IT 20 046
Examensarbete 45 hp
Augusti 2020

The Dark Side of IoT

Revealing Activities in the Smart Home Environment using Machine Learning

Ahmed Hussain

Abstract

The Dark Side of IoT

Ahmed Mohamed Hussain

With the emerging trend of smart IoT devices, providing and ensuring the privacy of the users'-devices interaction relationship is a must; risks such as identifying users' activities and detecting user behavior while at home or in office. We present a new machine learning attack that exploits network patterns to detect the presence of smart IoT devices, and services generated by these devices in the Wi-Fi radio spectrum. This is considered a major problem when it comes to users' privacy since it allows malicious users to infer on the other users' activities. We perform an extensive measurement campaign for data collection. We build up a model describing the traffic patterns characterizing three popular IoT smart home devices, i.e., Google Nest, Google Chromecast, Amazon Echo, and Amazon Echo Dot. We prove that detecting and identifying with overwhelming probability the presence of the aforementioned devices in a crowded Wi-Fi scenario is possible. Finally, we introduce a mitigation technique to preserve the users' privacy in the network. This work proves that standard encryption techniques alone are not sufficient to protect the privacy of the end-user. In contrast, more work is required to prevent non-trusted third parties to detect and identify the users' devices.

Handledare: Gabriele Oligeri
Ämnesgranskare: Thiemo Voigt
Examinator: Mats Daniels
IT 20 046
Tryckt av: Reprocentralen ITC

Acknowledgment

I would like to thank Dr. Gabriele Oligeri and Prof. Thiemo Voigt for all their efforts, support, and gaudiness in making this thesis possible. Additionally, their endless patience during the entire period of the project.

I want to thank my family, especially my parents, for their endless and continuous support. I will be forever grateful.

Finally, a big thanks to all my friends, Andreas, Mauro, Naram, Pietro, Sayyaf, and Simone, for all the good times and their feedback on this work. And whoever I forgot to mention, big thank you!

To my parents, brother and sister.

Contents

List of Figures

List of Tables

Acronyms

Echo	Amazon Echo.
ED	Amazon Echo Dot.
FPR	False Positive Rate.
IAT	Interarrival Times.
IoT	Internet of Things.
PZ	Packet Size.
TLS	Transport Layer.
TPR	True Positive Rate.
VPN	Virtual Private Network.

Chapter 1

Introduction

In recent years, the IoT technology has emerged rapidly and made it a buzzword. But not anymore, since it now has taken over and become the core of everything in humans daily lives, furthermore, it became a lifestyle. IoT devices are usually connected through a wireless connection to the base station, to be connected to the internet, and to execute users commands. The wireless connection between these devices is considered to be secured and encrypted. However, different techniques and methodologies have been introduced to infer on the packets, namely encrypted traffic classification using machine learning. Encrypted network traffic classification has emerged as a technique to detect and identify traffic patterns despite being anonymized by one or more encryption layers. The main goal of traffic classification is to distinguish different applications' network traffic (represented as flows) based on statistical features independently of the information that can be retrieved from the packet headers, such as network and physical layer addresses. Standard features can be packet size and interarrival times between consecutive packets, while more advanced features take into account statistics such as mean and variance computed over sequences of N consecutive packets. While encryption layers added by the transport layer (TLS) or applications such as VPN, protect the end user from adversaries willing to exfiltrate information from the content of the packet. This affects traffic patterns, as they are very difficult to hide without affecting services performance. The wireless scenarios addressed in this thesis explores the effectiveness of such an attack. That is due to the Wi-Fi intrinsic broadcast nature, wireless communications can easily eavesdrop without the consent of the user, and an adversary can exploit the traffic pattern generated by users to infer either on the devices they have or the services being used. This presents major issues for the users' privacy, not just because non-trusted third parties can infer the user behavior, but also detect the presence of specific devices to target in subsequent attacks.

1.1 Problem Statement

With the exponential growth rate of the Internet of Things (IoT) devices, the number of connected devices is anticipated to reach 500 billion by 2030. One popular category among users is the Smart Speakers such as Amazon Echo and Google Nest Mini. Due to the services and features they provide, these Smart IoT devices are found and used almost everywhere, in places such as homes, offices, schools, and sometimes in government offices. Services are performed based on the user vocal command(s), such as streaming music, news, making phone calls, buying products online, and many others. Users can also control some services through portable devices, to perform certain actions. With this definitive explosion rate of IoT devices, a broad majority of today's network traffic is encrypted to ensure the security and privacy of the communications.

Recent techniques involving machine learning and artificial intelligence made it possible to infer what type of application and services the users are running on their devices either at home or in the workplace network, without being physically present or connected to their network. This thesis report introduces an adversarial model that leverages information found in the Wi-Fi network packets header to distinguishing and identifying services running by 4 different smart IoT devices: Amazon Echo, Amazon Echo Dot, Google Nest Mini, and Google Chromecast, using machine learning.

1.2 Methodology

The method presented in this thesis report is of experimental computer science. We present an adversarial model that is capable of identifying services/applications running on smart IoT devices, leveraging machine learning techniques using packet size, and interarrival times between packets. In the initial stage, we have studied the field of machine learning and network traffic classification, to identify which technique is applicable. Thereafter, we collected wireless (2.4Ghz) traffic between four different IoT devices and internet access point, streaming music, news, and YouTube. At this stage, we extracted both packet sizes and computed interarrival times between packets then classified them using machine learning. Additionally, 8 statistical features were extracted from both features, using the windowing technique. The classification is done on several stages, first: the packet sizes, second: interarrival times, and finally both features combined. The results are quantified in terms of accuracy and the number of packets needed to achieve this accuracy.

1.3 Limitations

Due to the wide variety of smart IoT devices and the time needed for generating as well as collecting traffic, a limited set was chosen. This set is made of the most widely used devices, manufactured by Google and Amazon. The services/applications running on these devices vary from one device to another. That is due to configurations defined by the manufacturer and the compatibility of some of the devices.

1.4 Scientific Contributions

In this thesis, We propose a methodology based on machine learning classification to detect and identify the presence of smart devices in the Wi-Fi radio spectrum. We tested our approach against four different smart devices, i.e., Google Chromecast, Google Nest Mini, Amazon Echo, and Amazon Echo Dot, and three services, i.e., Music, YouTube, and News streaming. We proved that the aforementioned devices and services are affected by a critical privacy leakage since they allow unauthorized third parties to detect their presence, even in the case of standard multi-layer encrypted streams such as WPA and HTTPS. This thesis proves that the most commonly used smart devices in the market and their related services can be easily detected with overwhelming probability (> 0.99) from a crowded Wi-Fi link. We provide deep statistical insights about the traffic characterization, we show how standard machine learning techniques can be used to detect the presence of the devices, and finally, we highlight potential countermeasures against our attack.

1.5 Report Structure

The remaining chapters of this thesis report are as follows: Chapter 2 presents the background related to statistics, machine learning, basic computer networks, and IoT devices. Chapter 3 addresses the related work done on encrypted traffic classification in general, and IoT in particular. In Chapter 4 we present the adversarial model along with the traffic collection set-up, and analysis of the traffic collected. In Chapter 5 we classify the traffic for each service and present the techniques used to achieve the optimal accuracy. Chapter 6 address the countermeasures and techniques used for mitigation. Finally, the thesis is concluded in Chapter 7, along with a discussion of the results and suggestions for future work.

1.6 Publications

This thesis resulted in a peer-reviewed paper [6] that represents a compact form of this work in addition to various differences (e.g., evaluations, devices, and such).

Chapter 2

Background

This chapter presents an overview of methodologies and tools used throughout this thesis study.

2.1 Statistics

In this section we provide a brief overview on some of the statistical features found in [7, 8] and used for data analysis in this thesis.

2.1.1 Statistical Methods

1. **Mean (μ):** is the sum of all values in the data-set divided by their number as shown in Equation 2.1. It is used to find the middle value in the data-set.
2. **Variance:** is the differences between each of the numbers in the data-set and the mean, squared. Then divided by the total number of elements in the data-set, as shown in Equation 2.2. It is used to measure how far each of the values from the mean, and therefore from every other value in the data-set.
3. **Standard Deviation (σ):** is the square root of the *variance*, as defined in Equation 2.3. It measures the dispersion of a data-set relative to its mean.
4. **Minimum:** is the minimum value in the data-set.
5. **Maximum:** is the maximum value in the data-set.
6. **Summation:** is the sum of all the values in the data-set.
7. **Skewness:** measures the asymmetry of the data around the sample mean and defined as in Equation 2.4.
8. **Kurtosis:** measures how outlier-prone a distribution is, and defined as in Equation 2.5.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{2.1}$$

$$V = \frac{1}{N-1}\sum_{i=1}^{N} |A_i - \mu|^2 \tag{2.2}$$

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} |A_i - \mu|^2} \tag{2.3}$$

$$s = \frac{E(x-\mu)^3}{\sigma^3} \tag{2.4}$$

$$k = \frac{E(x-\mu)^4}{\sigma^4} \tag{2.5}$$

2.1.2 Sliding-Window Computational Method

Analysis using the raw data collected for any application is effective when the data source is identified and the data-set size is considerably small. When dealing with large data-sets, the sliding window technique is considered handy, as it allows for extracting and generating new features from the original data-set, and reduces the data-size. which is useful as it provides more insights on the data, with less number of samples. It relays on a predefined N number of samples that to be mapped and represented by one sample. Applying this technique can be done using the previously mentioned statistical methods.

2.2 Machine Learning

Machine learning is a branch of today's popular artificial intelligence, regarded as the primary method of understanding and analyzing data in the modern age of big data. Machine learning algorithms are built to model intelligence by learning from previous events and anticipating what may arise in the future. Machine learning methods have been widely applied in numerous fields, including pattern analysis, computer vision to engineering in spacecraft. Figure 2.2.1[1],[2] shows the categories and their subcategories learning methods, that fall under the machine learning umbrella. There's one category that is not addressed which is Reinforcement Learning.

[1]RNN: Recurrent Neural Networks
[2]CNN: Convolution Neural Networks

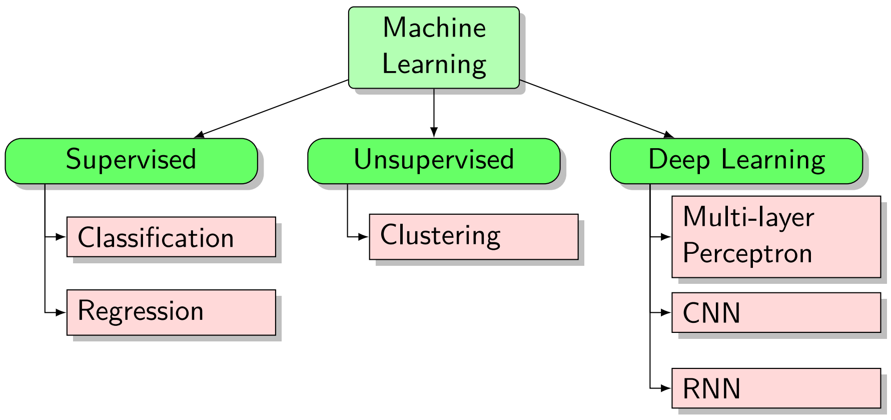


Figure 2.2.1: Machine Learning Categorization

2.2.1 Supervised Learning

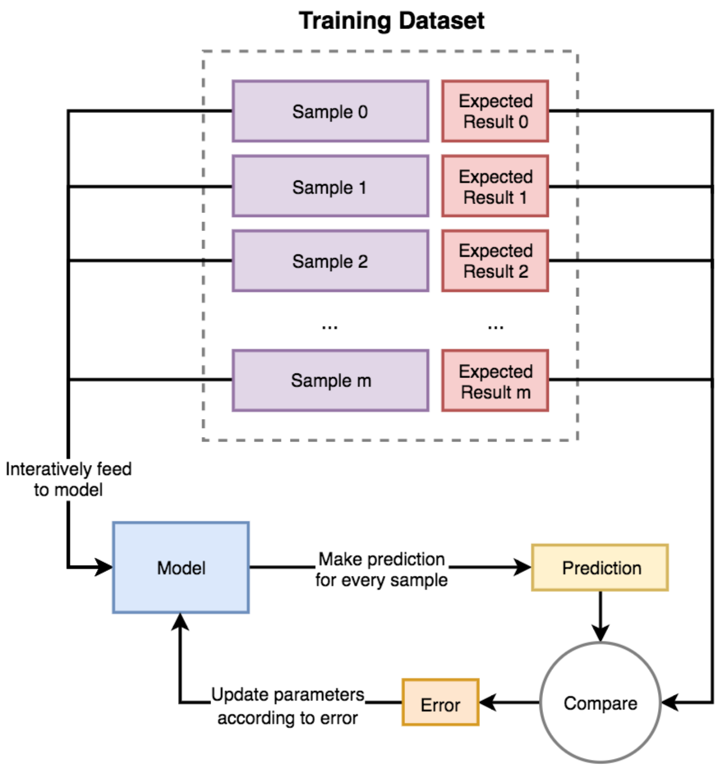


Figure 2.2.2: Supervised Learning Process [1]

Supervised learning a learning model based on mapping input data to labeled output data. Supervised learning algorithms such as linear regression or decision tree, includes training and building the model with training data, which consists of one or more input features as well as the desired output. These algorithms iteratively optimize the objective function to predict the desired output. Test data is then used to test how accurate the model is for predicting new unknown data. The test data-set is either from the training set or a new data-set. Supervised learning has two forms of tasks: *Regression* and *Classification*. Figure 2.2.3 illustrates the difference between the two approaches.

Regression

Regression techniques are typically used in predicting, forecasting, and finding relationships between quantitative data. It is widely used since it is one of the earliest learning techniques. Regression can be used to determine if a linear relationship exists between particular radiation therapy and tumor sizes.

Classification

Classification techniques are used in mapping different inputs to different outputs, i.e. $x \to y$. A typical example is whether or not an email is a spam. The algorithm can provide training data from e-mails that are spam and not spam utilizing a two-class data set. The model searches feature in the data that corresponds to either class and generates the mapping function that maps Y to $f(x)$. The built model uses this function to classify unknown emails, to determine if it is a spam or not.

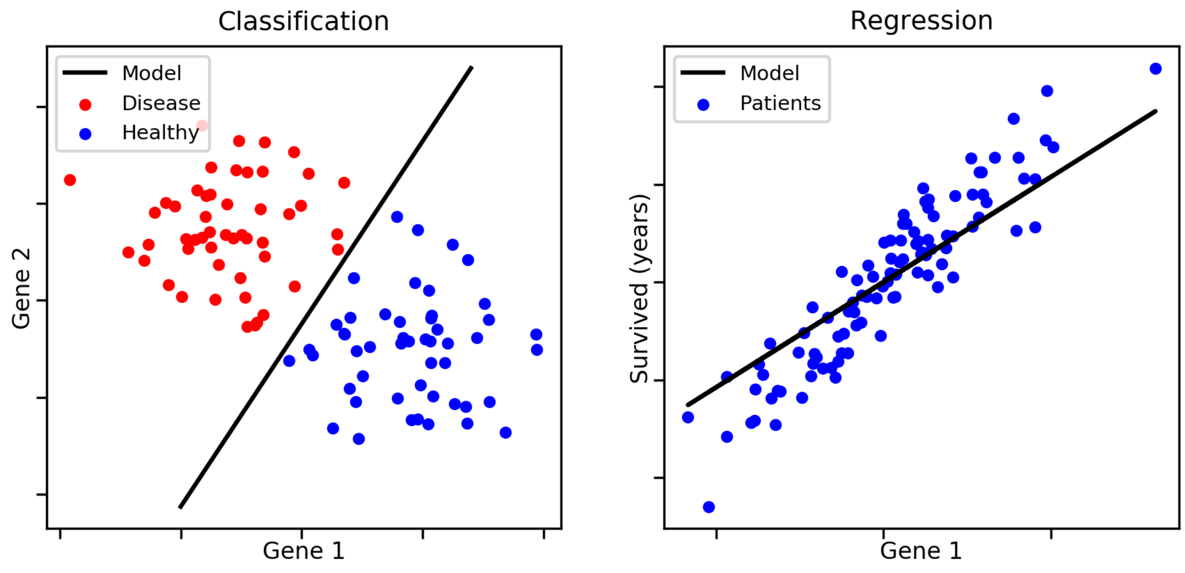


Figure 2.2.3: Gene example to illustrate the difference between Classification and Regression [2]

2.2.2 Decision Trees

Decision Trees apply either a sequential or multi-stage approach to assign labels to their leaves. Labeling leaves is based on a chain of decisions that are based on sequential test results rather than using single and complex decisions. The branches in the decision tree are made of a set that represents sequences of decisions, where the labels are represented by the leaves [9].

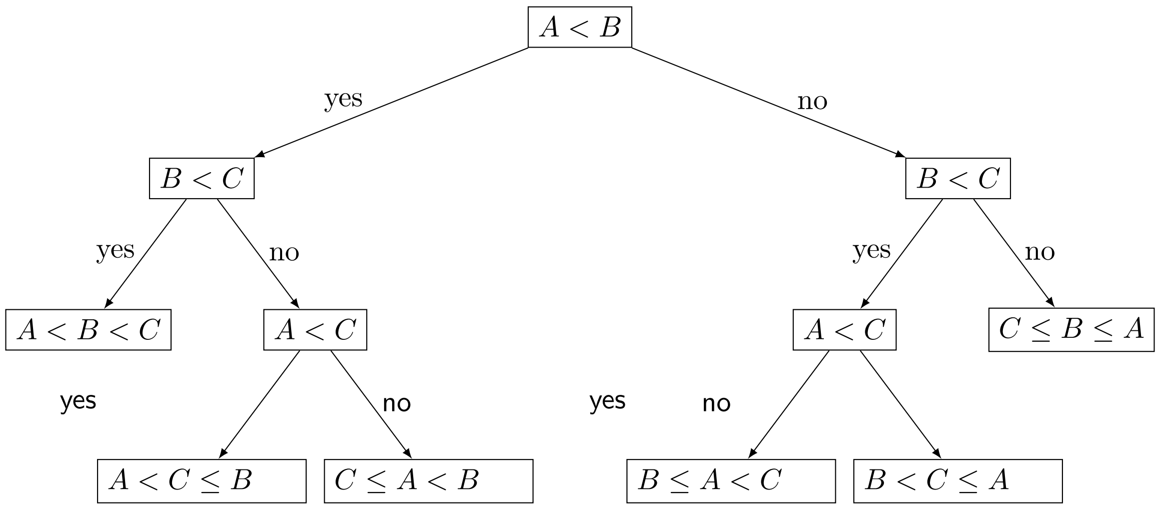


Figure 2.2.4: Decision Tree Representation

Random Forest

Random Forest [10] is an ensemble Machine Learning technique that is considered supervised and built as a combination of different tree predictors[3]. This type of ensemble classification relies on the fact that multiple classifiers (defined by the user) will provide a more reliable and generic classification rather than using one single classifier. This also ensures that the model is less susceptible to over-fitting. In *Random Forest*, each classifier is represented by a tree and depends on its own independently sampled random values vector. Once a large sufficient number of trees is generated, each tree selects the most popular class at input x. The randomization approach is used, to guarantee diversity among the decision trees.

2.2.3 Unsupervised Learning

In comparison with supervised learning, the output data is not labeled. Unsupervised algorithms aim to find structure and trends in the distribution of input data. For example, a clustering algorithm like K-Means attempts to cluster and partition the data based on the K number of centroids that define the clusters. Figure 2.2.5 shows how data is clustered into different groups.

