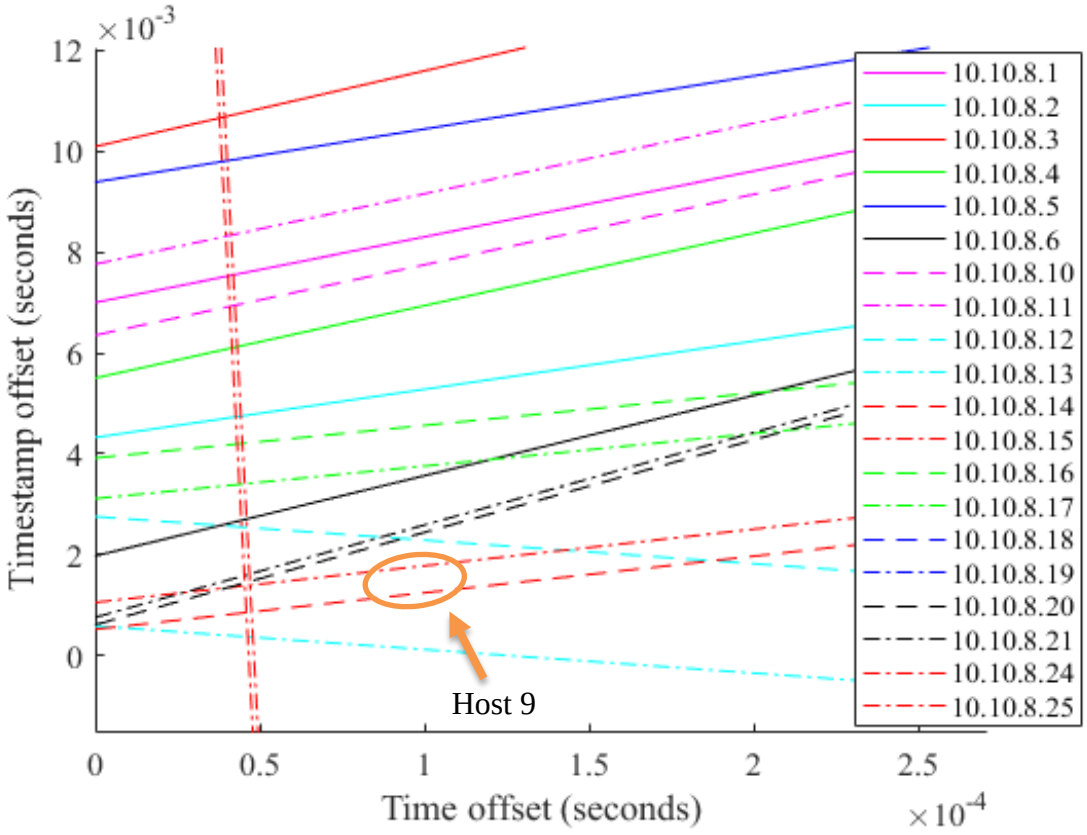

Figure 12. Zoomed-in View of All Clock Skews from Figure 11 over a Single Trial

The mean clock skew from both linear programming and linear regression for each IP address is listed in Table 2. It can be seen in Table 2 that the estimated clock skews from both the linear programming and linear regression appear to agree with one another with the exception of host 11. A deeper examination of host 11 is conducted later in this chapter.

Table 2. Mean Clock Skews for All Hosts Using Linear Programming and Linear Regression (in ppm)

Host	Device/Hardware	Operating System	IP Address	Linear Programming Clock Skew	Linear Regression Clock Skew
1	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.1	13.0434	13.0408
2	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.2	9.6767	9.5989
3	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.3	15.0640	15.0653
4	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.4	14.4238	14.4148
5	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.5	10.8369	10.5269
6	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.6	15.9289	15.9306
7	Raspberry Pi 3 Model B	Ubuntu MATE 16.04	10.10.8.10	13.9486	13.9480
			10.10.8.11	13.9493	13.9493
8	Dell Latitude E6430	Windows 7	10.10.8.12	-4.6413	-4.6403
			10.10.8.13	-4.6411	-4.6384
9	Dell Latitude E5420	Fedora 26	10.10.8.14	7.1764	7.1760
			10.10.8.15	7.1761	7.1715
10	Dell Latitude E6540	Linux Mint 18.1	10.10.8.16	6.5648	6.5479
			10.10.8.17	6.5657	6.5481
11	MacBook Pro	OSX 10.11.3 El Capitan	10.10.8.18	85.4578	90.8211
			10.10.8.19	85.4482	96.1752
12	Lenovo IdeaPad U430	Windows 10	10.10.8.20	18.2444	18.2516
			10.10.8.21	18.2450	18.3122
13	MacBook Pro	OSX 10.6.3 Snow Leopard	10.10.8.24	-1227.6546	-1230.4099
			10.10.8.25	-1227.6477	-1230.3975

The calculated clock skews are random variables and as such approach a Gaussian distribution when a large number of trials are conducted [5]. In this experiment, we conducted 174 trials for dual-homed devices. A histogram of host 3 clock skews is shown in Figure 13. It can be seen in Figure 13 that the clock skews reasonably resemble the Gaussian shape. Host 3 histogram is representative of all of the hosts tested with the exception of host 11.

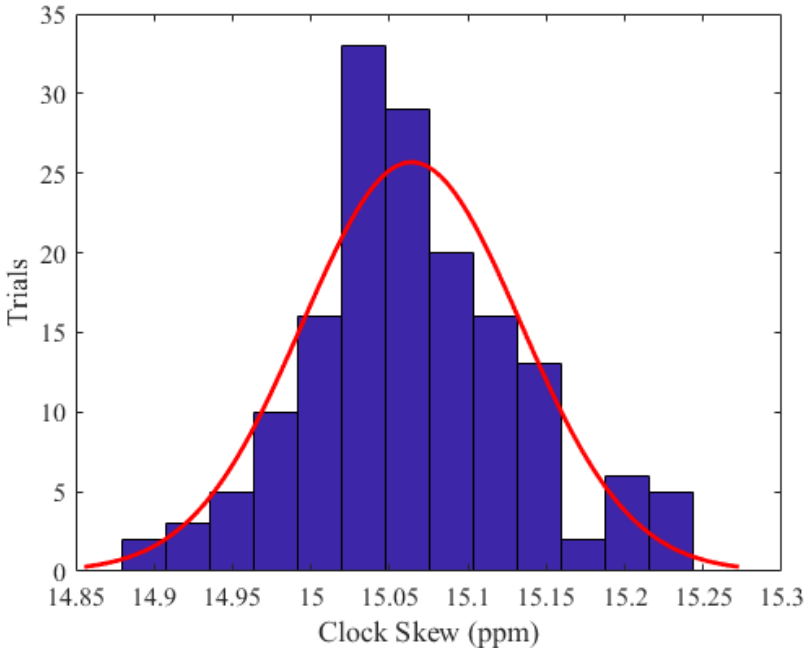

Figure 13. Histogram of the Estimated Clock Skews for Host 3 (10.10.8.3) after 174 Trials

The histogram of the clock skews of host 11 did not have a Gaussian shape, unlike the other hosts tested on the network. The two IP addresses assigned to host 11 were 10.10.8.18 and 10.10.8.19, and both yielded similar bi-modal histograms. The histogram from 10.10.8.19 is shown in Figure 14. This device was a MacBook Pro running OS X 10.11.3 El Capitan, which was released in 2015. Due to the instability of calculated clock skews, we were unable to fingerprint OS X 10.11.3 El Capitan. The instability of OS X is consistent with the findings from [7] where they observed clock skew changes on unknown occasions.

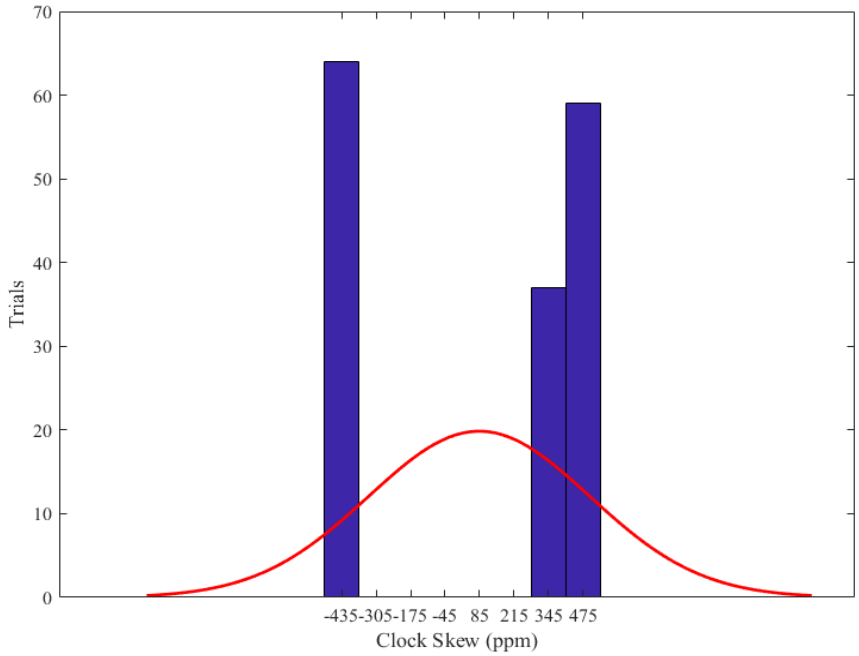

Figure 14. Histogram of the Estimated Clock Skews for Host 11 (10.10.8.19) Running OS X 10.11.3 Displaying Non-Gaussian Shape after 174 Trials

Comparatively, host 13 was a different MacBook Pro running OS X 10.6.3 Snow Leopard released in 2009. The histogram for the MacBook Pro running Snow Leopard is shown in Figure 15. Here, host 13 returned consistent clock skews whose histogram displayed a Gaussian shape similar to the non-Mac hosts. It did not have a bi-modal distribution as the El Capitan device presented in Figure 14. Based on the results, sometime between the release of Snow Leopard in 2009 and El Capitan in 2015, Apple altered the way timestamps are reported. Unfortunately, OS X is a proprietary operating system, and it was not possible to examine the kernel to determine how TCP timestamps are generated.

