2020

Profiling and Identification of Web Applications in Computer Network

Oudah, Hussein

http://hdl.handle.net/10026.1/16687

http://dx.doi.org/10.24382/775
University of Plymouth

UNIVERSITY OF PLYMOUTH

Profiling and Identification of Web Applications in Computer Network

by

Hussein Jaber Oudah

A thesis submitted to the University of Plymouth

in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering, Computing and Mathematics

April 2020

Acknowledgements

First and foremost, I would like to thank Allah (God) Almighty for giving me the strength, knowledge, ability, and opportunity to undertake this research study and to persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible.

I would like to express my appreciation and gratitude to my supervisor Dr Bogdan Ghita for his continuous support, interest, patience, and guidance throughout my studies. Thanks must also go to my other supervisors, Dr David Lancaster and David Walker, who has spent time proofreading papers and my thesis, in addition to providing helpful experience and guidance throughout my studies.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. Thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my mother and my father for his support and never-ending love.

My unreserved love, thanks, and appreciation must go to my wife (Zaman) and my sons who have been very patient, understanding, and inspiring to me throughout this endeavour, spending days, nights, and sometimes even holidays without me. I hope the potential success of this research will compensate some of what they have missed. May Allah bless them.

Many thanks to my colleague Dr Taimur for his support and for the motivating ideas and thoughts they provided during my PhD journey.

Finally, I would like to acknowledge, with thanks and appreciation, the government of Iraq and the Higher Committee for Education Development in Iraq, for granting me a scholarship and sponsoring my PhD studies.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

This study was financed with the aid of a scholarship from the Republic of Iraq.

Relevant seminars and conferences were attended at which work was often presented and published.

Word count of thesis: 43, 896 words

List of publications:

H. Oudah, B. Ghita, and T. Bakhshi, "Network Application Detection Using Traffic Burstiness," in *World Congress on Internet Security* (WorldCIS-2017), 2017.

https://www.researchgate.net/profile/Hussein_Oudah2/publication/321906101_Network_Application_Detection_Using_Traffic_Burstiness/links/5d42e8a392851cd046987a8b/Network-Application-Detection-Using-Traffic-Burstiness.pdf

H. Oudah, B. Ghita, and T. Bakhshi, "A Novel Feature Set for Application Identification," in International Journal for Information Security Research (IJISR), Volume 8, Issue 1, March 2018

DOI: https://doi.org/10.20533/ijisr.2042.4639.2018.0088

H. Oudah, B. Ghita, and T. Bakhshi, "A Novel Features Set for Internet Traffic Classification using Burstiness," in 5th International Conference on Information System Security and Privacy 2019.

DOI: https://doi.org/10.5220/0007384203970404

H. Oudah, B. Ghita, T. Bakhshi, A. Alruban, D. Walker "Using Burstiness for Network Applications Classification", Journal of Computer Networks and Communications, 2019.

DOI: https://doi.org/10.1155/2019/5758437

Signed Hussein Oudah

Date 13/04/2020

Abstract

Profiling and Identification of Web Applications in Computer Network

Hussein Jaber Oudah

Characterising network traffic is a critical step for detecting network intrusion or misuse. The traditional way to identify the application associated with a set of traffic flows uses port number and DPI (Deep Packet Inspection), but it is affected by the use of dynamic ports and encryption. The research community proposed models for traffic classification that determined the most important requirements and recommendations for a successful approach. The suggested alternatives could be categorised into four techniques: port-based, packet payload based, host behavioural, and statistical-based. The traditional way to identifying traffic flows typically focuses on using IANA assigned port numbers and deep packet inspection (DPI). However, an increasing number of Internet applications nowadays that frequently use dynamic post assignments and encryption data traffic render these techniques in achieving real-time traffic identification. In recent years, two other techniques have been introduced, focusing on host behaviour and statistical methods, to avoid these limitations. The former technique is based on the idea that hosts generate different communication patterns at the transport layer; by extracting these behavioural patterns, activities and applications can be classified. However, it cannot correctly identify the application names, classifying both Yahoo and Gmail as email. Thereby, studies have focused on using statistical features approach for identifying traffic associated with applications based on machine learning algorithms. This method relies on characteristics of IP flows, minimising the overhead limitations associated with other schemes. Classification accuracy of statistical flow-based approaches, however, depends on the discrimination ability of the traffic features used. *NetFlow* represents the de-facto standard in monitoring and analysing network traffic, but the information it provides is not enough to describe the application behaviour. The primary challenge is to describe the activity within entirely and among network flows to understand application usage and user behaviour. This thesis proposes novel features to describe precisely a web application behaviour in order to segregate various user activities. Extracting the most discriminative features, which characterise web applications, is a key to gain higher accuracy without being biased by either users or network circumstances. This work investigates novel and superior features that characterize a behaviour of an application based on timing of arrival packets and flows. As part of describing the application behaviour, the research considered the on/off data transfer, defining characteristics for many typical applications, and the amount of data transferred or exchanged. Furthermore, the research considered timing and patterns for user events as part of a network application session. Using an extended set of traffic features output from traffic captures, a supervised machine learning classifier was developed.

To this effect, the present work customised the popular *tcptrace* utility to generate classification features based on traffic burstiness and periods of inactivity for everyday Internet usage. A C5.0 decision tree classifier is applied using the proposed features for eleven different Internet applications, generated by ten users. Overall, the newly proposed features reported a significant level of accuracy (~98%) in classifying the respective applications. Afterwards, uncontrolled data collected from a real environment for a group of 20 users while accessing different applications was used to evaluate the proposed features. The evaluation tests indicated that the method has an accuracy of 87% in identifying the correct network application.

Table of Contents

List of Figures

List of Tables

Abbreviation List

ISPs	Internet Service Providers
VNI	Visual Networking Index
IANA	Internet assigned numbers authority
CAGR	Compound Annual Growth Rate
DPI	Deep packet inspection
MLAs	Machine-learning algorithms
TC	Traffic classification
SLFC	Session level flow classification
PSD	Packet size distribution
MSSC	Message size sequence classifier
MSSes	Message size sequences
STF	Small time scale flight
LTF	large time scale flight
T	burst_threshold
I	Idle_threshold
CSCAN	Centre for Security, Communications and Network Research
PCA	Principal Component analysis
CDN	Contents delivery network
QoS	Quality of service
QoE	Quality of experience
SVM	Support Vector Machine
GB	Gradient Boosting
SDN	Software defined network

1 Introduction

1.1 Introduction

In the context of ever-increasing network activity and reliance on the Internet, monitoring and characterizing network traffic is critical in providing network administrators with the necessary information for operational and security activities. A number of directions were explored by research community, such as establishing what are the websites that the users are interested in, how much traffic is generated by specific network applications, and whether these applications or services can be controlled in terms of network resource demands [1]. A report published by Cisco predicted that global IP traffic will raise to 4.8ZB per year by the end of 2022 [2]. In addition, characterising network traffic is a critical step for detecting network intrusion and traffic anomalies, both typically featuring in end-user and corporate environments. A UK-based survey from 2018 about cyber security breaches acknowledged that the majority of all organisations depend heavily on digital environments such as email, websites, online banking and shopping; therefore; providing a secure system in Internet environment is vital to keep people's life safer and easier. One of the reasonable solutions is to do traffic classification and labelling applications to set priority for significant traffic and dismiss the noise in order to maintain resources and keep optimal performance. It was observed that when a data is captured under windows, there are some traffic comes in the wire even a user not access Internet as these computers owned by the University. They run web-based services in the background that add noise to captured traffic.

