TRAFFIC PRIVACY STUDY ON INTERNET OF THINGS – SMART HOME APPLICATIONS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Ayan Patel

COMMITTEE MEMBERSHIP

TITLE: Traffic Privacy Study on Internet of Things – Smart Home Applications

AUTHOR: Ayan Patel

DATE SUBMITTED: August 2020

COMMITTEE CHAIR: DongFeng Fang
Professor of Computer Science

COMMITTEE MEMBER: Bruce DeBruhl
Professor of Computer Science

COMMITTEE MEMBER: Foaad Khosmood
Professor of Computer Science

ABSTRACT

Traffic Privacy Study on Internet of Things – Smart Home Applications

Ayan Patel

Internet of Things (IoT) devices have been widely adopted in many different applications in recent years, such as smart home applications. An adversary can capture the network traffic of IoT devices and analyze it to reveal user activities even if the traffic is encrypted. Therefore, traffic privacy is a major concern, especially in smart home applications. Traffic shaping can be used to obfuscate the traffic so that no meaningful predictions can be drawn through traffic analysis. Current traffic shaping methods have many tunable variables that are difficult to optimize to balance bandwidth overheads and latencies. In this thesis, we study current traffic shaping algorithms in terms of computational requirements, bandwidth overhead, latency, and privacy protection based on captured traffic data from a mimic smart home network. A new traffic shaping method - Dynamic Traffic Padding is proposed to balance bandwidth overheads and delays according to the type of devices and desired privacy. We use previous device traffic to adjust the padding rate to reduce the bandwidth overhead. Based on the mimic smart home application data, we verify our proposed method can preserve privacy while minimizing bandwidth overheads and latencies.

ACKNOWLEDGMENTS

I would like to acknowledge my advisor, Professor Dongfeng Fang, for her guidance at every stage of the process. I would like to acknowledge Dr. Khosmood for giving me access to the IoT dataset as well as serving on my committee. I would like to acknowledge Dr. DeBruhl for serving on my committee. I would like to acknowledge all the professors in the Computer Science department at Cal Poly who were instrumental in my educational experience and research path. I would like to thank our computer science lab technicians and my peers who supported me through my journey to obtain a degree.

I would like to thank my parents for encouraging me and giving me the opportunity to study at Cal Poly. I would like to thank my sister and family friends for supporting me through this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

This chapter first introduces the Internet of Things including their architecture and security issues. Then the motivation of the work is given which includes device identification and traffic analysis attacks. Finally, it introduces methods to prevent traffic analysis on IoT network traffic.

1.1 Internet of Things Background

The Internet of Things (IoT) is a system of connected computing devices that can communicate over the internet [1]. Prior to IoT most communication was human to human or human to machine, but IoT introduces machine to machine learning [2]. IoT devices for smart homes include speakers, TV's, game consoles, cameras, assistants, smart plugs, and more. All of these devices communicate over the home network constantly, whether it is with the cloud, other devices, or an individual. These devices have become very popular over the past few years. In 2018, 17 billion devices were connected to the internet worldwide, 7 billion of which were IoT devices [3]. The number of IoT devices connected to the internet by the end of 2020 is expected to be 10 billion, approaching 22 billion by 2025 [3]. McKinsey Global Institute predicts massive value growth for the Internet of Things in the U.S. reaching more than $11 trillion annually by 2025 [4].

IoT devices are not only applicable to smart homes. They are widely used in smart cities, healthcare, energy grids and utilities, manufacturing, transportation, and agriculture [5]. IoT devices are prevalent in healthcare services including health monitors, energy meters, and x-ray devices [5]. In industry, IoT robots have taken the place of

humans in manufacturing [5]. IoT devices add automation to tasks which improves efficiency as well as allows monitoring and control from remote locations [6]. IoT devices in a smart home provide convenience, safety, and efficiency for the users [7]. Some of these devices are connected to physical aspects of a home like locks. If an attacker is able to get into a smart lock, they will have access to the home itself [7].

1.1.1 IoT Architecture and Security

The Internet of Things architecture has three main layers: Cloud, Edge, and Things [7]. The Things layer consists of sensors and actuators connecting the cyber world to the physical world. Most devices in this layer are resource limited and cannot perform heavy computation [7]. Unlike the Things layer, the Cloud layer has enough resources to perform these heavy computations and run algorithms on large data-sets [7]. The Edge layer bridges the gap between the cloud and the things devices. The Edge devices have more computational power than the Things devices and can connect to the cloud more efficiently [7]. Sensitive data is constantly being sent between these layers to accomplish the ultimate goal of the IoT devices [8]. The nature of IoT devices including connection to the physical world, different operating systems and limited resources, and different communication methods and protocols make security more challenging than traditional devices [7]. Due to the vast number of IoT devices on the market, many of the low-end devices do not support strong security measures [9]. However, the majority of the consumers expect security to be built in to these devices and regard security as one of their top priorities [10]. As of January 2020, California has set into effect a law requiring all IoT devices to have 'reasonable security features' [11]. What this entails is still open-ended, but a step in the right direction.

There are three main levels at which security issues exist in an IoT network: computer attacks, software vulnerabilities, and data interception [8]. Computer attacks

include Distributed Denial-of-Service (DDoS) attacks, malware exploits, or modifying physical components of a device [8]. Software vulnerabilities are vulnerabilities in the IoT applications. Data interception has to do with the network traffic and hijacking a session or eavesdropping. Any security breach in any device can give an attacker access to the whole network. If an attacker gets access to the network traffic of these devices, there is a lot of private information the attacker can infer or predict. Capturing network packets can be done fairly easily with a network adapter in monitor mode and a tool such as Wireshark [12] or airodump-ng [13]. Since IoT devices communicate private information on the network, this data should not be read by third-parties [9]. A strong encryption scheme must be used to ensure confidentiality [9]. However, encrypting the data does not ensure privacy as malicious adversaries can still obtain sensitive information by capturing the encrypted network traffic.

1.2 Motivation

Even if the data is encrypted on the network, the metadata of the packets can reveal a lot of information that compromises user privacy, such as behavior patterns. Metadata is data about data [14]. In network packets, this includes the source and destination addresses, timestamp, size, and protocol [14]. The source addresses can be used to identify traffic streams of individual devices [15]. Queries are made to a Domain Name Server (DNS) to request the IP address of a server. The destination addresses in network packets can then be used to identify devices based on their DNS queries [16]. Smart speakers such as Google Home or Amazon Alexa are constantly communicating with the cloud to send information back and forth. The cloud endpoints usually have the device manufacturer's name it. For example, the Amazon Echo makes a DNS query to 'softwareupdates.amazon.com' for its software updates [15]. Based on this we can identify specific devices on the network. A Virtual Private

Network, VPN, creates a secure connection masking the true source or destination of a network packet [17]. The use of DNS queries to identify devices on a network can be prevented using a VPN. However, this does not prevent an attacker from looking at the pattern of the traffic.

The timestamp and the packet size is sufficient to plot time-series data and identify traffic patterns to determine user's activities. Time analysis and traffic burst correlation might allow the adversary to identify the relationship between devices on the network [18]. Traffic shape analysis can be used to infer the function of some devices. One study is able to identify websites that are accessed on the network by classifying and matching the traffic data using k-Nearest Neighbors algorithms, Support Vector Machines, and Fisher's Least Square Linear Discriminant classifiers [18]. Another study is able to classify new IoT devices using traffic analysis techniques [19]. The authors were able to classify new devices based on the fact that their traffic rate patterns are distinct and can be identified using machine learning [19]. Machine Learning classifiers can be trained using supervised learning to be able to classify certain types of IoT devices based on traffic patterns [20].

Once a specific device is identified, traffic patterns of that device can correlate to user activity [15]. Based on the network usage of a device, private user activities can be revealed or inferred [21]. A study shows that user presence in a smart home can be identified based on wireless camera's encrypted network traffic [22]. It was found that the camera's network usage increased and varied when there was movement in the home [22]. Extending this logic, attackers can use different IoT devices infer if a user is at home, watching tv, or even where in the home they may be. This compromises privacy of the user. To protect user privacy, there are many traffic shaping methods that attempt to obfuscate the network data by adding noise in cover traffic [23]. By doing so, it becomes more difficult to identify the user behavior.

1.3 Preventing Traffic Analysis

To prevent traffic analysis attacks, the traffic should not be able to correlate to user activity. A firewall can be used to block all traffic from IoT devices, but this would limit functionality of those devices [23]. A VPN connection can be used to hide the original IP header of packets. This would prevent some device identification techniques using DNS queries, but does not prevent analyzing the time and size patterns that would still correlate to user activity [23].

To prevent attackers from using traffic analysis techniques, the shape of the traffic can be altered so that it does not accurately correlate to user activity. This can be done by using traffic shaping algorithms. There are two methods widely used to add cover traffic. In one method, each individual device can produce random traffic using dummy packets. This method requires that the destination or the router be able to filter dummy packets out based on a flag in an encrypted header, which would require a change in the network protocol [24]. Another method uses an additional device on the network that produces cover traffic, or dummy packets, that is just dropped by the router [23]. This method requires the use of a VPN in order to prevent adversaries from separating out individual device traffic. Most existing traffic shaping methods require the use of a VPN connection.

There are different types of traffic shaping algorithms including Independent Link Padding (ILP) in which cover traffic is independent of the user device traffic and Dependent Link Padding (DLP) in which the cover traffic is dependent on the user device traffic [23]. A newer algorithm called Stochastic Traffic Padding (STP) introduces randomness into when to pad the traffic [25]. This paper will look into existing traffic shaping algorithms and identify their strengths and weaknesses based on computational requirements, bandwidth overhead, latency, and privacy protection.

Computational requirements are what additional processing or storage must go into padding traffic. Bandwidth overhead is the additional network usage due to dummy packets being sent across the network. Latency is the average amount of time that a packet is delayed. Privacy protection is a measure of how well user activity data is masked. Based on this, we will propose a new method of traffic shaping called Dynamic Traffic Padding that improves the balance of these metrics based on different requirements of devices. We will be testing these algorithms on captured network data from a list of IoT devices over several days.

1.4 Thesis Contributions

- Culmination of previous work on traffic shaping
- Evaluation of current traffic shaping algorithms
- Applying traffic shaping algorithms to a large IoT data-set
- Proposing a new traffic shaping algorithm called Dynamic Traffic Padding

1.5 Thesis Organization

The organization of the thesis is as follows:

Chapter	Title	Overview
1	Introduction	Background on IoT devices and their security issues. The motivation of the work which includes device identification and traffic analysis attacks. Introduction to methods that prevent traffic analysis on IoT network traffic.
2	Literature Review	Privacy vulnerabilities in traffic including lack of encryption, device identification, and traffic pattern analysis. Communications security protocols including IPSec with TFC, TLS, and DNS-over-HTTPS and DNS-over-TLS. Traffic shaping algorithms including Independent Link Padding, Dependent Link Padding, On-demand Link Padding, and Stochastic Traffic Padding.
3	Privacy vs Overhead	Metrics that influence an adversary's confidence that traffic patterns correlate to user activity. Metrics that influence bandwidth overhead for traffic padding. Introducing the privacy vs overhead trade-off.
4	System Model	Overview of our IoT network architecture and device list. Threat model including eavesdropping and traffic analysis attacks. Security objectives including confidentiality and privacy.
5	Data-set	Overview of our data-set containing captured network traffic from IoT devices. Data visualization of device's throughput vs time and how they correlate to user activity.
6	Current Solutions for Traffic Privacy	Solutions include Firewall, VPN, Dependent Link Padding, Independent Link Padding, and Stochastic Traffic Padding. More in-depth look into the algorithms for ILP and STP.
7	Evaluation of Current Traffic Shaping Algorithms	Evaluation of Independent Link Padding with a constant padding rate and a variable padding rate, and Stochastic Traffic Padding with a constant rate based on computational requirements, bandwidth overhead and latency, and privacy protection, using IoT device data.
8	Dynamic Traffic Padding	Introduction of new traffic shaping algorithm, DTP, that takes the type of device and privacy preference as input and determines tunable variables based on the input variables and previous traffic data. Overview of algorithm and evaluation based on computational requirements, bandwidth overhead and latency, and privacy protection.
9	Conclusion & Future Work	Summary of work and proposed future work including machine learning techniques and privacy vs overhead trade-off for different implementations of the algorithm.

Table 1.1: Thesis Organization Chart

Chapter 2

LITERATURE REVIEW

This thesis is based on a number of previous research papers looking into traffic shaping and how it can help to prevent network traffic analysis to predict user activities. Rate-shaping and traffic-injection are among other mitigations against traffic analysis including blocking with a firewall and tunneling with a VPN [15]. This section will start in a broader sense with privacy vulnerabilities in traffic. Next it will introduce security protocols and different traffic shaping algorithms that attempt to solve this problem.

2.1 Privacy Vulnerabilities in IoT Traffic

Since IoT devices are used for sensistive tasks, it means that private user data is stored and transmitted from and to these devices, especially in the medical field. A study was done to obtain sensitive data presented in cleartext in some medical devices [26]. In 2017, a paper was published by Princeton University researchers on the privacy vulnerabilities of encrypted IoT traffic [15]. They monitored the traffic of four smart home devices including Sense Sleep Monitor, Nest Security Camera, Amazon Echo, and WeMo Switch [15]. By utilizing DNS queries they were able to identify the devices on the network. They separated out the traffic to analyze traffic rate patterns on each device. Based on these patterns they were able to infer user activity. For example, spikes of traffic in the sleep monitor correlated to the user being awake or asleep. In addition, the Nest camera had higher network usage if the video was being streamed to a user's device. Therefore, it is possible to identify if the user is actively watching the video stream or not. For the Amazon Echo, there

is a network spike every time the user asks a question, therefore they were able to determine when the user interacts with their smart speaker [15]. This shows that traffic rate analysis is a major privacy and security risk that needs to be overcome, especially with the new age of smart homes. The majority of smart homes do not have any additional security mechanisms in place to combat traffic analysis attacks. To be able to prevent traffic analysis, we must obfuscate the traffic pattern and hide the original IP header of packets on the network.

2.2 Communications Security Protocols

There are several protocols that attempt to hide information in the IP layer or create a secure tunnel to prevent useful eavesdropping. We will look at IPSec for security at the IP layer and TLS for encrypting the payload. We will also look at two methods that encrypt DNS look-ups including DNS-over-TLS and DNS-over-HTTPS.

2.2.1 IPSec with TFC

Internet Protocol Security (IPSec) is a framework of open standards that provides data confidentiality, data integrity, and data authentication at the IP layer [27]. IPSec consists of two main protocols: Authentication Header (AH) and Encapsulating Security Payload (ESP) [27]. In a paper from 2008, a Traffic Flow Confidentiality (TFC) protocol was introduced as a sub-layer security protocol in the IPsec ESP protocol [24]. The TFC encapsulation will pad the payload to mask the actual size of the payload. In addition, it will perform packet aggregation and multiplexing which can combine packets into a larger data frame. It will also alter the timing pattern of the data which will add latency to the packet delivery. They also create dummy packets where packets would be generated and thrown away at the destination. Knowing if the packet it a dummy packet can be set using the next header field in the TFC pro-

tocol header [24]. The main overhead has to do with performance and cost. Adding dummy packets can increase network usage. Also, delaying packets add latency to packet delivery. In addition, the TFC module must be incorporated as a part of the IPSec protocol on every device in the network including the router. The TFC module hides packet sizes and packet inter-arrival times and therefore can prevent traffic analysis attacks [24]. The control logic must use a traffic shaping algorithm to be able to decide when and how much to pad traffic. Different traffic shaping algorithms have their benefits and drawbacks.