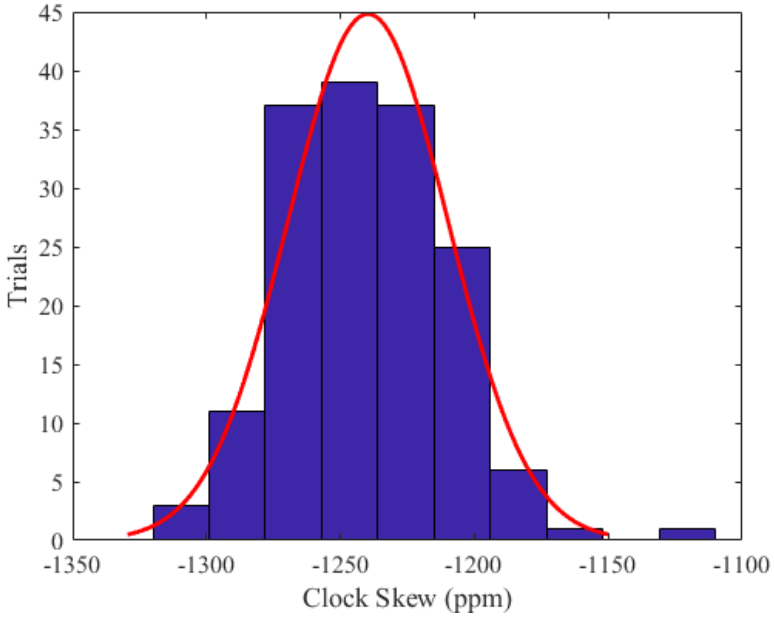

Figure 15. Histogram of Estimated Clock Skews for Host 13 (10.10.8.24) Running OS X 10.6.3 Displaying a Gaussian Shape after 174 Trials

After the clock skews of all IP addresses have been calculated, a confidence interval is determined for each IP address. We will use this interval to detect which IP addresses are dual-homed.

D. DETECTION OF DUAL-HOMED HOSTS

In Section E, Chapter III, we discussed the detection of multi-homed hosts using the estimated clock skews and a confidence interval of $1-\rho$. For detection purposes, ρ was selected to be 0.05 or 5%. The 95% confidence interval represents two standard deviations if the clock skews were a Gaussian distribution and is consistent with the confidence intervals used in [5]. After 174 trials were conducted and clock skews were calculated for each trial, a mean clock skew and confidence interval were calculated for each IP address. The upper and lower bounds of the confidence interval were determined using the *paramci* function in MATLAB. The code for this computation is included in

Appendix B. If the mean clock skew falls within the range of the bands, the IP addresses can be identified as coming from that host with a 95% percent confidence.

The results for the confidence interval testing are listed in Table 3. From the table, it can be shown that the single-homed devices had estimated clock skews that did not correspond to another device, indicating that they were not multi-homed. Also from the table, six of the seven dual-homed hosts had two clock skews that fell within the range of the 95% confidence interval of their two IP addresses, which indicate that they were multi-homed. The large width of the confidence interval for host 11 is a direct result of the instability of clock skews shown in Figure 14. For confidence interval detection, we assumed that the resultant clock skews were Gaussian in nature. Host 11 histogram was bi-modal and did not resemble a bell curve, thus we were unable to apply our proposed scheme for multi-homed detection.

Table 3. Linear Programming: Upper and Lower Bounds of the 95% Confidence Interval for the Clock Skews of All Hosts

Host	IP Address	C_{Upper}	Mean Clock Skew	C_{Lower}
1	10.10.8.1	13.0581	13.0439	13.0297
2	10.10.8.2	9.7448	9.6775	9.6103
3	10.10.8.3	15.0749	15.0641	15.0532
4	10.10.8.4	14.4429	14.4241	14.4052
5	10.10.8.5	11.0082	10.8393	10.6704
6	10.10.8.6	15.9405	15.9291	15.9177
7	10.10.8.10	13.9569	13.9489	13.9409
	10.10.8.11	13.9575	13.9496	13.9418
8	10.10.8.12	-4.6216	-4.6416	-4.6615
	10.10.8.13	-4.622	-4.6413	-4.6607
9	10.10.8.14	7.1852	7.1768	7.1685
	10.10.8.15	7.1849	7.1765	7.1681
10	10.10.8.16	6.5822	6.5643	6.5464
	10.10.8.17	6.5831	6.5651	6.5472
11	10.10.8.18	148.7956	83.5445	18.2935
	10.10.8.19	148.786	83.5335	18.2809
12	10.10.8.20	18.2564	18.2447	18.2331
	10.10.8.21	18.2571	18.2456	18.234
13	10.10.8.24	-1235.0783	-1239.7471	-1244.4159
	10.10.8.25	-1235.077	-1239.7429	-1244.4089

For the clock skews that were determined using the linear regression method, the results for the upper and lower bounds for the 95% confidence interval are shown in Table 5. Similar to the linear programming results, the single-homed devices all had estimated clock skews that did not correspond to another device. From Table 5, we can also see that six of the seven dual-homed devices were multi-homed. Host 11 confidence interval spans more than 130 ppm due to the bi-modal shape and is too wide for useful detection. Like the linear programming method, linear regression clock skew calculation did not allow us to fingerprint host 11.

Table 4. Linear Regression: Upper and Lower Bounds of the 95% Confidence Interval for the Clock Skews of All Hosts

Host	IP Address	C_{Upper}	Mean Clock Skew	C_{Lower}
1	10.10.8.1	13.0596	13.0449	13.0302
2	10.10.8.2	9.6086	9.5989	9.5893
3	10.10.8.3	15.0766	15.0654	15.0542
4	10.10.8.4	14.4274	14.4149	14.4025
5	10.10.8.5	10.5404	10.5273	10.5142
6	10.10.8.6	15.9427	15.9311	15.9194
7	10.10.8.10	13.9563	13.9483	13.9402
	10.10.8.11	13.9575	13.9495	13.9415
8	10.10.8.12	-4.6205	-4.6404	-4.6603
	10.10.8.13	-4.6178	-4.6387	-4.6597
9	10.10.8.14	7.1844	7.1763	7.1683
	10.10.8.15	7.1821	7.1719	7.1617
10	10.10.8.16	6.6142	6.5445	6.4748
	10.10.8.17	6.617	6.5449	6.4728
11	10.10.8.18	153.948	88.965	23.982
	10.10.8.19	159.1458	94.3519	29.5579
12	10.10.8.20	18.2666	18.2521	18.2375
	10.10.8.21	18.4375	18.3131	18.1888
13	10.10.8.24	-1238.0777	-1242.5625	-1247.0473
	10.10.8.25	-1238.0667	-1242.55	-1247.0333

Upon closer inspection of Table 3 and Table 4, it appears that the linear programming and the linear regression clock skew mean, upper, and lower confidence interval bounds are very similar. Due to these similarities, we will only discuss the graphical portion of the linear programming results in the following.

We plotted the mean clock skews and the confidence intervals for all IP addresses and displayed that data in Figure 16. Notably, the upper and lower confidence interval bounds for host 11 (IP address 10.10.8.18 and 10.10.8.19) cover a range of approximately 130 ppm. Due to the scale of Figure 16, the calculated confidence intervals appear as points rather than intervals.

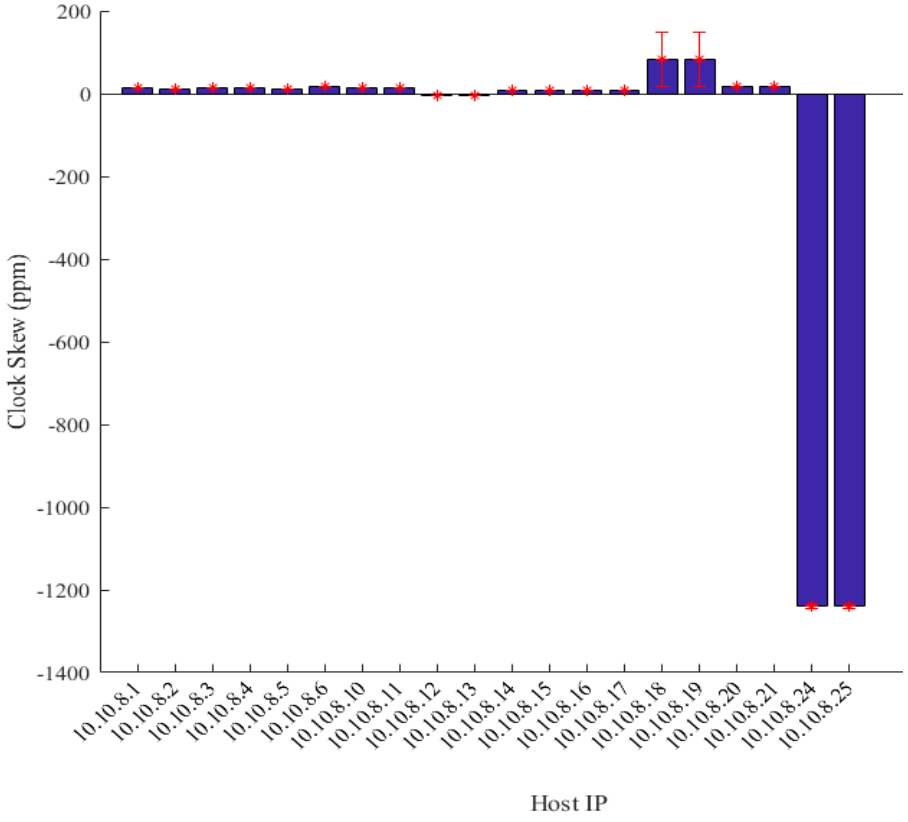

Figure 16. 95% Confidence Interval for the Estimated Clock Skews of All Hosts after 174 Trials

In Figure 17, we zoomed in on Figure 16 to focus on host 7 (IP addresses 10.10.8.10 and 10.10.8.11). The blue horizontal line is the confidence interval. Although it appears to be a single line, there are in fact two. The inset displayed further zooms in on the clock skews of host 7 showing the upper and lower bounds of the 95% confidence interval. Host 7 confidence interval only overlaps with its dual-homed counterpart and does not overlap with the other IP addresses.