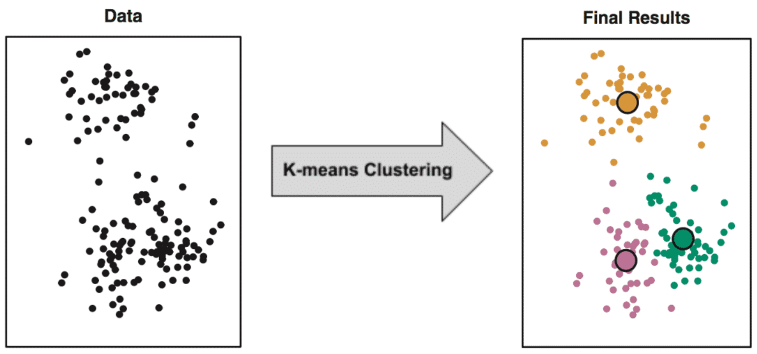


Figure 2.2.5: K-Means Clustering [3]

[3]Decision Tree(s)

2.2.4 K-Fold Cross Validation

Before starting with the classification procedures, the data-set is partitioned into the following three different sets:

1. training set to train the model,
2. validation set on to validate the models and
3. test set to evaluate the trained model.

Cross-Validation (CV) involves dividing the original data-set into k-folds of equal number of samples. From a set that has k samples, the first $k-1$ samples are to train the model, while the k sample is used for validation of the built model [11]. Figure 2.2.6 illustrates how the k-fold CV works.

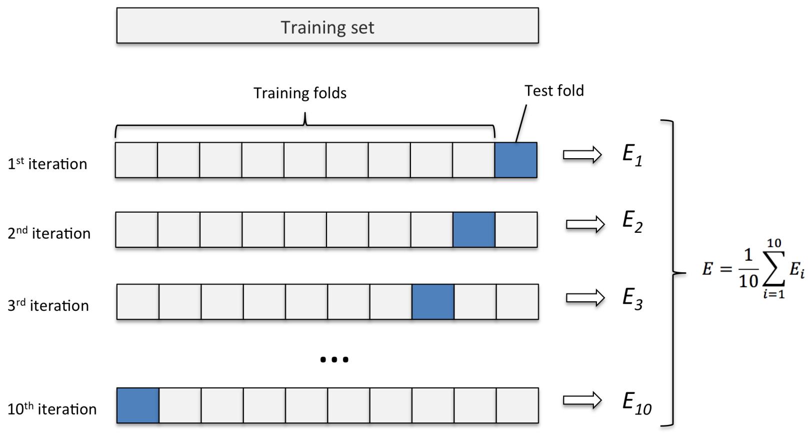


Figure 2.2.6: K-fold Cross Validation [4]

2.2.5 Deep Learning

Deep learning is a form of artificial intelligence and machine learning framework that is widely used today for a range of classification tasks. It allows multi-layered computational models to learn data by extracting more features with increasing the number of hidden layers (where training happens) in the network. Each layer consists of more than one artificial *neuron.* Neural networks operate by manipulating the input data by changing their weights, which are hypotheses about how the input features are mapped with each other and the object's class. As the network is trained, the weights are changed, and they converge on weights that accurately represent the relationships between features. Figure 2.2.8 illustrates a common deep learning network with N inputs, N hidden layers, and N Outputs. A deep learning network is also referred to as an artificial neural network (ANN).

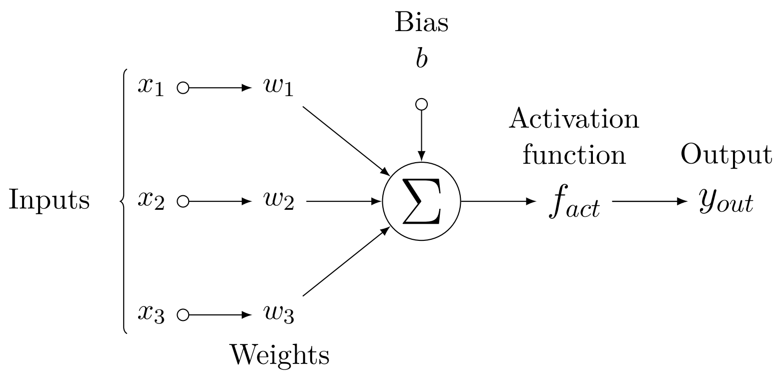


Figure 2.2.7: Neuron Node

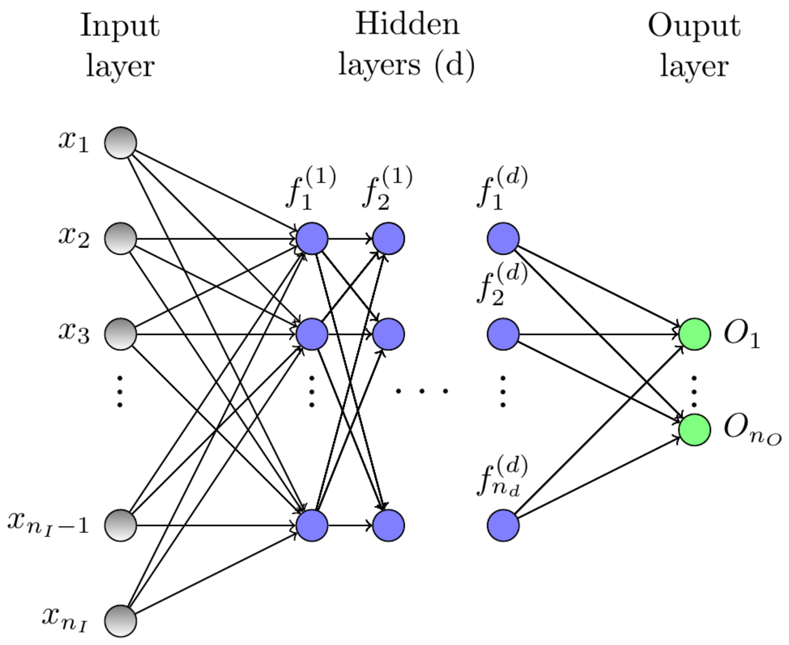


Figure 2.2.8: Artificial Neural Learning Network

Deep Learning Techniques

- **Multi-Layer Perceptron Neural Networks (MLP).** Is simply a deep artificial neural network that is made of multiple perceptrons. The structure of the network is simply made of 2 main layers, which are the input and output layers. The input layer is to receive the signal, and the output layer predicts which output perceptron the input relates to. Between these layers, one or more hidden layers that are the main computing engine behind MLP [12].
- **Convolutional Neural Networks (CNNs).** Similar to MLP, CNNs are a type of neural network that used mainly to classify images, cluster images by correspondence, and perform object recognition. For example, convolutional neural networks are used to recognize faces, individuals, street signs, and tumors [13].
- **Recurrent Neural Networks (RNNs).** Is an artificial neural network that recognizes patterns in data sequences. Such as data from sensors, financial markets, and government

agencies (but not limited to) numeric times series [14].

- **Generative Adversarial Networks (GANs).** These are systems of algorithms that use two neural networks to compete against each other, to construct new artificial data instances that can be passed on to true data. They are mainly used in the generation of pictures, videos, and voices [15].

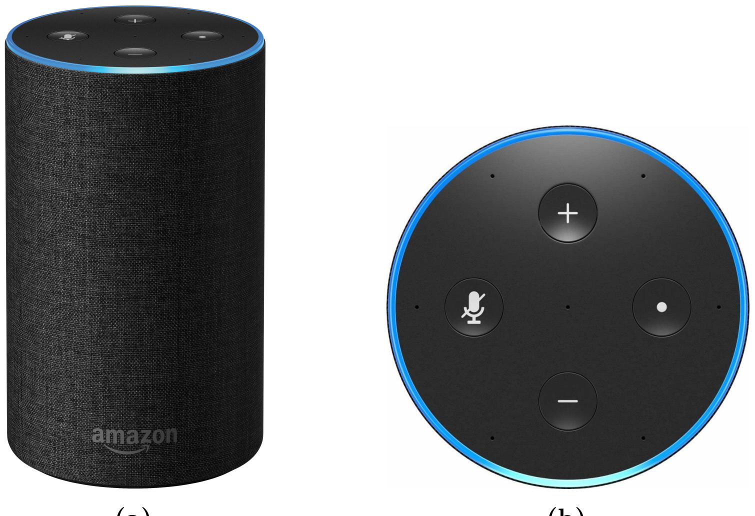
2.3 Internet of Things (IoT)

The IoT environment contains web-enabled, intelligent devices using embedded technologies, including processors, sensors, and communication hardware. The devices do most of the work, although people can interact with these devices. IoT can also use artificial intelligence (AI) and machine learning to facilitate and dynamically collect data. Linkage, networking, and communication protocols used with these devices depend largely on the specific IoT applications used. Connecting to these web-enabled systems, networking, and communication protocols ultimately depend on the IoT applications used. IoT can enhance communications also make it faster and through using artificial intelligence (AI) and machine learning.

2.3.1 IoT Home Devices

Amazon Echo

Echo is a lineup of smart speakers manufactured and sold by Amazon [16]. Echo is equipped with a total of 7 omnidirectional microphones designed to pick up the user voice from anywhere in the room. 6 distributed circularly and 1 microphone in the middle. Figure 2.3.1 (b) shows the locations of the exact microphones. The device is powered by a virtual assistant named ”Alexa” that takes the user vocal command/order and execute a variety of commands. Such as playing music, reading news or an e-book, reporting the environment and traffic, sports scores and schedule, calling, shopping, checking users’ bank account, film scheduling, checking the users’ calendar, setting up time-trip, laughing, hosting a game night and answering questions. It can also communicate with intelligent home devices such as Philips hue bulbs [17].



(a) (b)

Figure 2.3.1: Amazon Echo (a) and Microphones Locations (b)

Amazon Echo Dot

Echo Dot [18] is same as *Echo* (previously described) in term of features and functionalities. The only difference is size and number of microphones. Echo dot has only four far-field microphone array designed to pick up the user voice commands.

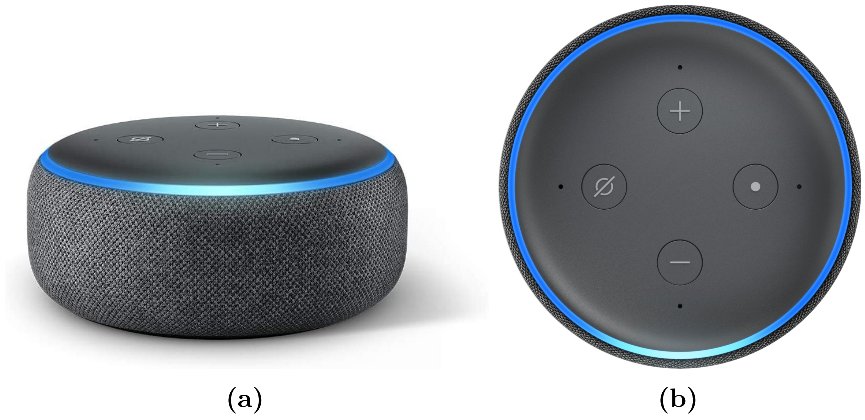


Figure 2.3.2: Amazon Echo Dot (a) and Microphones Locations (b)

Google Nest Mini

Nest Mini is a smart home device developed and sold by Google. Equipped with only three far-field microphones for voice detection and powered by Google Assistant, the device offers similar features provided by Echo, ranging from streaming music and news, to control smart home devices [19, 20].

Figure 2.3.3: Google Nest Mini (2nd Generation)

Google Chromecast

Chromecast is a streaming wireless device that connects users' phone, tablet, or computer with an HD monitor. If your monitor or TV screen has an HDMI port (which nearly every TV screen nowadays has), you can turn it into a smart device that can connect to all sorts of apps you have on your phone or computer.

Figure 2.3.4: Google Chromecast

Chapter 3

Related Work

This chapter provides an overview of the work that has been done in the area of network traffic classification. Followed by an in-depth and detailed review of the classification of the wireless encrypted traffic and its related works.

3.1 Network Traffic Classification

Classification of regular network traffic started in the mid-2000s without the use of the Machine Learning techniques, where authors in [21] introduced a payload-based algorithm for Peer to Peer (P2P) traffic identification. The identification is based on transport layer protocols TCP and UDP and identifying specific bit strings in the application level user data. Using features such as source/destination port and IP addresses, as well as payload size. This method is not a machine learning-based, it was comparison based. It was used to identify different P2P protocols.

Roughan et al. [22] proposed a supervised classification framework that uses nearest neighbors, linear discriminate analysis, and Quadratic Discriminant Analysis algorithms to classify different network applications traffic to predetermined QoS traffic classes. They used packet size as the only feature to distinguish between application classes.

Moore and Zuev [23] in 2005 used *Naïve Bayes* to classify traffic by application. Their traffic data-set consisted of applications for data transfer (FTP), HTTP protocol, database communications[1], ssh, mail protocols, worms, and virus attacks. For an accurate evaluation, they manually categorized each of the flows, based on its content. Features extracted from packets in each flow such as the total flow duration, TCP port, statistical features (such as mean and variance) for the inter-arrival time between packets, and the payload size. In addition to the *Fourier* transformation of the inter-arrival time between each of the packets, and effective bandwidth based on entropy. Naïve Bayes showed poor performance in comparison with the *Naïve Bayes kernel estimation method.* Naïve Bayes scored 65.26% while using the kernel estimation method achieved 93.50%.

Li and Moore [24] introduced an efficient approach to classify the traffic using the C4.5 decision tree classifier, without investigating the packet content. They achieved 99.8% total accuracy by collecting 12 features, from the first five packets of the flows. The majority of the traffic mainly had than 80% of web browsing traffic.

[1]User executing queries in the network.

3.2 Encrypted Network Traffic Classification

Security and privacy of communication are of paramount importance given the growing number of devices. The vast majority of today's network traffic is encrypted to guarantee the security and privacy of the communications. Different tools (such as VPN and Tor) is used to ensure *Confidentiality*, *Integrity* and *Availability* (C.I.A) of the communications between the machines[2] across different networks. Several studies showed that it is possible to identify the applications running by the user in the network.

3.2.1 Encrypted Website Traffic Identification

Kausar et al. [25] proposed an attack scenario that is based on a side-channel attack and leverages cluster classification to identify visited web pages. The attacker eavesdrops the wireless encrypted traffic between the user smartphone and the access point. Timestamp and packet direction is then extracted from the packet and used as the features to classify the traffic using K-Nearest Neighbor Classifier. The accuracy achieved of their attack is 96%.

3.2.2 Supervised Encrypted Application Traffic Classifications

Alshammari et al. [26] assessed the robustness of machine learning-based traffic classification for classifying encrypted traffic for SSH and Skype sessions. Five different Machine Learning methods were used and trained with data from one network. For testing, data from an entirely different network were used. The classifiers have been evaluated using flow-based features, where IP addresses, source/destination ports, and payload information have been anonymized. Results indicate the decision tree classifier performs much better than other algorithms on the identification of both SSH and Skype traffic on totally different networks.

Zeng et al. [27] introduced a general framework that is deep learning-based for traffic classification and intrusion detection. The (deep full range) DFR is a light framework is based on three different deep learning algorithms: Convolutional Neural Network, Long Short-Term Memory, and Stacked Auto-Encoder. The data-set consisted of 6 types of regular encrypted traffic such as email, chat, streaming, file transfer, VoIP, and P2P. The accuracy achieved using this framework was above 99.6% for all the 6, in comparison with classifying 1D-CNN and decision tree.

3.2.3 IoT Traffic classification

Acar et al. [28] demonstrated how to profile smart home IoT devices communications through multi-stage privacy attacks and applying machine learning on features extracted. The proposed adversary model can automatically detect the activities of IoT devices and identify the users' activities, even if the communication is encrypted. The classification of the types, actions, states, activities of the devices achieved very high accuracy, which is above 90%. [29] performed analysis on more than 20 IoT devices such as cameras, lights, appliances, and health monitors. Data-set used consisted of traces collected over 3 weeks. From the data-set, they extracted features such as data rates and burstiness, activity cycles, and signaling patterns. Using these features they tested two approaches developed a random forest-based classification method, that can distinguish IoT from non-IoT traffic, furthermore identifying specific IoT devices with over 95% accuracy.

Jackson et al. [30] investigated the use of six different supervised machine learning algorithms (C4.5 decision trees, random forests, linear kernel support vector machines, radial basis function kernel support vector machines, multilayer perceptron neural networks, and k-nearest neighbors

[2]A server or regular computer machine

classification), to extract information from the encrypted TCP packets. They implemented a request type classification that consists of classifying the encrypted information exchanged between Alexa cloud service and the Echo device by the type of user request that is being answered. Using three different feature vectors: TCPtrace [31] Features [32] Vector, Histogram Feature Vector, and Combined Feature Vector which combines the previous two features vectors. Classification results show that using Random Forests, they were able to achieve accuracy more than 93% for each of the features vectors, surpassing the accuracy of the other five algorithms.

Msadek et al. [33] proposed machine learning-based fingerprinting, to map IoT devices to their associated encrypted traffic flow. They assumed that the adversary is capable of passively collecting the traffic generated from the IoT devices while being remote, by either monitoring the gateway or through a compromised IoT device. The data-set used in their analysis [34] constitutes of 7 main categories: hubs, cameras, health care devices, light bulbs, electronics, air quality sensors, switches, and triggers. All of the devices were connected to an access point through wireless, except three, connected by wire. The network set-up in their study was adapted from [35]. The adversary is capable of obtaining the bidirectional flows generated by the victim. The classification of the traffic was done using five different classification algorithms. *Adaboost* [36] outperformed the other methods with 95.5% accuracy in identifying the type of the device.

Shahid et al. [37] presented a machine learning-based approach to identify the type of IoT devices connected to the network, by analyzing the bidirectional flows. From the first N packets exchanged between the internet and IoT devices, they extracted two features: packet size and inter-arrival times. They considered four different IoT devices: D-link Motion Sensor, Nest Security Camera, TP-Link Smart Plug, and a smart Bulb. Their devices were connected to an access point which in turn was not connected directly to the internet. The traffic from, and to the internet is redirected through a Raspberry Pi that is used for capturing the traffic. Which is not directly collected from the wireless link between the devices and the router. Random Forest, Decision Tree, Support Vector Machine, K-Nearest Neighbors, Artificial Neural Network and Naive Bayes were used for classification. The best performance has been achieved by using Random Forest with 99.9% accuracy.

Sivanathan et al. [29] applied machine learning on a data-set collected over three weeks, for more than 20 IoT devices. Such as cameras, lights, appliances, and health monitors. They extracted features such as data rates and burstiness, activity cycles, and signaling patterns. By using these features, they were able to exploit and distinguish IoT from non-IoT traffic, furthermore identifying specific IoT devices with about 95% accuracy.

Santos et al. [38] used a data-set that was collected within a smart campus [29] to identify and classify the traffic. The total number of devices in this data-set is 18. Only 11 of them are IoT devices, while 7 are none IoT devices. The considered features were maximum packet size in the forward direction, source port, destination port, and average packet size. They were able to identify each device network traffic flow, but not the application running, with 99% of accuracy by using the Random Forest algorithm.

Chapter 4

Adversarial Model, Measurements Set-up & Initial Observations

In this chapter, we introduce our measurement set-up by considering both the hardware and the software tools we adopted, the adversarial model, and scenarios. Finally, analysis of the collected data to identify if a pattern exists between the collected traffic for devices running different services.

4.1 Hardware Set-up

Hardware–Smart Devices. Our smart home set-up consists of the four smart home devices mentioned in chapter 2. These devices are distributed in different places and connected to a D-Link router (1200AC) via Wi-Fi.

Adversary–Hardware. Alfa Card (AWUS036NH) was adopted for capturing the traffic by switching to *monitor mode*, and therefore, collecting all the over-the-air data traffic.

Adversary–Software. We consider a standard Linux distribution (Kali Linux), and the tools *airodump-ng*/*Wireshark* for collecting and logging the Wi-Fi packets. Finally, we adopted *MATLAB 2019b* for data pre-processing and classification.

4.2 Data Collection Procedures

We would like to stress that we resort to MAC addresses to select the traffic of the smart devices and generate the ground truth with the machine learning algorithms. As it will be clear in the following chapters, the classification will be independent of the aforementioned information, while being rooted only on features like packet size, interarrival times, and statistical computations derived from them. The data collection has been performed accordingly to the following procedure:

1. We adopted *airodump-ng* to identify the victim access point and the devices connected to it.
2. Identifying[1] information such as MAC addresses and WiFi channel used by the devices in the network.