2.2.2 TLS

Transport Layer Security (TLS) is a successor of Secure Socket Layer (SSL) and provides data privacy, authentication, and data integrity [28]. It does this by encrypting traffic between the client and the server using a key derived from a handshake [29]. A data integrity check is applied to the data and authentication of the server can be done through a certificate [29]. TLS is used over a TCP connection while Datagram Transport Layer Security (DTLS) is used over UDP [29]. TLS encrypts the actual payload or data that is being sent over the network, but metadata of the IP header is still visible for traffic analysis attacks.

2.2.3 DNS-over-TLS and DNS-over-HTTPS

DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH) both aim to encrypt and hide DNS look-up queries [30]. DNS queries can be used to identify devices on a network so encrypting the DNS look-up adds privacy as it prevents an adversary from using the DNS requests to identify devices. DoT and DoH work by encrypting the messages between the client and the DNS resolver as well as hiding the metadata including the time and size by padding traffic and multiplexing with other HTTPS traffic using the

same TLS tunnel [30]. However, DoT or DoH traffic analysis attacks can still be used to identify devices on the network [30]. Therefore, DoT and DoH do not fully provide privacy.

2.3 Traffic Shaping Algorithms

To prevent traffic analysis, the same group from Princeton evaluated current defenses [23]. One strategy to prevent traffic analysis is to prevent an adversary from collecting traffic in the first place. This can be done by using a firewall to block traffic [23]. However, blocking traffic may limit the functionality of the smart devices. Another strategy is tunneling traffic using a VPN connection. VPN can wrap all traffic in an additional transport layer, similar to IPsec, that can prevent device identification [23]. A VPN connection may mask the devices and their DNS queries, but it does not mask the network usage at a given time. Therefore, traffic analysis can still be done on the traffic rates. The proposed defense is using a traffic shaping algorithm.

Traffic shaping algorithms can be implemented at a device level or can be implemented on a separate box or router. If it is implemented on a box or router and a VPN connection is used, then the VPN endpoints can drop dummy packets based on encrypted bytes in the packet header [25]. A library exists for some traffic shaping methods and requires the router or access point to run a padding script and a custom VPN endpoint [31]. The receiving endpoint can be based on an AWS server that takes care of dropping packets [31]. This method does not require each device to change their implementations and it gives the central box access to all the different device streams. If each device implements these algorithms, they would be sending dummy packets to their manufacturer's cloud endpoints, which need to drop the packets [25]. IPSec with TFC can be used to hide metadata and identify dummy packets [24]. The manufacturer would need to make changes in the code to accommodate for this

method. This method allows for a little more flexibility in shaping traffic based on the device requirements and can vary per device. However, to run traffic shaping algorithms on current IoT devices, a user can implement this on a router or middle box without waiting for device manufacturer's to update their systems.

2.3.1 Independent Link Padding

The group from Princeton propose a method of traffic shaping based on Independent Link Padding (ILP) in which cover traffic is injected into the network independent of device traffic [23]. In their implementation, they used constant rate shaping that shaped traffic to a constant rate. To do so, they implemented a queue with device traffic having higher priority than cover traffic. At a given time a fixed amount of data was sent from the queue. This method required a 40 KB/s network usage overhead and added a 10-15 second latency to some device traffic [23]. Another paper proposed a variant link padding algorithm for bursty traffic [32]. In this case, there is a constant length buffer and a traffic buffer. The constant length buffer is padded if not full and packets are sent out based on a heavy tail distribution [32]. Independent Link Padding can be accomplished with a constant rate or a variable rate based on a distribution, both independent of the user traffic rates.

2.3.2 Dependent Link Padding

There is another traffic shaping method called Dependent Link Padding (DLP) that is implemented in a ACM Conference Paper [33]. The DLP algorithm uses the packet arrival timing information to generate a matched schedule with the minimum sending rate for a given set of incoming packet flows [33]. DLP also requires the use of dummy packets to pad the actual packets. Due to the fact that DLP is based on incoming packet flows, the more flows that are incoming requires a greater overhead and delay

and drop rates increase. This means that an increase in network traffic from devices can increase network usage due to traffic pattern matching. This can be compared to constant rate shaping, where the overhead rate does not change or increase.

2.3.3 On-demand Link Padding

Another type of traffic shaping was proposed by a group from Harvard called On-Demand Link Padding [34]. On-demand Link Padding is based on providing cover traffic based on previous real traffic and only when traffic is present. There are two types of on-demand link padding proposed. On-demand Link Padding with Delay creates a traffic pattern based on the real traffic, once the padding is added to cover the traffic, the real traffic is sent out [34]. This method incurs a significant delay but has a reduced peak of usage. On-demand Link Padding with Headroom does not delay the packets but added enough headroom with padding and cover traffic for the real traffic to be masked [34]. This method predicts the usage of the next time period based on the previous time period. This method has a higher peak usage. However, this method does not hide the fact that devices are being used on the network.

2.3.4 Stochastic Traffic Padding

In 2019, another Princeton paper was published proposing an improved method of traffic shaping called Stochastic Traffic Padding (STP) [25]. STP shapes traffic based on a previously determined traffic pattern. Traffic is padded and sent out whenever it arrives so there is very little latency. A threshold is identified to only mask high peaks and not noise in traffic rates. In addition, if there is no traffic, dummy traffic will be created and sent out at random intervals that match a pre-defined pattern. This method hides peak activity by adding other peaks, which lowers the adversary's confidence in pulling useful information from the traffic rates [25]. This method does

require an overhead of network usage in order to send/receive dummy packets. STP was implemented using middle-boxes and a VPN connection. Both VPN end points can generate and drop cover traffic [25]. Alternatively, it could be implemented on each device, using an encrypted flag in the IPSec protocol to drop cover traffic [24].

Chapter 3

PRIVACY VS OVERHEAD

To be able to evaluate traffic shaping algorithms, we must understand what components affect privacy and overhead. We will first introduce what an adversary's confidence means, which correlates to privacy. Then, we will define bandwidth overhead. Lastly, we will show the privacy vs overhead trade-off.

3.1 Adversary Confidence

An adversary that is analyzing captured traffic aims to correlate time periods to user activities. Traffic shaping introduces additional traffic even when there is no user activity, which lowers the adversary's confidence in their correlation [25]. Given a time period, T, let p be the probability that user activity is occurring. Additionally, let q be the probability that a decision function chooses to pad traffic during no user activity. Given n time periods, let np represents the number of time periods with user activity. After using a padding algorithm, the expected number of padded periods is $np + n(1-p)q$. The adversary cannot distinguish between user activity traffic and padded traffic. Therefore, the adversary's confidence is

$$c = \frac{np}{np + n(1-p)q} = (1 + \frac{(1-p)q}{p})^{-1} \tag{3.1}$$

[25]. If the frequency of user activity is high, higher p, then a time period is more likely to correspond to user activity. If the frequency of user activity is low and more non-activity padding occurs, higher q, then a time period is less likely to correspond to user activity.

3.2 Bandwidth Overhead

Traffic shaping adds additional padding to the traffic, so there is going to be greater bandwidth usage. We can quantify bandwidth overhead using the following metrics. Let R be the padding rate. Let p and q be the probabilities introduced in the previous section. Let A be the average throughput during periods of user activity and let B be the average throughput of background traffic without user activity. The expected bandwidth overhead of a padding algorithm is

$$b = \frac{pR + (1-p)qR + (1-p)(1-q)A}{pA + (1-p)B} \tag{3.2}$$

[25]. Increasing q will increase overhead as more padding is required.

3.3 Privacy vs Overhead Trade-off

Increasing q will decrease the adversary's confidence in user activity correlation according to the power law, but will also increase bandwidth overhead linearly. The ratio of adversary confidence to bandwidth overhead is also a power law,

$$\frac{c}{b} = O(q^{-2}) \tag{3.3}$$

This means that just a little overhead can provide a drastic reduction in adversary confidence [25].

Chapter 4

SYSTEM MODEL

In this section, we introduce our system model in terms of the IoT network architecture, threat model, and security objectives.

4.1 The IoT Network Architecture

The IoT network architecture has multiple layers to it at shown in Figure 4.1. At the top of the IoT Test-bed, we have all the devices connecting to a Wemo Smart Plug to be able to monitor power usage. At the next level, we have all the IoT devices that we classify based on their functions shown in Table 4.1. All of these IoT devices are very common in any smart home. At the next level, all the devices are connected to the wireless access point. This access point functions as a router but also allows for the packets to be monitored. In the next layer, we have a PC connected to the same access point with its wireless card in monitor mode. This allows us to capture all the packets on the network produced by the IoT devices. A python script runs on the PC that captures network packets from the devices as well as the power usage from the smart plugs in real time and stores then into a MySQL database in an Amazon Web Services (AWS) cloud instance.

4.2 Threat Model

Based on the IoT network, we consider eavesdropping and traffic analysis attacks as follows.

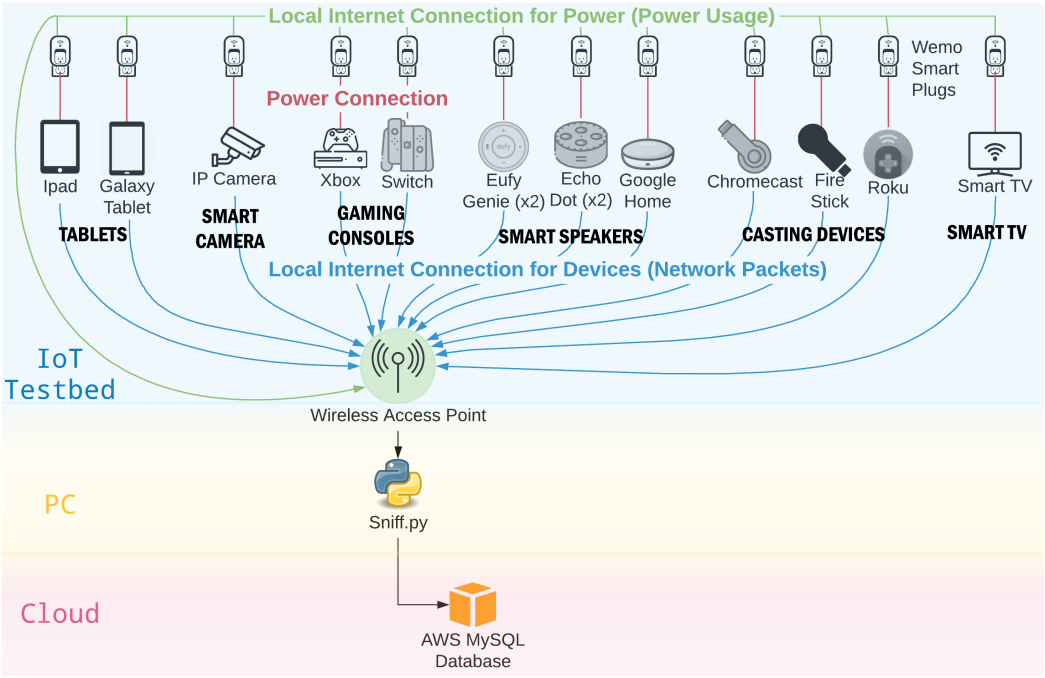

Figure 4.1: IoT Devices Network Diagram

4.2.1 Eavesdropping

An eavesdropping attack is the act of intercepting private information transmitted over a network by a connected device [35]. The adversary should be able to capture the network packets, which can be done by eavesdropping. The adversary does not need internal access to the wireless network to be able to capture metadata of network packets. A wireless card in monitor mode can accomplish this. Captured network packets can be stored in a pcap file using WireShark, a network analyzer, or in a MySQL Database using a custom Python script.

4.2.2 Traffic analysis

A traffic analysis attack consists of reviewing captured network packets to identify patterns correlating to user activity. We assume that the adversary has the necessary

Type	Function	Device
Tablets	Browse internet, stream/cast videos	Ipad Galaxy Tablet
Smart Camera	Record/stream video, send alerts	IP Camera
Gaming Consoles	Play online games, stream video	Microsoft Xbox One Nintendo Switch
Smart Speakers	Stream audio, search assistant, voice command	Eufy Genie Amazon Echo Dot Google Home
Casting Devices	Stream video/audio from the internet or another device	Google Chromecast Amazon Firestick Roku Express
Smart TV	Stream audio/video from the internet or another device	Samsung Smart TV
Smart Plugs	monitor power usage of plugged in device	Wemo Smart Plug

Table 4.1: IoT Devices Breakdown

computational power and knowledge to perform traffic analysis using machine learning techniques to identify user activity. In addition, we assume that an adversary has enough general knowledge to determine some user activities based on time of day. For example, it would be more likely for a TV turning off to mean a user is sleeping at night, but may mean the user is leaving the house in the morning.

4.3 Security Objectives

We consider confidentiality and privacy objectives for the traffic protection in this study.

4.3.1 Confidentiality

All the traffic is encrypted. Therefore, the attacker can not directly monitor the traffic details of each IoT device such as the payload. However, the metadata including the time, destination addresses, and size of the packets can still be obtained.

4.3.2 Privacy

Not only traffic details, but traffic patterns can be used to reveal user behavior patterns. Traditional encrypted traffic can still reveal the traffic pattern. To fully protect the privacy of the user, the network throughput cannot reveal network usage over time. In addition, the source and destination addresses can reveal information about what devices are on the network. These need to be masked as well. This can be done by using a VPN connection and shaping the traffic by adding padding to obfuscate the data. However, our goal is to shape the traffic to keep privacy intact while minimizing network overhead and computational resources.

Chapter 5

DATA-SET

This chapter goes over our data-set starting with an overview. Next, we outline some of its characteristics. Then we explain how and why we pre-processed the data. Finally, we visualize the data with respect to each IoT device captured.

5.1 Overview

The data-set we captured is based on network traffic of these IoT devices being used on the network. We have two tables stored in a My-SQL Database. The 'ip' table consists 116,830,878 rows of network packets captured over time on the network from these devices. The network packets contain meta data including the source, destination, size, timestamp, and the encrypted payload. The 'power' table consists of 61,240,223 rows including power usage in MW at a given time for each device. We can utilize the time stamp and size of each network packet to plot traffic rates over time. All the columns in the two tables are shown in Figure A.1 and Figure A.2 in Appendix A.

We compare these traffic rates to a spreadsheet containing user events at a given time on the network. Since this data was captured in a lab, it may not represent real smart home functionality. However, if separated by device, it shows clearly what normal user activity would look like on the network, interacting with an individual device from the list above. Since only certain devices were used over a few hours of the day, we should look at traffic rates within minutes or hours rather than looking at entire days or weeks.

5.2 Characteristics of Data-set

The data-set of captured traffic used to run traffic shaping algorithms is very important. There needs to be a variety of IoT devices that are being used on the network over a long period of time. Many of the previous papers introducing traffic shaping for IoT networks lack a strong data set to use for analysis of their algorithms. Our data-set consists of many different types of IoT devices as well as captured traffic based on events where a user uses one or multiple devices. Since this testbed was setup as a simulation, it may not reflect a smart home's usage throughout the day. However, we can still use the traffic in shorter increments and disregard the time that devices were being used.

5.3 Pre-processing Data

For both network packets and power usage captured, some of the fields of the meta data were missing so not all the data could be used as shown in Figure A.3 in Appendix A. For device identification, we would need the source and destination addresses. To identify network usage we need the time stamp and the size of each packet. By identifying the size grouped by time in ms, we can plot data throughput as a function of time. In order to manipulate the data, we imported the data from My-SQL into a Python Data frame using the Pandas module. Each entry in the data frame would represent a packet with its timestamp and size as shown in Figure A.4 in Appendix A. We can pull the packets from a 15-20 minute period where a couple events occurred for a given device. After running this data through a traffic shaping algorithm, we can group the data by time in ms and plot it against the size and compare our two graphs.