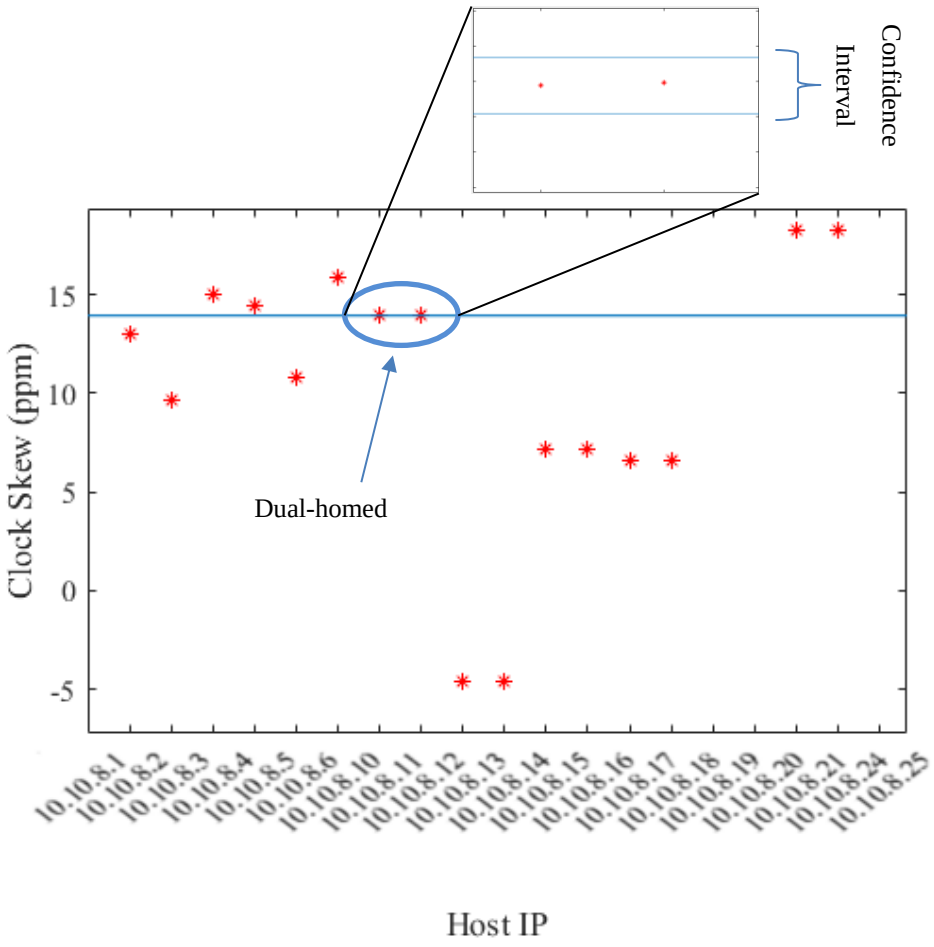

Figure 17. Confidence Interval of Host 7 (10.10.8.10) Compared to the Mean Values of All Clock Skews Calculated

A zoomed-in view of the confidence interval of host 11 (IP addresses 10.10.8.18 and 10.10.8.19) running OS X El Capitan is shown in Figure 18. The upper and lower bounds, which are represented by the blue horizontal lines, span approximately 140 ppm

and do not provide a reliable confidence interval for detection. The other hosts had intervals that generally spanned less than 0.3 ppm with the second largest span of 9 ppm for host 13. With the exception of host 11, the estimated clock skews calculated from linear programming and linear optimization provided reliable results that allowed us to successfully detect all multi-homed devices within the network.

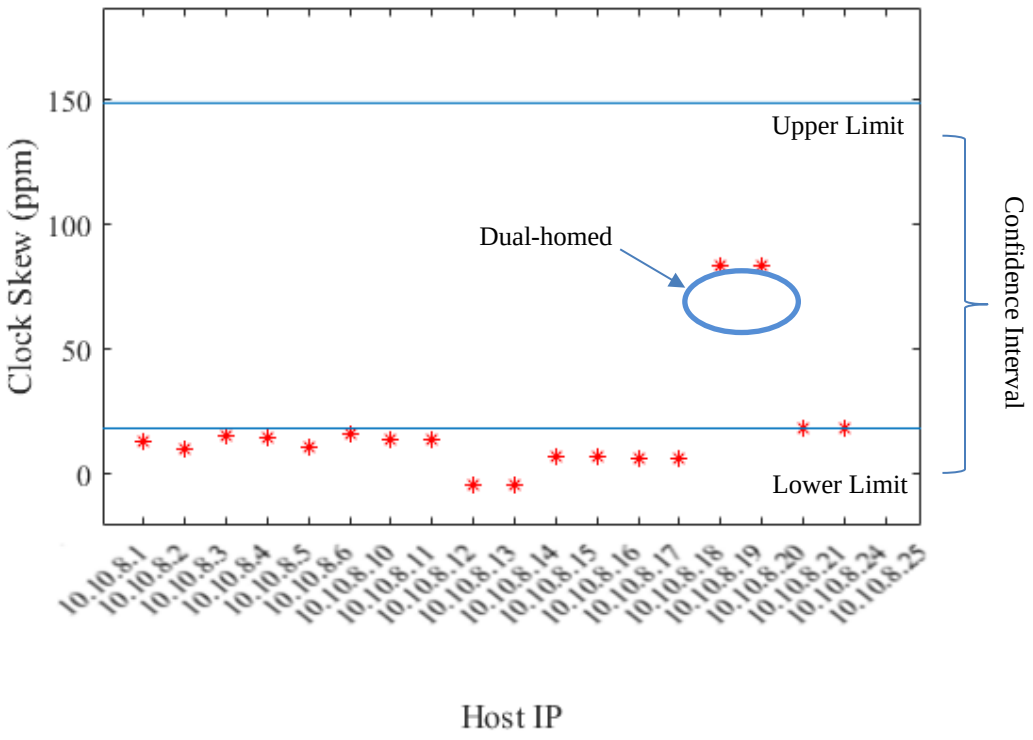

Figure 18. Confidence Interval of Host 11 (10.10.8.18) Compared to the Mean Values of All Clock Skews Calculated

This chapter focused on detecting dual-homed devices using calculated clock skews. A detailed description of the test bed, testing, and results was presented. The testing and analysis revealed that remote multi-homed detection is possible if the target clock skews result in a Gaussian shape. When the host histogram does not approach a Gaussian shape, the confidence interval detection method proposed in Chapter III is not sufficient. The next chapter focuses on expanding discussed detection methods to triple-homed devices and vertical testing.

V. TRIPLE–HOMED TESTING AND RESULTS

The dual-homed test bed was altered for triple-home and vertical testing. The chapter begins with a description of the triple-homed test bed. A triple-homed device consists of three separate Ethernet connections that are made using a combination of the built-in Ethernet port and two additional USB Ethernet NICs. Vertical testing is completed by maintaining constant hardware configuration on a device and varying the operating system.

The triple-homed testing was divided into three phases. In each phase, a different set of operating systems is tested while keeping the hardware configuration the same. Triple-homed vertical testing is conducted to determine the effects the operating system has on the clock skew and subsequent confidence interval based detection.

A. TRIPLE–HOMED TEST BED

In this network, we removed all single and dual-homed devices and tested three triple-homed devices. The testing was done in three phases. In each phase, a different operating system was installed on the host device. The first phase consisted of Windows 10, Windows 7, and Linux Mint 18.1. The second phase tested three versions of Ubuntu with various Linux kernels. The third phase consisted of freeBSD 12.0 installed on all devices. The setup for the network is summarized in Table 5, listing the host number, the various operating systems installed, and the IP addresses assigned to that device. To avoid confusion with the host numbers used in Chapter IV, the host numbers for testing in this chapter begin with 101.

A general network diagram for the triple-homed testing is shown in Figure 19, which lists the devices used, host number, and the IP addresses that were assigned. The switch and fingerprinter used for TCP traffic generation and traffic collection are the same as the ones used in Chapter IV.

Table 5. Summary of Hosts and Operating Systems for Triple-Homed and Vertical Testing

Host	Device/Hardware	Phase 1	Phase 2	Phase 3	IP Addresses
101	Lenovo IdeaPad U430	Windows 10	Ubuntu 16.04	freeBSD 12.0	10.10.8.31 10.10.8.32 10.10.8.33
102	Dell Latitude E6430	Windows 7	Ubuntu 14.04	freeBSD 12.0	10.10.8.34 10.10.8.35 10.10.8.36
103	Dell Latitude E6540	Linux Mint 18.1	Ubuntu 12.04	freeBSD 12.0	10.10.8.37 10.10.8.38 10.10.8.39