[1]Device type can be identified using: `https://macvendors.com/`

3. Setting the Alfa card channel to the one used by the smart device.
4. Using either *TCPDump* or *Wireshark* for traffic collection and filtering out the traffic generated from the access point to the IoT device. Using the MAC address obtained from step 1.
5. Extract the time and packet size associated with each over-the-air message.

The traffic forwarded by the access point to each of the smart devices is collected over approximately 8 hours of continuous news, music, and YouTube streaming. In our set-up, we ensure that no other devices are connected to the network other than the IoT devices. The total number of packets forwarded by the Wi-Fi access point to each device is approximately 280,000 for each service. In total, the overall data-set size sums up to 1,960,000 packets of multimedia streaming by all devices.

4.3 Adversary Model

The adversary model proposed in this thesis has two main characteristics:

- Being *Stealthy*, and
- *Resource Constrained.*

Based on the assumption that the adversary is capable of eavesdropping and collecting the Wi-Fi traffic, the adversary can be located within the premises of the victim (hiding in plain sight, which makes him stealthy) or far away (heavily depends on the equipment's capabilities). Resource-constrained means that the adversary has limited access to computational resources, to extract and generate features. Figure 4.3.1 illustrates the adversarial model. A general user case scenario is considered, where the user enjoys interacting with different IoT smart devices connected to the Internet. These devices provide different smart services such as streaming news and music based on the users' voice commands. Users' voice command is detected and processed by the smart device and sent to the cloud service. The cloud service replies to the smart device with the answer in the form of an audio stream, or action, based on the desired service.

The adversary is assumed to be far away from both the device and the Wi-Fi access point, but he can still eavesdrop and collect the Wi-Fi traffic. In particular, this adversary model takes into account the stream of messages from the cloud to the device (i.e. the in-going flow). For each eavesdropped message, the adversary is only interested in recording the packet size and the time at which the packet is received.

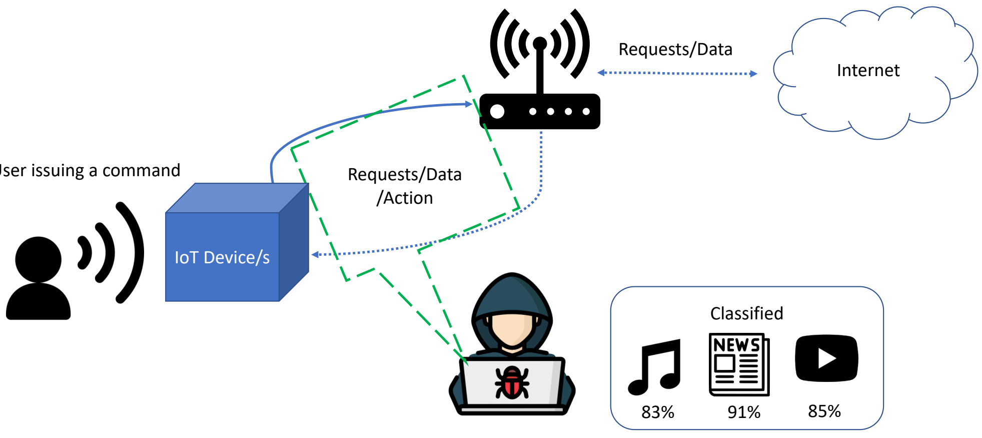


Figure 4.3.1: Adversary Model

The *adversary model* in Figure 4.3.1 is applied across the scenarios introduced in the following sections.

4.3.1 Features Extraction and Generation

Raw Features

The raw reception times and packet sizes are extracted from the traces. Inter-arrival times are calculated as $T_{x+1} - T_x$, where T_{x+1} is the time of the packet $x + 1$ that is received after packet x.

Statistical Features

Different statistical features have been adopted in the literature to uniquely identify patterns from encrypted traffic [30, 28, 38]. In this thesis, we consider 8 statistical features: standard deviation, sum, variance, maximum, minimum, mean, median, skewness, and kurtosis (see Section 2.1). These features are extracted by using a sliding window technique, where a window of N adjacent packets is considered for the computation of the aforementioned statistics.

4.4 Commands–Applications–Services

In this thesis, we focus mainly on the most widely used commands, applications, and services among regular users. Famous applications such as: *Spotify*, *YouTube* for streaming both music, videos. Additionally, we focus on *News* streaming by the *Bloomberg* skill on *Amazon* devices, since the majority of the users tend to have the news read by the smart IoT devices. The following is the services streamed by each of devices:

1. Amazon Echo: News and Music
2. Amazon Echo Dot: News and Music
3. Google Chromecast: YouTube and Music

4. Google Nest Mini: Music

4.5 Preliminary Data Collection and Initial Observations

4.5.1 Analysis Discussion

In this chapter, we introduced the adversarial model and described all the necessary steps for data collection. Traffic associated with each device and service were collected and analyzed to identify the possibilities of distinguishing a pattern. The two features analyzed are the interarrival times of the packets and the packet size for each device independently.

Interarrival Times Analysis. For each device and service, we extracted and analyzed the interarrival times as reported for each device running single service, and combined in Figure 4.5.17a which shows the interpolated inter-arrival times for the four devices and three services (except Google Nest Mini running one service, and Amazon Echo and Echo Dot running 2 different services). Although the samples have similar patterns, the interpolating polynomial functions have distinct patterns meaning that each service on each device can be potentially identified by just comparing the interarrival times. Table 4.5.1 lists all the interarrival times with the highest occurrence, associated with device and service. The values are obtained from the previously introduced interarrival times normalized histograms.

Device	**Service**	**Interarrival Time [Seconds]**	**Figure**
Amazon Echo	Music	2.15×10^{-7}	4.5.1b
	News	1.90×10^{-7}	4.5.9b
Amazon Echo Dot	Music	2.5×10^{-6}	4.5.3b
	News	1.6×10^{-7}	4.5.11b
Google Mini Nest	Music	1.13×10^{-5}	4.5.5b
Google Chromecast	Music	3.4×10^{-7}	4.5.7b
	YouTube	1.68×10^{-7}	4.5.13b

Table 4.5.1: *Interarrival Times* and their occurrences associated with devices and services

Packet Sizes Analysis. Packets sizes can be easily extracted and used directly for analysis (to identify the existence of a unique pattern) and classification. Figure 4.5.17b shows the packet size analysis as a function of the 7 classes considered in this paper. We observe that the frequency associated with the packet size is very different among the 7 considered categories, in particular when considering packet sizes larger than 800 bytes. Table 4.5.2 summarize the packet sizes with the highest occurrence in the traces associated with each device and service.

Device	**Service**	**Packet Size [Bytes]**	**Occurrences**	**Figure**	**Table**
Amazon Echo	Music	1571	85500	4.5.2	4.5.5
	News	1571	187280	4.5.10	4.5.9
Amazon Echo Dot	Music	1571	183828	4.5.4	4.5.6
	News	1402.1	129315	4.5.12	4.5.10
Google Mini Nest	Music	1571	249309	4.5.6	4.5.7
Google Chromecast	Music	1523	228935	4.5.8	4.5.8
	YouTube	1523	4771023	4.5.14	4.5.11

Table 4.5.2: *Packets Sizes* and their occurrences associated with devices and services

Noise

The collected noise consists of traffic generated by devices that are connected and not connected to the same network where services were streamed through. The trace consist of 2,130,318 packets, that represents different types of protocols including: **ACK**, **QoS**, **ICMP**, **DNS requests**, and other requests types. As illustrated in Figure 4.5.15b, Table 4.5.3 points out the most occurred interarival time within the noise traces. Similarly, the most occurred packet size represented in Figure 4.5.16 is addressed in Table 4.5.4.

Type	**Service**	**Interarrival Time [Seconds]**	**Figure**
Noise	Mixed	1.21×10^{-5}	4.5.15a

Table 4.5.3: Noise Most Occurred Interarrival Times

Type	**Packet Size [Bytes]**	**Occurrences**	**Figure**	**Table**
Noise	42	1364214	4.5.16	4.5.12

Table 4.5.4: Noise Most Occurred Packet Size

The remaining parts of this chapter provide the conducted analysis of the data collected associated to each use case.

4.5.2 Use case I: Streaming Music

In this case scenario, the user has two options to stream music using these devices through *Spotify*.

First, by vocal commands[2] requesting from the IoT device(s) to play a song, music list, or a custom mixed playlist generated by the device itself. The user initiates command such as *"Alexa, play music"* or *"Hey Google, play music"*.

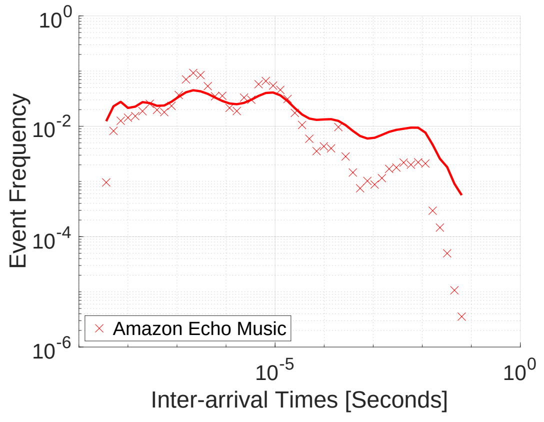
Second, the user can set the music to be played using the Spotify app through his devices such as phones or computers. For this use case, 4 traces were collected from each device, consisting of traffic generated from the access point to Device itself.

Each device traffic is represented by a trace that is approximately 8 hours of continuous music streaming.

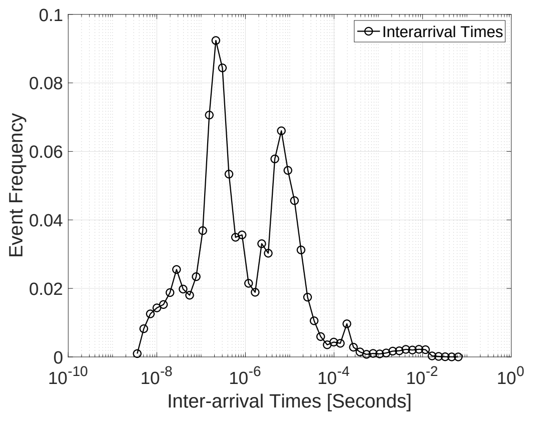
[2]All devices except Google Chromecast

Amazon Echo

Figure 4.5.1a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.1b shows the most occurred interarrival time 2.15×10^{-7}. For the packets sizes, Figure 4.5.2 and Table 4.5.5 shows the packets sizes and their occurrences. Packets with the size of 1571 bytes dominates the entire trace, with an amount of 249,309 packets of the total number of packets.



(a)



(b)

Figure 4.5.1: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

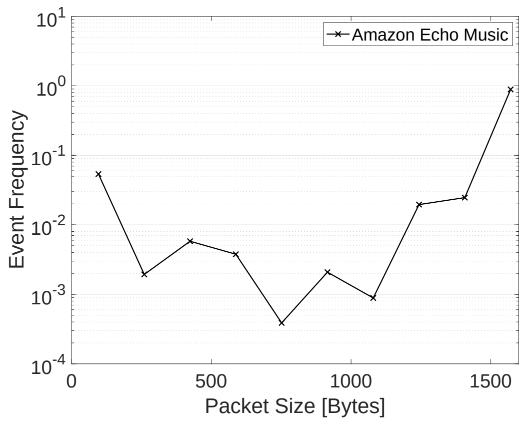


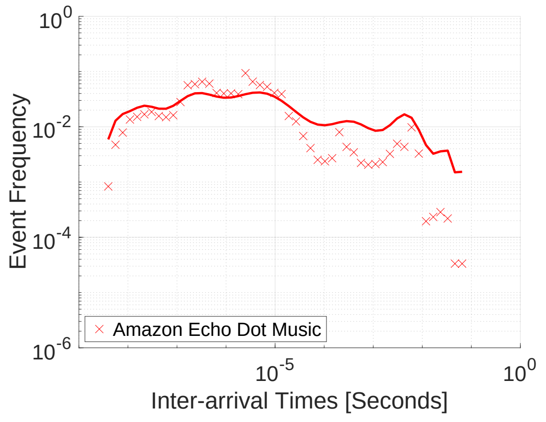
Figure 4.5.2: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	Frequency [Packets]
96	15100
259.89	543
423.78	1630
587.67	1060
751.56	109
915.44	585
1079.3	249
1243.2	5492
1407.1	6921
1571	249309

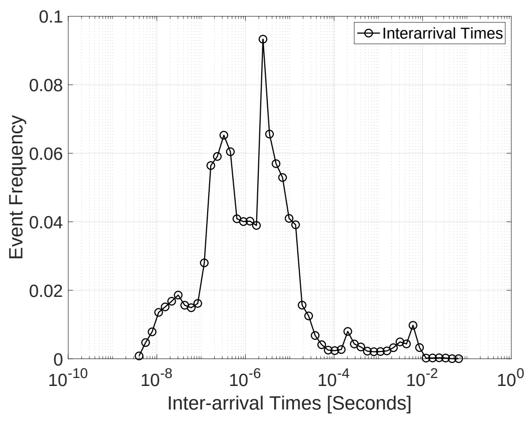
Table 4.5.5: *Packets sizes* and their frequencies

Amazon Echo Dot

Figure 4.5.3a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.3b shows the most occurred interarrival time 2.5×10^{-6}. For the packets sizes, Figure 4.5.4 and Table 4.5.6 shows the packets sizes and their occurrences. Packets with the size of 1571 bytes dominates the entire trace, with an amount of 187,280 packets of the total number of packets.



(a)



(b)

Figure 4.5.3: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

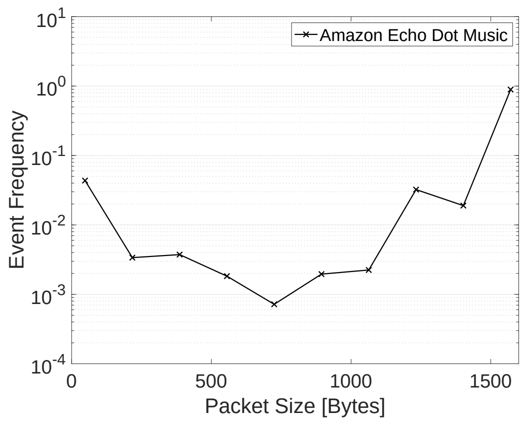


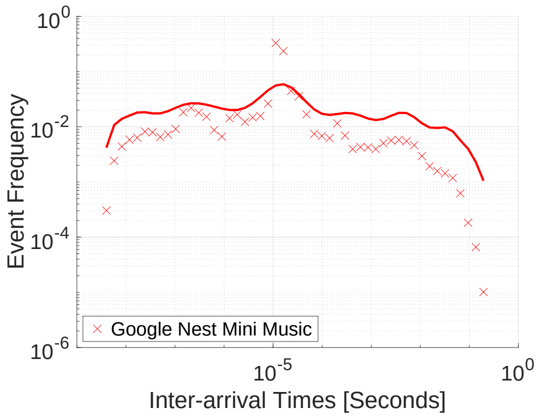
Figure 4.5.4: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	Frequency [Packets]
48	9143
217.22	710
386.44	785
555.67	384
724.89	151
894.11	411
1063.3	471
1232.6	6781
1401.8	3987
1571	187280

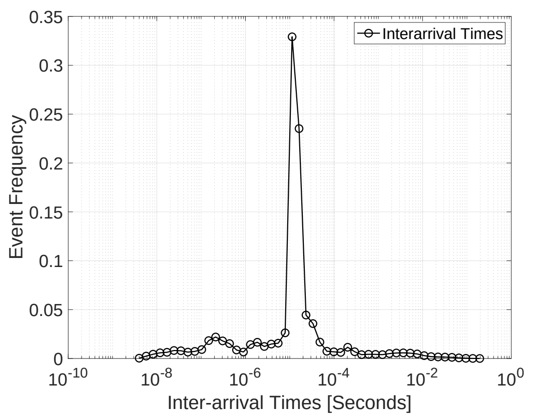
Table 4.5.6: *Packets sizes* and their frequencies

Google Nest Mini

Figure 4.5.5a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.5b shows the most occurred interarrival time 1.13×10^{-5}. For the packets sizes, Figure 4.5.6 and Table 4.5.7 shows the packets sizes and their occurrences. Packets with the size of 1571 bytes dominates the entire trace, with an amount of 187,280 packets of the total number of packets.



(a)



(b)

Figure 4.5.5: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

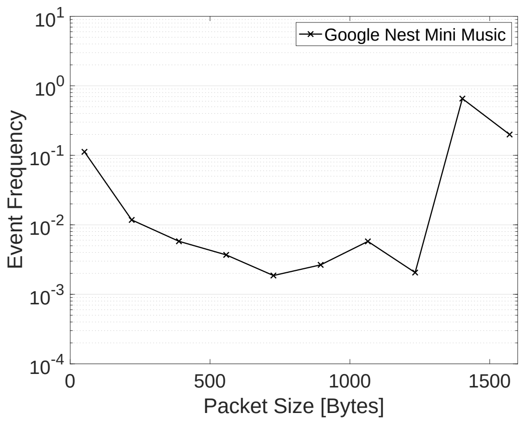


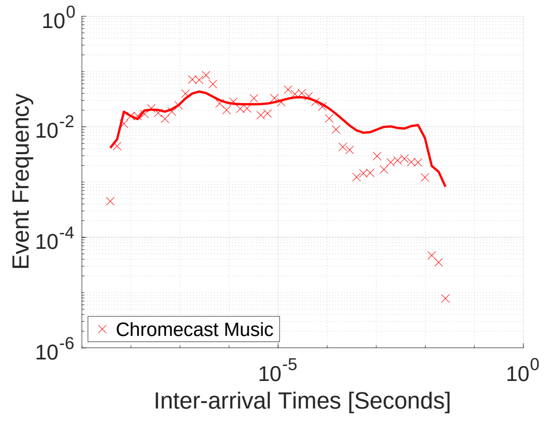
Figure 4.5.6: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	**Frequency [Packets]**
51	22166
219.89	2322
388.78	1144
557.67	729
726.56	368
895.44	524
1064.3	1140
1233.2	406
1402.1	129315
1571	39356

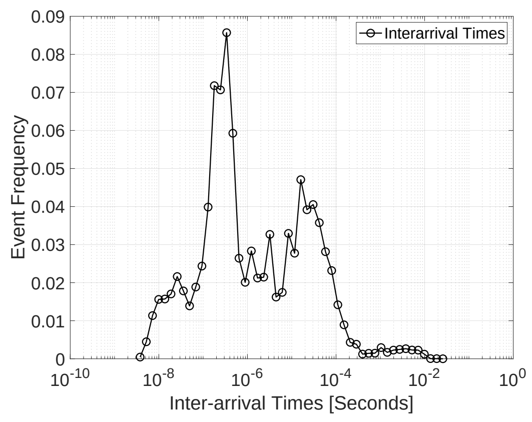
Table 4.5.7: *Packets sizes* and their frequencies

Google Chromecast

Figure 4.5.7a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.7b shows the most occurred interarrival time 3.4×10^{-7}. For the packets sizes, Figure 4.5.8 and Table 4.5.8 shows the packets sizes and their occurrences. Packets with the size of 1523 bytes dominates the entire trace, with an amount of 228,935 packets of the total number of packets.