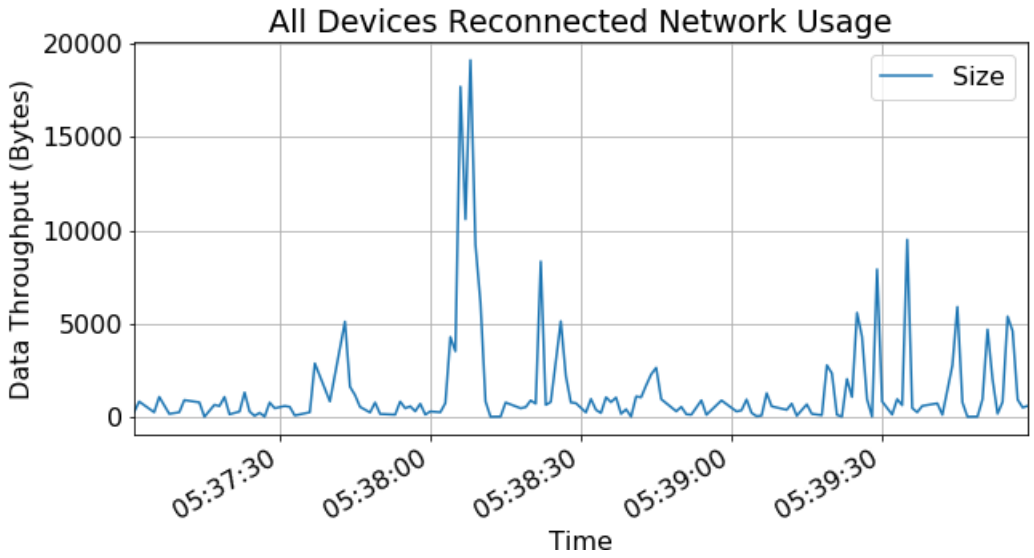

Figure 5.1: Network Usage of All Devices Reconnecting from 5:37pm to 5:40pm

5.4 Data Visualization

By plotting traffic rates we can discover user activity on the network. Spikes in traffic correlate to user activity and therefore privacy is not preserved. Figure 5.1 shows all devices on the network being disconnected and reconnected to the network. The spike at 5:38 correlates to all the devices reconnecting to the network. Figure 5.2 shows the network usage for an hour while multiple devices where being used on the network. As shown in the graph, the peak usage for our network is always less that 20 kB/s. The following graphs have different time periods based on the length of an event consisting of user activity.

5.4.1 Google Chromecast

Figure 5.3 shows data throughput plotted against time for the Chromecast device. The 10 minutes that are plotted were captured during the initial setup of the device.

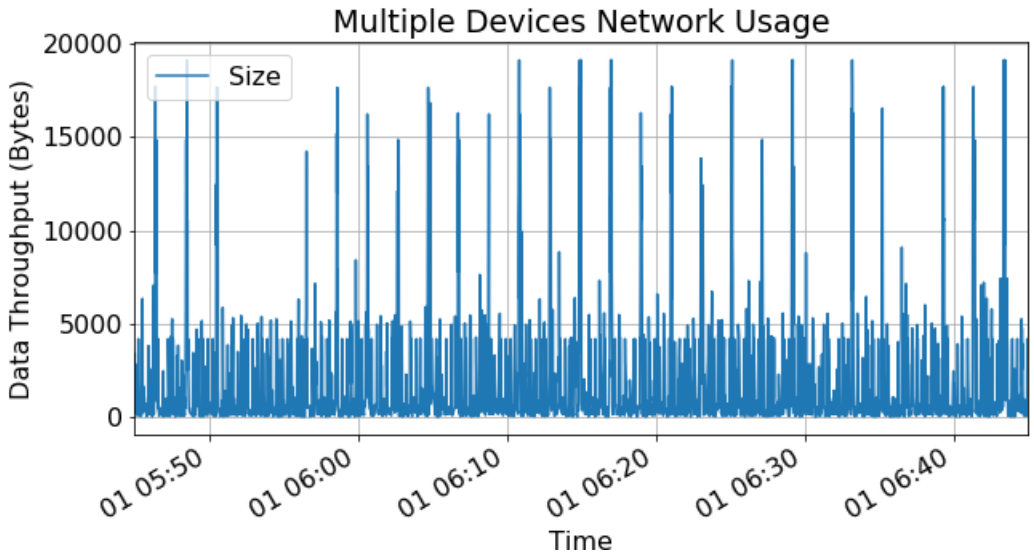

Figure 5.2: Network Usage of Multiple Devices from 5:50pm to 6:50pm

Between 12:39 and 12:42, there is a spike in network usage. This can be correlated to the minute long intro video and the device updating followed by two reboots.

5.4.2 Google Home

Figure 5.4 shows the the data throughput against time for the Google Home device. The spikes in usage correlate to the user asking 'what's the weather?' to the google home device. In other words, this can show that the user is home, awake, and interacting with the device.

5.4.3 Microsoft Xbox One

Figure 5.5 shows the data throughput against time for the Xbox. The spikes in usage correlate to the Xbox being used. In this instance, the user was playing a game on the Xbox. This can also show that the user is home, awake, and sitting in front of the TV.

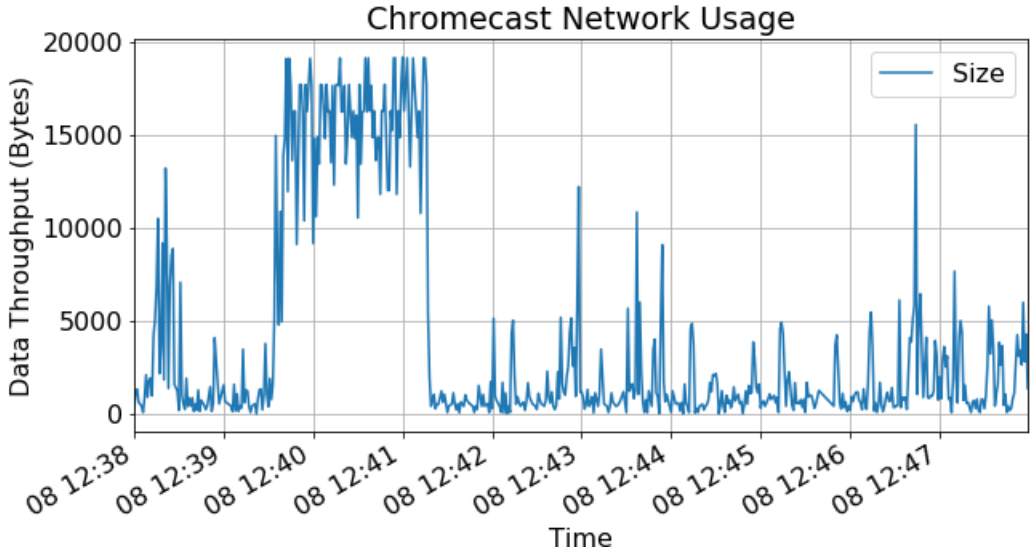

Figure 5.3: Chromecast Network Usage During Setup from 12:38pm to 12:48pm

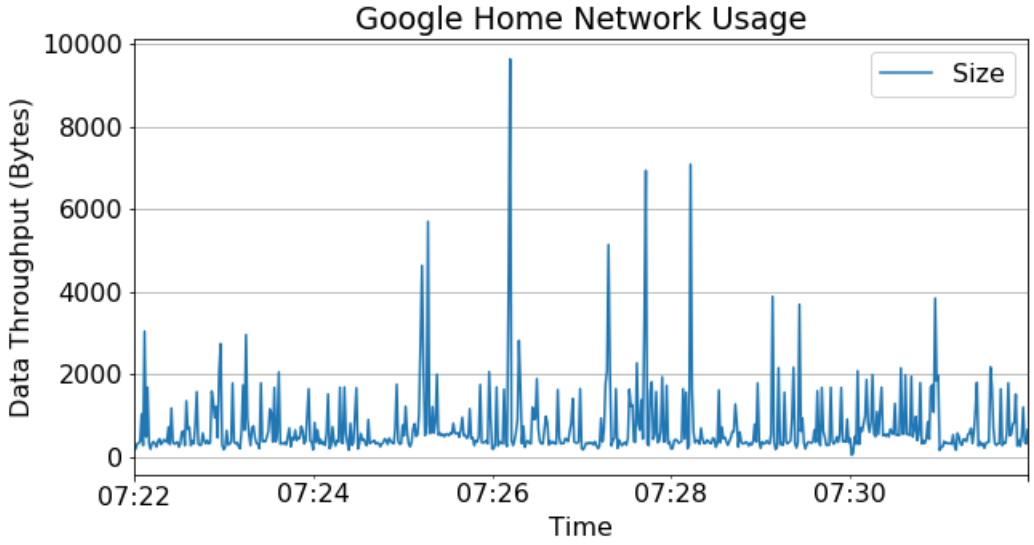

Figure 5.4: Google Home Network Usage from 7:22pm to 7:32pm

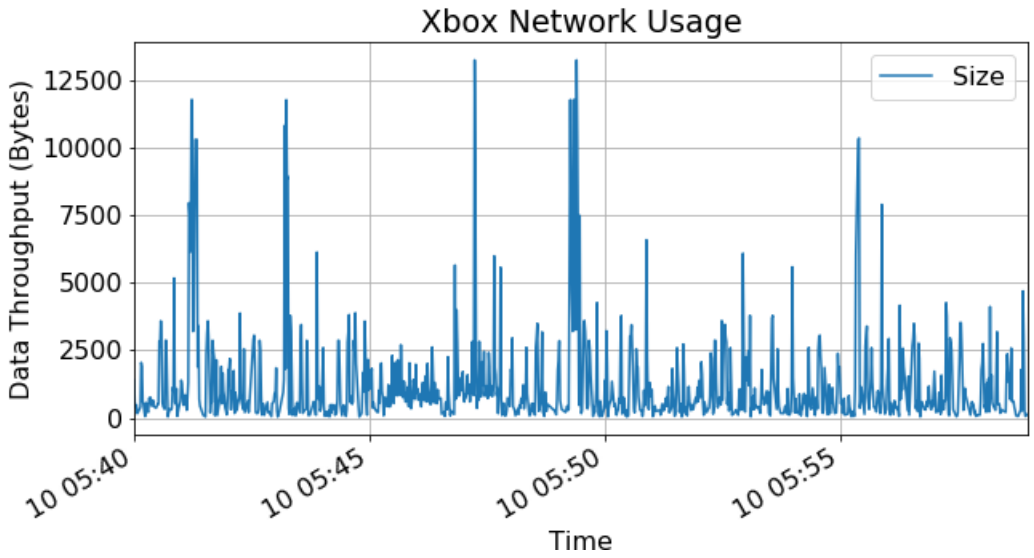

Figure 5.5: Xbox Network Usage from 5:40pm to 6:00pm

5.4.4 Ipad Tablet

Figure 5.6 shows the data throughput against time for an Ipad tablet. During this period in time, the user was updating apps on the Ipad tablet. The spikes correlate to downloading updates for different apps.

5.4.5 Samsung Smart TV

Figure 5.7 shows the data throughput of a Samsung Smart TV begin used. At 11:38, smart features were being setup on the TV. At 11:41, the spikes correlate to streaming YouTube on the TV. This can be used to infer that the user is using the TV to stream a show or movie and is sitting in front of the TV.

5.4.6 IP Camera

Figure 5.8 shows the data throughput of an IP Camera recording video and streaming it. The camera was restarted at 4:40. The spike at 4:42 is the camera reconnecting

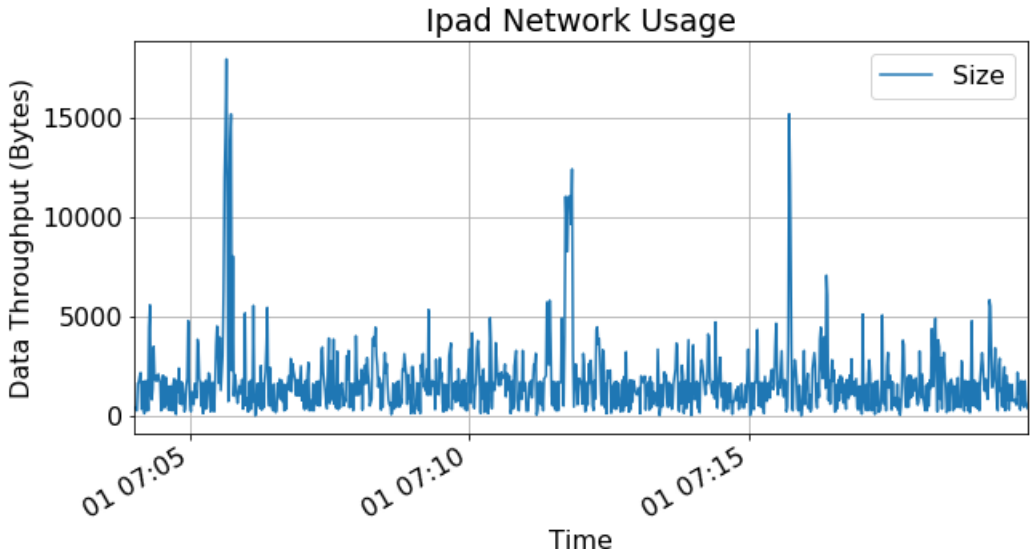

Figure 5.6: Ipad Tablet Network Usage from 7:04pm to 7:20pm

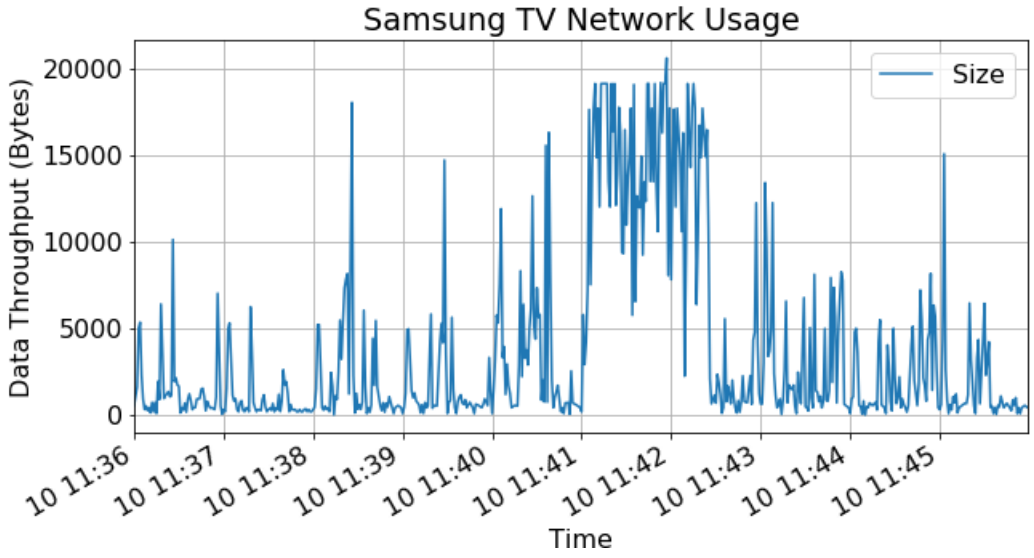

Figure 5.7: Samsung Smart TV Network Usage from 11:36am to 11:46am

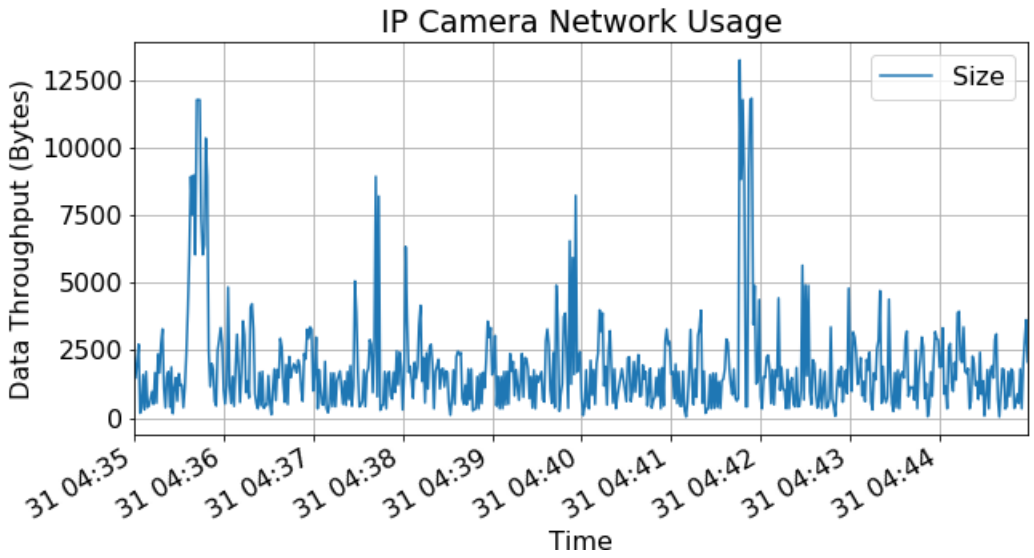

Figure 5.8: IP Camera Network Usage from 4:35pm to 4:45pm

to the internet and streaming video. This can be used to identify if the camera is on and if it is recording and streaming live video.