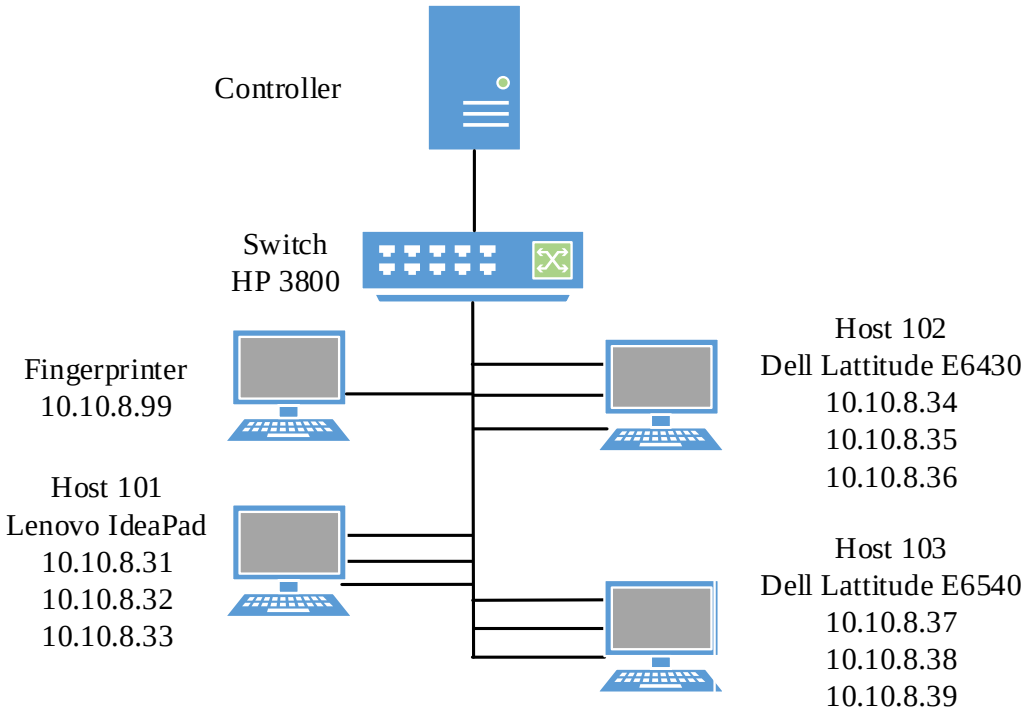

Figure 19. SDN Test Bed for Triple-Homed Testing

B. TRIPLE-HOMED CLOCK SKEW RESULTS

TCP timestamp data was generated, collected, and processed in the same manner as the dual-homed case. Each trial had a minimum of 3600 samples per IP address, which was approximately 60 minutes of data collection in a given trial. Using linear programming and linear regression methods, clock skews were calculated. The results from each IP address were compared to one another to determine which device the estimated clock skews originated from. The same linear programming and linear regression methods used for dual-homed testing were also used for the triple-homed case. For consistency between the results, only the upper solution was considered. The results from phase one and phase two will be presented together as they had similar results.

1. Phase One and Phase Two Results

The estimated clock skews for all IP addresses in the first two phases behaved similarly to the dual-homed cases. An example of the output from host 103 (IP address 10.10.8.38), running Linux Mint 18.1, is shown in Figure 20. The blue line is the upper bound solution of linear programming, and the red dots are the individual observed offsets.

Clock skews originating from the same device will have the same slope [5]. All hosts clock skews from phase one testing are shown in Figure 21. Due to the similarities and consistent results between phase one and two, we will only discuss the results of phase one testing. Each of the lines represents an IP address and similar colored lines originate from the same device. Graphically, it can be seen that the parallel clock skews for the triple-homed hosts have the same slope. Looking at host 101, the three blue lines all had a slope of about 16.0 ppm. Hosts 102 (green) and 103 (magenta) had negative slopes of -8.1 and -10.9, respectively. Similar to the dual-homed case, the difference in the location of the *y*-intercept is due to the measured timestamp delay $\epsilon(t)$.

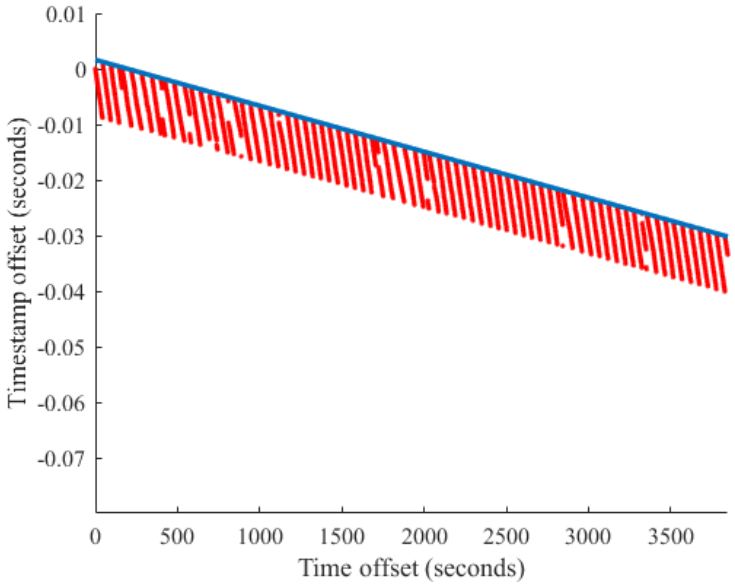

Figure 20. Upper-Bound Clock Skew Solution for Triple-Homed Host 103 (10.10.8.11) Running Linux Mint 18.1 for a Single Trial during Phase One Testing

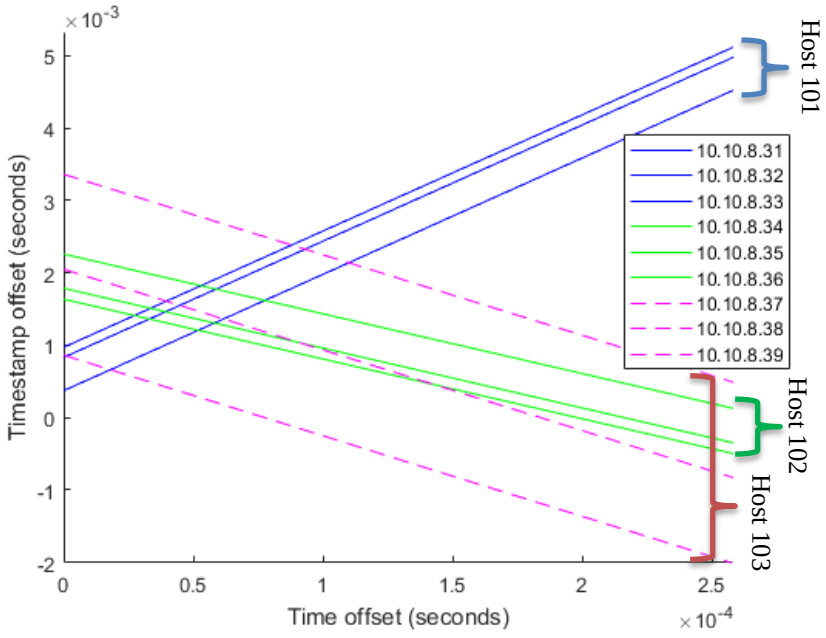

Figure 21. Triple-Homed Clock Skews of All Hosts from Phase One Testing

The mean clock skew values for all hosts in phase one and phase two testing are listed in Table 6 and Table 7, respectively. The clock skews for hosts 102 and 103 were negative values in phase one testing and switch to positive values for phase two testing as shown in Tables 6 and 7. The hardware configuration remained the same for all triple-homed hosts, thus the change in slope can be attributed to a change in the operating system. As expected, the IP addresses that share a host all had similar clock skews to one another. It can be noted that in these two phases, the estimated clock skews from both the linear programming and linear regression methods agree with one another.

The MATLAB functions *tic* and *toc* were used to calculate the computational time to execute the linear programming and linear regression methods. Similar to the dual-homed case, the linear regression computation time was 2-3 times faster than the linear programming method. This time difference is attributed to the differences in complexity of the two methods. Before we move onto triple-homed detection using confidence intervals, we will discuss the clock skew histograms of phase one and phase two results after 150 trials.

Table 6. Phase One: Mean Clock Skews for All Hosts Using Linear Programming and Linear Regression (in ppm)

Phase One					
Host	Device	Operating System	IP Address	Linear Programming Clock Skews	Linear Regression Clock Skews
101	Lenovo IdeaPad U430	Windows 10	10.10.8.31	16.0558	16.0680
			10.10.8.32	16.0573	16.0687
			10.10.8.33	16.0571	16.0684
102	Dell Latitude E6430	Windows 7	10.10.8.34	-8.1262	-8.1360
			10.10.8.35	-8.1252	-8.1582
			10.10.8.36	-8.1231	-8.1275
103	Dell Latitude E6540	Linux Mint 18.1	10.10.8.37	-10.9307	-10.9795
			10.10.8.38	-10.9333	-10.9730
			10.10.8.39	-10.9352	-10.9724

Table 7. Phase Two: Mean Clock Skew for All Hosts Using Linear Programming and Linear Regression (in ppm)

Phase Two					
Host	Device	Operating System	IP Address	Linear Programming Clock Skews	Linear Regression Clock Skews
101	Lenovo IdeaPad U430	Ubuntu 16.04	10.10.8.31	17.1122	17.1429
			10.10.8.32	17.1114	17.1468
			10.10.8.33	17.1133	17.2005
102	Dell Latitude E6430	Ubuntu 14.04	10.10.8.34	13.7592	13.6835
			10.10.8.35	13.7600	13.6574
			10.10.8.36	13.7587	13.6665
103	Dell Latitude E6540	Ubuntu 12.04	10.10.8.37	10.3061	10.3024
			10.10.8.38	10.3077	10.3023
			10.10.8.39	10.3061	10.3043

After 150 trials were conducted for each phase, a histogram for each IP address was reviewed. Histograms of all of the hosts had a Gaussian-like shape and are similar to those of the dual-homed hosts. A representative histogram generated is presented in Figure 22 from host 101 (IP address 10.10.8.33) running Windows 10. Before we move to confidence interval based detection, we will review the results of phase three testing.

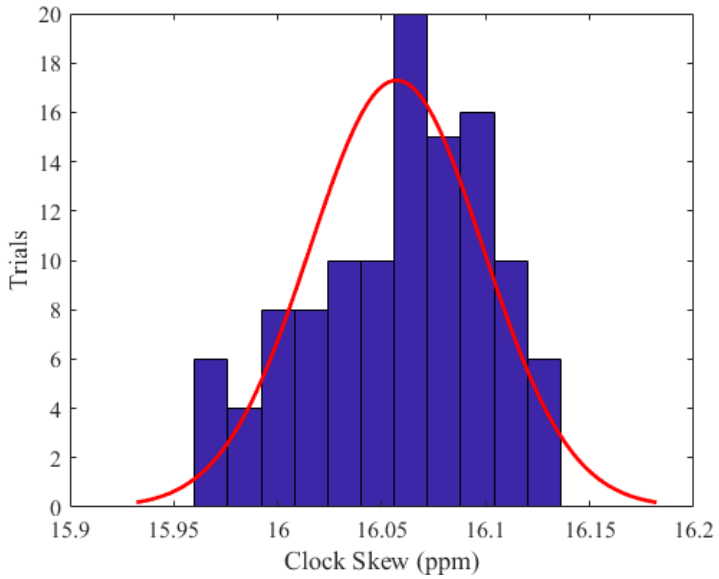

Figure 22. Histogram of Estimated Clock Skews for Host 101 (10.10.8.33) Running Windows 10 Displaying a Gaussian Shape after 150 Trials

2. Phase Three Results

FreeBSD 12.0 was installed on all three triple-homed devices for phase three testing. TCP timestamp data was collected in the same manner as in the previous two phases. The estimated clock skews varied on the order of $\pm 10^6$ ppm, unlike the triple-homed clock skews from phases one and two, which presented stable and consistent clock skews. The upper-bound solution for host 103 (IP address 10.10.8.38) with timestamp offsets varied between -4×10^6 and 1.6×10^6 seconds and is shown in Figure 23. All IP addresses produced similar upper-bound clock skew solutions as the one in Figure 24.