(a)



(b)

Figure 4.5.7: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

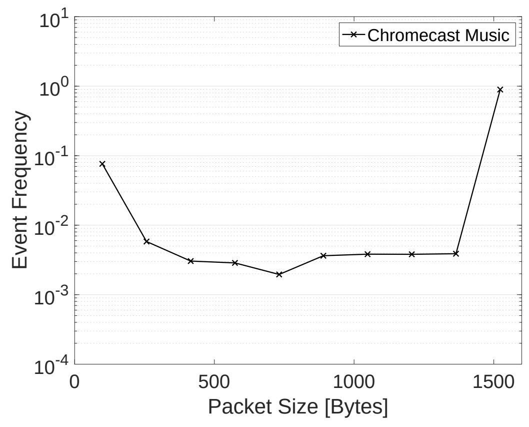


Figure 4.5.8: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	**Frequency [Packets]**
99	19523
257.22	1489
415.44	779
573.67	733
731.89	500
890.11	930
1048.3	979
1206.6	975
1364.8	994
1523	228935

Table 4.5.8: *Packets sizes* and their frequencies

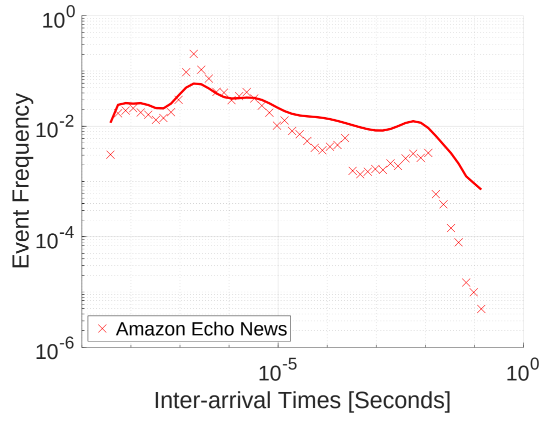
4.5.3 Use case II: Streaming News

In this use case scenario, the user has only *one* option to stream *News* using these devices.

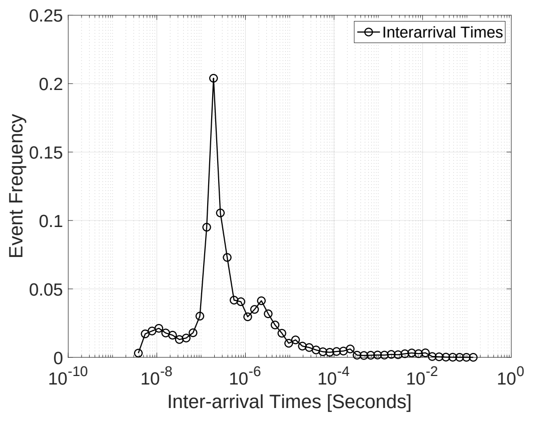
The user has to initiate a vocal command[3] requesting from the IoT device(s) to stream the news from the default source, or by specifying which source. The user initiates command such as *"Alexa, play the news"* or *"Alexa, play the news from CNN"*.

Amazon Echo

Figure 4.5.9a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.9b shows the most occurred interarrival time 1.9×10^{-7}. For the packets sizes, Figure 4.5.10 and Table 4.5.9 shows the packets sizes and their occurrences. Packets with the size of 1571 bytes dominates the entire trace, with an amount of 183,828 packets of the total number of packets.



(a)



(b)

Figure 4.5.9: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

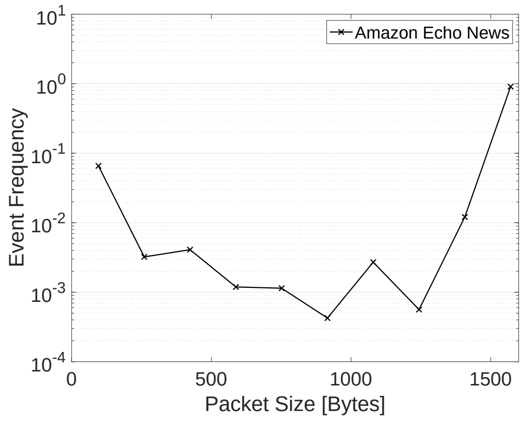


Figure 4.5.10: Log-scale Normalized Histogram associated with *packets sizes*

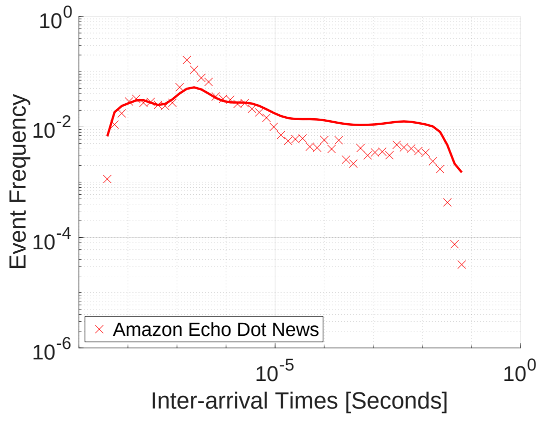
Size [Bytes]	Frequency [Packets]
96	13289
259.89	652
423.78	828
587.67	241
751.56	231
915.44	86
1079.3	547
1243.2	114
1407.1	2447
1571	183828

Table 4.5.9: *Packets sizes* and their frequencies

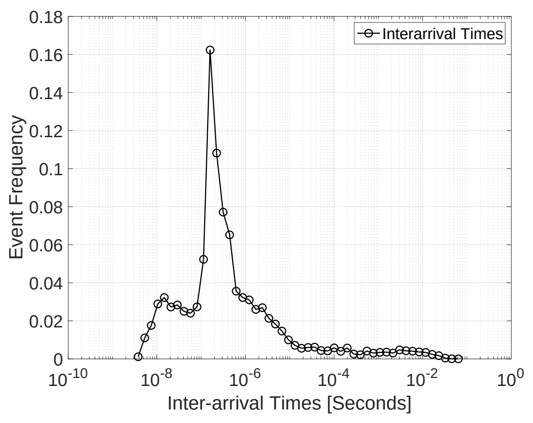
[3]All devices except Google Chromecast and Mini Nest

Amazon Echo Dot

Figure 4.5.11a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.11b shows the most occurred interarrival time 1.6×10^{-7}. For the packets sizes, Figure 4.5.12 and Table 4.5.10 shows the packets sizes and their occurrences. Packets with the size of 1571 bytes dominates the entire trace, with an amount of 183,828 packets of the total number of packets.



(a)



(b)

Figure 4.5.11: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

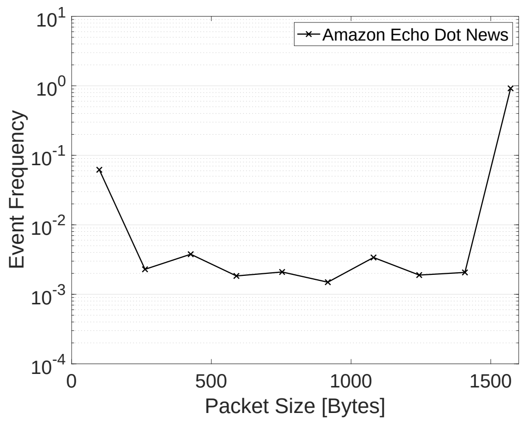


Figure 4.5.12: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	**Frequency [Packets]**
99	5761
262.56	213
426.11	351
589.67	171
753.22	195
916.78	139
1080.3	315
1243.9	176
1407.4	192
1571	85500

Table 4.5.10: *Packets sizes* and their frequencies

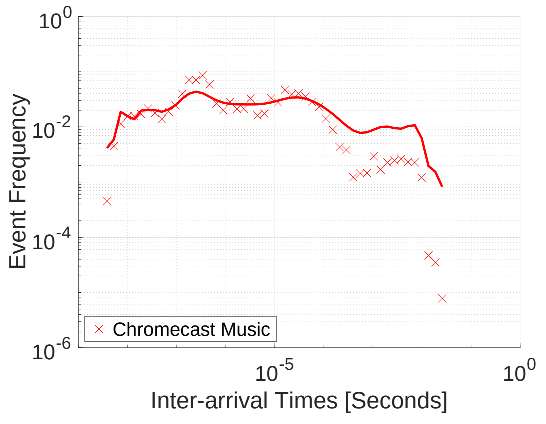
4.5.4 Use case III: Streaming YouTube

In this case scenario, the user has only one option to stream *YouTube*, which is done using Google Chromecast.

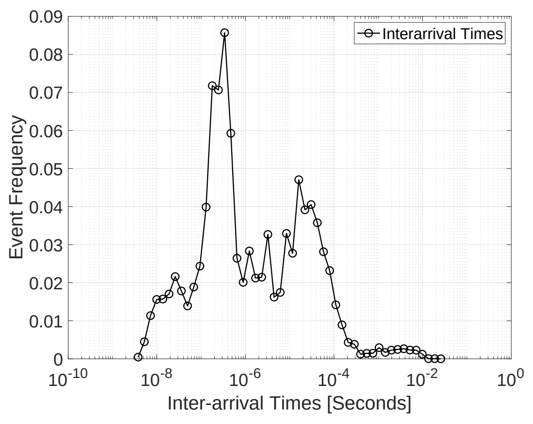
The user has to open the YouTube app on any of his devices (phone, tablet, laptop...etc) that is connected to the same network as the Chromecast. The user then has the option to ”cast” any video to Chromecast, and automatically start streaming.

Google Chromecast

Figure 4.5.13a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.13b shows the most occurred interarrival time 1.68×10^{-7}. For the packets sizes, Figure 4.5.14 and Table 4.5.11 shows the packets sizes and their occurrences. Packets with the size of 1571 bytes dominates the entire trace, with an amount of 4,771,023 packets of the total number of packets.



(a)



(b)

Figure 4.5.13: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the *interarrival times*

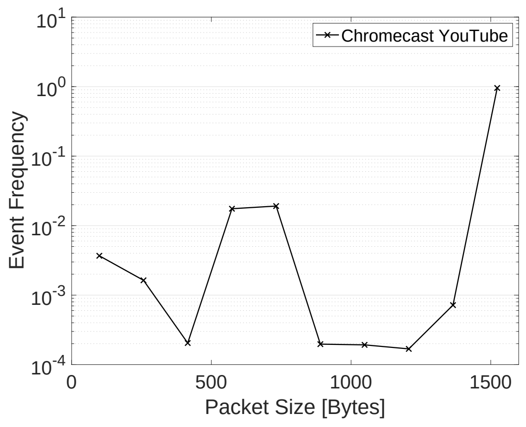


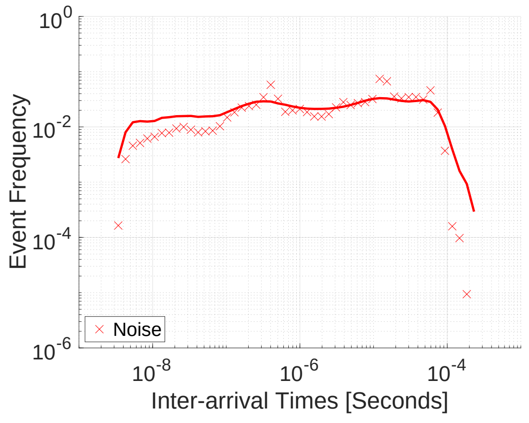
Figure 4.5.14: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	Frequency [Packets]
99	18362
257.22	8148
415.44	1020
573.67	87319
731.89	95358
890.11	983
1048.3	960
1206.6	840
1364.8	3581
1523	4771023

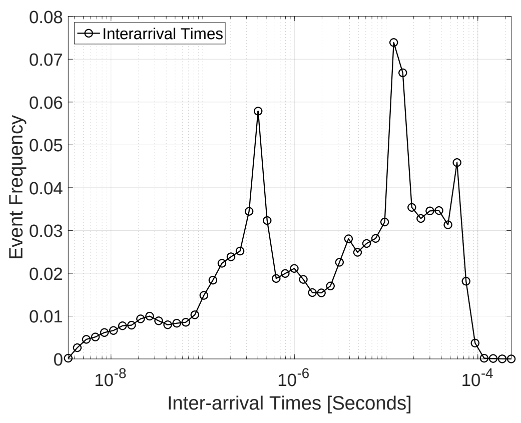
Table 4.5.11: *Packets sizes* and their frequencies

4.5.5 Noise

Figure 4.5.15a illustrate the distribution of the interarrival times and their occurrences. Figure 4.5.15b shows the most occurred interarrival time 1.21×10^{-5}. For the packets sizes, Figure 4.5.16 and Table 4.5.12 shows the packets sizes and their occurrences. Packets with the size of 42 bytes dominates the entire trace, with an amount of 1,364,214 packets of the total number of packets.



(a)



(b)

Figure 4.5.15: Probability distribution function (a), and Log-scale normalized histogram (b) associated with the noise *interarrival times*

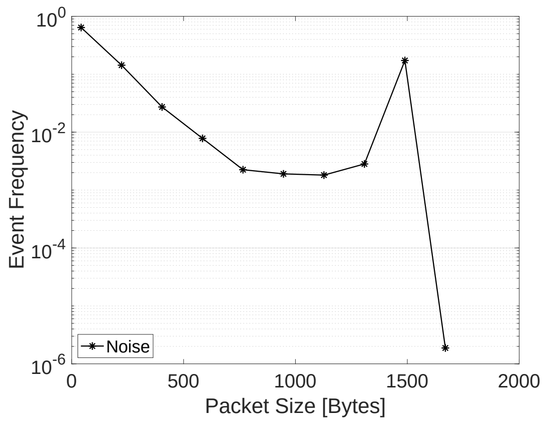


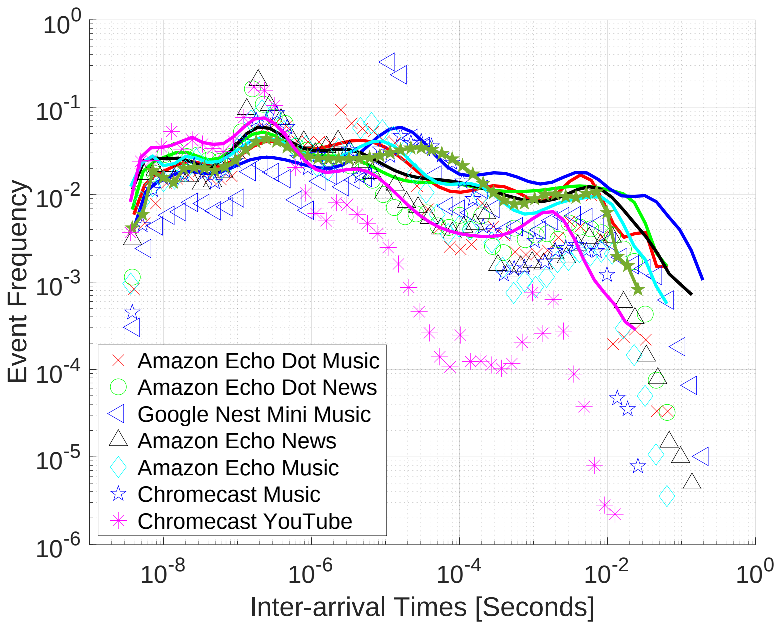
Figure 4.5.16: Log-scale Normalized Histogram associated with *packets sizes*

Size [Bytes]	Frequency [Packets]
42	1364214
222.89	305151
403.78	57955
584.67	16624
765.56	4786
946.44	4056
1127.3	3861
1308.2	6014
1489.1	367653
1670	4

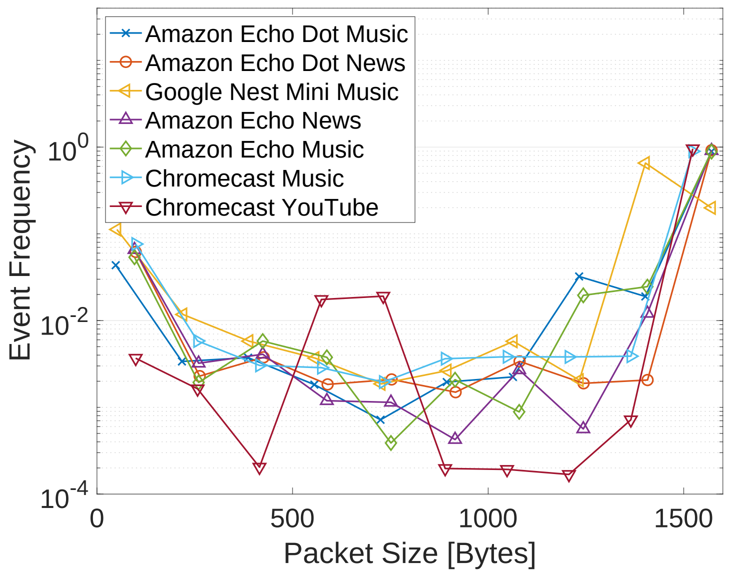
Table 4.5.12: *Packets sizes* and their frequencies

4.5.6 Traces Combined

Figure 4.5.17a illustrates the *interarrival times* probability distribution function for each service and device. Figure 4.5.17b also shows the normalized histogram of the *packets sizes* associated with each services and devices.



(a)



(b)

Figure 4.5.17: All Traces Combined

Chapter 5

Classification

In this chapter, we provide all the techniques for classifying and building the trained model from the data-sets collected.

Features Engineering. We introduce two different feature engineering techniques: *raw* and *statistical* for classifying services streamed on each device. The raw features consist of interarrival times and packets sizes, while the statistical features are those extracted from the interarrival times and packets sizes using statistics methods introduced in chapter 2. For each technique, the classification is done using the packets sizes, and interarrival times independently and combined.

Noise. For each of the feature engineering set-ups, the classification is done with and without the presence of noise. The reason for introducing noise is to see its effect on classification performance.

Classification. The vast majority of our results adopts the *Random Forest* algorithm for all our tests being the best performer from related works in the literature [33, 28, 30, 37, 38, 29]. Random Forest implementation [39] was adopted and modified to fit this thesis requirement. We consider the following configuration for the random forest algorithm: *k-folds* = 15, *number of trees/bags* = 30. All classes have been balanced (by taking the smallest trace size) to ensure that the number of samples is equal among all the traces.

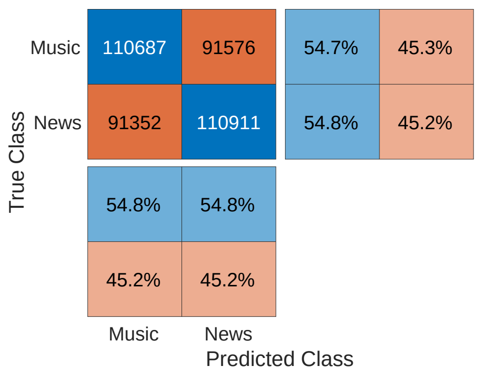
To summarize all the previous points related to feature engineering, the following set-ups are considered:

1. Raw Features:
    a. Classification using only packets sizes.
    b. Classification using interarrival times.
    c. Classification using packets sizes and interarrival times.
2. Statistical Features
    a. Classification using statistical features extracted from the packets sizes.
    b. Classification using statistical features extracted from the interarrival times.
    c. Classification using statistical features extracted from both the interarrival times and packets sizes.

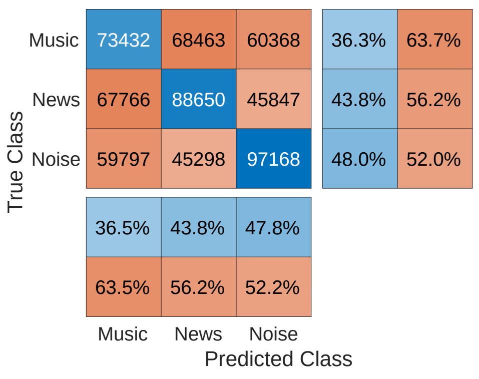
5.1 Amazon Echo

5.1.1 Interarrival Times

When classifying the *interarrival times* of the packets arriving at Amazon Echo, with and without the presence of noise (Figure 5.1.1a, 5.1.1b), the total number of samples per class is 202,263 samples in both scenarios. The accuracy achieved when classifying the two classes/services is 54.8%. While with the presence of noise, we achieve an accuracy of 42.7%.