Chapter 6

CURRENT SOLUTIONS FOR TRAFFIC PRIVACY

To prevent an adversary from inferring user activities from traffic rate analysis, the traffic can be obfuscated by using a method called traffic shaping. There exists traffic shaping algorithms which add padding to user traffic. If the padded rate is less than the user activity rate, latency is introduced. It is important to note that in addition to these algorithms, to prevent identification of devices, a VPN should be used. We will give an overview of 3 padding algorithms: Dependent Link Padding, Independent Link Padding, and Stochastic Traffic Padding.

6.1 Firewall

A firewall is used to block incoming or outgoing traffic from a particular source or destination [23]. To prevent an adversary from analyzing IoT traffic, one could simply block outgoing traffic from the smart home devices. However, this is a very naive approach as it drastically reduces functionality of the devices as many IoT devices communicate with cloud servers for computations [23].

6.2 VPN

A Virtual Private Network (VPN) can be deployed which hides all traffic under a VPN header, masking the original header including source and destination and type of packet [17]. A VPN can help prevent device identification through DNS queries [25]. However, a VPN alone does not prevent against traffic analysis attacks to infer user activities because the time and size of the packet remain known and no padding

is added [23]. Therefore, a VPN must be paired with a traffic shaping algorithm to provide the desired privacy.

6.3 Dependent Link Padding

Dependent Link Padding (DLP) [33] adds cover traffic only when there is user activity. In addition, the traffic shape is based on the incoming traffic flows. A higher traffic rate incurs a greater overhead, delay, and dropped packet rates increase. DLP does not protect the privacy of a user because different user activities can be correlated to traffic patterns revealed by DLP. In addition, an adversary can still identify if no user activity is occurring. Therefore, we will not be evaluating DLP as a solution in our model.

6.4 Independent Link Padding

Independent Link Padding (ILP) adds cover traffic at a pre-determined rate, independent of the device traffic [25]. The simplest ILP algorithm, Constant Interval Timing (CIT), matches the traffic to a constant rate. This protects privacy of user activity because there is no changes in traffic rates to infer any information. However, this method adds latency to some device traffic as well as imposes a bandwidth overhead. A modification of this method, Variable Interval Timing (VIT), uses a varying rate, still independent of the device traffic. The overhead and latency is still similar to the previous ILP method.

Taking a look at the ILP algorithm, we must check and pad traffic constantly. For each time interval, we send user packets until we reach the pre-determined rate, the rest of the user packets must be delayed. We then find the difference between the sent packet sizes and the pre-determined rate. This is the amount of padding we must

add through a dummy packet. In this algorithm, the constant rate or the variable rate must be pre-determined, but can vary based on the device.

Algorithm 1 Independent Link Padding

```
Input: constantRate, userPackets
currentSize ← 0
for each time interval do
  while currentSize < constantRate do
    if userPackets is not empty then
      p ← userPackets.next()
      len ← p.size
      if len + currentSize < constantRate then
        currentSize ← currentSize + len
        sendPacket(p)
      else
        padLen ← constantRate − currentSize
        padding ← dummyPacket(padLen)
        currentSize ← currentSize + padLen
        sendPacket(padding)
      end if
    else
      padLen ← constantRate − currentSize
      padding ← dummyPacket(padLen)
      currentSize ← currentSize + padLen
      sendPacket(padding)
    end if
  end while
  currentSize ← 0
end for
```

6.5 Stochastic Traffic Padding

A newer method of traffic shaping is called Stochastic Traffic Padding (STP) was introduced in a 2019 Princeton paper [25]. STP adds cover traffic with randomness, based on existing traffic rate patterns, and based on the peak rates of the particular device. This method has been shown to be slightly better than ILP and DLP algorithms. STP still requires the use of a VPN in order to prevent device identification. STP sets a peak rate to pad to and always pads user traffic to that rate. It will only

pad user traffic above a given threshold to prevent padding noise in traffic and focus on peak traffic rates. If no user activity is occurring it will randomly pad traffic to that rate based on the padding probability q. A constant rate can be set or rates can be pulled from previous traffic patterns.

Algorithm 2 Stochastic Traffic Padding

```
Input: rate R, userPackets, padProbability q, timePeriod T
padStart ← 0
padEnd ← 0
padOffset ← 0
if tmodT = 0 and decisionFunc(q) then
  padOffset ← rand(0, T)
  if t + padOffset > padEnd then
    padStart ← t + padOffset
    padEnd ← padStart + T
  else
    padEnd ← padEnd + T
  end if
end if
if padStart ≤ t ≤ padEnd then
  padTraffic(R)
else
  if userActivityOccuring then
    padStart ← t
    padEnd ← t + T
    padTraffic(R)
  end if
end if
```

Taking a look at the STP algorithm [25], we take the rate R, probability q, and time period T as pre-determined inputs. We call the STP function at every time interval with current time, t. If we choose to pad, we set our padding period to the size of T. A decision function based on q determines if we should pad while there is no user activity. If there is user activity, then we pad regardless. The traffic is padded to the rate R, which is set to the peak usage of a device, so that we do not delay any packets.

Chapter 7

EVALUATION OF CURRENT TRAFFIC SHAPING ALGORITHMS

We analyzed both constant and variable independent link padding, as well as stochastic traffic padding to identify how well they preserve privacy. We can identify if privacy is preserved based on if peak rates are identifiable and if these peaks actually correlate to user activity. We also looked at how plausible they were to be implemented based on computational requirements, bandwidth overhead, and latency.

Algorithm	Bandwidth Overhead	Latency	Privacy
ILP-CIT	ConstPadRate/AvgUserRate	Varies w/ ConstPadRate	max
ILP-VIT	AvgPadRate/AvgUserRate	Varies w/ AvgPadRate	max
STP	Varies w/ q	none	Varies w/ q^{-1}

Table 7.1: Comparing Traffic Shaping Algorithms, where q is the padding probability in STP

Device	ILP-CIT		ILP-VIT		STP (q=0.15)	
	Overhead	Latency	Overhead	Latency	Overhead	Latency
Chromecast	138.67%	87s	149.88%	79s	95.90%	0s
Google Home	40.73%	600s	32.15%	253s	239.5%	1s
Xbox	919.07%	0s	958.22%	0s	192.04%	0s

Table 7.2: Comparing Bandwidth Overhead and Latency, where latency is the maximum delay of a packet

7.1 Independent Link Padding

Both constant interval timing and variable interval timing have the same average predetermined rate. Therefore, the overhead and latency between them are very similar, if not the same. For VIT, an additional tunable variable is how much to deviate from this average rate.

7.1.1 Computational Requirements

For ILP, traffic needs to either be padded or delayed at every time step. Delayed packets need to be stored in a queue. If the padding rate is variable, the pre-determined shape needs to be stored as well. ILP can either run at a device level or router level.

7.1.2 Bandwidth Overhead and Latency

ILP shapes traffic to a pre-determined rate. The average rate to pad to can be determined in several ways. The chosen rate could be the average rate of prior user activity traffic. However, this may vary based on the amount of time the user is on the network. Constantly changing the rate would reveal when a user is on the network. If the chosen rate is too low, as shown by using the average user activity rate in Figure 7.1 for the Google Home, then it increases the latency when there is user activity. A high latency would result in some devices not working, especially during streaming or playing a video game. On the contrary, if the rate is too high such as Figure 7.3, then it requires a higher bandwidth overhead when there is no user activity, but latency is minimized. Latency is a bigger problem compared to bandwidth overhead. Therefore, a slight increase in overhead is acceptable if it lowers latency. In VIT, to be able to provide no latency, it requires the minimum padding to always be above the peak usage of a device, which requires greater bandwidth overhead.

7.1.3 Privacy Protection

Since ILP always pads to a pre-determined rate that is either constant or varies randomly, there are no user activity patterns revealed. In addition, the peaks of user traffic are hidden and an ILP peak does not correlate to user activity. If the padded rate is the peak rate of a device, the peak usage may be obtained but it cannot

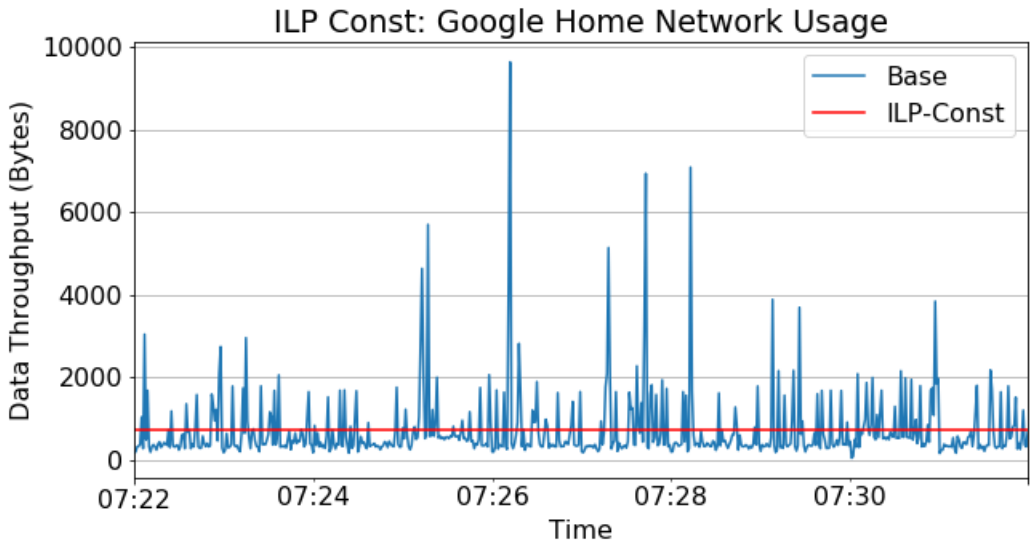

Figure 7.1: Google Home ILP Constant

correlate to user activity occurring at a given time. Therefore, ILP has maximum privacy protection. In terms of adversary confidence, this will vary based on p. For example, Chromecast has a c value for ILP of 0.375 while Xbox has a c value of 0.175 due to less user activity.

7.2 Stochastic Traffic Padding

7.2.1 Computational Requirements

STP pads traffic based on the probability, q. At each time step, the algorithm checks for user activity, and if there is no activity, makes a decision to pad or not based on q. If padding is required, then it must send cover traffic. There are no delayed packets, so there is no queue to be stored. However, if a variable rate is set, the shape of the pre-determined rate must be stored.