To generate TCP traffic, h*ping3* is called once per second. Each time it is called, it creates a TCP connection with the host, sends one packet to retrieve a TCP timestamp, and immediately closes the connection. It appears that freeBSD 12.0 is randomly assigning a timestamp value each time a TCP connection is made to the host causing large variations in the offset. Randomized TCP timestamps explains the extreme variations of timestamp offsets observed in Figure 23.

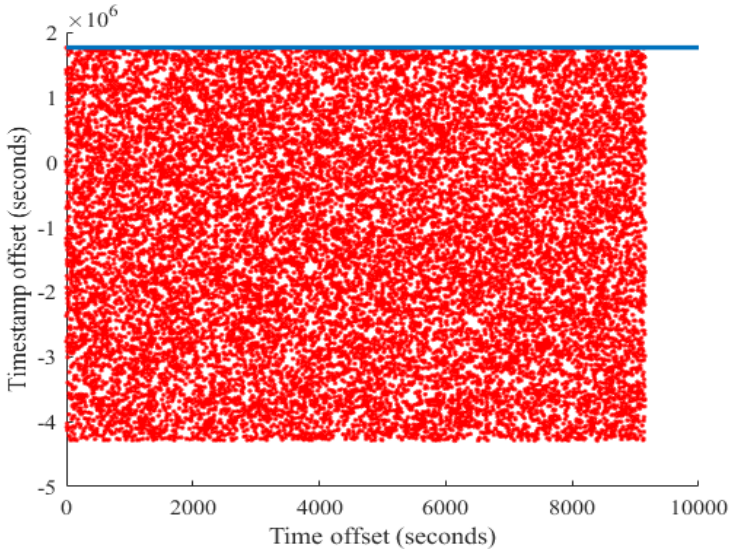

Figure 23. Upper-Bound Solution for Host 102 (10.10.8.38) Running freeBSD 12.0 over a Single Trial

The instability of the clock skews for freeBSD 12.0 is apparent when we plot all of the estimated clock skews from a single trial as shown in Figure 24. Similar to the color scheme used in Figure 21, each color represents a different host. Looking at host 103 (magenta lines), they clearly are not parallel as the clock skews cross one another between 3 and 4 seconds. Host 101 only has two lines plotted as the third line had a *y*-intercept below -3×10^4. Unlike the clock skews from phase one and phase two, the clock skews originating from the same host are not parallel. Due to the instability of the clock skews, the current method of confidence interval detection is not sufficient to fingerprint devices running freeBSD 12.0.

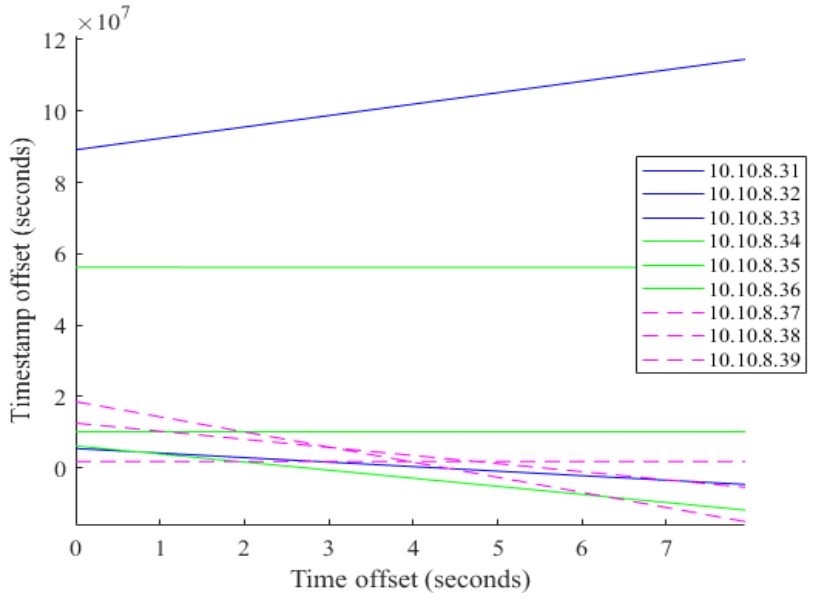

Figure 24. Non-parallel Clock Skews from a Single Trial during Phase Three Testing

The results are summarized in Table 8. It can be seen that the clock skews from the same host are not similar. For example, looking at host 101, the mean clock skews from linear programming ranges from a high of 1234354.91 ppm to a low of –2167570.96 ppm. For the same host, the linear regression mean clock skews range between 75923751.99 ppm and –32972297.96 ppm. Hosts 102 and 103 also had a wide

variation in the mean clock skews from both linear programming and linear regression. Now that we have calculated mean clock skew data from all of the phases, we can apply them to our detection scheme to determine which IP addresses are triple-homed.

Table 8. Phase Three: Mean Clock Skews for All Hosts Using Linear Programming and Linear Regression (in ppm)

Phase Three					
Host	Device	Operating System	IP Address	Linear Programming Clock Skews	Linear Regression Clock Skews
101	Lenovo IdeaPad U430	FreeBSD 12.0	10.10.8.31	1234354.91	75923751.99
			10.10.8.32	-867187.70	-32972297.96
			10.10.8.33	-2167570.96	7362498.01
102	Dell Latitude E6430	FreeBSD 12.0	10.10.8.34	-598920.03	1667652.00
			10.10.8.35	-1985387.07	7238501.11
			10.10.8.36	-519820.73	26827071.88
103	Dell Latitude E6540	FreeBSD 12.0	10.10.8.37	-816815.26	-53436558.43
			10.10.8.38	-1878372.71	-22952129.85
			10.10.8.39	-927646.91	7604539.66

C. DETECTION OF TRIPLE-HOMED HOSTS

In the previous chapter, we looked at detecting dual-homed devices by determining the upper and lower bounds of the confidence interval from multiple trials. We apply the same method to detect triple-homed hosts. The confidence intervals were calculated for both the linear programming and linear regression methods, and like the dual-homed confidence interval results, both methodologies produced similar results for phases one and two. Due to the similarities in interval values, only the linear programming results will be shown here. The upper and lower confidence interval bounds from both methods are summarized in Tables 9–11.

Looking at Tables 9 and 10, the mean clock skews from phase one and phase two testing are stable and consistent within a host. For example, in phase two testing, host 103 mean clock skew was 10.3061, 10.3077, and 10.3061 ppm across different interfaces. These clock skews are within 0.002 ppm with one another. The other hosts within phase

one and phase two shared consistent results within the same host. The upper and lower confidence bounds were also consistent within a host from phase one and phase two. Looking at Tables 9 and 10, we can see that the upper and lower confidence intervals are all within 0.4% of their associated mean clock skew. As discussed in the previous section, the phase three results (shown in Table 11) were not as consistent as the phase one and phase two results.

Table 9. Phase One: Upper and Lower Bounds of the 95% Confidence Interval for the Clock Skews of All Hosts

Phase One						
Host	Device	Operating System	IP Address	C_{Upper}	Mean Clock Skew	C_{Lower}
101	Lenovo IdeaPad U430	Windows 10	10.10.8.31	16.0631	16.0558	16.0485
			10.10.8.32	16.0653	16.0573	16.0493
			10.10.8.33	16.0648	16.0571	16.0493
102	Dell Latitude E6430	Windows 7	10.10.8.34	-8.1139	-8.1262	-8.1384
			10.10.8.35	-8.1125	-8.1252	-8.1379
			10.10.8.36	-8.1103	-8.1231	-8.1360
103	Dell Latitude E6540	Linux Mint 18.1	10.10.8.37	-10.9110	-10.9307	-10.9505
			10.10.8.38	-10.9141	-10.9333	-10.9525
			10.10.8.39	-10.9159	-10.9352	-10.9546

Table 10. Phase Two: Upper and Lower Bounds of the 95% Confidence Interval for the Clock Skews of All Hosts

Phase Two						
Host	Device	Operating System	IP Address	C_{Upper}	Mean Clock Skew	C_{Lower}
101	Lenovo IdeaPad U430	Ubuntu 16.04	10.10.8.31	17.1208	17.1122	17.1036
			10.10.8.32	17.1202	17.1114	17.1027
			10.10.8.33	17.1219	17.1133	17.1048
102	Dell Latitude E6430	Ubuntu 14.04	10.10.8.34	13.7732	13.7592	13.7452
			10.10.8.35	13.7737	13.7600	13.7462
			10.10.8.36	13.7724	13.7587	13.7450
103	Dell Latitude E6540	Ubuntu 12.04	10.10.8.37	10.3250	10.3061	10.2873
			10.10.8.38	10.3268	10.3077	10.2887
			10.10.8.39	10.3252	10.3061	10.2870

Table 11. Phase Three: Upper and Lower Bounds of the 95% Confidence Interval for the Clock Skews of All Hosts

Phase Three						
Host	Device	Operating System	IP Address	C_{Upper}	Mean Clock Skew	C_{Lower}
101	Lenovo IdeaPad U430	FreeBSD 12.0	10.10.8.31	4537083.91	1234354.91	-2068374.10
			10.10.8.32	250934.89	-867187.70	-1985310.30
			10.10.8.33	971101.31	-2167570.96	-5306243.23
102	Dell Latitude E6430	FreeBSD 12.0	10.10.8.34	-256229.61	-598920.03	-941610.45
			10.10.8.35	-638958.50	-1985387.07	-3331815.63
			10.10.8.36	2918937.35	-519820.73	3958578.82
103	Dell Latitude E6540	FreeBSD 12.0	10.10.8.37	754059.84	-816815.26	-2387690.36
			10.10.8.38	-57932.72	-1878372.71	-3698812.70
			10.10.8.39	-383871.51	-927646.91	-1471422.30

The phase three mean clock skew did not produce consistent results. Due to the inconsistent clock skews, the upper and lower bounds of the confidence intervals varied widely within a host. As shown in Table 11 for host 102, the upper bounds ranged from 2918937.35 ppm to –638958.50 ppm and the lower bounds ranged from –941610.45 ppm to 3958578.82 ppm. With these extreme ranges, it is not possible to fingerprint a device, which will be further shown in the confidence interval plots.