(a)

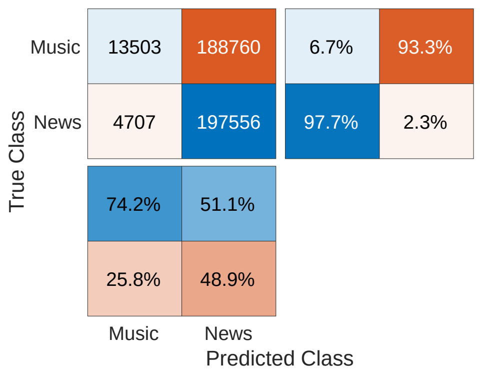


(b)

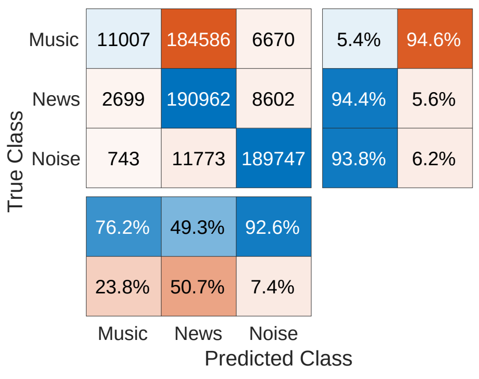
Figure 5.1.1: Confusion Matrix with only *interarrival times* for (a) using two classes, and (b) three classes classification.

5.1.2 Packets Sizes

For classifying the *packets sizes*, a similar classification approach (as in previous section) is adopted. The results obtained is as following: without noise (Figure 5.1.2a), the accuracy is 52.2%. With noise presence (Figure 5.1.2b), the accuracy achieved is 64.6%.



(a)

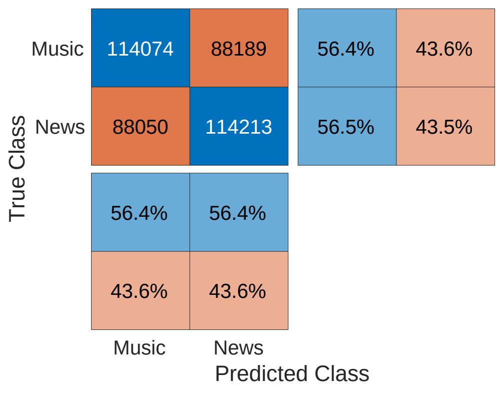


(b)

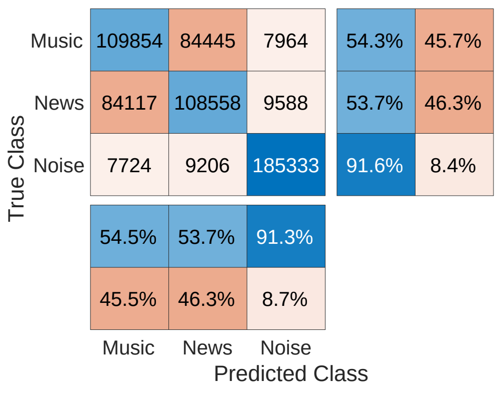
Figure 5.1.2: Confusion Matrix with only *packets sizes* for (a) using two classes, and (b) three classes classification.

5.1.3 Interarrival Times and Packets Sizes

Theoretically, when combining both features the classification should result in a better accuracy. Without the presence of noise (Figure 5.1.3a), the accuracy achieved is 56.4%. With noise being present (Figure 5.1.3b) the model accuracy is at 66.5%.



(a)

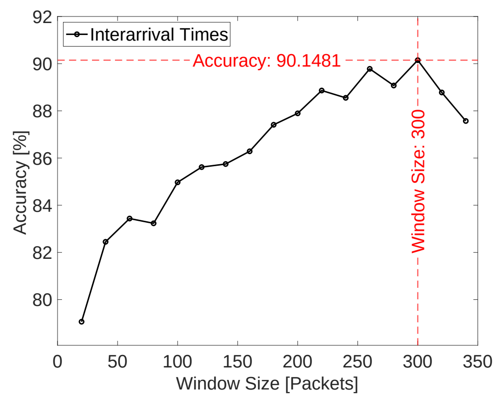


(b)

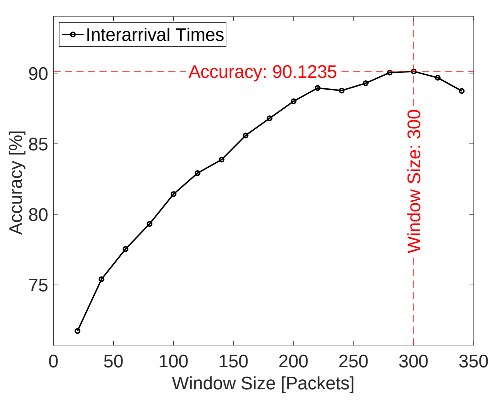
Figure 5.1.3: Confusion Matrix with both *interarrival times* and *packets sizes* for (a) using two classes, and (b) three classes classification.

5.1.4 Interarrival Times Statistical Features

The 8 statistical features are extracted from the *interarrival times* with different window sizes, starting from 20 to 340 packets. Figures 5.1.4a, 5.1.4b shows the models accuracies, with and without the presence of noise. In both scenarios, the best achieved accuracy $> 90\%$, when the window size is set to 300 packets.



(a)

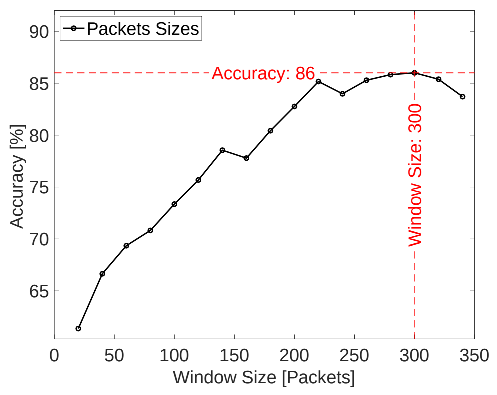


(b)

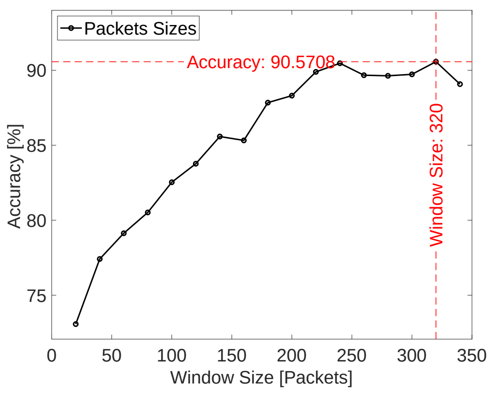
Figure 5.1.4: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *interarrival times* statistical features.

5.1.5 Packets Sizes Statistical Features

Once the 8 statistical features extracted from the *packets sizes* and classified, the accuracy obtained without considering the noise is 86% at window size of 300 packets. While with the presence of noise, the accuracy achieved is approximated to be 90.6%, at window size of 320 packets.



(a)

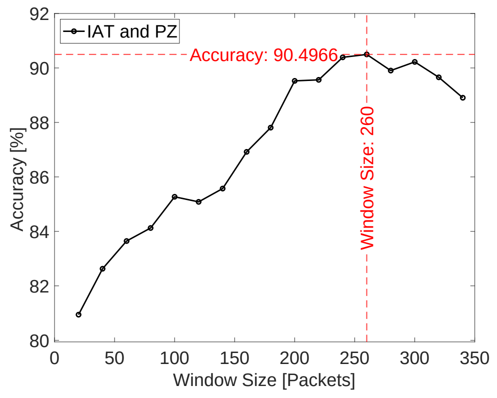


(b)

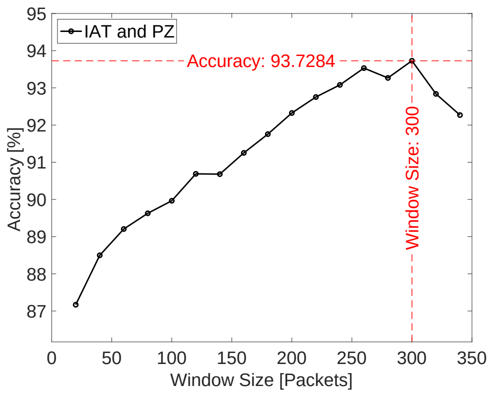
Figure 5.1.5: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *packets sizes* statistical features.

5.1.6 Interarrival Times and Packets Sizes Statistical Features

Similarly to the combined *raw* features, we combined 16 statistical features extracted from both *packets sizes*, and *interarrival times*. The accuracy obtained in a noiseless environment is 90% at a window size of 260. While in a noise environment, the accuracy achieved is 93.7% with a window size of 300 packets.



(a)



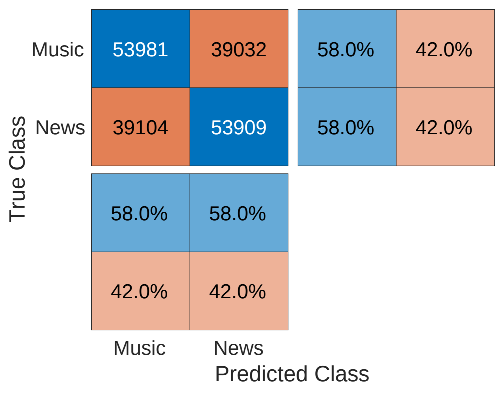
(b)

Figure 5.1.6: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing both *interarrival times* and *packets sizes* statistical features.

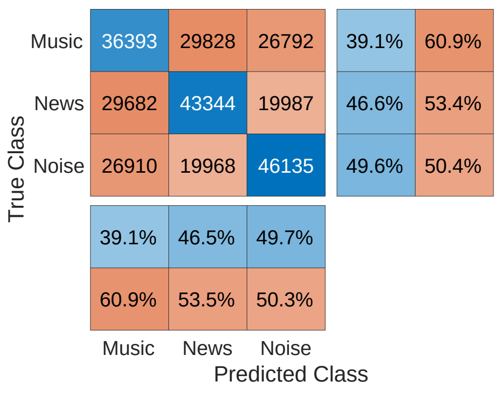
5.2 Amazon Echo Dot

5.2.1 Interarrival Times

The total numbers of samples per class is 93,013 when classifying 2 and 3 classes. When classifying the *interarrival times* of the two classes (music and news, Figure 5.2.1a), the obtained accuracy is 57.9%. With the presence of noise (Figure 5.2.1b), the accuracy achieved is 45.1%.



(a)

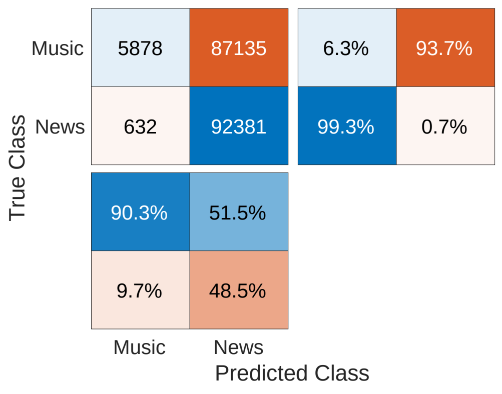


(b)

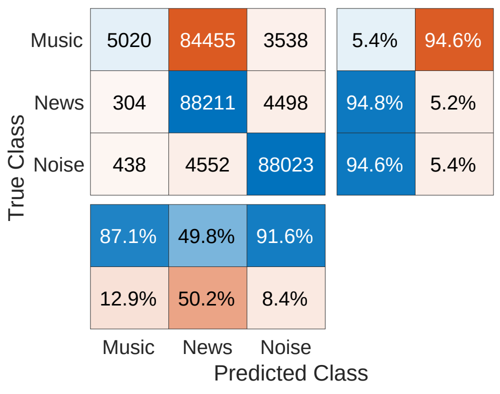
Figure 5.2.1: Confusion Matrix with only *interarrival times* for (a) using two classes, and (b) three classes classification.

5.2.2 Packets Sizes

The total numbers of samples per class remains the same (93,013) as previously mentioned. The classification accuracy achieved when using the *packets sizes*, without the noise being present (Figure 5.2.2a) is 52.8%. While in the noisy environment, the accuracy is 64.9%.



(a)

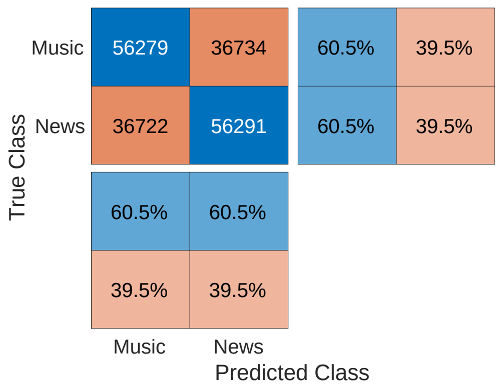


(b)

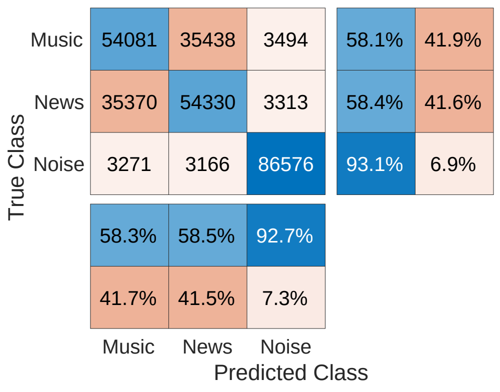
Figure 5.2.2: Confusion Matrix with only *packets sizes* for (a) using two classes, and (b) three classes classification.

5.2.3 Interarrival Times and Packet Sizes

The accuracy achieved when combining both *raw* features without the noise presence is 60.5%. With the noise being present, the total accuracy is 69.9%.



(a)

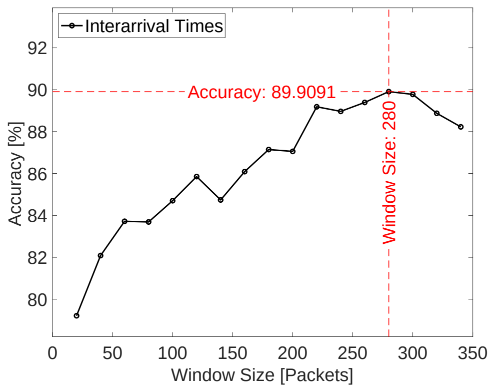


(b)

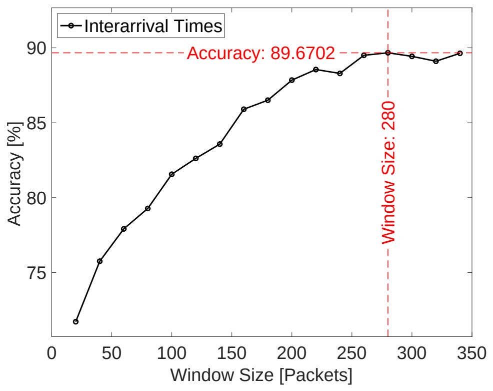
Figure 5.2.3: Confusion Matrix with *interarrival times* and *packets sizes* for (a) using two classes, and (b) three classes classification.

5.2.4 Interarrival Times Statistical Features

Once the 8 statistical features extracted from the *interarrival times* using different window sizes (from 20 to 340), the results obtained with and without the presence of noise is 89.7% and 89.9% respectively. Figures 5.2.4a, 5.2.4b shows that both models achieve a very similar accuracy (> 89%) with the same window size set at 280 packets.



(a)

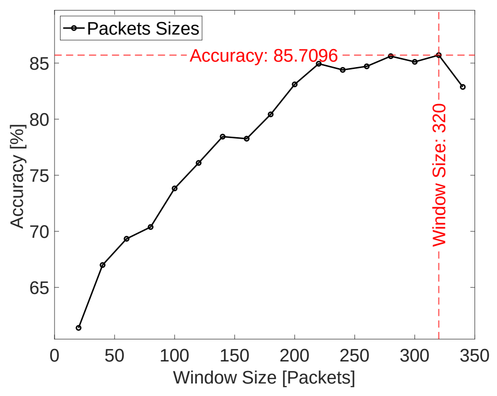


(b)

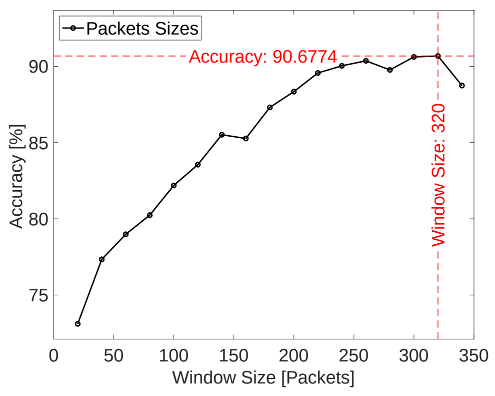
Figure 5.2.4: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *interarrival times* statistical features.

5.2.5 Packets Sizes Statistical Features

The accuracy achieved when using the 8 statistical features extracted from the *packets sizes* without the presence of noise (Figure 5.2.5a) is 85.7%, with a window size of 320 packets. In the noisy environment (Figure 5.2.5b), the classification accuracy achieved is 90.7% when the window size is also set to 320 packets.



(a)

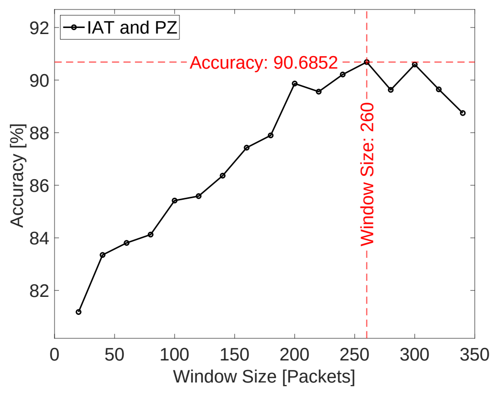


(b)

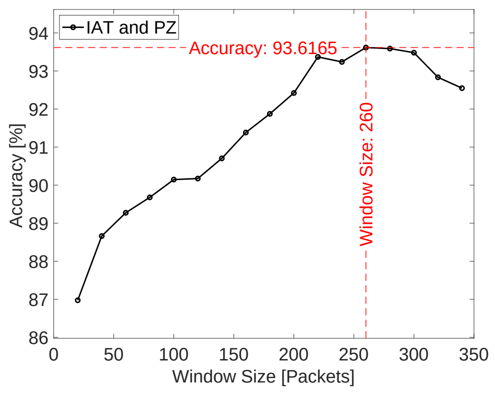
Figure 5.2.5: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *packets sizes* statistical features.

5.2.6 Interarrival Times and Packets Sizes Statistical Features

Combining all 16 statistical features results in accuracy of 90.7% at a window size set to 260 packets, without the presence of noise (Figure 5.2.6a). With the presence of noise, the accuracy is 93.6% at window size of 260 packets (Figure 5.2.6b).



(a)



(b)

Figure 5.2.6: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing both *interarrival times* and *packets sizes* statistical features.

5.3 Google Nest Mini

On this smart IoT device, we only streamed music since it does not stream other services (such as news) for a longer period as the other devices. Thus, we test the trace against the noise.

5.3.1 Interarrival Times

The total number of samples per class is 197,470. The accuracy achieved with the presence of noise, using only *interarrival times* (Figure 5.3.1) is 64.9%.

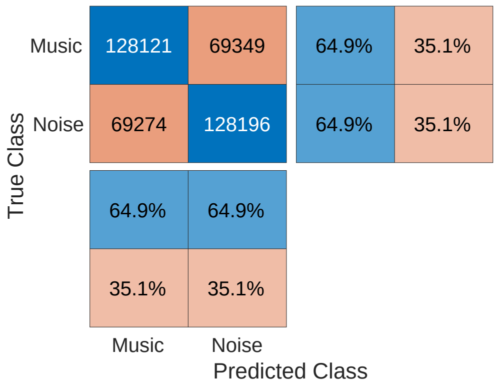


Figure 5.3.1: Confusion Matrix with only *interarrival times* using two classes.

5.3.2 Packets Sizes

The accuracy achieved with the presence of noise, using only *packets sizes* (Figure 5.3.2) is 91.9%.

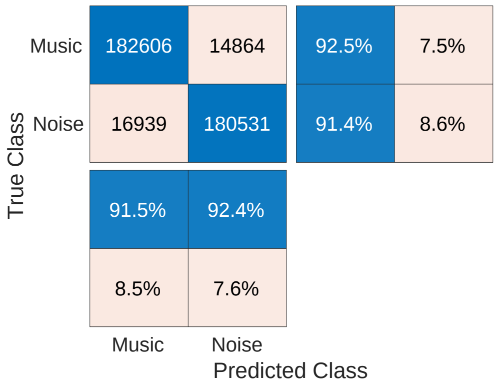


Figure 5.3.2: Confusion Matrix with only *packets sizes* using two classes.