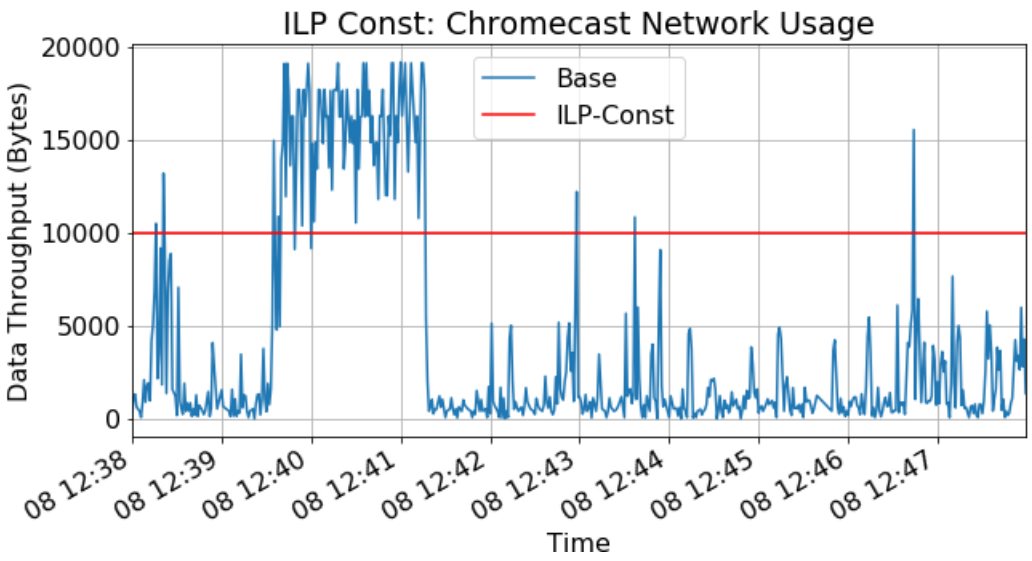

Figure 7.2: Chromecast ILP Constant

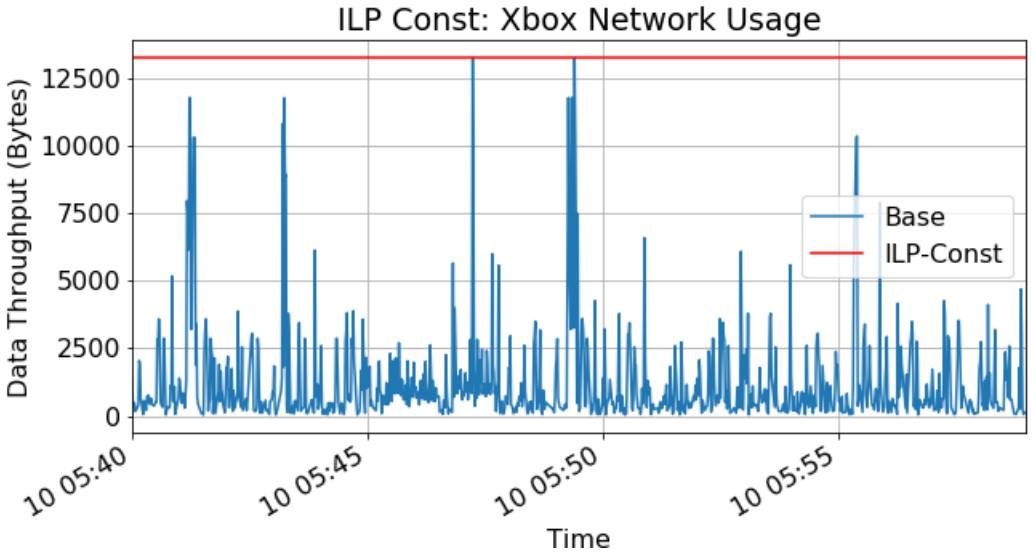

Figure 7.3: Xbox ILP Constant

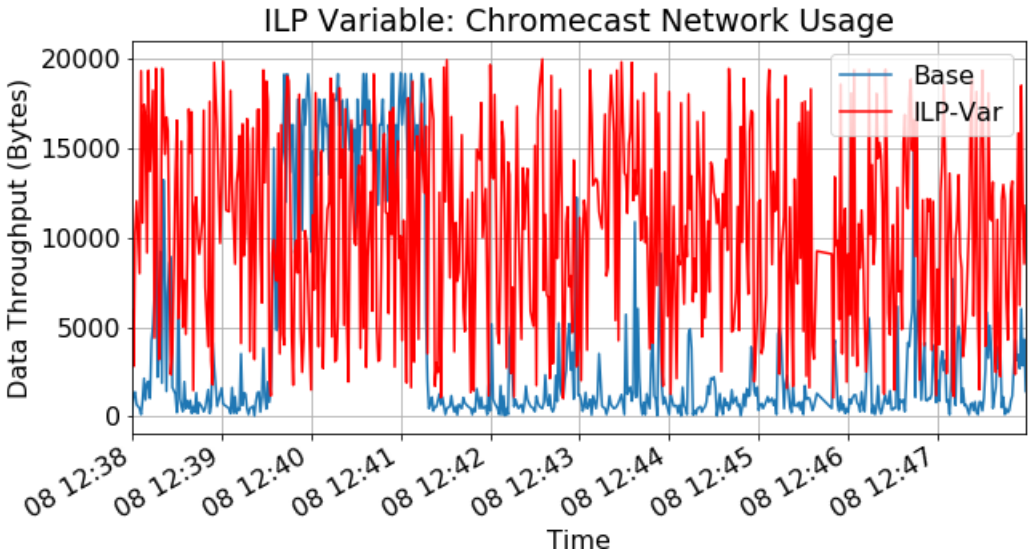

Figure 7.4: Chromecast ILP Variable

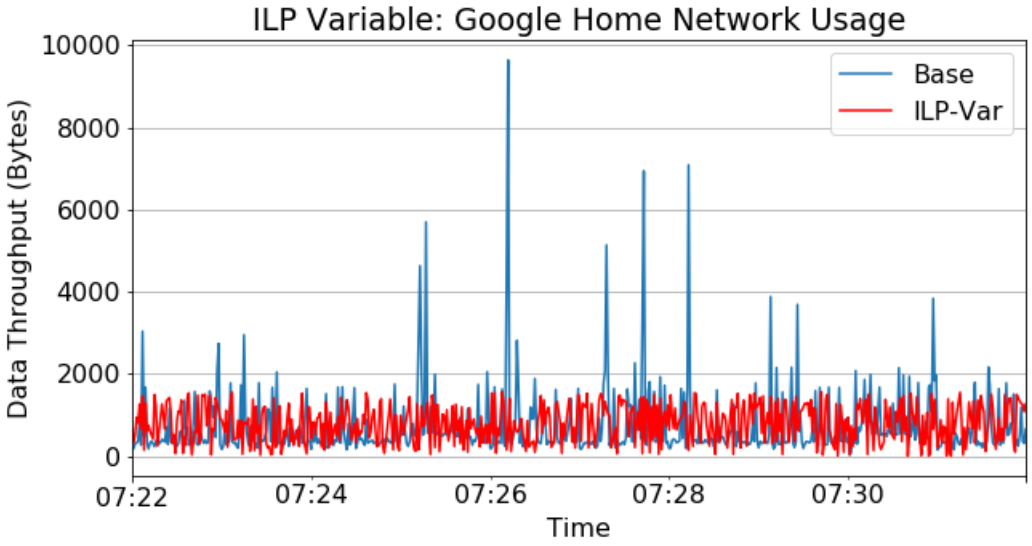

Figure 7.5: Google Home ILP Variable

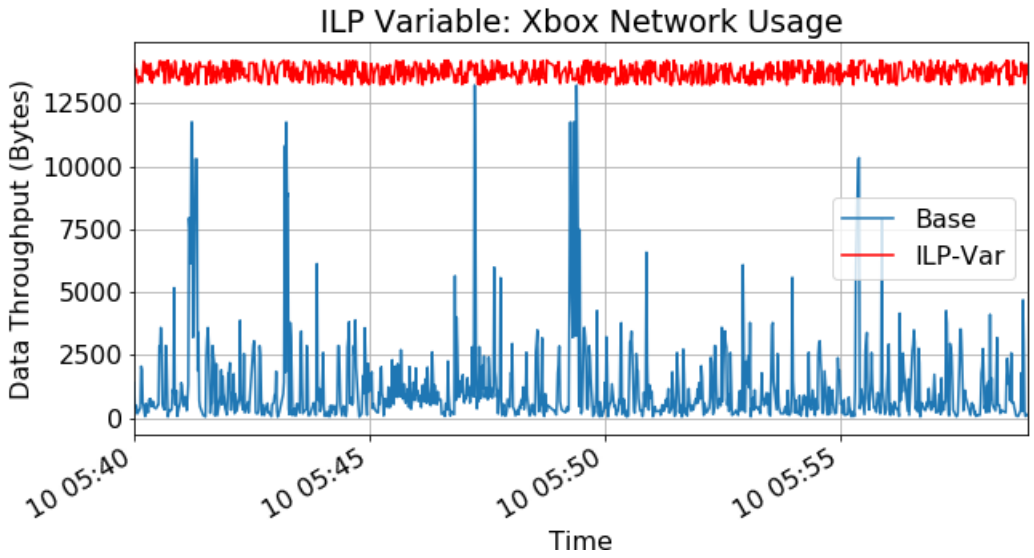

Figure 7.6: Xbox ILP Variable

7.2.2 Bandwidth Overhead and Latency

For STP, we used a peak padding rate of the maximum usage of that device as shown in Figure 7.7. STP is intended to provide no latency by setting a higher peak rate, however if a lower rate is chosen like in Figure 7.8, there can be delayed packets. In addition, it adds cover traffic at random intervals. The bandwidth overhead of STP is related to the frequency of cover traffic, dependent on padding probability q. If q is increased, then more padding is required increasing the bandwidth overhead. However, the bandwidth overhead does not exceed that of Independent Link Padding Algorithms such as constant rate padding at the peak usage rate. In other words, if q is set to 1, then it is equivalent to ILP with a constant rate of the peak usage, which will require the most bandwidth.

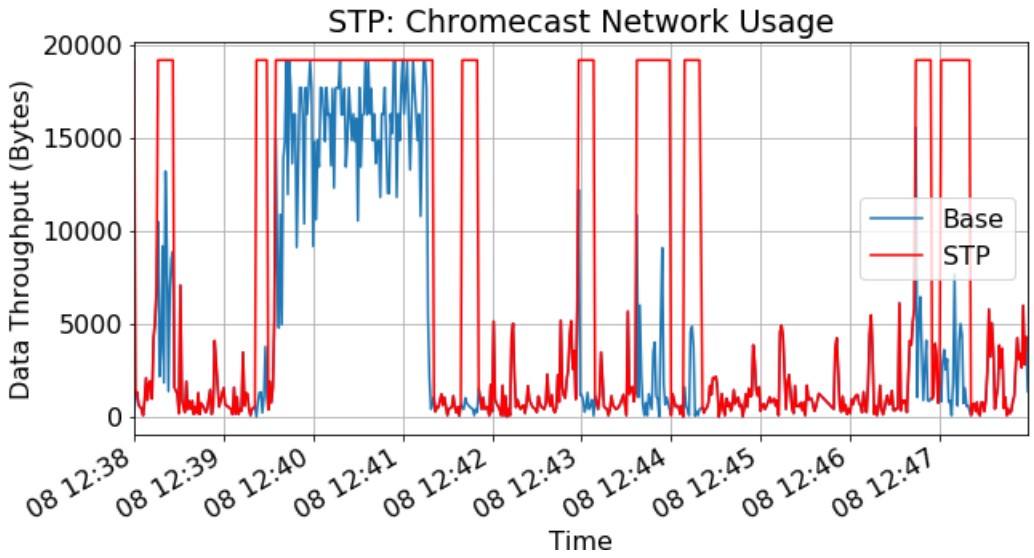

Figure 7.7: Chromecast STP

7.2.3 Privacy Protection

Since STP provides minimal to no latency, this means that a time period with high traffic rates would be based on high user traffic. Privacy protection increases as q increases because it increases the chance of non-user activity to be padded. Privacy is fully protected if q is set to 1, and will be equivalent to ILP. Lowering q to 0, will provide the lowest protection as all peaks will correlate to user activity. However, it still reveals the peak usage rate because that is what it is padded to. In addition, setting the variables q, R, and T, in STP, changes the overhead and privacy protection drastically and is difficult to optimize. For example, Chromecast with q of 0.15 has a c value of 0.8 while Xbox with a q value of 0.15 has a c value of 0.59.

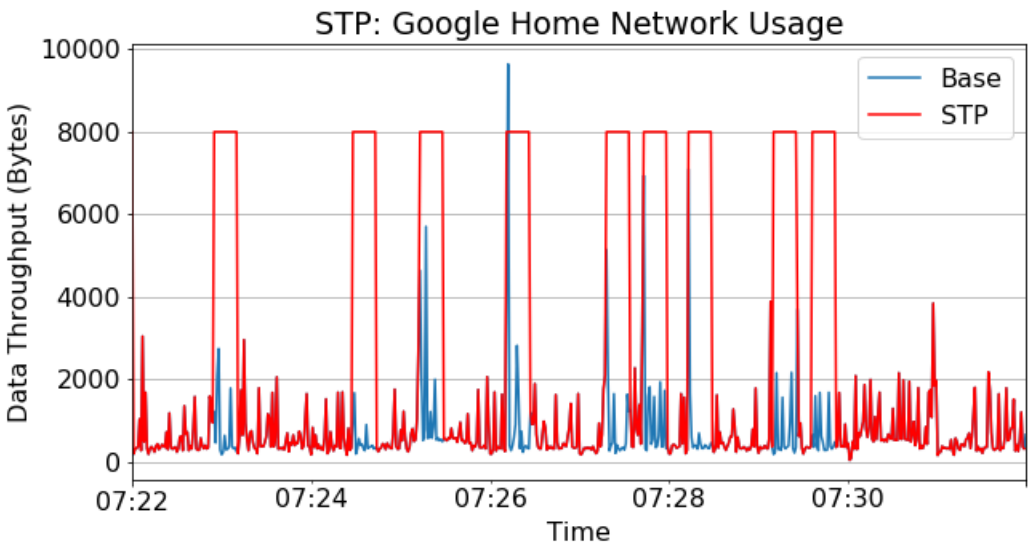

Figure 7.8: Google Home STP

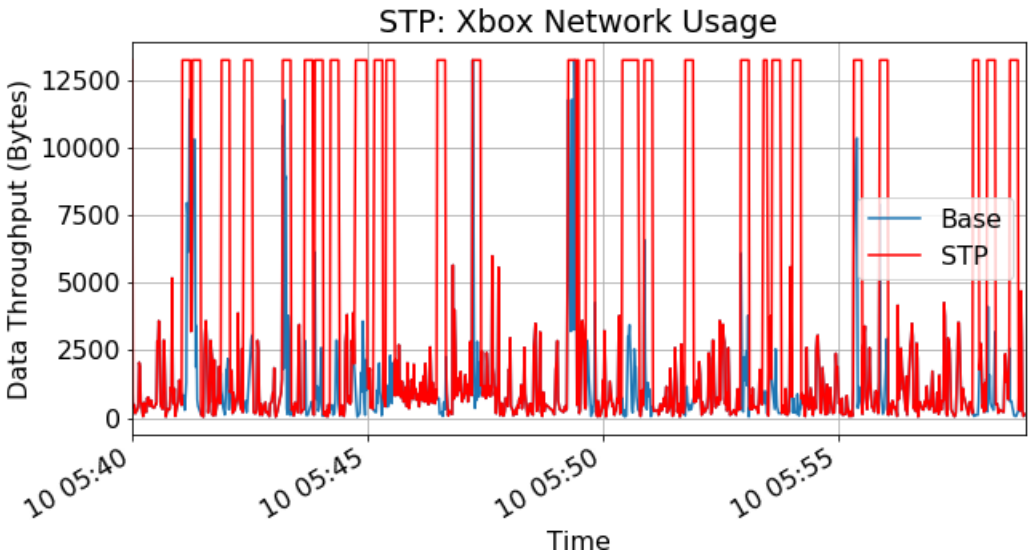

Figure 7.9: Xbox STP

Chapter 8

DYNAMIC TRAFFIC PADDING

We are proposing a dynamic traffic padding algorithm, DTP, that determines tunable variables for STP using device information such as the type of device and how much latency it can tolerate, prior traffic rates and patterns, and desired privacy level or overhead. Our goals include maximizing privacy, minimizing overhead, and minimizing latency. As additional input we need to know the level of latency tolerable by a device. In addition, we need to know user preference of privacy vs overhead. With these variables, and previous user activity patterns, we can tune the variables for STP.

The previous algorithms, ILP and STP, have the following issues: optimizing variables is not user-friendly and they do not allow for changing the variables based previous traffic and different days of the week. Our proposed algorithm is based on STP, but makes improvements to the user input variables so that they are more user friendly and saves bandwidth by adjusting the variables based on previous traffic.

8.1 Tunable Variables

The padding algorithm requires the following three parameters, which can be dynamically set using DTP.

- q is the padding probability when there is no user activity. The higher we set q, the greater overhead we have, but the more privacy we obtain. We can use the amount of user activity per day to determine this. If there is not much activity, we can lower q, to lower overhead.

- R is the padding rate. We will use a variable STP algorithm, which means the padding rate will be set to a pre-determined pattern based on peak user activity rates and tolerable latency. A gaming console streaming a game would need close to no latency, so a R greater than the peak rate would be set. However, a tablet browsing the web may tolerate some latency and so the R can be set slightly below the peak rate, lowering the overhead.
- T is the time period for padding. This can be adjusted based on the average duration of user activity from previous traffic. The pattern that we shape to must be longer than T. Also, T must be long enough so that we cannot distinguish user traffic and cover traffic.

8.2 Input Variables

We propose two user input variables to the DTP algorithm: device type, dT, privacy preference, pP.

- dT: Level of Latency tolerance; (Ex: highLatency, lowLatency)
- pP: Level of Privacy; (Ex: 0, 1, 2, 3; where 0 is minimum privacy prioritizing minimizing overhead, and 3 is maximum privacy)

8.3 Algorithm

In Dynamic Traffic Padding, we have two main input variables: dT and pP. In addition, we can use the previous day's activity to identify the peak rate, average activity duration, and activity frequency. These metrics can then be used to determine q, T, R, for STP. In DTP, we use the previous day's metrics to set variables for the current day's STP algorithm, while simultaneously keeping track of the current days activity for the next day.

A low-latency device such as streaming devices or gaming consoles, require close to no delay in packets. Therefore, we set the padding rate R to be the peak rate from the previous day. However, a high-latency device such as smart plugs or tablets, can withstand a slight delay in packets. We can set the padding rate to be slightly lower than the previous day's peak rate to save bandwidth. The value of $decR$ can be adjusted based on the time interval t and the average size of a packet.