A plot of the 95% confidence interval is shown in Figure 25 for host 101 (IP address 10.10.8.31) during phase one testing. Here, we can see that the confidence intervals for the triple-homed case suggest that the IP addresses 10.10.8.31, 10.10.8.32, and 10.10.8.33 originated from host 101. It can also be seen that 10.10.8.34, 10.10.8.35, and 10.10.8.36 originated from host 102. And finally, 10.10.8.37, 10.10.8.38, and 10.10.8.39 originated from host 103. This is consistent with the network configuration in Figure 19.

The blue horizontal line is the confidence interval. Although it appears to be a single line, there are in fact two. The inset box displayed further zooms in on the clock skews of host 101 showing the upper and lower bounds of the 95% confidence interval. Host 101 confidence interval only overlaps with its triple-homed counterpart and does not overlap with the other IP addresses. The scheme for multi-homed detection works for

phase one and phase two testing; however, the same scheme does not work for the freeBSD devices in phase three.

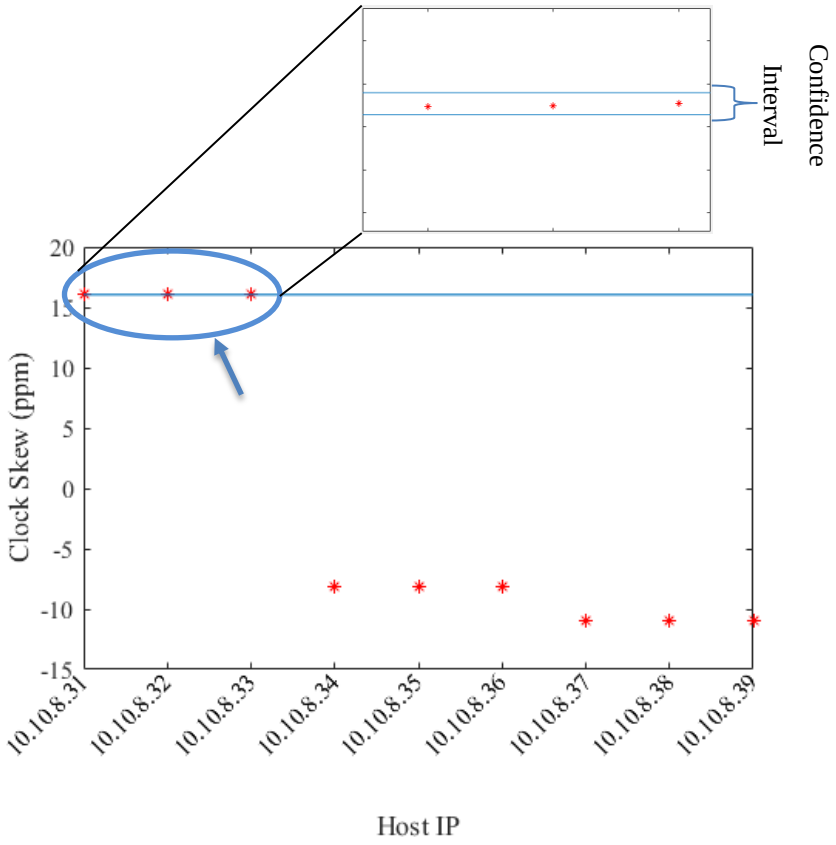

Figure 25. Confidence Interval of Host 101 (10.10.8.31) Compared to the Mean Values of All Clock Skews Calculated

When looking at the same 95% confidence interval for phase three, the upper and lower bounds of the confidence intervals cover a wide clock skew range that is not useful for discriminating instances of multi-homed connections. The mean estimated clock skews and 95% confidence intervals for all hosts in phase three testing were plotted in Figure 26. In the figure, the blue boxes and red stars are the mean clock skews. The whiskers are the upper and lower bands of the 95% confidence intervals centered about the mean clock skew. The IP addresses in Figure 26 correspond to the hosts listed in Table 11.

From Figure 26, we see that IP address 10.10.8.34 (host 102) has the smallest 95% confidence interval of 685381 ppm. Compare this to the average interval size from phase one and phase two testing of 0.027 ppm. The largest confidence interval, in Figure 26, originates from host 101 (IP address 10.10.8.31) with a value of 6.605×10^6 ppm. As shown, all hosts running freeBSD 12.0 experienced inconsistent clock skews, which resulted in a wide range of confidence interval values.

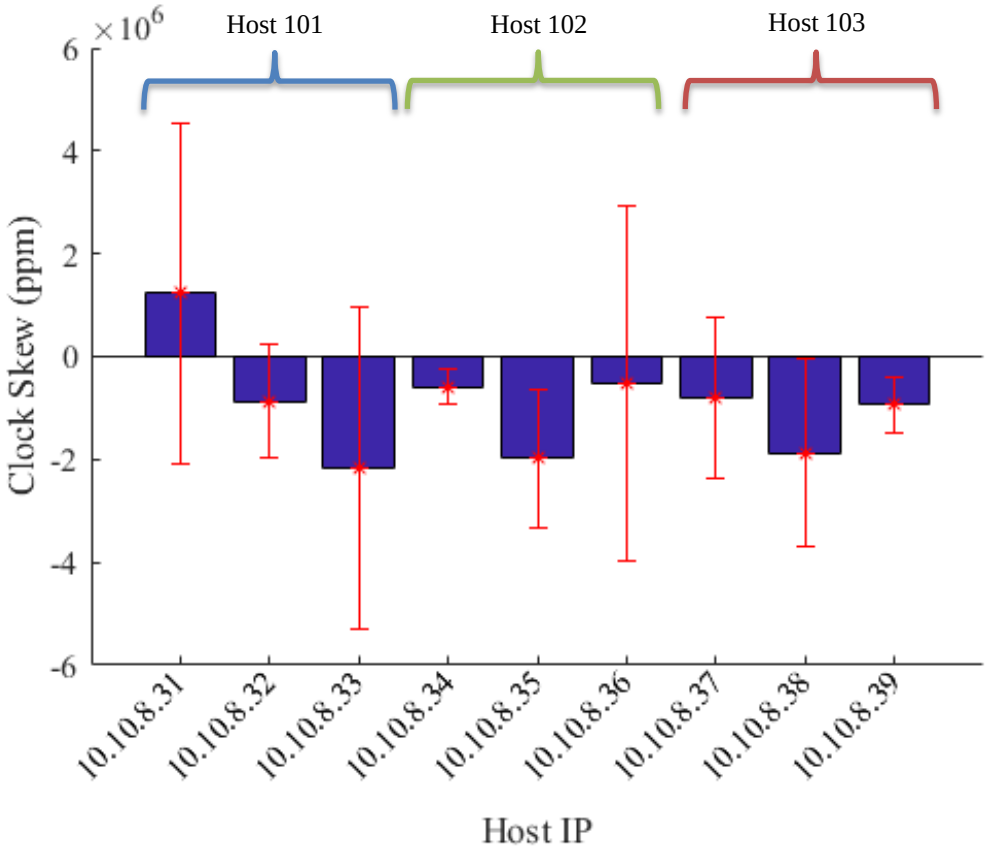

Figure 26. Mean Clock Skews and Confidence Interval Bounds of All Hosts during Phase Three Testing

D. VERTICAL TESTING RESULTS

The results of the vertical tests for host 101 (Lenovo IdeaPad) are listed in Table 12. When a device is rebooted to a new operating system, the clock skews change for that

host. In Table 12, host 101 has three separate clock skews for each of the operating systems that are installed. This supports the idea that the estimated clock skew depends on both the imprecisions of the CPU (in manufacturing) and the operating system that is running on the device.

Table 12. Mean Linear Programming and Linear Regression Clock Skews for Host 101 – Lenovo IdeaPad U430 Running Various Operating Systems

Operating System	IP address	Linear Programming Clock Skews	Linear Regression Clock Skews
Windows 10	10.10.8.31	16.0558	16.0680
	10.10.8.32	16.0573	16.0687
	10.10.8.33	16.0571	16.0684
Ubuntu 16.04	10.10.8.34	17.1122	17.1429
	10.10.8.35	17.1114	17.1468
	10.10.8.36	17.1133	17.2005
freeBSD 12.0	10.10.8.37	1234354.91	75923751.99
	10.10.8.38	-867187.70	-32972297.96
	10.10.8.39	-2167570.96	7362498.01

Both host 102 (Dell Latitude E6430) and host 103 (Dell Latitude E6540) yielded similar results in that they showed an analogous trend when the device was rebooted to a different operating system; the estimated clock skews differed from operating system to operating system while on the same device. The results for host 102 and 103 are in Tables 13 and Table 14, respectively. From Table 13, we can see the stability of clock skews when the host 102 was running Windows 7 or Ubuntu 14.04. The three clock skews within the same operating system are consistent with one another. When host 102 was running freeBSD 12.0, the clock skews become erratic. The results from host 102 show that the device displayed different clock skews when a new operating system was installed on the device.

The results in Table 14 show a similar trend that Table 12 and 13 held. The clock skews for host 103 also changed when a new operating system was booted onto the device. All three hosts shared similar results when booted to a new operating system. The

results of vertical testing is consistent with the findings from [7] where they observed differences in the clock skews of computers when booted from a different operating system.