5.3.3 Interarrival Times and Packets Sizes

The accuracy achieved with the presence of noise, using both features (Figure 5.3.2) is 91.8%.

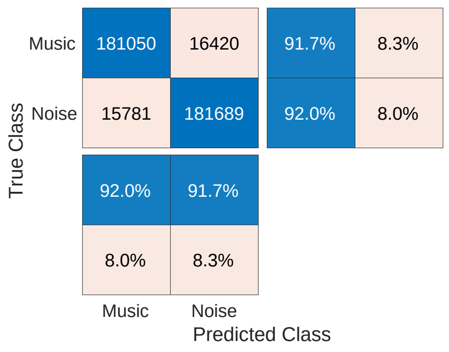


Figure 5.3.3: Confusion Matrix with both *interarrival times* and *packets sizes* combined, using two classes.

5.3.4 Statistical Features

Classification using the extracted statistical features from *interarrival times*, results in model with accuracy of 99.9% when window size is set to 320 packets (Figure 5.3.4a). On the contrary, *packets sizes* the model built achieved similar accuracy (99.9%) but with a window size of 240 packets (Figure 5.3.4b). When combining both statistical features, the model accuracy is 100% at window size of 140 packets (Figure 5.3.4c).

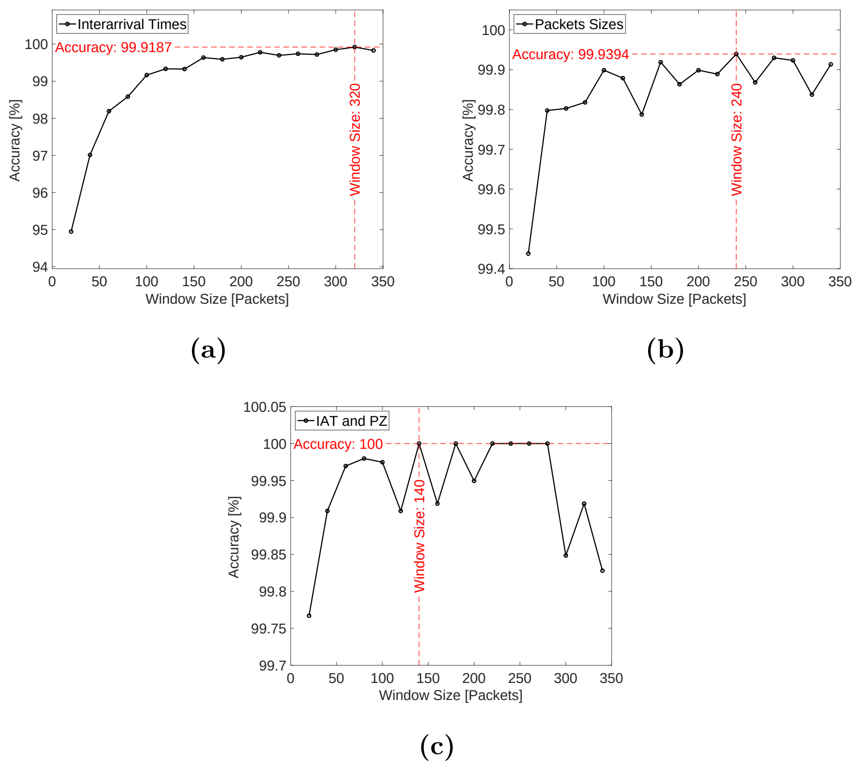
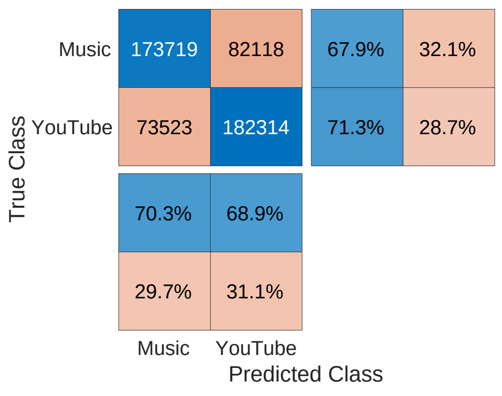


Figure 5.3.4: Services classifications accuracies associated with window sizes achieved when utilizing (a) *interarrival times*, (b) *packets sizes*, and (c) both combined with the presence of noise.

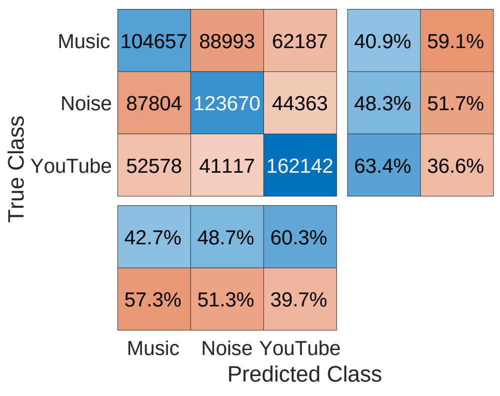
5.4 Google Chromecast

5.4.1 Interarrival Times

The total numbers of samples per class is 255,837 in both scenarios. When classifying the *interarrival times* of the two classes (music and YouTube), the obtained accuracy is 69.6% (Figure 5.4.1a). With the presence of noise, the accuracy achieved is 50.9% (Figure 5.4.1b).



(a)

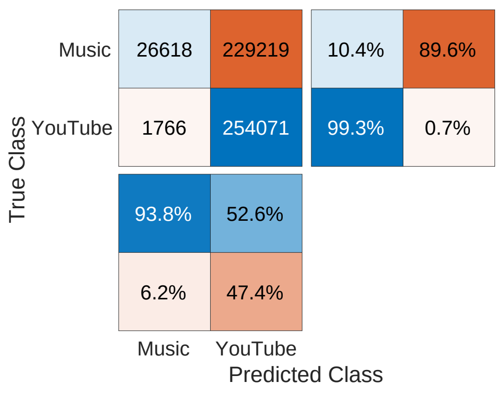


(b)

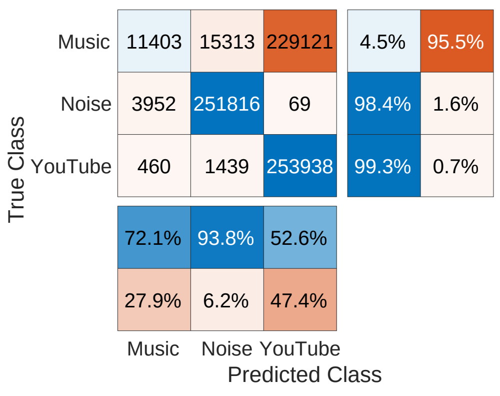
Figure 5.4.1: Confusion Matrix with only *interarrival times* for (a) using two classes, and (b) three classes classification.

5.4.2 Packets Sizes

Classifying the *packets sizes* results in an accuracy of 54.9% (Figure 5.4.2a). With the presence of noise, the accuracy achieved of the model is 67.4% (Figure 5.4.2b).



(a)

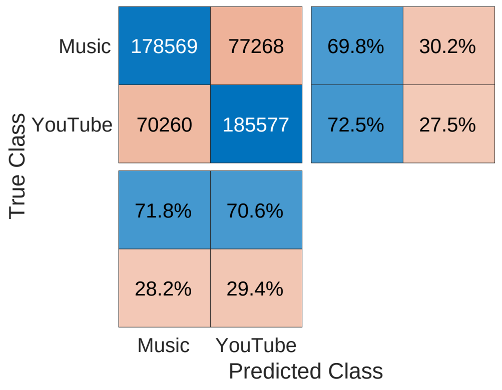


(b)

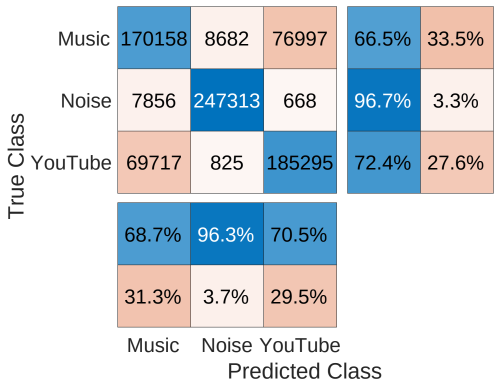
Figure 5.4.2: Confusion Matrix with only *packets sizes* for (a) using two classes, and (b) three classes classification.

5.4.3 Interarrival Times and Packets Sizes

Combining both *raw* features, the accuracy achieved without noise presence is 71.2% (Figure 5.4.3a). With noise being present, the total accuracy achieved is 78.5% (Figure 5.4.3b).



(a)

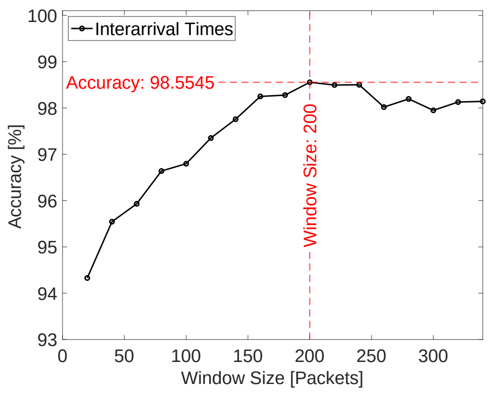


(b)

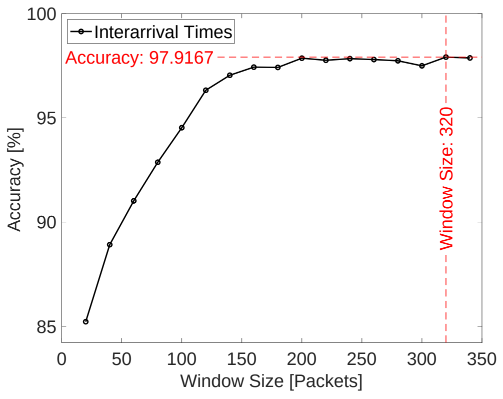
Figure 5.4.3: Confusion Matrix with both *interarrival times* and *packets sizes*, for (a) using two classes and (b) three classes classification.

5.4.4 Interarrival Times Statistical Features

Classifying using the statistical features extracted from the *interarrival times* without the presence of noise, results in an accuracy of 98.5% with window size of 200 packets (Figure 5.4.4b). In the noisy scenario, the accuracy is 97.9% at window size of 320 packets.



(a)

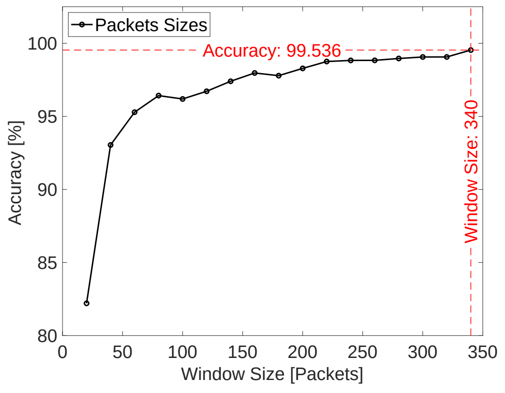


(b)

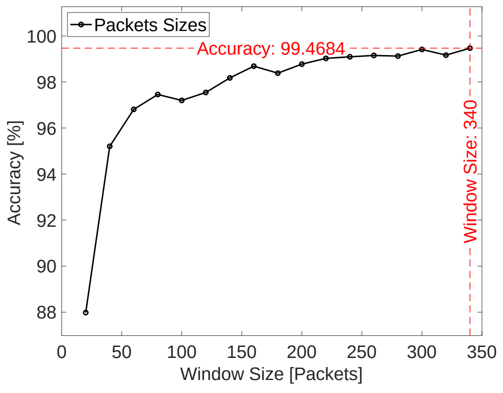
Figure 5.4.4: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *interarrival times* statistical features.

5.4.5 Packets Sizes Statistical Features

Using statistical features extracted from the *packets sizes*, the accuracy achieved is 99.5% at a window size of 340 packets, without the presence of noise (Figure 5.4.5a). With noise, the accuracy and window size are also the same as without the noise (Figure 5.4.5b).



(a)

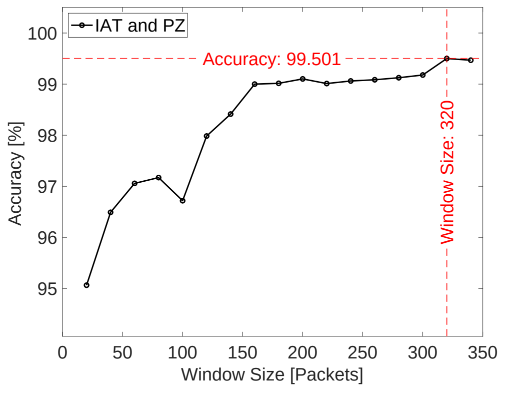


(b)

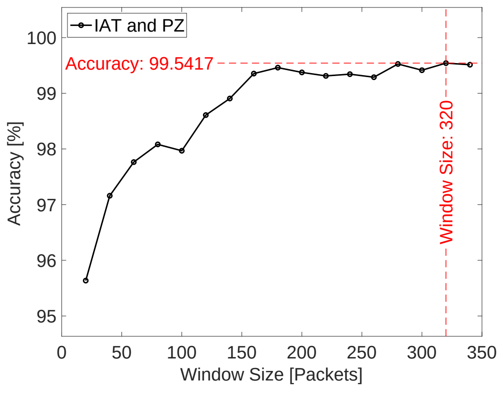
Figure 5.4.5: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *packets sizes* statistical features.

5.4.6 Interarrival Times and Packets Sizes Statistical Features

Combining both statistical features extracted from the *raw* feature, the accuracy achieved is 99.5% at a window size of 320 packets, without the presence of noise (Figure 5.4.6a). With noise, the accuracy and window size are also the same as without the noise (Figure 5.4.6b).



(a)



(b)

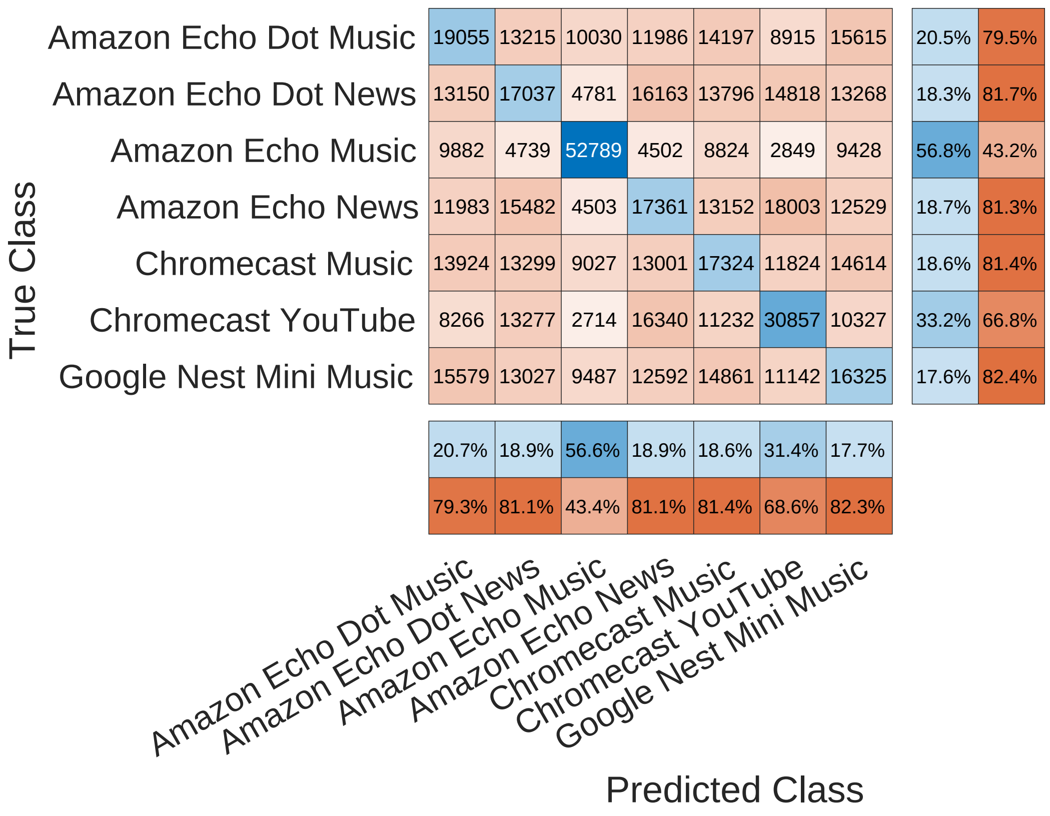
Figure 5.4.6: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing both *interarrival times* and *packets sizes* statistical features.

5.5 All Devices and Services

In the following classification experiments, we combined all the data-sets and classified each *raw* and *statistical* features individually, with and without the presence of noise. The total number of samples per class in each experiment is 93013.

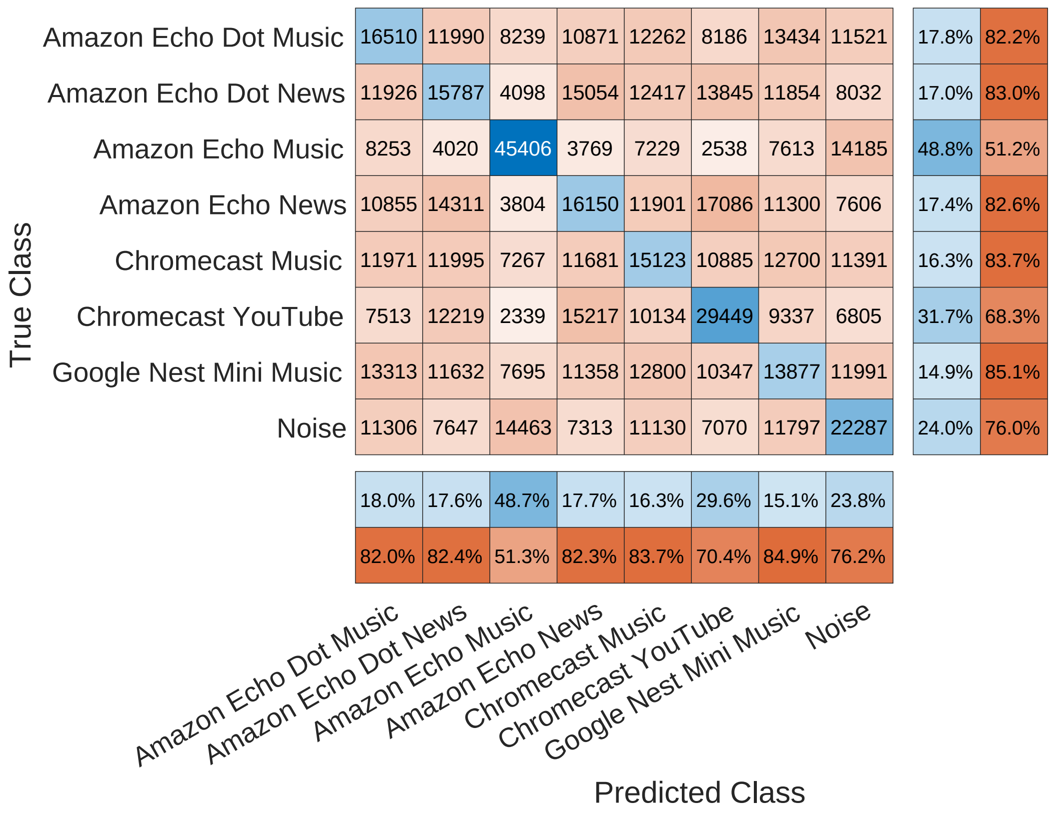
5.5.1 Interarrival Times

Starting with the *interarrival times*, the classification accuracy is 23.5% and 26.2%, with and without the presence of noise respectively (Figure 5.5.1, and 5.5.2).