There are four main approaches (port-based, packet payload based, host behavioural, and statistical-based) that have been used for characterizing Internet traffic and giving the administrators, ISPs (Internet Service Providers), and engineers a better view of the network activity. In the early days of the Internet, applications were identified based exclusively on port number [3]. However, due to the continuous growth of Internet applications, this is no longer an option, as applications have been moved towards a web-based front-end (i.e., they used http or https with port 80 and 443 respectively) or used dynamic ports [4]. Consequently, this method becomes inaccurate in identifying applications and typical performance ranging between 30-70% [5]. A more accurate method is Deep packet inspection (DPI) [6] that relies on the contents of the packets to identify signatures of applications or protocols. This method is also proved to be inefficient in recent years as most applications use encryption methods, moreover, it breaches the privacy of the users and needs more computational resources [7, 8]. The research community has therefore introduced two techniques, focusing on host behaviour and statistical methods, to avoid these limitations. The former technique is based on how an application behaves depending on a variety of communication patterns at transport layer generated from this application. Despite the high accuracy of this method (over 90%) that was considered by many studies [9-15], it is unable to identify application name such as YouTube or Netflix while classifying them as streaming. However, this technique is primary used to identify P2P applications with high accuracy as the approach relies on the connection patterns that are generated from the peers. In other words, this approach based on analysing parameters that are collected from different flows in the end-point before successful application identification.

In contrast, the statistical approach tends to outperform previous methods with high accuracy (over 95%) and it is widely used by the recent studies [16-18] [19-28]. This method uses packet header rather than payload information, which makes the approach efficient even with encrypted traffic and does not breach the user's privacy; it achieved a relatively high accuracy while employing machine-learning algorithms MLAs. Different techniques have been used in this method from supervised to unsupervised and semi-supervised ML. Whilst the supervised approach outperforms the other techniques, building robust ground truth data for training a machine-learning model is required. In addition, it is apparent that most papers tried to do coarse classification. In other words, most studies identified either application class such as streaming and browsing, or protocols such as HTTP and FTP, or P2P applications such as Bit Torrent and skype. Few papers tried to do fine-grained classification such as the one in 2018 [29] that identified application type such as Facebook and Google services. Such studies typically employ machine learning approaches to classify Internet traffic based on recycling conventional features, focusing on the amount of data transferred in the network or the arrival timing for packets, flows or session. These features are calculated statistically and are therefore subject to change due to the continuously changing in the content of web pages. The features that are introduced in this thesis are based on timing between packets within a flow or between flows within a session based on burstiness and idle time. In other words, they are counting the activities of a user when he/she is browsing internet websites to represent the behaviour of the application.

Flow accounting methods such as *NetFlow* [30] represent the de-facto standard in monitoring and analysing network traffic. A *NetFlow* record, however, comprises limited aggregate information about packets traversing the network and is usually considered inadequate to describe application behaviour. This project aims to propose and investigate a novel mechanism to define web applications as seen through the generated network traffic using tcptrace tool. Therefore, this thesis proposes novel features to describe precisely a web application behaviour in order to segregate various user activities. Extracting the most discriminative features, which characterise web applications, is a key to gain higher accuracy without being biased by either users or network circumstances. This work investigates novel and superior features that characterize a behaviour of an application based on timing of arrival packets and flows.

While the application does indeed exhibit a different signature in terms of packet arrival distribution, user behaviour may also influence this distribution, particularly in relation to long-term activity, as idle times are a factor of user behaviour too. The results showed that some features can be affected by a user behaviour when different users browse the same application. Using different feature or set of features could lead to different results, therefore, more investigations are needed to prove whether a user's behaviour is affected or not by the proposed features.

The rest of the chapter is organized as follows: Section 1.2 illustrates the growth and rapid evolution of Internet traffic over the past decade. Section 1.3 identifies the need for traffic classification. Section 1.4 discusses the methods of current traffic classification approaches and challenges. Section 1.5 highlights the aims and objectives of this thesis, and section 1.6 presents the thesis structure.

1.2 Context – Internet Traffic Classification

In 2019 the Cisco Visual Networking Index (VNI)[2], which is responsible for tracking and forecasting networking applications, published a report that predicted the anticipated growth in global IP traffic and the number of connected devices from (2017-2022). The report forecasts that the annual global IP traffic will stand at around 4.8 ZB per year by the end of 2022, while the annual rate was 1.5 ZB per year by the end of 2017. Figure 1-1 shows the yearly consumption of IP traffic between the years 2017-2022. The boost in compound annual growth rate (CAGR) that surpasses 26% is a substantial increase in yearly Internet traffic. Moreover, the report also highlights the following key findings with respect to the growth in user Internet activity. Broadband speeds will double by 2022, the