The privacy preference adjusts the overhead vs privacy trade-off. A higher pP will increase the padding probability q. To save some bandwidth, we can adjust this padding probability q based on the previous day's activity frequency. If there is minimal activity, we do not need to constantly send packets when there is no activity. We can do this by slightly lowering the padding probability by $decQ$ if the activity frequency is under a certain threshold.

For the DTP algorithm, the variables are represented in Table 8.1.

Variable	Definition
pR	previous day's category's peak rate
pD	previous day's category's average user activity duration
pF	previous day's category's user activity frequency
cR	current user activity rate
D	duration total
t	time interval (1 second)
$decR$	size in bytes to reduce the padding rate pR based on dT
$decQ$	float between 0 and 1 to reduce q based on pF
$dayCategories$	Categories for day classification; (Ex: Weekday, Weekend)

Table 8.1: DTP Variables

Algorithm 3 Dynamic Traffic Padding

```
INPUT: userPackets, dT, pP
pR, pD, pF ← 0
for each day do
    peakRate, duration, D, activity, count ← 0
    T ← pD
    R ← setR(pR, dT, decR)
    q ← setq(pF, pP, decQ)
    STP(userPackets, q, R, T) {Run STP in separate thread}
    for each t do
        if cR > peakRate then
            peakRate ← cR
        end if
        if userActivityOccuring then
            duration ← duration + 1
            activity ← activity + 1
        else
            if duration ≠ 0 then
                D ← D + duration
                count ← count + 1
                duration ← 0
            end if
        end if
    end for
    pR ← peakRate
    pD ← D/count
    pF ← activity/(t per day)
end for
```

8.4 Evaluation

8.4.1 Computational Requirements

Since DTP uses prior user activity to determine tunable variables for the padding algorithm, it does require slightly more memory or storage, as well as additional computations on each time interval to keep track of the peak rate and duration. The padding algorithm needs to run in parallel with the DTP algorithm, keeping track of each day's category, updating the metrics based on user activity while simultaneously padding the current user traffic. We are using a variable rate padding algorithm based on STP. Using a random rate between the set rate R and the actual activity rate, can remove the need for storing a fixed pattern to shape to.

DTP requires more computational power and memory than previous methods like ILP and STP, especially if machine learning is implemented using previous traffic. However, this is an acceptable amount as in return we lower overhead and optimize tunable variables in STP.

8.4.2 Bandwidth Overhead and Latency

In DTP, the type of device and privacy preference are taken as input to the algorithm. The padding rate R can be set slightly lower if the device can withstand latency, lowering the bandwidth overhead. Lowering the padding probability q if there is minimal user activity will also save bandwidth while providing the same privacy level. Since we use a variable padding rate, if the peak rate is higher than the user activity rate, we can pad to a random rate between the actual user activity rate and the peak rate R. This will also lower overhead, as we are not constantly padding to the highest rate.

Since DTP uses the same underlying STP algorithm, it is difficult to compare them. STP can achieve similar overhead if the same variables are set. But the difficulty is setting these variables based on previous device traffic. DTP makes this easier and can optimize variables while the algorithm is running over time. DTP does lower overhead and is definitely better than ILP. Latency may be introduced based on the type of device which will also lower overhead.

8.4.3 Privacy Protection

DTP has a privacy preference that can either prioritize privacy or overhead. The q variable in STP relates to privacy protection and can be tuned based on preference. Lowering the q value would decrease privacy but would also decrease the overhead. We can also use the previous day's activity frequency and the given preference to determine the q value. Setting the privacy preference to the highest setting is equivalent to the ILP algorithm, where traffic is constantly padded. Prioritizing privacy over bandwidth is highly recommended, as the bandwidth overhead is not as significant.

DTP and STP can have the same privacy protection based on the q value that is set. However, determining this value is difficult and varies based on desired privacy and the frequency of user activity. Based on these metrics, DTP can adjust q to provide the desired privacy level and adversary confidence while minimizing overhead.

8.4.4 Data Visualization

Since DTP can tune the variables in the STP algorithm, it is unfair to directly compare DTP to STP. However, we can compare different settings of DTP and how it affects overhead, latency, and privacy. There are several cases that can occur when shaping the traffic. First, the privacy level can be low or high. Next, the peak rate

of the previous day's traffic can either be higher or lower than the current peak rate. The time period, T, can either be less than or greater than the user activity time period. We can simulate the occurrence of each of these and their effects on overhead, latency, and privacy.

Fig	dT	pP	p	q	c	R	T	TH	A	B	b	OH%
8.2	loL	1	0.393	0.25	0.72	19188	15	5000	8718	1055	3.55	241.9
8.3	loL	3	0.393	1.0	0.393	19188	15	5000	8718	1055	4.72	548.3
8.4	loL	2	0.393	0.647	0.5	13230	15	5000	8718	1055	3.02	245.5
8.5	hiL	2	0.393	0.647	0.5	12718	15	5000	8718	1055	2.53	277.4
8.7	loL	1	0.3	0.25	0.63	13780	15	4000	6486	604	4.20	296.6
8.8	loL	2	0.3	0.5	0.46	13780	10	4000	6486	604	4.74	573.3
8.9	loL	3	0.3	1.0	0.3	13780	10	4000	6486	604	5.82	881.6

Table 8.2: Comparing DTP with different variables

Fig	p	q	c	R	T	TH	A	B	b	OH%
8.2	0.393	0.0	1.0	20000	15	5000	8718	1055	3.23	310.2
8.3	0.393	1.0	0.393	20000	15	5000	8718	1055	4.92	1004.6
8.4	0.393	0.5	0.564	14000	15	5000	8718	1055	3.05	430.1
8.5	0.393	0.5	0.564	12000	15	5000	8718	1055	2.71	366.5

Table 8.3: Comparing STP with different variables

8.4.4.1 Chromecast

Here we use the Chromecast throughput from Figure 5.3 to shape the traffic for Chromecast in Figure 8.1. It is important to note than the peak rate from Figure 5.3 is higher than the peak rate of Figure 8.1. Padding traffic to a higher rate results in no latency because packets are not delayed, only padding is added. Chromecast DTP1 in Figure 8.2 shows minimum privacy protection with adversary confidence of 0.72 while Figure 8.3 shows maximum privacy protection with adversary confidence of 0.393, which is similar to using ILP with a variable rate. Chromecast DTP 3 and 4 pad to a peak rate of the current traffic to show optimal rate settings. They show

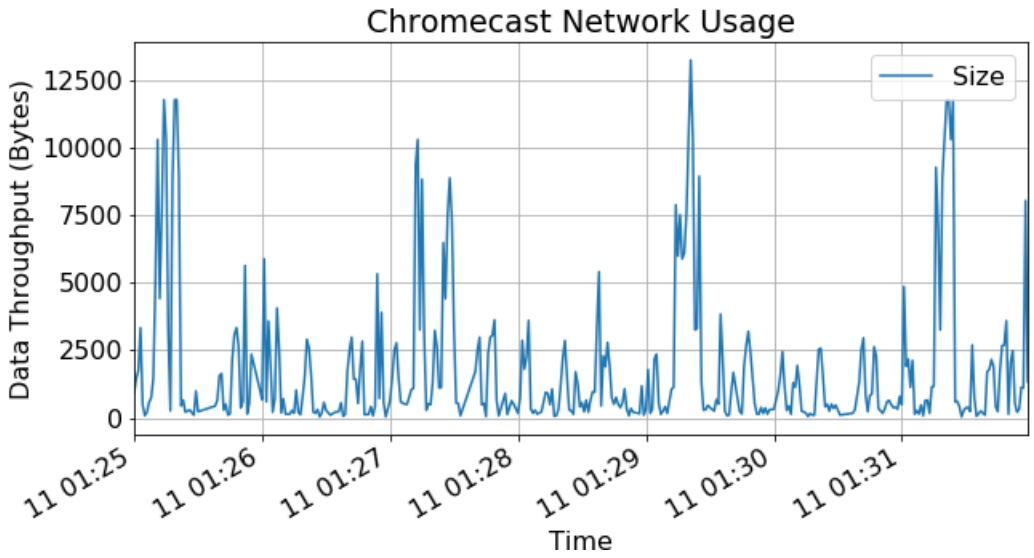

Figure 8.1: Chromecast Throughput 2

what q value is required for a desired adversary confidence of 0.5, which varies based on p. Figure 8.5 has slightly lower overhead than Figure 8.4 as changing the device type from low to high latency tolerance can lower the peak rate. A slight latency is introduced, in this case only for one packet for one time step. In all four of the graphs, STP is run with similar variables that a user may input, not fully optimized. The privacy protection or adversary confidence may not be what the user intends. Additionally, STP shows a higher bandwidth overhead overall than DTP due to fine tuning variables and using a variable rate. In the case of this Chromecast traffic, we were able to reduce overhead on average by 200%.

8.4.4.2 Google Home

Next, we use the Google Home 2 throughput from Figure 8.6 to shape the Google Home throughput shown earlier in Figure 5.4. The peak rate of the previous traffic is higher than the traffic being shaped so there is no latency. Google Home DTP3 shows minimum privacy in Figure 8.7. As we increase the q value for DTP 4 and 5 we can see that privacy and overhead both increase. In this case, the overhead is higher due