True Class \ Predicted Class	Amazon Echo Dot Music	Amazon Echo Dot News	Amazon Echo Music	Amazon Echo News	Chromecast Music	Chromecast YouTube	Google Nest Mini Music		
Amazon Echo Dot Music	19055	13215	10030	11986	14197	8915	15615	20.5%	79.5%
Amazon Echo Dot News	13150	17037	4781	16163	13796	14818	13268	18.3%	81.7%
Amazon Echo Music	9882	4739	52789	4502	8824	2849	9428	56.8%	43.2%
Amazon Echo News	11983	15482	4503	17361	13152	18003	12529	18.7%	81.3%
Chromecast Music	13924	13299	9027	13001	17324	11824	14614	18.6%	81.4%
Chromecast YouTube	8266	13277	2714	16340	11232	30857	10327	33.2%	66.8%
Google Nest Mini Music	15579	13027	9487	12592	14861	11142	16325	17.6%	82.4%
	20.7%	18.9%	56.6%	18.9%	18.6%	31.4%	17.7%		
	79.3%	81.1%	43.4%	81.1%	81.4%	68.6%	82.3%		

Figure 5.5.1: Confusion Matrix with only *interarrival times* for all possible classes without noise



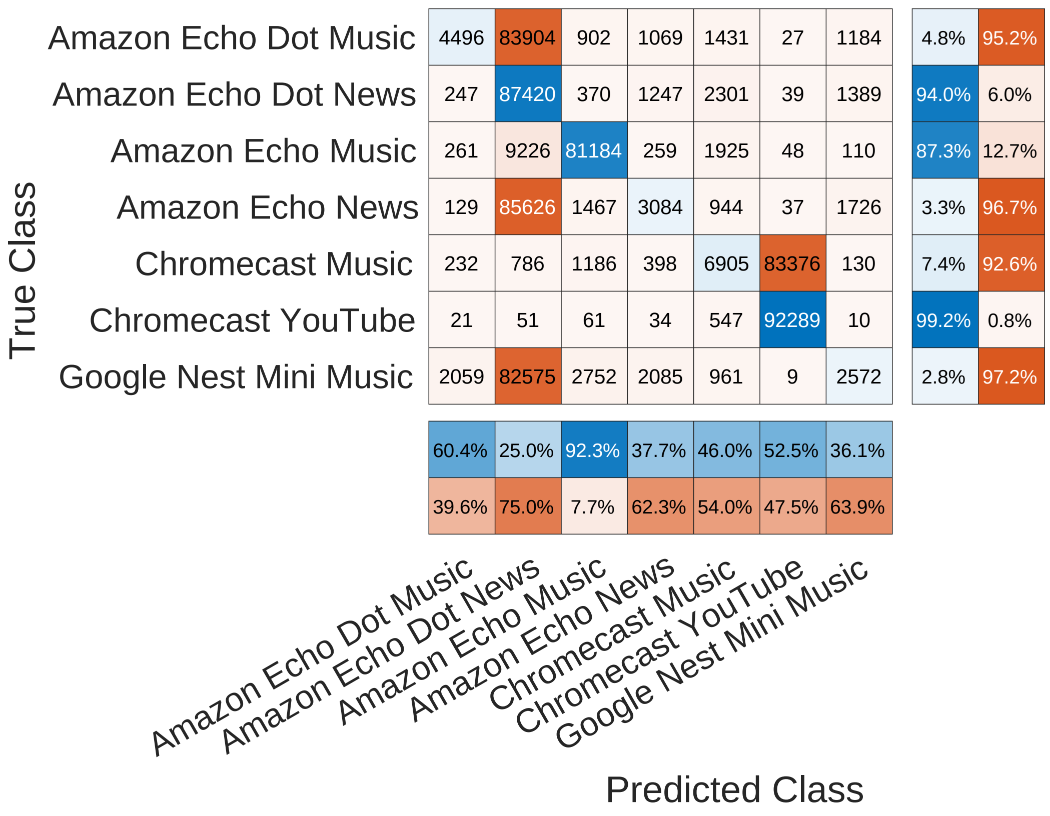
True Class \ Predicted Class	Amazon Echo Dot Music	Amazon Echo Dot News	Amazon Echo Music	Amazon Echo News	Chromecast Music	Chromecast YouTube	Google Nest Mini Music	Noise		
Amazon Echo Dot Music	16510	11990	8239	10871	12262	8186	13434	11521	17.8%	82.2%
Amazon Echo Dot News	11926	15787	4098	15054	12417	13845	11854	8032	17.0%	83.0%
Amazon Echo Music	8253	4020	45406	3769	7229	2538	7613	14185	48.8%	51.2%
Amazon Echo News	10855	14311	3804	16150	11901	17086	11300	7606	17.4%	82.6%
Chromecast Music	11971	11995	7267	11681	15123	10885	12700	11391	16.3%	83.7%
Chromecast YouTube	7513	12219	2339	15217	10134	29449	9337	6805	31.7%	68.3%
Google Nest Mini Music	13313	11632	7695	11358	12800	10347	13877	11991	14.9%	85.1%
Noise	11306	7647	14463	7313	11130	7070	11797	22287	24.0%	76.0%
	18.0%	17.6%	48.7%	17.7%	16.3%	29.6%	15.1%	23.8%		
	82.0%	82.4%	51.3%	82.3%	83.7%	70.4%	84.9%	76.2%		

Figure 5.5.2: Confusion Matrix with only *interarrival times* for all possible classes with noise

Based on these 2 confusion matrices, we can conclude that classifying using only the *interarrival times* results in a model with very poor accuracy.

5.5.2 Packets Sizes

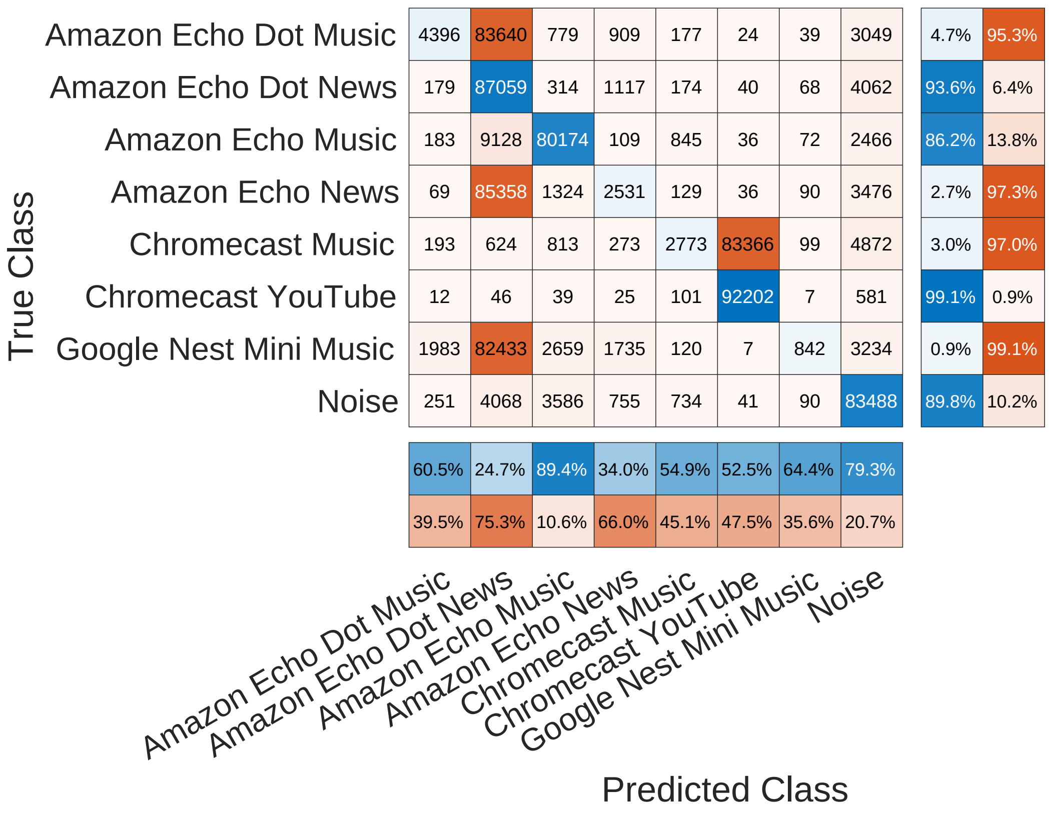
The *packet sizes* are classified, and the accuracy of the model, without the presence of noise is 54.9% (Figure 5.5.3).



True Class \ Predicted Class	Amazon Echo Dot Music	Amazon Echo Dot News	Amazon Echo Music	Amazon Echo News	Chromecast Music	Chromecast YouTube	Google Nest Mini Music		
Amazon Echo Dot Music	4496	83904	902	1069	1431	27	1184	4.8%	95.2%
Amazon Echo Dot News	247	87420	370	1247	2301	39	1389	94.0%	6.0%
Amazon Echo Music	261	9226	81184	259	1925	48	110	87.3%	12.7%
Amazon Echo News	129	85626	1467	3084	944	37	1726	3.3%	96.7%
Chromecast Music	232	786	1186	398	6905	83376	130	7.4%	92.6%
Chromecast YouTube	21	51	61	34	547	92289	10	99.2%	0.8%
Google Nest Mini Music	2059	82575	2752	2085	961	9	2572	2.8%	97.2%
	60.4%	25.0%	92.3%	37.7%	46.0%	52.5%	36.1%		
	39.6%	75.0%	7.7%	62.3%	54.0%	47.5%	63.9%		

Figure 5.5.3: Confusion Matrix with only *packet sizes* for all possible classes without noise

While the presence of noise results in a classification accuracy of 47.5% (Figure 5.5.4).



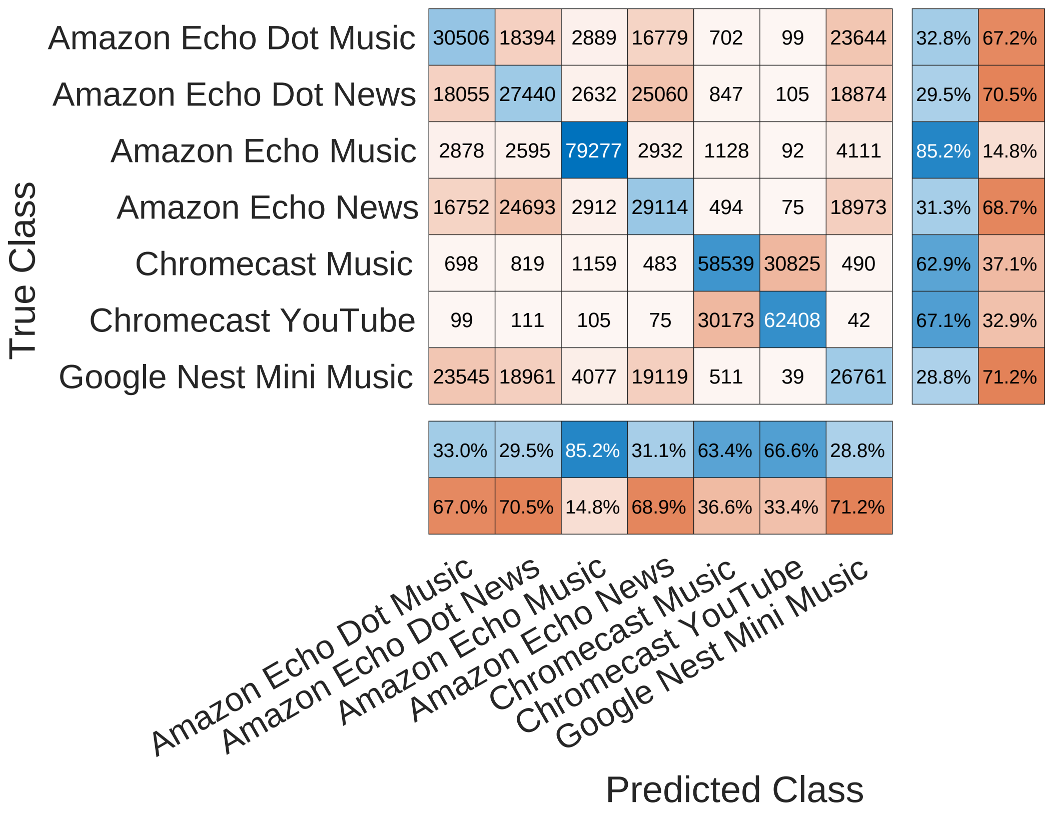
True Class \ Predicted Class	Amazon Echo Dot Music	Amazon Echo Dot News	Amazon Echo Music	Amazon Echo News	Chromecast Music	Chromecast YouTube	Google Nest Mini Music	Noise		
Amazon Echo Dot Music	4396	83640	779	909	177	24	39	3049	4.7%	95.3%
Amazon Echo Dot News	179	87059	314	1117	174	40	68	4062	93.6%	6.4%
Amazon Echo Music	183	9128	80174	109	845	36	72	2466	86.2%	13.8%
Amazon Echo News	69	85358	1324	2531	129	36	90	3476	2.7%	97.3%
Chromecast Music	193	624	813	273	2773	83366	99	4872	3.0%	97.0%
Chromecast YouTube	12	46	39	25	101	92202	7	581	99.1%	0.9%
Google Nest Mini Music	1983	82433	2659	1735	120	7	842	3234	0.9%	99.1%
Noise	251	4068	3586	755	734	41	90	83488	89.8%	10.2%
	60.5%	24.7%	89.4%	34.0%	54.9%	52.5%	64.4%	79.3%		
	39.5%	75.3%	10.6%	66.0%	45.1%	47.5%	35.6%	20.7%		

Figure 5.5.4: Confusion Matrix with only *packet sizes* for all possible classes with noise

Overall, the accuracy achieved when using the *packet sizes* is considered 50/50 to identify the service and device.

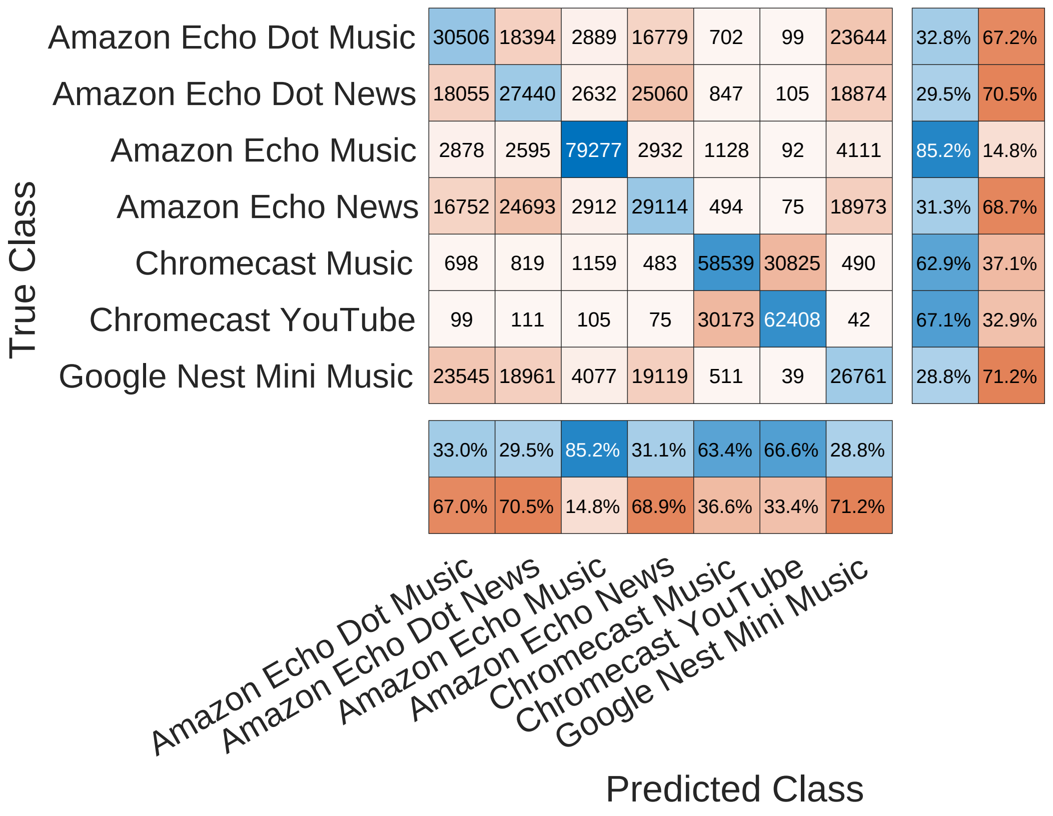
5.5.3 Interarrival Times and Packets Sizes

Combining both features results in accuracy of 48.2% without the involvement of noise (Figure 5.5.5). With the noise, the model accuracy is 52.2% (Figure 5.5.6).



	Amazon Echo Dot Music	Amazon Echo Dot News	Amazon Echo Music	Amazon Echo News	Chromecast Music	Chromecast YouTube	Google Nest Mini Music		
Amazon Echo Dot Music	30506	18394	2889	16779	702	99	23644	32.8%	67.2%
Amazon Echo Dot News	18055	27440	2632	25060	847	105	18874	29.5%	70.5%
Amazon Echo Music	2878	2595	79277	2932	1128	92	4111	85.2%	14.8%
Amazon Echo News	16752	24693	2912	29114	494	75	18973	31.3%	68.7%
Chromecast Music	698	819	1159	483	58539	30825	490	62.9%	37.1%
Chromecast YouTube	99	111	105	75	30173	62408	42	67.1%	32.9%
Google Nest Mini Music	23545	18961	4077	19119	511	39	26761	28.8%	71.2%
	33.0%	29.5%	85.2%	31.1%	63.4%	66.6%	28.8%		
	67.0%	70.5%	14.8%	68.9%	36.6%	33.4%	71.2%		

Figure 5.5.5: Confusion Matrix with both *interarrival times* and *packet sizes* for all possible classes without noise



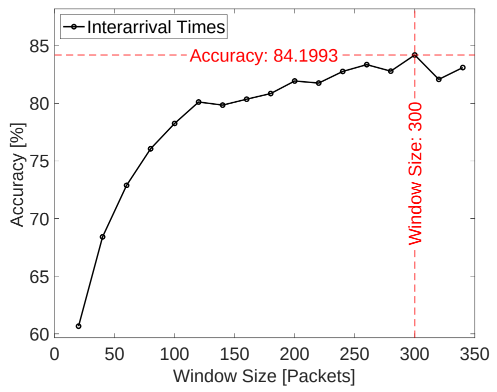
True Class \ Predicted Class	Amazon Echo Dot Music	Amazon Echo Dot News	Amazon Echo Music	Amazon Echo News	Chromecast Music	Chromecast YouTube	Google Nest Mini Music		
Amazon Echo Dot Music	30506	18394	2889	16779	702	99	23644	32.8%	67.2%
Amazon Echo Dot News	18055	27440	2632	25060	847	105	18874	29.5%	70.5%
Amazon Echo Music	2878	2595	79277	2932	1128	92	4111	85.2%	14.8%
Amazon Echo News	16752	24693	2912	29114	494	75	18973	31.3%	68.7%
Chromecast Music	698	819	1159	483	58539	30825	490	62.9%	37.1%
Chromecast YouTube	99	111	105	75	30173	62408	42	67.1%	32.9%
Google Nest Mini Music	23545	18961	4077	19119	511	39	26761	28.8%	71.2%
	33.0%	29.5%	85.2%	31.1%	63.4%	66.6%	28.8%		
	67.0%	70.5%	14.8%	68.9%	36.6%	33.4%	71.2%		

Figure 5.5.6: Confusion Matrix with both *interarrival times* and *packet sizes* for all possible classes with noise

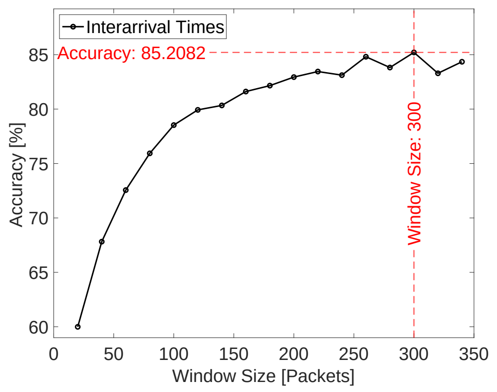
Classification using the raw features combined results in an approximated probability of 50%, in identifying the device and services running.