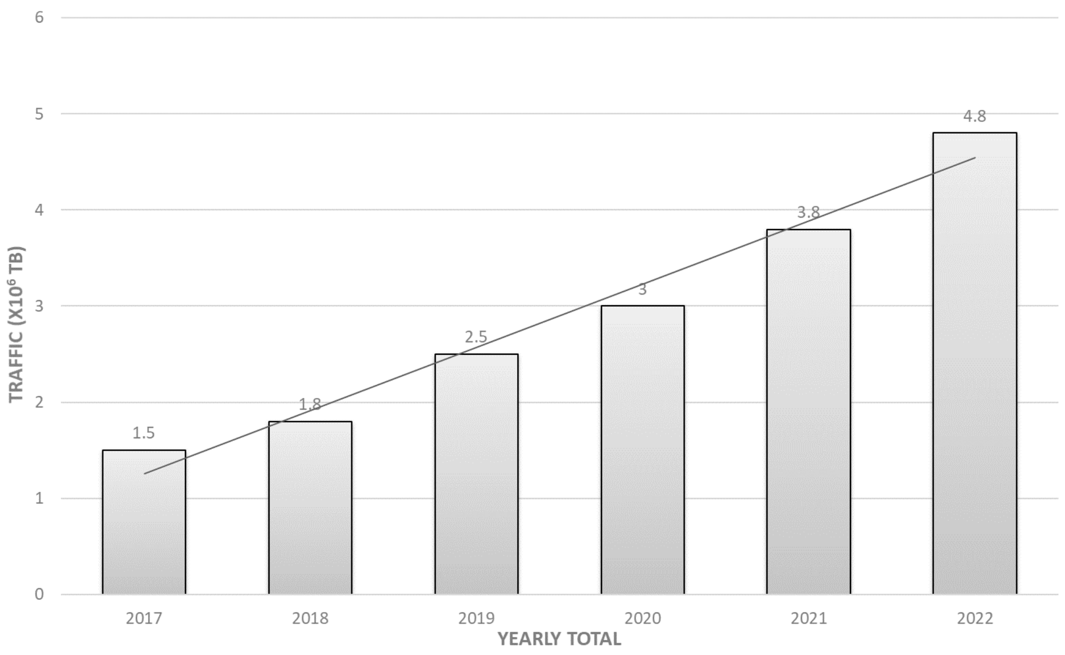


Figure 1-1: Cisco VNI Forecast Report: Growth in IP Traffic (2017-2022)

globally fixed broadband speeds could increase from 39 Mbps in 2017 to reach up to 75.4 Mbps in 2022. Wireless traffic will overtake wired traffic by 2022; the percentage of wireless and mobile devices traffic will be about 71% of IP traffic; however, only 29% of IP traffic will be generated by wired devices by the end of 2022. Smartphone traffic will also exceed PC traffic by 2022; in 2017, the traffic generated by the PCs was about 41% of total IP traffic, while by 2022 this percentage will decline to approximately 19%. In contrast, the IP traffic generated by the smartphones will be 44 percent of total IP traffic by 2022, up from 18 percent in 2017. The growth of PC traffic will be 8%, while the percentage of other devices such as TVs, tablets, M2M, and smartphones will be around 17%, 39%, 44%, and 58% respectively. Two sources of web traffic are generated across the computer networks. Traffic that is being generated by devices such as TVs, tablets, PCs and smartphones, which is mentioned by Cisco VNI Forecast Report and is emerging from people browsing the Internet. In contrast, there is another type of web traffic that is generated by search engine, good bot traffic, hacking tools, and scrapers, which is belong to non-human sources. The later one represents a majority of web traffic according to a report that was published by Incapsula [31], which is a provider of cloud-based security for web sites. Figure 1-2 shows the distribution and composition of Internet traffic in different categories. In addition to the growing constraints on existing networks, a profound increase in Internet traffic also affects storage devices and application servers, influencing the overall performance and efficiency of network infrastructures [31]. This makes the task of classifying Internet traffic for subsequent policy implementation even more pertinent, requiring a sophisticated yet scalable traffic classification approach to manage network traffic efficiently. The following section discusses the need for traffic classification in more detail.

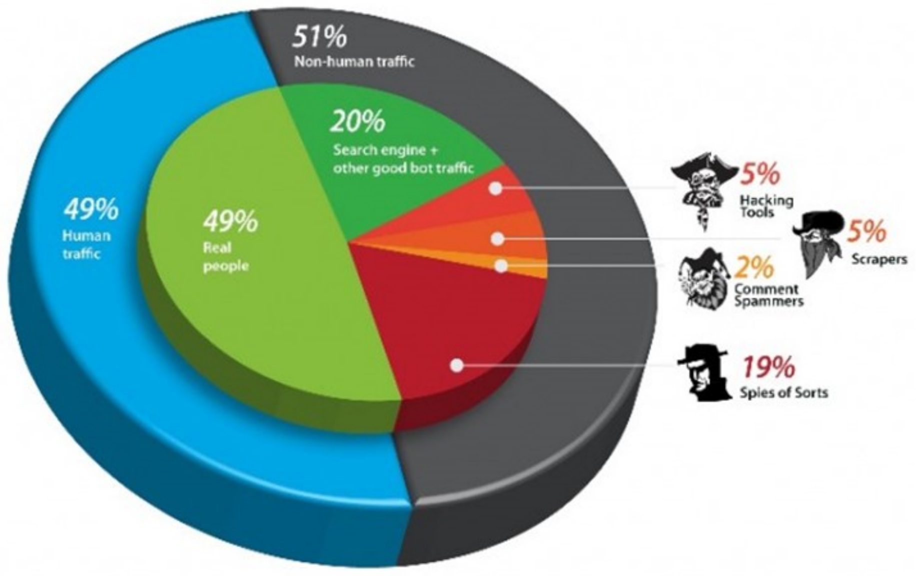


Figure 1-2: Web Traffic (Type) Distributions [32]

1.3 Traffic Classification Importance

Traffic classification can be considered as an initial task of analysing different patterns of applications and protocols in the network and subsequently utilising classification information to manage different tasks such as monitoring, service discovery, routing control, and resource optimisation [33]. The existing solutions for traffic monitoring and management such as Solarwinds, Nmap, spiceworks, Zabbix, and Cacti are only used to monitor network devices (i.e., switches, routers, and firewalls). In other words, they provide visibility into the devices on the managed networks. These tools provide detail information about the CPU, temperature, fan and etc. Other tools such as ntop [34] is a traffic probe that capturing packet using libpcap to display information on network traffic. This tool provides information regarding volume, bytes, and IP addresses and classify traffic based on IP, port, and protocols. Also, Wireshark is an open source packet analyser that capturing packets at wire speed or reading existing dump files. It is able to filter, group and annualize network traffic. IP SLA is a tool to detect jitter, packet loss, and MOS (Mean Opinion Score). This tool can use DNS to verify protocols such as FTP and HTTP. As can be noticed that these solutions are providing only information about volume, IP addresses or protocols such as FTP and HTTP. Therefore, the method proposed in this thesis is to classify traffic into different web applications such as Facebook, YouTube and Gmail. As an example, application identification helps Internet Service Providers (ISPs) in managing and prioritising Internet traffic classes and appropriating network resources. Traffic classification, therefore, aids network administrators in accurately distributing limited network resources in an effective manner. Also, traffic classification is helping the network designers to understand different types of traffic to apply quality of service (QoS). The requirements of applications and services are different according to bandwidth, delay, packet loss and other parameters. Therefore, knowing what application or service is associated with network flows is essential. The next section reviews some of the limitations of existing solutions to traffic monitoring and managing.

1.4 Existing Methods and Challenges

Several prior studies have discussed a range of traffic classification mechanisms focusing on port-based mappings for traffic classification to the use of machine learning (ML) techniques for accurate application identification. A summary of prominent methods along with their limitations is presented in Table 1-1. In the early stages of traffic characterisation, Internet assigned numbers authority (IANA) port-based mapping was used to classify Internet traffic type [35]. Being a relatively simple approach, it yielded high accuracy in the early days of the Internet when all applications were assigned and utilised known (documented) port numbers. After the rapid evolution of the Internet and the subsequent increase in the number of available applications, port-based traffic identification became increasingly obsolete. Moreover, the existence of firewalls, address translation, port forwarding and protocol tunnelling makes it challenging to match service with a particular port [10].