Table 13. Mean Linear Programming and Linear Regression Clock Skews for Host 102 – Dell Latitude E6430 Running Various Operating Systems

Operating System	IP address	Linear Programming Clock Skews	Linear Regression Clock Skews
Windows 7	10.10.8.31	-8.1262	-8.1360
	10.10.8.32	-8.1252	-8.1582
	10.10.8.33	-8.1231	-8.1275
Ubuntu 14.04	10.10.8.34	13.7592	13.6835
	10.10.8.35	13.7600	13.6574
	10.10.8.36	13.7587	13.6665
freeBSD 12.0	10.10.8.37	-598920.03	1667652.00
	10.10.8.38	-1985387.07	7238501.11
	10.10.8.39	-519820.73	26827071.88

Table 14. Mean Linear Programming and Linear Regression Clock Skews for Host 103 – Dell Latitude E6540 Running Various Operating Systems

Operating System	IP address	Linear Programming Clock Skews	Linear Regression Clock Skews
Linux Mint 18.1	10.10.8.31	-10.9307	-10.9795
	10.10.8.32	-10.9333	-10.9730
	10.10.8.33	-10.9352	-10.9724
Ubuntu 12.04	10.10.8.34	10.3061	10.3024
	10.10.8.35	10.3077	10.3023
	10.10.8.36	10.3061	10.3043
freeBSD 12.0	10.10.8.37	-816815.26	-53436558.43
	10.10.8.38	-1878372.71	-22952129.85
	10.10.8.39	-927646.91	7604539.66

In summary, this chapter focused on multi-homed detection of triple-homed devices. A detailed description of triple-homed test bed, testing, and results was discussed. The testing and analysis of this chapter further confirmed that the remote

multi-homed detection method outlined in Chapter III is feasible if the target clock skews are consistent and have a Gaussian shape. When the timestamp offset values of the host do not produce a Gaussian shape, the confidence interval detection method proposed is not sufficient to identify which IP addresses originated from a given host. This chapter also showed that the clock skew from a device depends on the hardware and the operating system that is installed. The next chapter summarizes the findings and proposes possible future work.

VI. CONCLUSION

Multi-homed devices pose a threat to networks protected by firewalls and physical separation schemes. The primary motivation for this research was to identify hosts that are capable of bypassing firewalls and physical separation security measures by employing multi-homed connections. The idea of using clock skews to detect multi-homed devices was first presented in [5] as the clock skews were shown to be unique and remained relatively constant over long periods of time. Continuing this work, we validated the detection scheme proposed in [5] and applied it to a larger set of devices and operating systems.

To expand the applicability of the detection scheme, we built an SDN and connected 13 multi-homed devices running 11 different operating systems. We performed dual-homed testing by passively collecting TCP timestamps and calculating an estimated clock skew using linear programming and linear regression methods. The clock skews were then used to calculate a mean clock skew and a confidence interval. The confidence interval detection was used to determine which IP addresses were multi-homed.

For completeness, we tested the detection scheme on triple-homed devices by altering the dual-homed test bed to support triple-homed testing and vertical testing. Vertical testing was conducted to determine the effects the operating system has on the clock skew and subsequent confidence interval based detection. The operating systems that were selected represented a wide range of what was available at the time of testing.

A. SIGNIFICANT RESULTS

We were able to remotely collect traffic and determine the clock skews for devices using both linear programming and linear regression methods. To determine computation times for clock skew calculations, we used the built-in *tic* and *toc* functions in MATLAB. The linear regression method was 2–3 times faster at calculating clock skews when compared to linear programming. As long as network delays are minimized,

the use of linear regression may prove beneficial, as it was less computationally intensive and less complex to implement than linear programming. When clock skews were stable and consistent, the results from linear programming and linear regression were consistent with one another.

We were able to identify dual and triple-homed hosts when running various operating systems as long as the calculated clock skews from a host were consistent. Through vertical testing, we determined that clock skew consistency depends on the operating system installed and not on the hardware. Our results showed that a device can exhibit consistent clock skews under one operating system and erratic and inconsistent clock skews under a different operating system while keeping the hardware configuration the same.

All multi-homed devices were detected with clock skews, with the exception being OS X 10.11 El Capitan and freeBSD 12.0. In an older release of OS X 10.6, the clock skew was consistent and able to be fingerprinted. This led to the conclusion that Apple changed the way timestamp data is handled following the release of Snow Leopard in 2009. All attempts to fingerprint freeBSD 12.0 were unsuccessful as the resultant clock skews were inconsistent. Ultimately we were able to show that multi-homed detection proposed in [5] depends on the operating system installed and the consistency within the generation of clock skews.

B. RECOMMENDATIONS AND FUTURE WORK

We tested the proposed detection scheme on an SDN that was on the same subnet being controlled from one controller. The methodology used in this research requires the user to use a fingerprinter on each subnet installed on a network. The ultimate goal of this work would be to move the fingerprinting process to the controller and remove the need of using a stand-alone fingerprinting device. Moving the fingerprinting and detection to the controller has many benefits, which includes the capability to fingerprint all devices on a network regardless of the subnet.

The proposed scheme to identify if a host is connected to two physically separated networks simultaneously, as depicted in Figure 2, requires the use of two fingerprinters. The concern of using different fingerprinters is that the estimated clock skews depend on the target device and the fingerprinter. Since the two fingerprinters have independent clocks, a method must be determined in order for the results to be directly compared. Future effort would be to create a detection scheme that can be used between two fingerprinters to detect hosts connected to different networks.

We were unsuccessful in detecting multi-homed connections when the device was running newer versions of OS X and freeBSD 12.0. These two operating systems are BSD-based, however, another BSD-based operating system (Snow Leopard) did not exhibit these issues. Further investigation into the way in which TCP timestamps are generated within these operating systems will prove useful when designing a more robust detection scheme.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB CODE FOR CALCULATING CLOCK SKEW

The MATLAB code to calculate clock skews (see Section D, Chapter III) is contained below. The script imports an excel file that contains the observed times and raw timestamp data. From this data, the observed time offset x_i and the timestamp offset w_i are used to determine the observed offset y_i. The MATLAB function *linprog* is used to calculate linear programming clock skews, and the MATLAB function *fit* is used to calculate linear regression clock skews. The output data is a matrix of estimated clock skews for each IP address for one trial. The clock skews are then collected and saved in an EXCEL data table for confidence interval calculations.

```
%% Thesis Clock Skew Calculation Test Runs
close all
clear all
clc
format compact
format long
% Change default axes fonts to Times New Roman
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
% Change default text fonts to Times New Roman
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)
% Determine the time to execute the calculations
tic
%% Load the data from a single test run
run_name = 'z_Run_072.xlsm';
Data = xlsread(run_name);
% List of the host's last IP address, 10.10.8.*
hosts = [1,2,3,4,5,6,10,11,12,13,14,15,16,17,18,19,20,21,24,25];
L = length(hosts);
%% Calculate the Clock Skew and plot the data
for n = 1:L
  % Prints the current calcuation for the host and debugging
  fprintf('****************** %d ******************\n',hosts(n));
  % Checks the entire row for the the hosts IP address
  [row,~] = find(Data == hosts(n));
  % Extract data for a given IP address
  Hosts = Data(row,:);
  % Create a matrix for that data
  for k = 1:length(Hosts)
    x(k) = Hosts(k,1) - Hosts(1,1);
    % Calculate the time offset relative to the initial point listed
    v(k) = Hosts(k,3) - Hosts(1,3);
```

```
    % Calculate the timestamp offset relative to the initial point listed
  end
  b = ones(length(x),1);
  a = [x' b];
  f = [sum(x)/length(x) 1];
  I = linprog(f, -a, -v);
  % Solve the linear programing solution for frequency (Hz)

  for k = 1:length(Hosts)
    w(k) = v(k)/round(I(1));
    % Adjust v based on frequency the difference between observed and
    % actual time
    y(k) = w(k) - x(k);
  end
  z = linprog(f, -a, -y);
  % Linear programming solution for which reports the slope of O,
  % which is the clock skew
  Z(n) = z(1);
  B(n) = z(2);

  % plot of the reflines
  figure
  hold on
  plot(x,y,'r.')
  %plotting the upper bound limit of O
  h = refline(z(1),z(2));
  get(h, 'linewidth');
  set(h, 'linewidth', 2.5);
  title(['Clock Skew for host 10.10.8.' num2str(hosts(n)) ' '])
  xlabel('Time offset (seconds)','FontName','Times New Roman')
  ylabel('Timestamp offset (seconds)','FontName','Times New Roman')

  f = fittype('poly1');
  [fit1, gof, fitinfo] = fit(x',10^6*y','poly1');
  fit1

  % clear variables
  clear Hosts row x v b a f I w y z h
end

% Converts Z into ppm
Z = Z*1000000';
B = B';

% Output the clock skew values
format long
fprintf('Clock Skew \n')
fprintf('%8.10f\n' ,[Z']')

%% Compare Clock Skews between hosts
figure
axis([0 length(Data) 0 0.02])
axis([0 2000 0 0.02])
hold on

%% Compare Clock Skews between hosts - The example shown is only for 1 IP
address. The remaining code was omitted for brevity.
C1 = refline(Z(1),B(1));
```

```
get(C1, 'color');
set(C1, 'color', 'm');


% Plot all of the clock skews from a given run on single plot
legend('10.10.8.1 ','10.10.8.2 ','10.10.8.3 ','10.10.8.4 ','10.10.8.5 ',...
    '10.10.8.6 ','10.10.8.10','10.10.8.11','10.10.8.12','10.10.8.13',...
    '10.10.8.14','10.10.8.15','10.10.8.16','10.10.8.17','10.10.8.18',...
    '10.10.8.19','10.10.8.20','10.10.8.21','10.10.8.24','10.10.8.25',...
    'Location','best');
title('Comparison of clock skews for all hosts')
xlabel('Time offset (seconds)','FontName','Times New Roman')
ylabel('Timestamp offset (seconds)','FontName','Times New Roman')

fprintf(run_name);
fprintf('\n');
% End timing function
toc
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB CODE FOR CALCULATING CONFIDENCE INTERVALS

The MATLAB code to calculate the mean, upper bound, and lower bound of a confidence interval for a given IP address (see Section E, Chapter III) is listed below. The script imports an excel file that contains all of the estimated clock skews from each trial. The MATLAB function *paramci* is used to calculate the confidence intervals. The output data is a matrix of the mean, upper bound, and lower bound for each IP address. The histograms and confidence interval graphs, for each IP address, are also produced within the code.