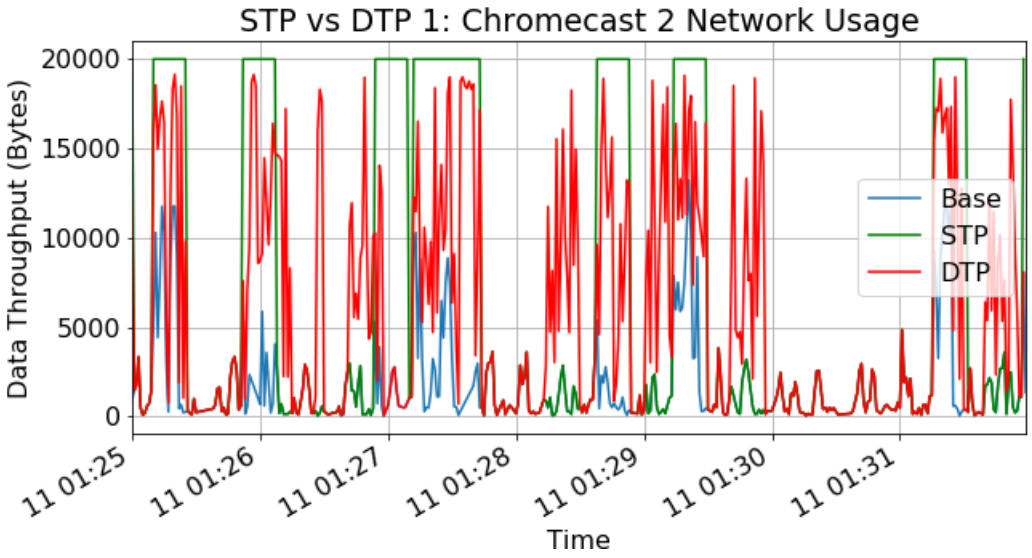

Figure 8.2: Chromecast STP vs DTP 1

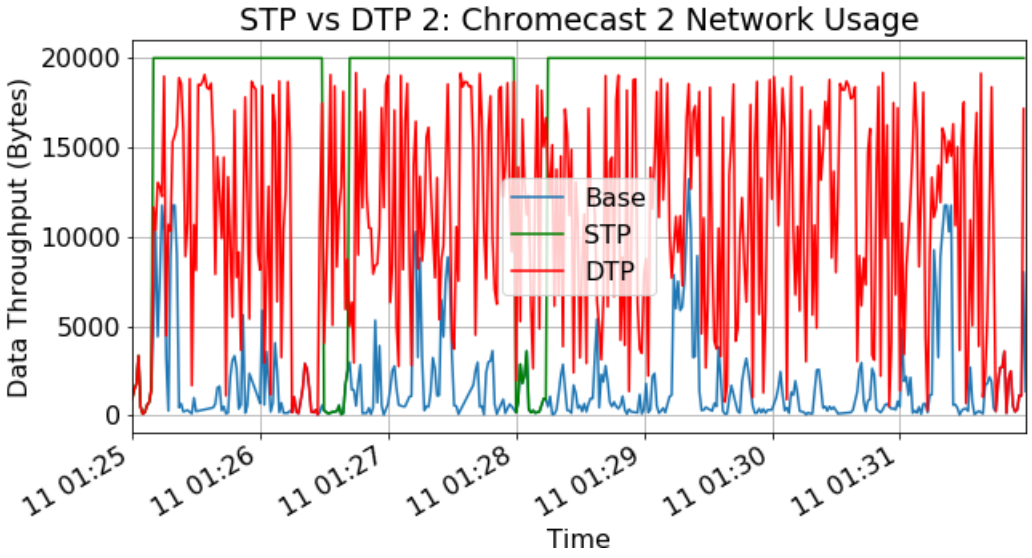

Figure 8.3: Chromecast STP vs DTP 2

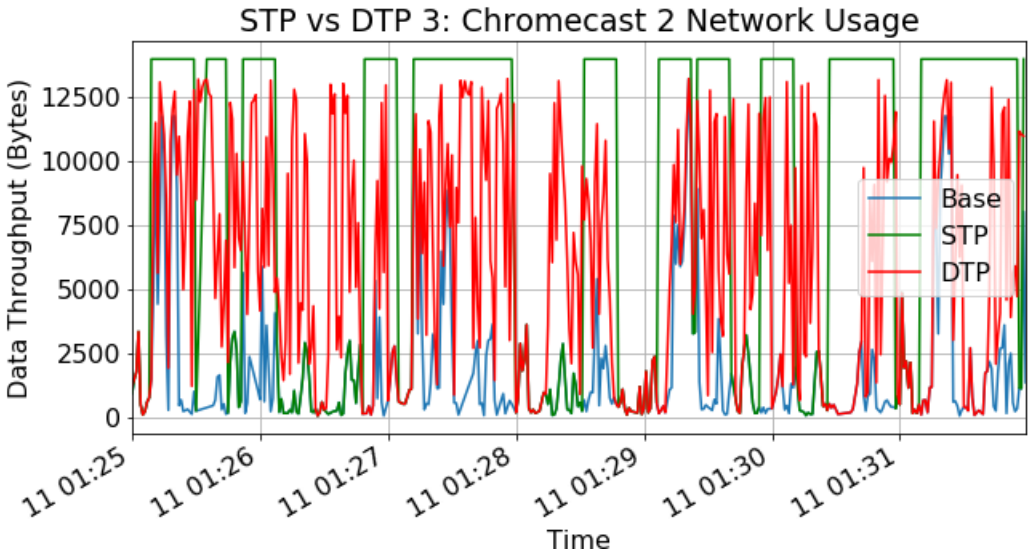

Figure 8.4: Chromecast STP vs DTP 3

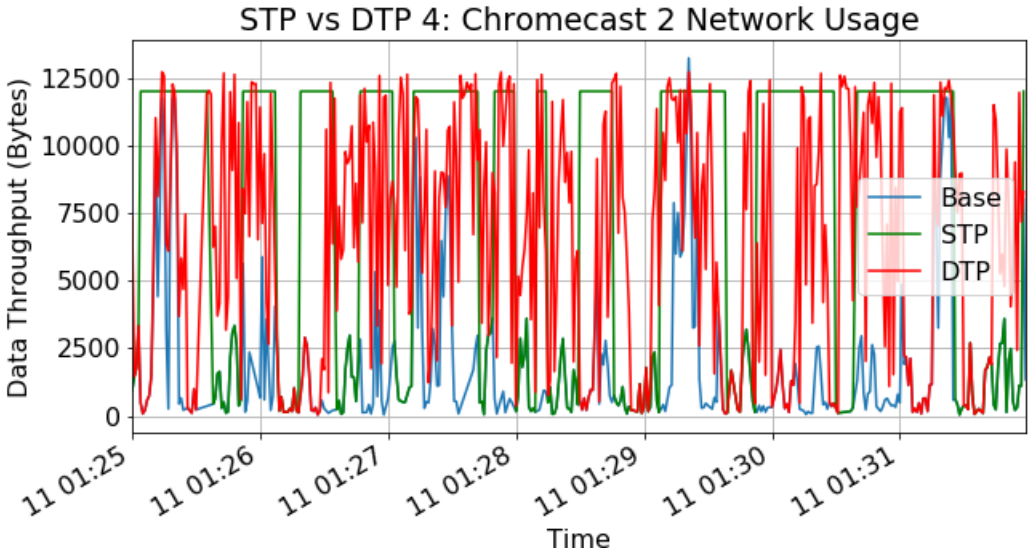

Figure 8.5: Chromecast STP vs DTP 4

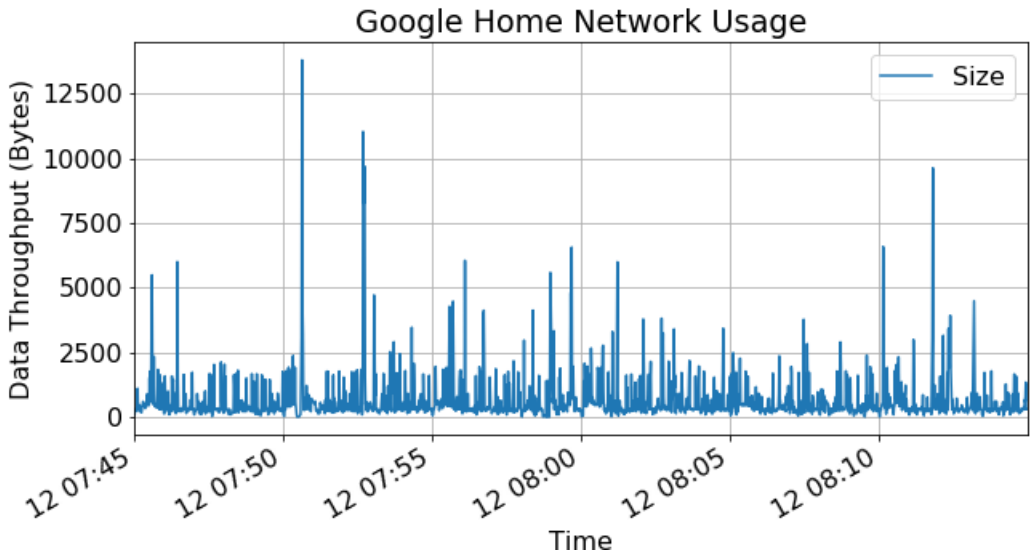

Figure 8.6: Google Home Throughput 2

to the fact that the frequency of user activity was relatively low. The actual overhead is slightly different than the b value as these are derived through averages.

8.4.5 Privacy vs Overhead Trade-off

Figure 8.10 shows adversary confidence and bandwidth overhead for different values of q. This graph is based on the Chromecast traffic in Figure 8.1 with a p value of 0.393. As shown, increasing q will increase b linearly, but will decrease c based on a power law. This means that initially, small changes in q will lower c significantly, but towards the end, it takes a bigger change in q to make small decreases to c. The value of p and the desired level of privacy or adversary confidence can help determine the q value to set with minimum overhead.

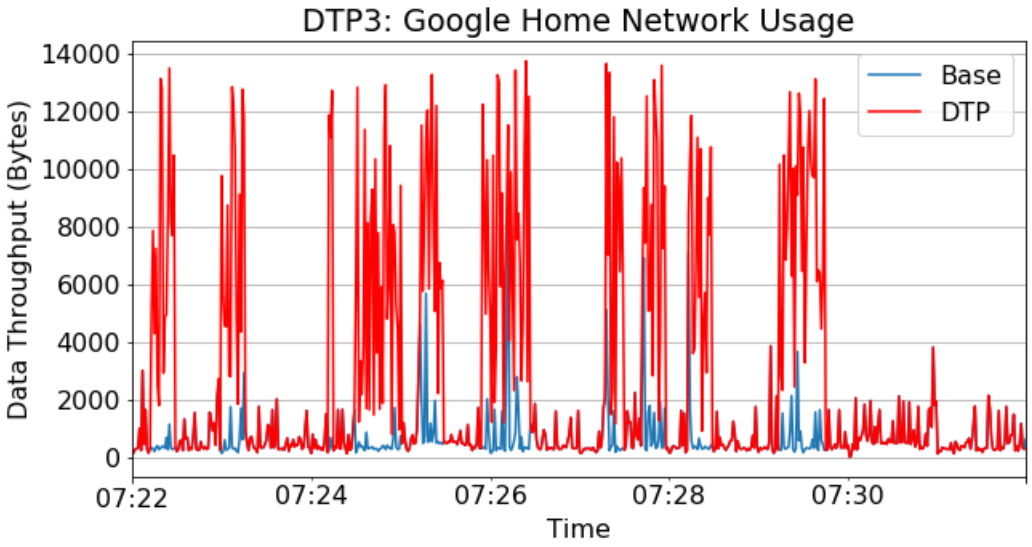

Figure 8.7: Google Home DTP3

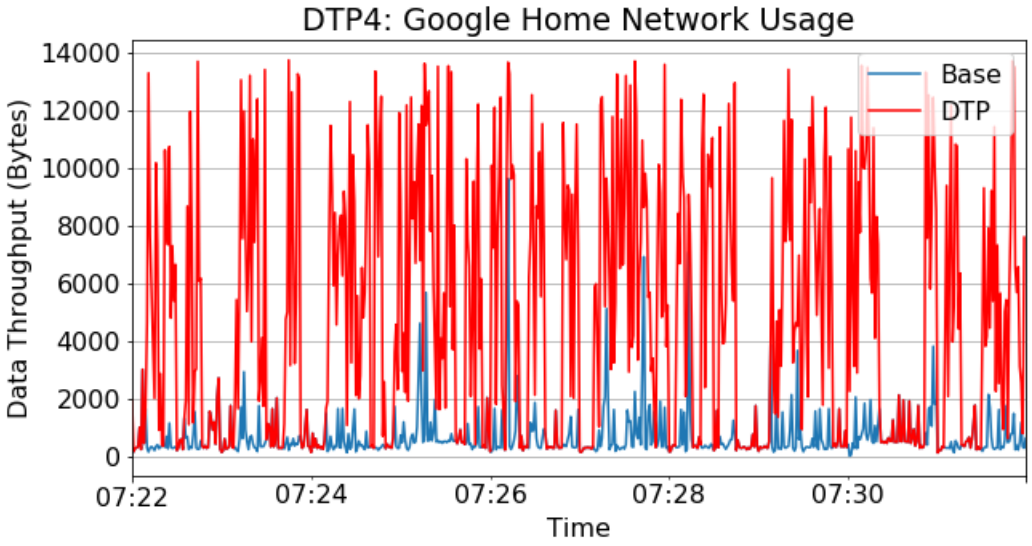

Figure 8.8: Google Home DTP4

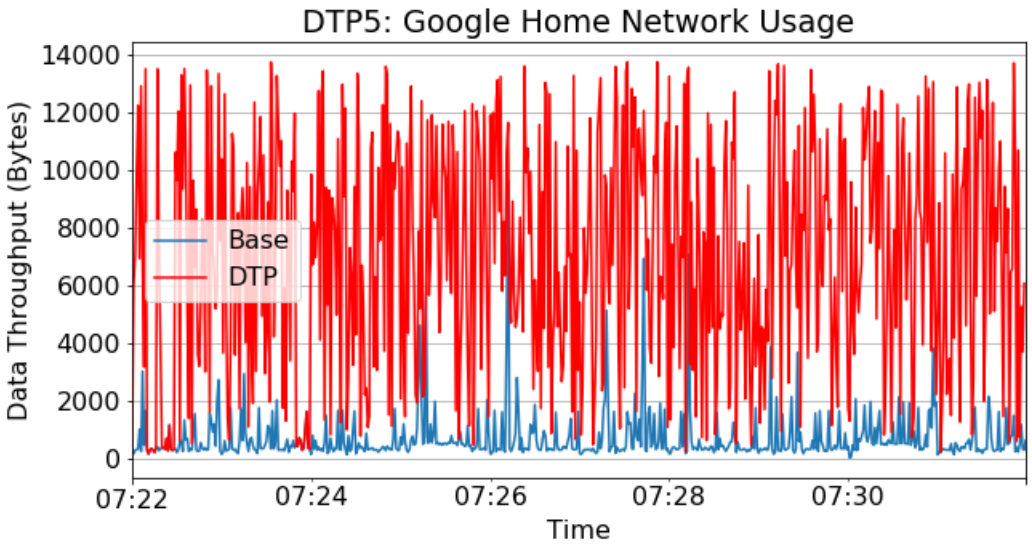

Figure 8.9: Google Home DTP5

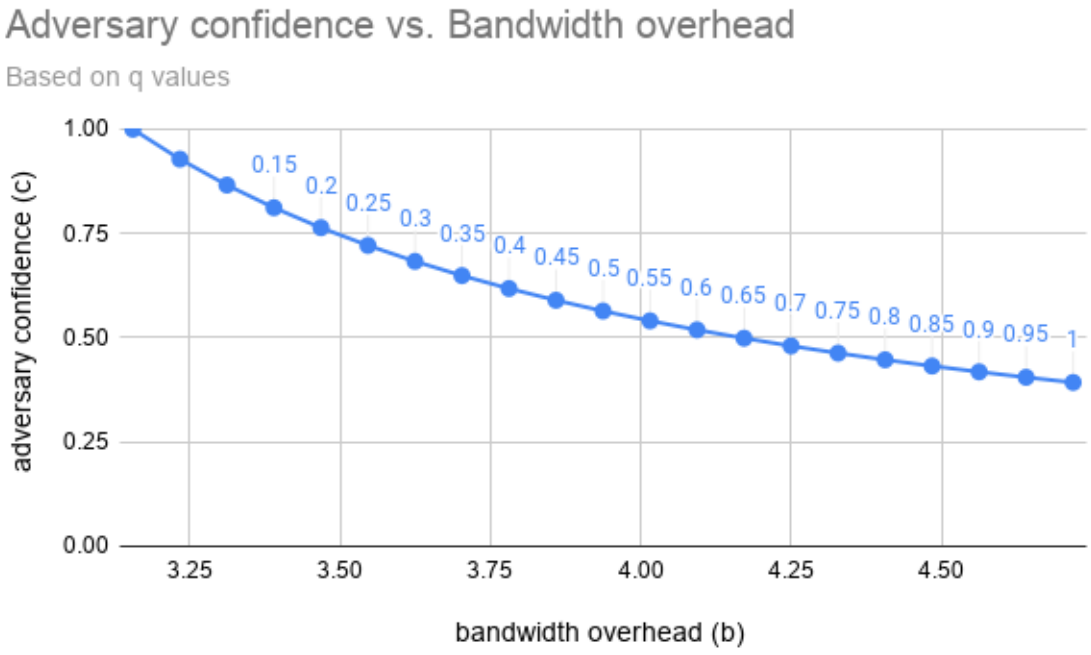

Figure 8.10: Chromecast Traffic Adversary Confidence vs Bandwidth Overhead

Chapter 9

CONCLUSION & FUTURE WORK

9.1 Conclusion

We examined traffic rates from different IoT devices in a smart home to analyze their privacy. We found that privacy is not entirely preserved due to the fact that even encrypted traffic metadata can be used to plot time series data and infer user activity. Traffic shaping preserves privacy by obfuscating the traffic rates. We evaluated current solutions including Independent Link Padding algorithms and the Stochastic Traffic Padding algorithm.

STP can also be tuned based on the padding probability and the padding rate, which in turn correlate with overhead and adversary confidence [25], but these are difficult to determine. We introduced Dynamic Traffic Padding, which uses latency tolerance and privacy preference to tune variables set in the STP algorithm making it more user-friendly. We were able to reduce bandwidth overhead, for Chromecast by 200% on average, while maintaining privacy.

9.2 Future Work

Future work would need to be done to improve the machine learning aspect to achieve an optimal set of values for the tunable variables. In addition, running these algorithms in a live IoT network would be beneficial to achieve more accurate results based on previous week's traffic. Next, there needs to be work done to identify the trade-off between using a single box or router to shape traffic or to run the algorithms on a device level.

BIBLIOGRAPHY

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, 2010.

[2] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in *2012 10th International Conference on Frontiers of Information Technology*, 2012, pp. 257–260.

[3] K. L. Lueth, "State of the iot 2018: Number of iot devices now at 7b – market accelerating," 2018.

[4] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "The internet of things: Mapping the value beyond the hype," Tech. Rep., 2015.

[5] S. Cook, "60+ iot statistics and facts," 2020.

[6] (2019) Iot smart home automation. [Online]. Available: https://www.digiteum.com/iot-smart-home-automation

[7] K. Sha, W. Wei, T. Andrew Yang, Z. Wang, and W. Shi, "On security challenges and open issues in internet of things," *Future Generation Computer Systems*, vol. 83, pp. 326 – 337, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17324883

[8] A. MERELLA, "Iot security issues and risks," 2018.

[9] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot

devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[10] L. Mukherjee, "The owasp iot top 10 list of vulnerabilities," 2020.

[11] R. Lemos, "California's iot security law causing confusion," 2019.

[12] C. Maynard. (2019) Wireshark capture setup: Wlan. [Online]. Available: https://wiki.wireshark.org/CaptureSetup/WLAN

[13] (2020) Airodump-ng. [Online]. Available: https://www.aircrack-ng.org/~~V:/doku.php?id=airodump-ng

[14] J. McDermott, "What are traffic analysis and metadata?" 2016.

[15] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[16] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 559–564.

[17] "Virtual private networking: An overview," Microsoft, Tech. Rep., 2009.

[18] I. Sanchez, R. Satta, I. N. Fovino, G. Baldini, G. Steri, D. Shaw, and A. Ciardulli, "Privacy leakages in smart home wireless technologies," in *2014 International Carnahan Conference on Security Technology (ICCST)*, 2014.

[19] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of internet of things," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018.

[20] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. Association for Computing Machinery, 2017, p. 506–509.

[21] A. Sivanathan, "Iot behavioral monitoring via network traffic analysis," *arXiv preprint arXiv:2001.10632*, 2020.

[22] W. Li, X. Ji, Y. Cheng, W. Xu, and X. Zhou, "User presence inference via encrypted traffic of wireless camera in smart homes," *Security and Communication Networks*, 2018.

[23] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," *arXiv preprint arXiv:1708.05044*, 2017.

[24] C. Kiraly, S. Teofili, G. Bianchi, R. Lo Cigno, M. Nardelli, and E. Delzeri, *Traffic Flow Confidentiality in IPsec: Protocol and Implementation*, 2008, vol. 262.

[25] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) iot traffic shaping," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, 2019.

[26] D. Wood, N. Apthorpe, and N. Feamster, "Cleartext data transmissions in consumer iot medical devices," in *Proceedings of the 2017 Workshop on*

Internet of Things Security and Privacy, ser. IoTSP '17. Association for Computing Machinery, 2017, p. 7–12. [Online]. Available: https://doi.org/10.1145/3139937.3139939

[27] "Internet protocol security (ipsec)."