Deep packet inspection (DPI) techniques emerged when port-based classification technique was deemed ineffective. DPI investigates the payload and the header of the packet searching for virus, spam, intrusion or signatures that belong to specific applications [36]. DPI is robust and gives highly accurate traffic identification, but also requires relatively high processing time and adds to the management overhead. Additionally, DPI schemes do not conserve user privacy and more importantly cannot deal with encrypted applications [36]. To address the above limitations of traffic classification, research studies also focused on techniques which analyse the host behaviour by observing the traffic patterns generated by different end-user applications through the network to reveal the application type [10]. Although being more resource efficient in comparison with DPI, behavioural classification also presented some challenges. Applications have somewhat similar network behaviour, for example, VoIP and P2P could not be accurately classified using host behaviour alone and required heuristic-based approaches using different machine learning techniques to increase classification accuracy. Another body of work in traffic classification employed statistical analysis for identifying application traffic types recording numerical features such as packet size, inter-arrival time of the packets, byte size, etc. Statistical analysis, coupled with machine learning algorithms incorporating supervised and unsupervised training methods, can be used to build ground truth classification data for individual applications. The accuracy of the machine learning approaches requires significant effort in obtaining high-quality ground truth data for supervised classifier derivation [37].

Table 1-1: Existing Traffic Classification Approaches and Challenges

Classification Approach	Method	Limitations
Port-Based	IANA assigned port-mappings	Dynamic port-assignments and tunnelling
Deep Packet Inspection	Packet content and header analysis	Computational overhead encrypted payload
Host Behaviour Analysis	Analyse host behaviour and application traffic pattern	Applications with similar behaviour are difficult to classify
Statistical Analysis	Identify applications using numerical traffic features	Difficult to obtain high quality ground-truth training data
Combinatorial/Hybrid	Multiple approaches, combination of machine learning techniques	Specific to individual network settings

Many hybrid approaches have, therefore, been implemented in several prior studies to design an optimal traffic classifier. The trade-offs between high classification accuracy, the specific approach used and system (hardware) requirements are highly dependent on business needs and the implementation scenario. Each of the proposed solutions focuses on or is suitable for a specific network setting, meaning that no global classification scheme can be deployed for at least many network environments [38]. The primary reasons contributing to the challenges in designing a generalised traffic classification model can be summarised as follows:

1. **Resource constraints:** The first reason associated with the limited applicability of any solution is the rate of traffic traversing computer networks and somewhat inadequate computational resources such as memory, storage, etc. in implementing real-time traffic classification. As mentioned earlier, while techniques such as DPI are highly accurate in identifying traffic using extracted patterns and features from packet payloads, the underlying equipment required for classifying traffic in even a modestly vast enterprise network is costly.
2. **Regular re-evaluation:** Once an optimal traffic classifier has been built using statistical, DPI or hybrid ML-based approaches; it needs to be regularly updated to identify newer applications (signatures) accurately. The classifier design, therefore, needs to account for and consider the real-time data collection mechanism, specifically the method for continuously acquiring ground-truth data and regularly updating/re-training the derived classifier. This adds further management and computational overhead to the classification system. Techniques such as offline training of the classifier followed by online classification have been used in prior studies to circumvent resource constraints; however, an optimal method for regular re-training and evaluation of classification system is still required.

3. **Limited datasets:** The third reason for the difficulty of designing traffic classification model is the inability to accurately compare among the several presently available methods of traffic classification. Limited public availability of data sets and lack of open source classification systems led researchers to either build their own training datasets (or databases) that make an accurate comparison among the available techniques even more challenging. Furthermore, where such datasets have been made available, training data are usually labelled using basic techniques such as port-based application mappings resulting in low-quality training data.

Finally, as the complexity of the Internet continually evolves, the composition and volume of the traffic characteristics will alter continuously. Therefore, new methods are being continuously introduced for accurate traffic classification and Internet traffic identification will remain a prevalent research problem in future.

1.5 Aims of the Project

This project aims to propose and investigate novel mechanisms to define web applications as seen through the generated network traffic. The project is divided into the following distinct stages.

1. Display the real Internet traffic nowadays and how it is predicted to grow in the future (chapter 2).
2. Review prior research in Internet traffic classification, identifying means of recording network application traffic patterns and characterising traffic (chapter 3).
3. Define novel traffic metrics for application and user traffic profiling and recording. To accurately describe the application behavior, the project will consider parameters such as the on/off data transfer, defining characteristics for a number of typical applications considering timing and

patterns for user events as part of a network application session (chapter 4).

4. Collect datasets appropriate for studying application behavior, under controlled environment to build the ground truth data and real traffic network to investigate the feasibility of the proposed method (chapter 5).
5. Perform an analysis of the proposed features to determine whether they are discriminant for identifying network applications based on the traffic that they exchange. Data analysis aims to find out the correlation and variability between the proposed features; consequently, an application behavior could be represented by few features rather than applying many features which enhance the classification accuracy (chapter 6).
6. Use machine learning techniques with an extended set of traffic features as input to derive an Internet traffic classifier that will be validated and evaluated against a number of applications (chapter 7).
7. Displays SDN (Software-defined network) technology to build an architecture to identify different applications based on IP addresses matching (chapter 8).

1.6 Thesis Structure

The remainder of thesis is structured as follows: Chapter 2 introduces the Internet infrastructure and overviews the technologies that have accelerated Internet performance, such as cloud computing and CDN. These technologies make the traffic classification harder as such environments increase the number of Internet applications and the possibility of continuous developing by the applications owners. Therefore, the behaviour of the applications could be different during the time that requires new definition for the existing metrics and propose new ones. In addition, this chapter provides an indication of what applications/traffic exists on this environment and connected devices. Moreover, this chapter focuses on network performance and challenges that could be faced during data transmission such as throughput, delays and loss of packets.

Chapter 3 presents the methods that are proposed by the research community for classification Internet applications with the emerging of Internet and how the early methods became inapplicable with nowadays applications. This chapter is ended with comprehensive discussion and conclusion for the most challenges that face the traffic classification.

Chapter 4 presents the main principle of burstiness and idle time and how the proposed features are generated. This principle identifies an additional set of features that can be used to discriminate between network applications, based on the statistical differences between inter-arrival times of packets and flows. The burstiness principle defined in two levels, the first level is in the context of packet analysis and the second level is in the context of flow analysis. Finally, the chapter highlights on a preliminary study that is conducted to determine whether the distribution of arrival time does indeed differ when using different applications

Chapter 5 shows a methodology and a collection of two types of data sets to test the feasibility of the proposed features mentioned in chapter 4. The first data set contained 10 users that were browsing 11 applications. The second data set was real data that was collected from a lab at Plymouth University for 20 users and different Internet applications; the chapter also presents the methodology of the proposed design for traffic classification. Moreover, the chapter details the pre-processing steps that were carried out on the data before evaluation by the classifiers.