```
%% Confidence Interval Calculation
close all
clear all
clc
format compact
format long
% Change default axes fonts.
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 12)
% Change default text fonts.
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 12)
%% Load the data from trials
data_name = 'ConfInt_linprog_data.xlsx';
data_name = 'ConfInt_regress_data.xlsx';
TrialData = xlsread(data_name)';
% List of the host's last IP address, 10.10.8.*
hosts = [1,2,3,4,5,6,10,11,12,13,14,15,16,17,18,19,20,21,24,25];
%% Caclulate the CI - only the first 2 IP address calculations are shown
here for brevity
pd1 = fitdist(TrialData(:,1),'Normal');
ci1 = paramci(pd1);
pd2 = fitdist(TrialData(:,2),'Normal');
ci2 = paramci(pd2);

%% Plots
%
mu = [pd1.mu; pd2.mu; pd3.mu; pd4.mu; pd5.mu; pd6.mu; pd10.mu; pd11.mu;...
  pd12.mu; pd13.mu; pd14.mu; pd15.mu; pd16.mu; pd17.mu; pd18.mu;...
  pd19.mu; pd20.mu; pd21.mu; pd24.mu; pd25.mu];
e = [pd1.mu-ci1(1,1); pd2.mu-ci2(1,1); pd3.mu-ci3(1,1); pd4.mu-ci4(1,1);...
  pd5.mu-ci5(1,1); pd6.mu-ci6(1,1); pd10.mu-ci10(1,1); pd11.mu-
ci11(1,1);...
  pd12.mu-ci12(1,1); pd13.mu-ci13(1,1); pd14.mu-ci14(1,1);...
  pd15.mu-ci15(1,1); pd16.mu-ci16(1,1); pd17.mu-ci17(1,1);...
  pd18.mu-ci18(1,1); pd19.mu-ci19(1,1); pd20.mu-ci20(1,1);...
```

```
  pd21.mu-ci21(1,1); pd24.mu-ci24(1,1); pd25.mu-ci25(1,1)];


hold on
bar(mu)
errorbar(mu,e,'r*')
set(gca,'XTickLabel',{'' '10.10.8.1' '10.10.8.2' '10.10.8.3' '10.10.8.4'...
  '10.10.8.5' '10.10.8.6' '10.10.8.10' '10.10.8.11' '10.10.8.12' ...
  '10.10.8.13' '10.10.8.14' '10.10.8.15' '10.10.8.16' '10.10.8.17' ...
  '10.10.8.18' '10.10.8.19' '10.10.8.20' '10.10.8.21' '10.10.8.24' ...
  '10.10.8.25'});
set(gca,'XTick',1:20,'XTickLabel','')
lab=[{'10.10.8.1'}; {'10.10.8.2'}; {'10.10.8.3'}; {'10.10.8.4'};
{'10.10.8.5'};...
  {'10.10.8.6'}; {'10.10.8.10'}; {'10.10.8.11'}; {'10.10.8.12'};...
  {'10.10.8.13'}; {'10.10.8.14'}; {'10.10.8.15'}; {'10.10.8.16'};...
  {'10.10.8.17'}; {'10.10.8.18'}; {'10.10.8.19'}; {'10.10.8.20'};...
  {'10.10.8.21'}; {'10.10.8.24'}; {'10.10.8.25'}];

hx = get(gca,'XLabel');
set(hx,'Units','data');
pos = get(hx,'Position');
y = pos(2);
X=1:20;
for i = 1:size(lab,1)
 t(i) = text(X(i),y,lab(i,:));
end

set(t,'Rotation',45,'HorizontalAlignment','right')
xlabel({'','','','','Host IP'});
ylabel('Clock Skew (ppm)')

%% Table of all hosts and their confidence interval
D=[mu+e,mu,mu-e]'

%% Display means of clock skews
fprintf('Clock Skew Mean\n')
fprintf('%8.10f\n' ,[mu]')

%% Dual Homed Detection
for i=1:length(mu)
  for N=1:length(mu)
    if mu(i)-e(i)<=mu(N) && mu(i)+e(i)>=mu(N)
      x(i,N)=1;
      x(N,N)=0;
    end
  end
end
%display(x)

%% Comparing Clock Skews
for i=1:length(mu)
  for N=1:length(mu)
    Y(i,N)=abs(mu(i)-mu(N));
  end
end

%% Graphic Depiction
```

```
x=1:20;

% plot 10.8.8.* - The code for
figure
plot(x,mu,'r*')
set(gca,'XTickLabel',{'' '10.10.8.1' '10.10.8.2' '10.10.8.3' '10.10.8.4'...
  '10.10.8.5' '10.10.8.6' '10.10.8.10' '10.10.8.11' '10.10.8.12' ...
  '10.10.8.13' '10.10.8.14' '10.10.8.15' '10.10.8.16' '10.10.8.17' ...
  '10.10.8.18' '10.10.8.19' '10.10.8.20' '10.10.8.21' '10.10.8.24' ...
  '10.10.8.25'});
set(gca,'XTick',1:20,'XTickLabel','')
lab=[{'10.10.8.1'}; {'10.10.8.2'}; {'10.10.8.3'}; {'10.10.8.4'};
{'10.10.8.5'};...
  {'10.10.8.6'}; {'10.10.8.10'}; {'10.10.8.11'}; {'10.10.8.12'};...
  {'10.10.8.13'}; {'10.10.8.14'}; {'10.10.8.15'}; {'10.10.8.16'};...
  {'10.10.8.17'}; {'10.10.8.18'}; {'10.10.8.19'}; {'10.10.8.20'};...
  {'10.10.8.21'}; {'10.10.8.24'}; {'10.10.8.25'}];
hx = get(gca,'XLabel');
set(hx,'Units','data');
pos = get(hx,'Position');
y = pos(2);
X=1:20;
for i = 1:size(lab,1)
 t(i) = text(X(i),y,lab(i,:));
end
set(t,'Rotation',45,'HorizontalAlignment','right')
xlabel({'','','','','Host IP'});
ylabel('Clock Skew (ppm)')
title(['Confidence Interval for Host 10.10.8.1'])
refline(0,ci1(1,1))
refline(0,ci1(2,1))

%% Histfit
for i=1:20
  d=TrialData(:,i);
  figure
  h=histfit(d,10);
  xlabel('Clock Skew (ppm)')
  ylabel('Trials')
  %title(['Trial Histogram for host 10.10.8.' num2str(hosts(i)) ' '])
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] D. Kreutz, F. Ramos, P. E. Verissimo, C. Rothenberg, and S. Azodolmolky, "Software-defined networking a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, April 2014.

[3] T. Parker, J. Jones, J. Mayberry, G. Chanman, Z. Staples, M. Tummala, and J. McEachen, "Defensive cyber operations in a software-defined network," in *49th Hawaii International Conference on Cyber Systems*, Koloa, HI, USA, 2016, pp. 5561–5568.

[4] *(U)RED/BLACK Installation Guidance*, CNSSAM TEMPEST/01-13, National Security Agency, Ft. Meade, MD, 2014, pp. 1–13.

[5] B. J. Martin, "Detecting a multi-homed device using clock skew," M.S. thesis, Dept. Elec. Eng., Naval Postgraduate School, Monterey, 2016.

[6] T. Khono, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, Jun. 2005.

[7] L. Polcak and B. Frankova, "On reliability of clock-skew-based remote computer identification," in *Security and Cryptography (SECRYPT), 2014 11th International Conference*, Vienna, Austria, 2014, pp. 18.

[8] E. Byres, "Dual homed machines are the juiciest targets," Tofino Security, 2010. https://www.tofinosecurity.com/blog/dual-homed-machines-are-juiciest-targets

[9] T. Benson, A. Aella, and D. Maltz, "Unraveling the complexity of network management"" in *Proceedings of the 6th USENIX symposium on Networked Systems Design and Implementation*, Berkeley, CA, 2009, pp. 335–348.

[10] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, pp. 36–43, July 2013.

[11] K. Bakshi, “Considerations for software defined networks (SDN): approaches and use cases,” in *Aerospace Conference*, Big Sky, MT, USA, 2013, pp. 1–8.

[12] J. Wang, W. M. Vokkarane, R. Jothi, X. Qi, B. Raghavachari, and J. P. Jue, "Dual-homing protection in IP-over-WDM networks," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3111–3124, October 2005.

[13] Jacobson, V., Braden, R., and D. Borman, “TCP extensions for high performance,” RFC 1323, DOI 10.17487/RFC1323, May 1992, http://www.rfceditor.org/info/rfc1323.

[14] Kernel.org. (2017, Aug. 1). Timestamping. [Online]. Available: https://www.kernel.org/ doc/Documentation/networking/timestamping.txt

[15] S. Zander and S. J. Murdoch, “An improved clock-skew measurement technique for revealing hidden services,” in *SS'08 Proceedings of the 17th Conference on Security Symposium*, San Jose, CA, 2008, pp. 211–225.

[16] NTP.Org. (2017, June). Network time protocol. [Online]. Available: http://www.ntp.org/ntpfaq/NTP-s-sw-clocks-quality.htm

[17] F. Lanze, A. Panchenko, B. Braatz, and A. Zinnen, “Clock skew based remote device fingerprinting demystified,” in *Global Communications Conference, 2012 IEEE*, Anahiem, CA, USA, 2012, pp. 813–819.

[18] M. Tummala and C. Therrien, *Probability and Random Processes for Electricaland Computer Engineers*. Boca Raton, FL: CRC Press, 2012.

[19] D. Kreutz et al., “Software-defined networking a comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[20] L. Polcak, J. Jirasek, and P. Matousek, “Comment on remote physical device fingerprinting,” *IEEE Transactions on Dependable and Secure Communications*, vol. 11, no. 5, pp. 494–496, Sep. 2014.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California