5.5.4 Interarrival Times Statistical Features

Using statistical methods to extract features from the raw data, the accuracy achieved using the *interarrival times* without the presence of noise is 84.2% at window size of 300 packets (Figure 5.5.7a). With the presence of noise the accuracy is 85.2% with window size of also 300 packets (Figure 5.5.7b).



(a)

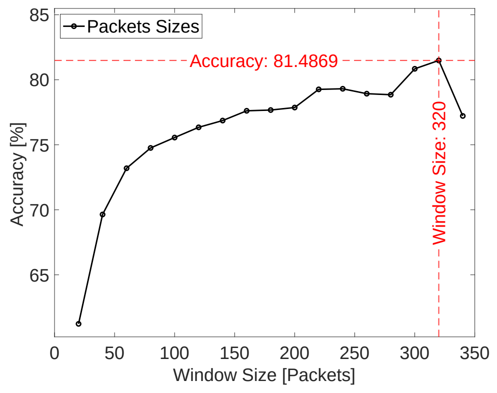


(b)

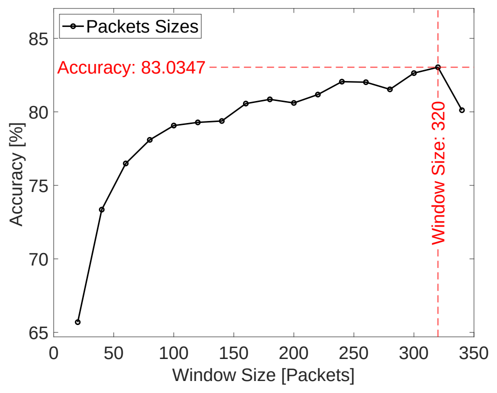
Figure 5.5.7: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *interarrival times* statistical features.

5.5.5 Packets Sizes Statistical Features

For the *packet sizes*, the accuracy in noiseless environment is 81.5% with window size of 320 (Figure 5.5.8a). While in a noisy link, the accuracy achieved is 83.0%, with window size of 320 (Figure 5.5.8b).



(a)

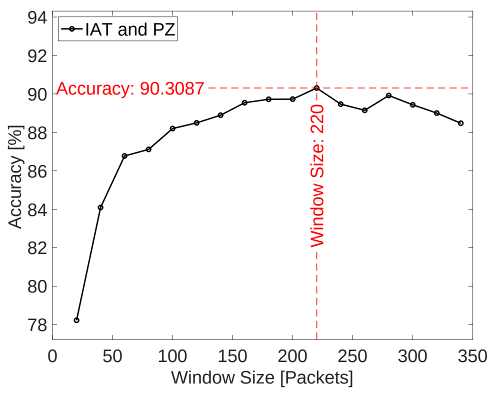


(b)

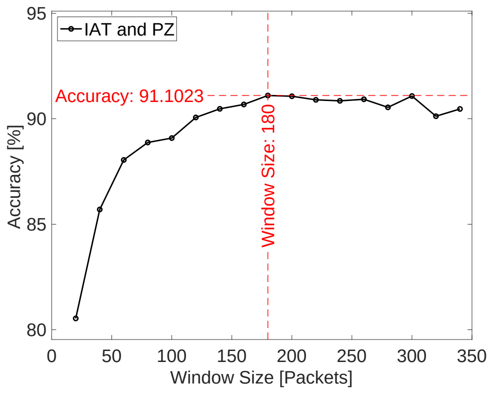
Figure 5.5.8: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing *packets sizes* statistical features.

5.5.6 Interarrival Times and Packets Sizes Statistical Features

The accuracy achieved in a noiseless environment is 90.3% with a window size of 220 (Figure 5.5.9a). With the noise scenario, the accuracy is 91.1% with window size of 180% packets (Figure 5.5.9b).



(a)



(b)

Figure 5.5.9: Services classifications accuracies associated with window sizes achieved (a) without, and (b) with the presence of noise, when utilizing both *interarrival times* and *packets sizes* statistical features.

5.6 Results Discussion

In this chapter, we introduced 2 different features engineering techniques: *Raw* and *Statistical*, to classify traffic associated with 3 services running on four different smart home IoT devices, using *Random Forest* machine learning algorithm. First we classified raw features (*interarrival times* and *packets sizes*) individually and combined. Similarly, we classified based on the statistical features extracted from the raw features. For both techniques, the classification was done with and without the presence of noise.

5.6.1 Raw Features

In Table 5.6.1, all the classification accuracies associated with each device running different service(s) and feature(s) used is summarised. The classification is ran against two services streamed by the same device, with and without the presence of noise. It can be noticed that in most cases, when classifying using both features combined (interarrival times and packets sizes), the accuracy achieved is much higher than classifying each independently, even with the presence of noise.

Device	Service(s)	Feature(s)	Noise	Accuracy [%]	Figure
Amazon Echo	Music vs. News	Interarrival Times	✗	54.8	5.1.1a
			✓	42.7	5.1.1b
		Packets Sizes	✗	52.2	5.1.2a
			✓	64.6	5.1.2b
		Combined	✗	60.6	5.1.3a
			✓	66.5	5.1.3b
Amazon Echo Dot	Music vs. News	Interarrival Times	✗	57.9	5.2.1a
			✓	45.1	5.2.1b
		Packets Sizes	✗	52.8	5.2.2a
			✓	64.9	5.2.2b
		Combined	✗	60.5	5.2.3a
			✓	69.9	5.2.3b
Google Nest Mini	Music vs. Noise	Interarrival Times	✓	64.9	5.3.1
		Packets Sizes	✓	91.9	5.3.2
		Combined	✓	91.8	5.3.3
Google Chromecast	Music vs. YouTube	Interarrival Times	✗	69.6	5.4.1a
			✓	50.9	5.4.1b
		Packets Sizes	✗	54.9	5.4.2a
			✓	67.4	5.4.2b
		Combined	✗	71.2	5.4.3a
			✓	78.5	5.4.3b
All Devices	**All Services**	**Interarrival Times**	✗	26.2	5.5.1
			✓	23.5	5.5.2
		Packets Sizes	✗	54.9	5.5.3
			✓	47.5	5.5.4
		Combined	✗	48.2	5.5.5
			✓	52.2	5.5.6

Table 5.6.1: Using Raw Features

5.6.2 Statistical Features

When extracting statistical features from the raw features, the accuracy is dramatically increased along with increasing the window size N. Table 5.6.2 illustrates the obtained accuracies for all the devices and services streamed with and without the presence of noise. The classification was done using different window sizes, starting from $N = 20$ to 340. When classifying interarrival times or packets sizes independently, it can be see that the model accuracy is $> 85.7\%$. With the presence of noise, the results obtained are $> 93.6\%$. In Table 5.6.3, we extracted statistical features from all traces, and tested each individually against different chunks of the noise data-set. The results obtined shows that using window size of 180 packets, with the presence of noise, gives very high and accurate classification results.

Device	**Service**	**Feature**	**Noise**	**Accuracy [%]**	**Window Size [Packets]**	**Figure**
Amazon Echo	Music vs. News	Interarrival Times	✗	90.1	300	5.1.4a
			✓	90.1	300	5.1.4b
		Packets Sizes	✗	86.0	300	5.1.5a
			✓	90.6	320	5.1.5b
		Combined	✗	90.5	260	5.1.6a
			✓	93.7	300	5.1.6b
Amazon Echo Dot	Music vs. News	Interarrival Times	✗	89.9	280	5.2.4a
			✓	89.7	280	5.2.4b
		Packets Sizes	✗	85.7	320	5.2.5a
			✓	90.7	320	5.2.5b
		Combined	✗	90.7	260	5.2.6a
			✓	93.6	260	5.2.6b
Google Nest Mini	Music vs. Noise	Interarrival Times	✓	99.9	320	5.3.4a
		Packets Sizes	✓	99.9	240	5.3.4b
		Combined	✓	100	140	5.3.4c
Google Chromecast	Music vs. YouTube	Interarrival Times	✗	98.5	200	5.4.4a
			✓	97.9	320	5.4.4b
		Packets Sizes	✗	99.5	340	5.4.5a
			✓	99.5	340	5.4.5b
		Combined	✗	99.5	320	5.4.6a
			✓	99.5	320	5.4.6b
All Devices	**All Services**	**Interarrival Times**	✗	84.2	300	5.5.7a
			✓	85.2	300	5.5.7b
		Packets Sizes	✗	81.5	320	5.5.8a
			✓	83.0	320	5.5.8b
		Combined	✗	90.3	220	5.5.9a
			✓	91.1	180	5.5.9b

Table 5.6.2: Using Statistical Features

5.6.3 Testing against different Noise segments

In this experiment, the best performing window size selected and used to test the traces against different chunks of noise. The noise is divided into chunks based on the number of samples of the trace being evaluated. The trace is then classified against each noise chunk and the mean of all confusion matrices is taken. Table 5.6.3 shows the obtained accuracies associated with devices and services. It can be seen that all accuracies are > 0.99.

Device	**Service**	**TPR**	**FPR** $[\cdot 10^{-4}]$	**Accuracy**
Amazon Echo Dot	Music	0.9997	0	0.9999
	News	0.9996	5.28	0.9996
Amazon Echo	Music	0.9995	2.64	0.9996
	News	0.9996	4.40	0.9996
Google Chromecast	Music	0.9993	4.40	0.9994
	YouTube	0.9982	4.40	0.9989
Google Nest Mini	Music	0.9987	5.30	0.9991

Table 5.6.3: Performance for Devices/Services detection with window size set to 180 packets and presence of Noise

Chapter 6

Countermeasures

Preventing network traffic classification is a challenging task since it strongly depends on the features adopted by the machine learning algorithm used for the classification. Thus, the adversary model (see Section 4.3) is critical since the attack performance can be significantly mitigated if the adversary is not able to build a ground-truth related to the device/service traffic. The best-case scenario (from the user perspective) is a completely *flat traffic* with no patterns, e.g., by generating traffic with packet sizes and interarrival times that cannot be detected as anomalies. Nevertheless, such an approach can hinder the quality of experience associated with the service, while traffic reshaping might require a different calibration of the device's network buffers.

6.1 Leaky Bucket

Imagine a bucket with a small hole at the bottom, no matter how much water is poured into the bucket, the amount or rate at which water flows out of the small hole is constant, this is how leaky bucket algorithm works. The leaky bucket is one of the congestion control mechanisms used to reduce congestion in the network, by reshaping the traffic at a constant rate. The algorithm works as follows:

- Packets that will be transmitted over the network are added to the bucket.
- If the bucket is full, any new packets are discarded.
- The bucket ”leaks”/transmit the packets at a predefined constant rate.

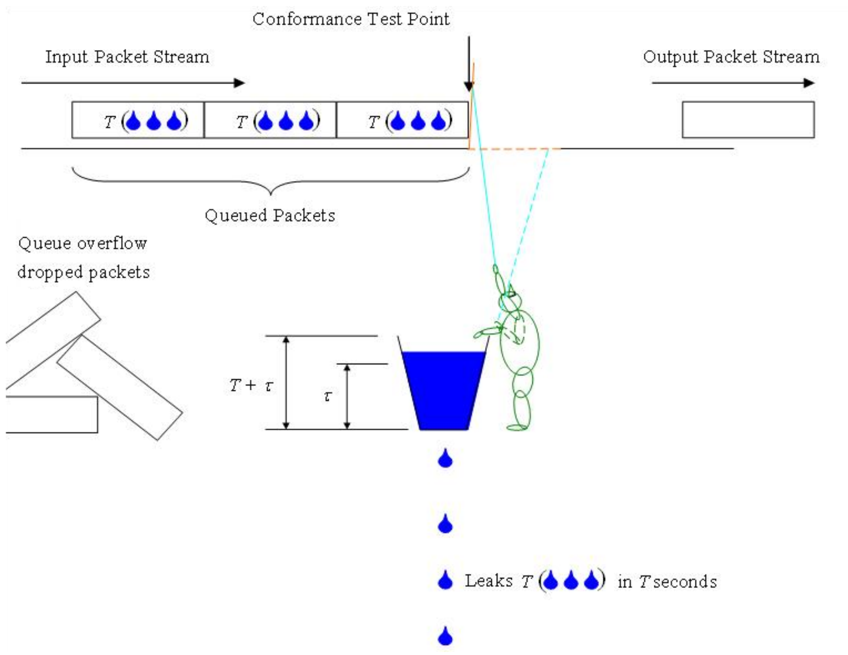


Figure 6.1.1: How Leaky Bucket Works [5]

Mitigation Technique

Leaky Bucket algorithm can be used to mitigate the pattern created by the interarrival times. Since its capable of sending out packets at a constant rate, which will make it difficult (almost impossible) for the adversary to infer by using or just looking at the interarrival times. That is because the arrival times for N packets will be identical since they are processed at the same fixed rate. The algorithm could be implemented in different places, for example on the sender (server) side, where the streaming data generated and sent to the receiver (user). Or on the receiver side, or between both sender and receiver. However, this method is considered a trade-off, as it will affect the quality of service delivered to the user, due to the delay it will cause when for example streaming videos.

Chapter 7

Conclusion

This work presents a major privacy leakage problem associated with the usage of the smart IoT assistant devices. The adversarial model introduced in this study is capable of identifying services and devices running within the user network, by utilizing two features from the traffic flow: interarrival times and packets sizes. These features are extracted and calculated from packets transmitted over the Wi-Fi link.

We presented a step by step procedures for data collection over the Wi-Fi and performed massive data collection campaign from four different widely used devices (Amazon Echo, Amazon Echo Dot, Google Nest Mini, and Google Chromecast) running three popular services: music, news, and YouTube. An in-depth analysis of the collected data was performed, and the results of this analysis show that each of the services associated with its devices, has its unique pattern by just examining the most frequent interarrival times and packets sizes.

For classification, the Random Forest algorithm was used, considering it the best performer among other supervised and unsupervised machine learning algorithms. Two different features engineering approaches were used for classification: 1. by using *raw* features, 2. computing and extracting 8 different *statistical* features from the raw features, over a predefined window size of packets.

The presented methodology has proved the existence of the aforementioned privacy leakage and proved the feasibility of the attack with accuracies > 0.99 for identifying and classifying devices running several services within a noisy environment. Finally, we introduced a mitigation technique named "leaky bucket", that process the packets sent from the streaming source to the receiving device at a constant rate. Thus, mitigating the phenomena created by the interarrival times and makes it impossible for flows to be identified.

7.1 Future Work

For future work, portable implementation of the adversarial model is considered. Where the usage of mini-computers such as Raspberry PI, powered by a portable power bank could be more critical. As it could be left anywhere without it being noticed and collect data for hours. Furthermore, increasing the number of devices and services is considered.

Bibliography

[1] Kamil Krzyk, “Coding Deep Learning For Beginners,” https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d, accessed: April 2020.

[2] Alexandre Drouin, “Microbiome Summer School 2017, Introduction to Machine Learning,” https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/, accessed: April 2020.

[3] “Detecting Patterns with Unsupervised Learning,” https://mc.ai/detecting-patterns-with-unsupervised-learning/, accessed: April 2020.

[4] Karl Rosaen, “K-fold cross-validation,” http://karlrosaen.com/ml/learning-log/2016-06-20/, accessed: April 2020.

[5] Wikipedia, “Leaky bucket,” https://en.wikipedia.org/wiki/Leaky_bucket, accessed: May 2020.

[6] A. M. Hussain, G. Oligeri, and T. Voigt, “The Dark (and Bright) Side of IoT: Attacks and Countermeasures for Identifying Smart Home Devices and Services,” in *Security, Privacy, and Anonymity in Computation, Communication, and Storage: SpaCCS 2020 International Workshops, Nanjing, China, December 18-20, 2020, Proceedings 13.* Springer, 2021, pp. 122–136.

[7] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers.* John Wiley & Sons, 2014.

[8] MATLAB, “Matlab statistics and machine learning toolbox,” https://se.mathworks.com/help/stats/, accessed: February 2020.

[9] M. Pal and P. M. Mather, “An assessment of the effectiveness of decision tree methods for land cover classification,” *Remote sensing of environment*, vol. 86, no. 4, pp. 554–565, 2003.

[10] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] J. Ashfaque and A. Iqbal, “Introduction to support vector machines and kernel methods,” 04 2019.

[12] M. W. Gardner and S. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[13] N. Buduma and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms.* ” O’Reilly Media, Inc.”, 2017.

[14] D. P. Mandic and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability.* John Wiley & Sons, Inc., 2001.

[15] J. Langr and V. Bok, *GANs in Action: Deep Learning with Generative Adversarial Networks.* Manning Publications, 2019.

[16] Amazon, "Amazon Echo (2nd Generation)," https://www.amazon.com/all-new-amazon-echo-speaker-with-wifi-alexa-dark-charcoal/dp/B06XCM9LJ4, accessed: March 2020.

[17] ——, "Hue: Alexa Skills," https://www.amazon.com/Philips-Hue/dp/B01LWAGUG7, accessed: March 2020.

[18] Nick Pino, "Amazon Echo Dot (3rd Generation)," https://www.techradar.com/reviews/audio-visual/hi-fi-and-audio/audio-systems/amazon-echo-dot-1328595/review, accessed: March 2020.

[19] Will Greenwald, "Google Nest Mini Review," https://www.pcmag.com/reviews/google-nest-mini, accessed: April 2020.

[20] Erika Rawes, "Google Nest Mini vs. Amazon Echo Dot: Which is better?" https://www.digitaltrends.com/home/google-home-mini-vs-amazon-echo-dot/, accessed: April 2020.

[21] T. Karagiannis, A. Broido, M. Faloutsos *et al.*, "Transport layer identification of p2p traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement.* ACM, 2004, pp. 121–134.

[22] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement.* ACM, 2004, pp. 135–148.

[23] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 50–60.

[24] W. Li and A. W. Moore, "A machine learning approach for efficient traffic classification," in *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems.* IEEE, 2007, pp. 310–317.

[25] F. Kausar, S. Aljumah, S. Alzaydi, and R. Alroba, "Traffic analysis attack for identifying users' online activities," *IT Professional*, vol. 21, no. 2, pp. 50–57, 2019.

[26] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications.* IEEE, 2009, pp. 1–8.

[27] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$deep-full-range$: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45 182–45 190, 2019.

[28] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and A. S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" *arXiv preprint arXiv:1808.02741*, 2018.

[29] 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)*. IEEE, 2017, pp. 559–564.

[30] R. B. Jackson and T. Camp, "Amazon echo security: Machine learning to classify encrypted traffic," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–10.

[31] S. Ostermann, "tcptrace," http://www.tcptrace.org, accessed: December 2019.

[32] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 2003, pp. 25–36.

[33] N. Msadek, R. Soua, and T. Engel, "Iot device fingerprinting: Machine learning based encrypted traffic analysis," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–8.

[34] U. Sydney, "Iot traffic analysis," https://iotanalytics.unsw.edu.au/, accessed: February 2020.

[35] A. Sivanathan, D. Sherratt, H. H. Gharakheili, V. Sivaraman, and A. Vishwanath, "Low-cost flow-based security solutions for smart-home iot devices," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2016, pp. 1–6.

[36] T. Jan, "Ada-boosted locally enhanced probabilistic neural network for iot intrusion detection," in *Conference on Complex, Intelligent, and Software Intensive Systems*. Springer, 2018, pp. 583–589.

[37] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Iot devices recognition through network traffic analysis," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5187–5192.

[38] M. R. Santos, R. M. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in iot ecosystems," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 304–00 309.

[39] J. Too, "Simple machine learning algorithms for classification," https://www.mathworks.com/matlabcentral/fileexchange/71461-simple-machine-learning-algorithms-for-classification, accessed: February 2020.