Chapter 6 presents a feasibility study of using statistical techniques for selecting potential features by a thorough examination and preliminary testing. This analysis aims to determine whether the proposed features have a positive impact in discriminating between applications. This chapter aims to determine the possible correlations between input features, exploring the possible relationship between input and output features and investigating the minimum set of input features that maximize the accuracy for output prediction.

Chapter 7 presents an in-depth investigation into approaches that classify Internet traffic to evaluate the performance of the proposed features and to determine the validity of the present features. Building upon the previous chapters that investigated the features and the proposed design, this chapter proceeds to evaluate appropriate classifiers to determine the overall performance that can be achieved.

Chapter 8 displays SDN (Software-defined network) technology to build an architecture to identify different applications based on IP addresses matching. This chapter explains the main components of this architecture and the possible advantages and disadvantages.

Chapter 9 Presents the main conclusions from the research, highlighting the key achievements and limitations. The chapter also discusses the future research and development.

2 Internet Traffic Review

2.1 Introduction

Today's Internet is a massive engineering system that contains of hundreds of millions of servers, communication links, routers and switches; with billions of users that are accessing this environment via laptops, tablets, and smartphones [39]. Accessing the Internet enables users to buy and sell goods, watch movies or TV programs, play games, communicate and share information with friends and others. Companies and employers try to exploit the Internet for advertising their services and goods to customers based on their requirements. Therefore, any online activities that occur in this environment can be monetised. The success of such online environment is based on the availability of high-bandwidth and low-latency network connectivity that triggered of emerging new services such as social networking, content delivery, and e-commerce at large scale. This environment opens the doors for new technology to appear such as the Internet of things, M2M, gaming network and smartphones that run different applications and causes a massive of Internet traffic. Scheduling such massive traffic with the existing resources for a diverse set of applications is a challenging problem that needs a scalable and dynamic approach to manage and classify each application.

This chapter aims to provide a general introduction to the Internet infrastructure and overviews the technologies that have accelerated Internet performance, such as cloud computing and CDN – in addition, presenting the main points of applying traffic engineering and the appropriate tools in capturing, analysing and reduction traffic.

2.2 Internet Connectivity, Applications, and Traffic

The Internet is a collection of massive number of networks that contains hardware and software equipment that provide a global communication [40]. There are different Internet applications have been emerged recently such as social networks (e.g., Facebook, Twitter), video applications (e.g., Netflix, YouTube), and personal applications (e.g., iCloud, Dropbox). These applications need various requirements such as availability of resources and response time due to an enormous number of users access them over the Internet. For example, hundreds of processing units with thousands of servers spread over the world to provide a high quality of service for Google's users or Facebook. Therefore, many invented technologies have been built in the recent of years to fulfil this demand (e.g., cloud computing and contents delivery network (CDN)) [41]. Cloud computing means that the resources are available in data centres and everywhere with infinite scale and high response time to provide services on demand with low cost to users over the Internet [42]. The National Institute of Standards and Technology (NIST) describes cloud computing based on five Primary features which include on-demand service, easy remote access even from mobile devices, cloud resources are shared by customers, flexible in providing and release resources, and services are priced based on usage [43]. Applications in cloud computing have the advantage of an automatic-scaling feature which is not available in the traditional applications that provides these applications with high performance, availability, and lowest cost. Multiple applications in the cloud-based are dissimilar from the traditional applications in that share on a virtual machine (i.e., computing, memory, storage, and resources of a network) that provided by cloud infrastructure service provider.

On the other hand, CDN technology consists of servers that are connected to the origin server and located at massive load points. The primary goal is to deliver contents to clients from the nearest server that decreases not only the distance of carrying the contents from the main cloud but also reduces the number of hops in the packet travelling from point to point. This increases the performance of the system as it provides low latency and low packet loss [44]. The CDN consists of many geographical locations called PoPs (points of presence) that are cached with the contents to cover as much as possible users. For instance, when a user tries to access a web site that is hosted in the US, the contents of this web site are delivered from the PoP that is located in London [45] as Figure 2-1 shows this case clearly.

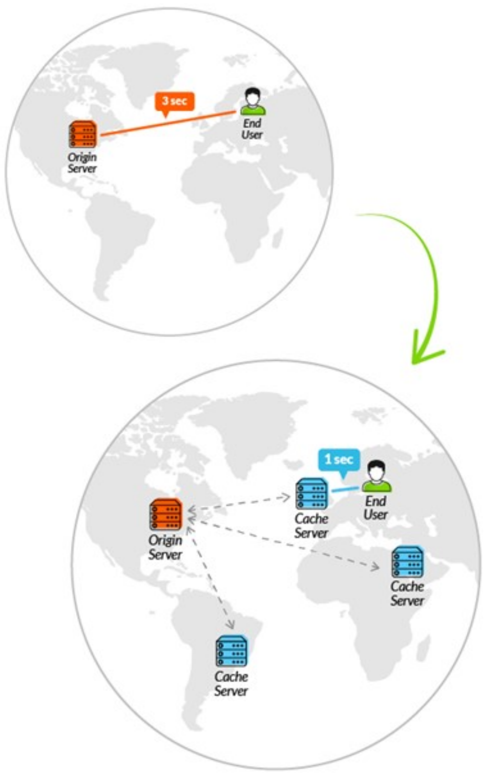


Figure 2-1: Low Latency for Applying CDN[45]

2.2.1 Internet Traffic

Internet traffic has grown dramatically during the last decades, based on a study published by Cisco [2] showing that the global traffic on the Internet networks was about 100 GB per day in 1992. In 2002, after only ten years, it raised to 100 GB per second, the raising nearly 86,440-fold within one decade. In 2017, global Internet traffic extended more than 46.6 TB per second. This study predicted that the traffic will reach up to 150.7 TB per second in 2022. There are many reasons behind this growth in such traffic; mainly, the increasing number of Internet users which has been growing from 500 million users within the past 15 years to more than 4 billion users [46]. Moreover, each person is expected to have about 3.6 of connected devices in 2022 up from 2.4 in 2017. Besides, the emerging of M2M applications such as healthcare monitoring, traffic control (vehicles), security in business and transportation which increase the growth of connected devices in the Internet environment. Further, other devices such as TVs, Non-Smartphones, PCs and others are also contributing to this growth, and the amount of generating traffic differs from one device to another. Figure 2-2 shows that smartphones will be the main source of global traffic (39 percent) in 2022 [2]. Although the M2M devices represent the majority of the connected devices, they are less generating from others. On the other hand, content delivery network technology (CDN) caches content in local servers which provide Internet availability for users and satisfy their requests [47]. For instance, a user from North America was able to access a third percent of his traffic from CDN area in 2017 and this figure will raise up to half percent by 2022. Universally, the average internet traffic, which delivered from CDN, was 56% in 2017 and it expects to be 72% by 2022 as shown in Figure 2-3. Another important factor is a broadband speed that also would be increased from 39 Mbps to 75.4 Mbps during the period from 2017-2022.