[28] "What is transport layer security (tls)?" [Online]. Available: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

[29] A. Hussein, I. Elhajj, A. Chehab, and A. Kayssi, "Securing diameter: Comparing tls, dtls, and ipsec," 11 2016, pp. 1–8.

[30] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted dns–¿ privacy? a traffic analysis perspective," *arXiv preprint arXiv:1906.09682*, 2019.

[31] T. Datta, N. Apthorpe, and N. Feamster, "A developer-friendly library for smart home iot privacy-preserving traffic obfuscation," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, ser. IoT SP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 43–48. [Online]. Available: https://doi.org/10.1145/3229565.3229567

[32] W. Yan, E. Hou, and N. Ansari, "Defending against traffic analysis attacks with link padding for bursty traffics," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, 2004, pp. 46–51.

[33] W. Wang, M. Motani, and V. Srinivasan, "Dependent link padding algorithms for low latency anonymity systems," 2008.

[34] C.-M. Cheng, H. Kung, and K.-S. Tan, "On-demand link padding in traffic anonymizing," vol. 6, 2005.

[35] J. Frankenfield. (2020) Eavesdropping attack. [Online]. Available: https://www.investopedia.com/terms/e/eavesdropping-attack.asp

[36] "Cal Poly Github," http://www.github.com/CalPoly.

[37] M. binti Mohamad Noor and W. H. Hassan, "Current research on internet of things (iot) security: A survey," *Computer Networks*, vol. 148, pp. 283 – 294, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618307035

[38] A. Capossele, V. Cervo, G. De Cicco, and C. Petrioli, "Security as a coap resource: An optimized dtls implementation for the iot," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 549–554.

[39] R. Hummen, J. Hiller, M. Henze, and K. Wehrle, "Slimfit — a hip dex compression layer for the ip-based internet of things," 10 2013.

[40] Tokyoneon, "Stealthfully sniff wi-fi activity without connecting to a target router," 2019. [Online]. Available: https://null-byte.wonderhowto.com/how-to/stealthfully-sniff-wi-fi-activity-without-connecting-target-router-0183444/

[41] F. Le, J. Ortiz, D. Verma, and D. Kandlur, *Policy-Based Identification of IoT Devices' Vendor and Type by DNS Traffic Analysis*. Cham: Springer International Publishing, 2019, pp. 180–201. [Online]. Available: https://doi.org/10.1007/978-3-030-17277-0_10

[42] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.1901

APPENDICES

Appendix A

DATA-SET BACKGROUND

The data-set we used in our thesis came from a previous group that captured both network packets and power usage of each IoT device connected to their network. They recorded user events in a spreadsheet that correlated what device was being used, for what reason, and at what time. They began plotting graphs of data throughput against time to show what user activity information can be drawn. The events ranged from using a single device or multiple devices for short periods of time like 20-30 minutes. Several events took place within a couple hours and there was hours if not days breaks in between these events due to the fact that a student used the devices at certain times of the week.

The data-set consisted of an ip table and a power table. Figure A.1 shows the columns for the ip table and Figure A.2 shows the columns for the power table.

This work uses the ip table as it consists of captured network packets from the IoT devices on the network. Some of the fields in the ip table were not filled out for several of the packets. Figure A.3 shows a few rows from the ip table.

For traffic analysis and traffic shaping, only the timestamp and size are required to be able to plot a time-series graph of data throughput against time. Therefore, in our pre-processing of the data we removed all the other columns so our dataframe consisted of only time and size as shown in Figure A.4.

```
mysql> describe ip;
+-------------+--------------+
| Field       | Type         |
+-------------+--------------+
| time        | datetime     |
| source      | varchar(45)  |
| src_host    | varchar(100) |
| destination | varchar(45)  |
| dst_host    | varchar(100) |
| protocol    | varchar(100) |
| type        | varchar(45)  |
| src_port    | varchar(100) |
| dst_port    | varchar(100) |
| size        | int(11)      |
| hexdump     | longtext     |
| dst_geo     | varchar(100) |
| src_geo     | varchar(100) |
+-------------+--------------+
13 rows in set (0.00 sec)
```

Figure A.1: ip table columns

```
mysql> describe power;
+---------------+--------------+
| Field         | Type         |
+---------------+--------------+
| name          | varchar(100) |
| power_mw      | int(11)      |
| time          | datetime     |
| today_kwh     | varchar(100) |
| on_for        | varchar(100) |
| today_on_time | varchar(100) |
+---------------+--------------+
6 rows in set (0.00 sec)
```

Figure A.2: power table columns

	time	source	src_host	destination	dst_host	protocol	type	src_port	dst_port	size	hexdump	dst_geo	src_geo
0	2018-02-02 15:16:18	0.0.0.0	None	255.255.255.255	None	UDP	outgoing	None	None	390	None	None	None
1	2018-02-02 15:16:33	0.0.0.0	None	255.255.255.255	None	UDP	incoming	None	None	344	None	None	None
2	2018-02-02 15:16:33	0.0.0.0	None	255.255.255.255	None	UDP	outgoing	None	None	344	None	None	None
3	2018-02-02 15:16:35	0.0.0.0	None	255.255.255.255	None	UDP	incoming	None	None	344	None	None	None
4	2018-02-02 15:16:35	0.0.0.0	None	255.255.255.255	None	UDP	outgoing	None	None	344	None	None	None
5	2018-02-02 15:16:36	192.168.12.1	None	192.168.12.199	None	ICMP	outgoing	None	None	48	None	None	None
6	2018-02-02 15:16:36	192.168.12.1	None	192.168.12.199	None	UDP	outgoing	None	None	328	None	None	None
7	2018-02-02 15:16:36	192.168.12.1	None	192.168.12.199	None	UDP	outgoing	None	None	328	None	None	None
8	2018-02-02 15:16:36	0.0.0.0	None	255.255.255.255	None	UDP	incoming	None	None	356	None	None	None
9	2018-02-02 15:16:37	0.0.0.0	None	255.255.255.255	None	UDP	outgoing	None	None	356	None	None	None

Figure A.3: ip table rows

	Time	Size
0	2018-05-11 01:25:00	918
1	2018-05-11 01:25:01	1460
2	2018-05-11 01:25:02	1752
3	2018-05-11 01:25:03	3332
4	2018-05-11 01:25:04	476

Figure A.4: processed ip table

Appendix B

CODE

B.1 Pre-Processing Data

```
#connect to MySQL database
conn = MySQLdb.connect(host="localhost", user="root", passwd="", db="iot")
cursor = conn.cursor()
#fetch chromecast packets time and size
cursor.execute('SELECT time, size FROM ip WHERE time >="2018-05-11 1:25:00" AND time <"2018-05-11 1:32:00"')
rows = cursor.fetchall()
#create a dataframe of packets
df = pd.DataFrame( [[ij for ij in i] for i in rows] )
df.rename(columns={0: 'Time', 1: 'Size'}, inplace=True)
df['Size'] = pd.to_numeric(df['Size'])
#this is the individual packets
chromecast_packets_df = df
#these are packets grouped by time
df = df.groupby(['Time']).sum().reset_index()
#plotting time series graph
plt.rcParams.update({'font.size': 16})
ax = df.plot(x='Time', y='Size', kind='line', grid=True, title='Chromecast Network Usage', figsize=(10,5))
ax.set_xlabel('Time')
ax.set_ylabel('Data Throughput (Bytes)')
plt.savefig('figures/chromecast_throughput.png')
plt.show()
```

B.2 ILP-Const Algorithm

```
def ilp_constant_alg(input_df, constant_rate):
    output_df = pd.DataFrame(columns=['Time', 'Size'])
    stats = {'max_delay' : 0, 'num_packets_delayed' : 0, '
   total_padding' : 0, 'num_times_padded' : 0}
    padding = 0
    delayed_time = 0
    current_size = 0
    current_time = 0
    delayed_packets = []
    for index in range(len(input_df)):
        t = input_df['Time'].iloc[index]
        s = input_df['Size'].iloc[index]
        if t != current_time:
            if current_size < constant_rate and current_time != 0:
                padding = constant_rate - current_size
                output_df = output_df.append({'Time': current_time,
   'Size': padding}, ignore_index=True)
                stats['total_padding'] += padding
                stats['num_times_padded'] += 1
            current_time = t
            current_size = 0
        pop_count = 0
        for dt, ds in delayed_packets:
            if ds + current_size <= constant_rate:
                current_size += ds
                output_df = output_df.append({'Time': current_time,
   'Size': ds}, ignore_index=True)
                delayed_time = (current_time - dt).total_seconds()
                if delayed_time > stats['max_delay']:
                    stats['max_delay'] = delayed_time
```

```
                pop_count += 1
            else:
                break
        for i in range(pop_count):
            delayed_packets.pop(0)
        if s + current_size <= constant_rate:
            output_df = output_df.append({'Time': t, 'Size': s},
    ignore_index=True)
            current_size += s
        else:
            delayed_packets.append((t,s))
            stats['num_packets_delayed'] += 1
    if current_size < constant_rate and current_time != 0:
            padding = constant_rate - current_size
            output_df = output_df.append({'Time': current_time, '
    Size': padding}, ignore_index=True)
            stats['total_padding'] += padding
            stats['num_times_padded'] += 1
    return output_df, stats
#running ILP-Const on the chromecast packets
chromecast_ilp_const_df, chromecast_ilp_const_stats =
    ilp_constant_alg(chromecast_packets_df, 10000)
chromecast_ilp_const_df = chromecast_ilp_const_df.groupby(['Time']).
    sum().reset_index()
```

B.3 ILP-Var Algorithm

```
def ilp_variable_alg(input_df, min_rate, max_rate):
    output_df = pd.DataFrame(columns=['Time', 'Size'])
    stats = {'max_delay' : 0, 'num_packets_delayed' : 0, '
    total_padding' : 0, 'num_times_padded' : 0}
```

```
    padding = 0
    delayed_time = 0
    current_size = 0
    current_time = 0
    delayed_packets = []
    current_rate = 0
    for index in range(len(input_df)):
        t = input_df['Time'].iloc[index]
        s = input_df['Size'].iloc[index]
        if t != current_time:
            if current_size < current_rate and current_time != 0:
                padding = current_rate - current_size
                output_df = output_df.append({'Time': current_time, 'Size': padding}, ignore_index=True)
                stats['total_padding'] += padding
                stats['num_times_padded'] += 1
            current_time = t
            current_size = 0
            current_rate = random.randint(min_rate, max_rate)
        pop_count = 0
        for dt, ds in delayed_packets:
            if ds + current_size <= current_rate:
                current_size += ds
                output_df = output_df.append({'Time': current_time, 'Size': ds}, ignore_index=True)
                delayed_time = (current_time - dt).total_seconds()
                if delayed_time > stats['max_delay']:
                    stats['max_delay'] = delayed_time
                pop_count += 1
            else:
                break
        for i in range(pop_count):
```

```
            delayed_packets.pop(0)
        if s + current_size <= current_rate:
            output_df = output_df.append({'Time': t, 'Size': s},
    ignore_index=True)
            current_size += s
        else:
            delayed_packets.append((t,s))
            stats['num_packets_delayed'] += 1
    if current_size < current_rate and current_time != 0:
            padding = current_rate - current_size
            output_df = output_df.append({'Time': current_time, '
    Size': padding}, ignore_index=True)
            stats['total_padding'] += padding
            stats['num_times_padded'] += 1
    return output_df, stats
#running ILP-Var on chromecast packets
chromecast_ilp_var_df, chromecast_ilp_var_stats = ilp_variable_alg(
    chromecast_packets_df, 1000, 20000)
chromecast_ilp_var_df = chromecast_ilp_var_df.groupby(['Time']).sum
    ().reset_index()
```

B.4 STP Algorithm

```
def stp_alg(input_df, R, q, threshold):
    output_df = pd.DataFrame(columns=['Time', 'Size'])
    stats = {'max_delay' : 0, 'num_packets_delayed' : 0, '
    total_padding' : 0, 'num_times_padded' : 0}
    T = 10 #Time period length in sec
    padStart = 0
    padEnd = 0
    padOffset = 0
```

```
    current_time = input_df['Time'].iloc[0]
    totalTime = int((input_df['Time'].iloc[len(input_df)-1] -
current_time).total_seconds())
    for t in range(totalTime):
        if t % T == 0 and random.random() < q:
            padOffset = random.randint(0, T)
            if t + padOffset > padEnd:
                padStart = t + padOffset
                padEnd = padStart + T
            else:
                padEnd = padEnd + T
        temp_df = input_df[input_df['Time']==current_time]['Size']
        if t >= padStart and t <= padEnd:
            stats['num_times_padded'] += 1
            if temp_df.size > 0:
                s = temp_df.iloc[0]
                stats['total_padding'] += R - s
            else:
                stats['total_padding'] += R
            output_df = output_df.append({'Time': current_time, '
Size': R}, ignore_index=True)
        elif temp_df.size > 0:
            s = temp_df.iloc[0]
            if s > threshold:
                padStart = t
                padEnd = t + T
                stats['num_times_padded'] += 1
                stats['total_padding'] += R - s
                output_df = output_df.append({'Time': current_time,
'Size': R}, ignore_index=True)
            else:
```

```
                output_df = output_df.append({'Time': current_time, 'Size': s}, ignore_index=True)
            current_time = current_time + datetime.timedelta(seconds=1)
    return output_df, stats
#running STP on chromecast packets
chromecast_stp_df, chromecast_stp_stats = stp_alg(df, int(df['Size'].max()), 0.15, 7500)
```

B.5 DTP/STP-Var Algorithm

```
def dtp_stp_var_alg(input_df, R, q, threshold):
    output_df = pd.DataFrame(columns=['Time', 'Size'])
    stats = {'max_delay' : 0, 'num_packets_delayed' : 0, 'total_padding' : 0, 'num_times_padded' : 0}
    T = 15 #Time period length in sec
    padStart = 0
    padEnd = 0
    padOffset = 0
    current_time = input_df['Time'].iloc[0]
    totalTime = int((input_df['Time'].iloc[len(input_df)-1] - current_time).total_seconds())
    min_rate = R - 1000
    for t in range(totalTime):
        if t % T == 0 and random.random() < q:
            padOffset = random.randint(0, T)
            if t + padOffset > padEnd:
                padStart = t + padOffset
                padEnd = padStart + T
            else:
                padEnd = padEnd + T
        temp_df = input_df[input_df['Time']==current_time]['Size']
```

```
20        if t >= padStart and t <= padEnd:
21            stats['num_times_padded'] += 1
22            if temp_df.size > 0:
23                s = temp_df.iloc[0]
24                current_rate = random.randint(s, R)
25                stats['total_padding'] += current_rate - s
26            else:
27                current_rate = random.randint(min_rate, R)
28                stats['total_padding'] += current_rate
29            output_df = output_df.append({'Time': current_time, '
    Size': current_rate}, ignore_index=True)
30        elif temp_df.size > 0:
31            s = temp_df.iloc[0]
32            if s > threshold:
33                padStart = t
34                padEnd = t + T
35                stats['num_times_padded'] += 1
36                current_rate = random.randint(s, R)
37                stats['total_padding'] += current_rate - s
38                output_df = output_df.append({'Time': current_time,
    'Size': current_rate}, ignore_index=True)
39            else:
40                output_df = output_df.append({'Time': current_time,
    'Size': s}, ignore_index=True)
41        current_time = current_time + datetime.timedelta(seconds=1)
42     return output_df, stats
```