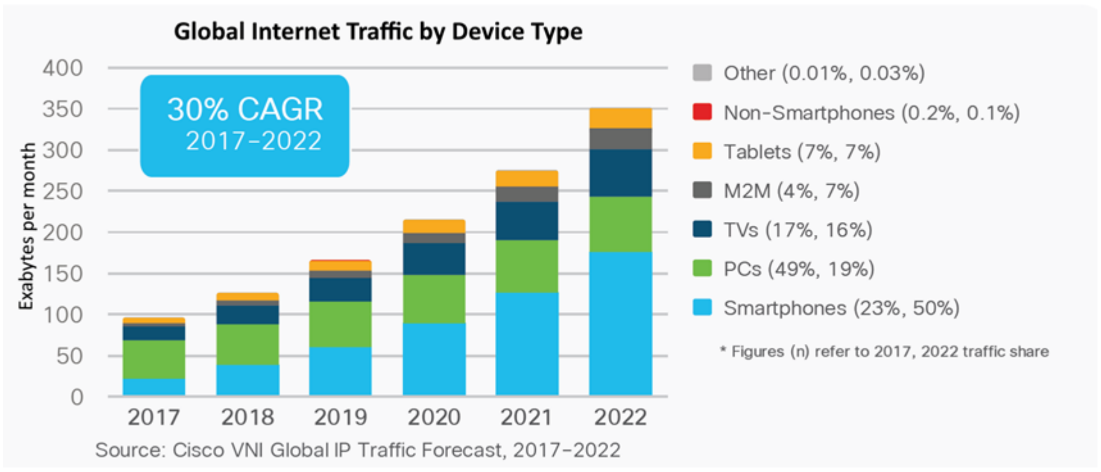


Figure 2-2: The Forecast of Global Traffic, 2017-2021 [2]

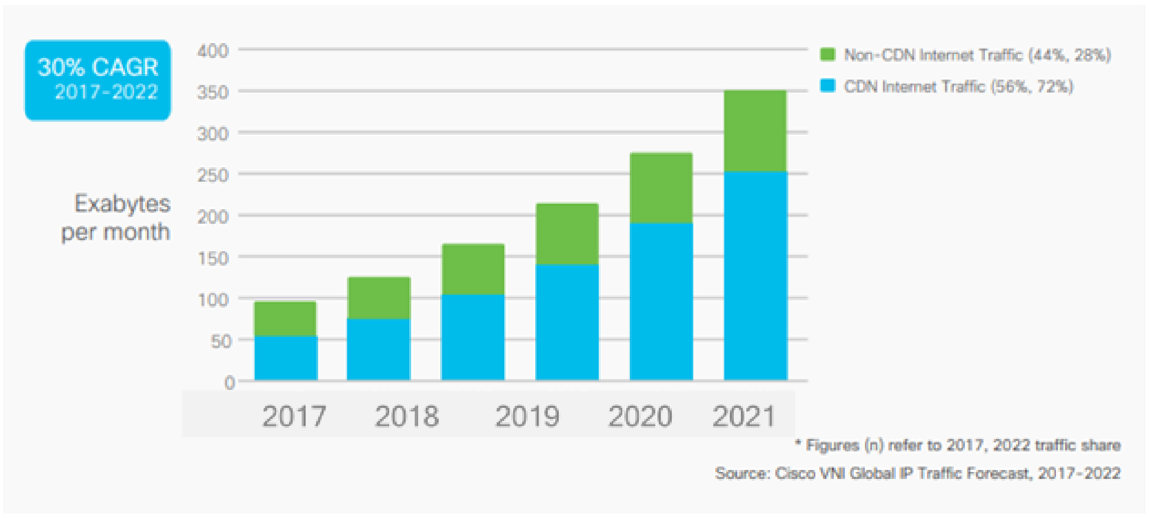


Figure 2-3: CDN Internet Traffic Growth, 2017-2022 [47]

A consumption of user for different Internet applications certainly raise when he has more bandwidth. Internet service providers found that with more bandwidth more traffic generates. Consequently, and due to the enormous traffic and users, there are concerns of breaching the security. For example, the FBI IC3 (Internet Crime Complaint Centre) received on average about 22,000 incidents of cyber attacks per month in 2014, with total loss of approximately $800 million [48]. Also, another report from Data Breach Investigations found about 80,000 incidents around the world in the same year and causing losing about $400 million [49]. In May 2017, the cyber attackers released a phishing program known as WannaCry through victims' email, which encrypted all victim's files. The program affected more than 200,000 computers around the world and they asked the infected users to pay $300 to control back on their files according to the Europol [50]. The biggest impact was in the UK in the NHS sectors which were unable to access their digital information caused cancellation of operations and appointments as patients' information were encrypted.

2.3 Network performance and applications

The aim of building a robust network is to enable the Internet services to move higher data between the clients and servers rapidly and without any loss in the data. However, there are some challenges that limit this aim such as throughput, delays and loss of packets. When the packet begins his journey from the source host, crosses many routers, and finishes in the destination host, it suffers from different types of delays at each node during this route. These delays are processing delay, queuing delay, transmission delay and propagation delay, which are in total give the actual delay that happens in the network. Consequently, such a delay will impact on the performance of Internet applications such as email, browsing, and video streaming[51]. The individual value of these delays changes from significant to a value that could be negligible. For example, the propagation delay could be a few microseconds within local connections while this delay could be higher for hundreds of milliseconds for geographical connections. The transmission and processing delays nowadays are negligible as the majority of the routers have high transmission speed and throughput. On the other hand, the queuing delay that is unlike the others and it is more interesting by the research community can alter among different packets as it is harnessed by the policy first-come-first-served. For example, when an empty buffer of a router receives 10 packets at ones, the first two or three packets could be sent without delay while the remaining packets might be sent in different delay. Therefore, this delay impacts by different factors which are the arriving rate of the packets to the router, the transmission rate of the packets from the router, and whether the arriving traffic comes in discrete form or bursts form. When the ratio between the arriving packets rate to the transmission bits rate is greater than 1, the queue would increase gradually until the router begins to lose packets; therefore, the traffic engineer tries to make this ratio less or equal to 1. In the opposite scenario, when the router is set to the ideal case, then the queueing delay would be formed based on the nature of the arriving packets (periodic or burst). The second case would be the worst when the traffic comes in burst forms and the queuing delay would also increase gradually.

2.3.1 **Packet Loss and throughput**

Traffic intensity in telecommunication networks denotes to the number of occupied resources (servers) at a given instant of time. When the traffic intensity is nearly 1 or less, the queuing delay will not reach infinity. In contrast, when the traffic density is greater than 1, the queueing buffer would be full and there are no space to store packets, therefore, the packet is dropped by the router and this is the packet loss. This phenomenon happens when there are high traffic density and the packets losing increases with increasing the intensity. Therefore, the delay and the probability of packet loss determine the network performance [51]. Throughput is also considered a measure of performance in a computer network and could be defined as the amount of data that the end node can receive per time. In voice applications, the throughput is very important and should be no less than 24 kbps for voice and 256 kbps for video applications with low delay. To understand the throughput, two scenarios are taken to explain this concept. The first scenario, when a server starts transferring data to a client, the rate of transmission should not exceed the minimum transmission rate of any node within a single link. In a different scenario, when a data is transferred from 10 servers to 10 clients and all the server shares the same link. The throughput rate is no longer calculated from the Min of the transmission links rate; instead, it is calculated from the transmission common link rate divided by 10 [52]. Therefore, the Internet networks could be impacted by the bottlenecks circumstances due to the bandwidth shared as shown in the above example, and this leads to high latency, packet loss and network outages.

2.3.2 Popular web applications

Internet traffic contains a variety of applications, such as online search, e-entertainment, online social networking, and gaming, all parts of people's lives. Most popular web applications [53] are selected to explore the fact that different applications can generate different characteristics based on application type and usage. Table 2-1 shows the properties of the web applications, which were browed by human being with the most common activities for users when he/she accesses the Internet. In spite of the common activities for applications that belong to the same class such as Facebook and Instagram, there are different characteristics for others when comparing among different classes. From the network traffic perspective, Facebook and Instagram web traffic could be classified into three clusters as were reported by [28], the first cluster contains the biggest payload such as streaming video, the second cluster contains information to control and establish connections, while the third one relates to the background and live information which is updated frequently. As shown from the Cisco report, that the video data represents a major part of traffic with volume reaching to terabits per second TB/s [53], such traffic cannot be provided from one or few servers to end users, rather it is provided by CDN that is available to user' location

Table 2-1: Different Activities for Eleven Web Applications

App/Type	Web browsing	Instance messages	Streaming	VoIP	Email	Search engine
Facebook & Instagram	×	×	×	×		
You Tube	×		×			×
Skype		×	×	×		
Gmail & Yahoo mail					×	
BBC news & CNN	×		×			×
Google search & Bing						×
Amazon	×					×

Due to the popularity of this application, the ISP (Internet service provider) must offer a good service to their clients in particular with large bandwidth demand. When the **YouTube** web application is requested by the client, several communications occur between clients, YouTube server, and cached contents server (CDN) [54][55]. According to study[28], the authors showed in practical that the YouTube traffic is not just a streaming, but also include other two classes which are video searches and messages between the YouTube servers. Looking at **Skype** traffic, two distinct clusters could be noticed, according to [28]; the first cluster is produced due the connections between client and super host which are basically low level data rate, while the other cluster is generated from connections between two clients which contains the actual calling with high-level data rate. For Email applications such as **Gmail** and **Yahoo email**, two clusters could be noticed, one for exchange email messages between client and server and this type of flows could be easily identified by well-known destination port such as

SMTP, POP, and IMAP protocols. The second cluster regarding the flows of directory lookup for the client and they have a low data rate compared with the first cluster [28]. For news websites such as **BBC news** and **CNN**, three types of flows that are generated when the client accesses such websites. The first type is video streaming and this flows include a high volume of data, the second type represents browsing with low bit rate flows and the last type forms search engine with also low bit rate. For **Google search** and **Bing** websites contain only one type of flows for a searching engine that accesses an external different website with a low bit rate. In conclusion, different applications generate different traffic in which different requirements need to be provided by the ISPs. The fact that the ISP needs to identify applications in order to profile users depending on their interactions. Therefore, providing standard QoS for the customers is not an easy task as the end-to-end path contains several networks that introduce packet loss and delay. The ISP offers good quality by utilizing bandwidth and availability in particular for live applications such as VoIP, gaming, and video conference streaming. For example, in real-time gaming that needs instance updating of a game information, a quality of experience (QoE) depends on latency and packet loss. Also for VoIP and video, the jitter (variation of latency over time) and packet loss are important for providing consistent service. Moreover, a new trend appears for giving the best quality based on specific application, for example, the ISP Australian iiNet [56] increased bandwidth for customers who accessed the Netflix application. In contrast, the application provider does efforts to minimize packet size requirements to reduce application burden which leads to minimum bandwidth allocation and improves QoS for packet loss and latency. Therefore, labelling flows/packets based on the application that using traffic classification (TC) techniques is vital for better QoS by routing appropriate traffic [57]. For example, the SDN (software-defined network) updates the network parameters based on requirements in the flows which are identified using TC engine, therefore the success of SDN operation relies on accurate classification [58]. Although new technologies have been introduced in this area such as cloud computing and CDN, the continuous increase of the traffic could influence performance, reliability and scalability of many Internet applications. Poor handling of these parameters could cost companies a lot of money as well as their reputation [59], therefore, the quality of Internet applications and services must be under a strict policy. The Internet success is dependent on providing sufficient resources and suitable performance requirements for present and future applications. For that reason, the internet service providers (ISPs) need to accommodate these requirements, but all traffic is encrypted, consequently, it is difficult to differentiate between flows. This project aims to propose and investigate a novel mechanisms to define web applications as seen through the generated network traffic to provide services with good quality.

2.4 Traffic and performance monitoring

The process of monitoring transmitted or received traffic within an Internet network is called internet traffic monitoring that aims to the following benefits:

1. Characterizing Internet traffic and giving the administrators, ISPs (Internet Service Providers), and engineers a better view of the network activity
2. Setting priority for significant traffic and dismiss the noise in order to maintain resources and keep optimal performance.
3. establishing what are the websites that the users are interested in, how much traffic is generated by specific network applications, and whether these applications or services can be controlled in terms of network resource demands

4. Identifying new applications and protocols, detect malicious or suspicious activities and provide a policy to delete or block such activities.

This section contains information regarding packets, flows, and tools that are used to capture and analysis traffic.

2.4.1 Packets

Traffic characteristics are important in performance analysis to calculate throughput, packet loss, packet delay in the links and routers, moreover, it is vital in network engineering that is concerned to know the network capacity and demand, monitoring and enhancing the operation of the network. The simplest form, which is valuable in traffic, is a packet, which is a principal unit in the IP protocol. Monitoring a collection of packets at some point of the network can reveal different activities. The most essential information in the packets is the manner of packet arrivals at observation point such as router or link. The arrival packet times could be summarized through the distribution of the characterization of inter-arrival process $\{I_n, n=1, 2…\}$ where $(I_n) = A_n - A_{n-1}$, where A_n refers to the arrival of current packet time and A_{n-1} refers to arrival of previous packet time. The packet size is also important which is equal to a total number of bytes in the packet, using time series of packet arrival with the size of the packets could reveal very essential information as the packet size varies during the time[60]. Packets could be also defined as a collection of packets during the active time, and zero packets during idle time, the state is similar to on/off process [61]. This traffic is generated when a number of packets form a train, which is defined as pulses of packets that are separated by an interval greater than a defined threshold between inter-arrival packet times. The precise of determining the right threshold does not change from the distributions of packets when it is greater than a typical value [62]. This definition is important to understand traffic properties and the transport protocols that are responsible for generating such phenomena when the main source of traffic (applications) is accessing. Observing and examining this structure helps understanding the traffic characteristics and their applications. A collection of trains called a session, the session could be defined as a single activity for the user when he/she accesses an application such as browsing page or sending an email.

tcpdump: tcpdump is a program for capturing the packets that travel through network interfaces. The libpcap library is an application programming interface (API) includes pcap which is implemented by Unix-like systems and used for capturing network traffic [63]. The interfaces in the network could be monitored by this library that contains entry points for that purpose and collects the desired packets. If the interface is set in the promiscuous mode, all packets would be collected included the host packets. The raw packet data is delivered using libpcap library to a higher software that is responsible for analysing packet header fields and interpreting protocols. This tool provides different tasks over capturing and presenting statistical information of packets such as debugging and troubleshooting issues. Capturing packets in the local area network (LAN) is easier from capturing within links of the Internet as the state becomes more complicated. With higher data bit rate, higher traffic is aggregated with more diversity and volume, hence, special requirements should be met for such data collection [64].

2.4.2 **Flows**

Packet levels are required to identify applications; however, a preferred approach and input would be traffic summarization. Instead of processing and storing information about individual packets, analysis may focus on packets transferred between endpoints that share the same attributes. This term is called a flow, where packets have the same source and destination addresses, source and destination ports, and protocol [65]. The information in the flow is valuable as it presents some traffic characteristics as follows:

- The source address shows who is initializing the traffic.
- The destination address displays who is receiving traffic.
- Ports refer to some protocols such as http or https as port numbers are 80 or 443.
- Priority of traffic could be examined by the class of service.
- Flow timestamps that show flow life.
- TCP handshakes flags.

From an application perspective, a flow can be defined as packets exchange between a sending application and receiving application. Labelling a packet that belongs to an application leads to label all packets in the flow consequently, this mechanism speeds up the classification process in high link networks and requires no additional resources. The IP flow could be collected at a various level, it might be collected by the port number, protocol type, IP address or combination of these attributes. For instance, VoIP applications have two protocols, H.323 that is setting up a call and RTP that is carrying the voice data. Marking the H.323 flow leads to tag all RTP/RTCP flows that share the same source IP and destination IP [66]. In recent years, the researchers and operators have used flow-based techniques in different complex applications such as management of resources, traffic classification, and intrusion detection rather than simple diagnosing and accounting. They are carried out easy, scalable as well as their wide availability in existing hardware using standardized export formats such as *NetFlow*. **Capturing flows:** reducing the volume of data traffic that requires more resources is the key to collect and manage packets in high-speed links. Different methods have been emerged to achieve data reduction. Simple Network Management Protocol (SNMP) is the most common approaches that use counter methods for data reduction, and the collection of flow records [67]. The first type is based on counting bytes or packets with time series, this approach is applicable in all routers. However, there are problems in data collection that utilized SNMP, the first problem regarding packet loss as the approach uses UDP for transmission. Secondly, losing synchronization in the time series through different network interfaces as the polling used for data collection. Similarly, sFlow which is a protocol that used in high-speed monitoring as it selects one packet for every sampling rate and gathering the total size of all the selected packets and send them using UDP to the collector. In spite of the SNMP and sFlow protocols provides critical information about the bandwidth and how it is being utilized by the IP network, the operators cannot rely on this tool to characterize Internet applications and patterns which is important in business thrive. The most powerful approach than counters, which shows network activities in simple form with losing important traffic characteristics, is capturing data traffic via packet trains or flows, which provides valuable information to the ISPs and in data analysis field. The concept of packet train was first introduced in [62] and it provided summary information about the Internet traffic that used for uncovering basic network activities, applications and users monitoring, network design, and security analysis. The packet trains can be captured using the tools that are embedded in the main routers. The drawback in the packet trains is the difficulty of determining a general definition for the end time of the packet train. There are different criteria to determine this time, either by setting time out the threshold for the inter-arrival packets or by the whole flow or by observation the FIN or RST packet [68].

- **Tcptrace:** Capturing packets which share the same 5-tuples within time period called flows as presented early where most routers and switches nowadays export flows in the form of NetFlow. *NetFlow* represents the de-facto standard in monitoring and analysing network traffic that invented by Cisco and embedded in their equipment [69]. However, the information it provides is not enough to describe the application behaviour. Modifying this tool is not applicable as it is owned by cisco, to this effect, the present work customised the popular *tcptrace* utility to generate classification features based on traffic burstiness and periods of inactivity (idle time) for everyday Internet usage. The collected Internet traffic can be analysed using the *tcptrace* tool [70], developed at Ohio University and is widely considered a useful tool for identifying network flows [71]. The *tcptrace* utility segregates traffic sent between client and server and vice-versa while other tools such as Wireshark group the sent and received traffic in a single stream [72]. It is used specifically to analysis TCP connections by filtering dump files from tcpdump as input and output summary report with the separated flow. The research community has previously used *tcptrace* to extract a lot of features to classify Internet traffic as well as for intrusion detection [73, 74]. Features were extracted from *tcptrace* tool directly by making some modification inside this tool to generate more features, or indirectly by writing an external script based on features taken from the tool. In [58][71], they used the same attributes which were focusing on flow-group and time occupancy. Flow-group is generated during the first few seconds of communication and based on the same IP address, while time occupancy depends on the ratio of flow duration over the entire duration. The state of occupancy could be high when the data transfers continuously while the state could be low when the data transfer occurs in a short duration of time chunks. The main drawbacks in capturing trains of packets and flows are the absence of the inter-arrival packet time and the difficulty of determining precise time-scale[68].

2.4.3 Challenges

The previous sections presented clearly the massive volume and complex properties of traffic that existing in the computer networks. The development of the Internet from single backbone before 1995 to enormous interconnections nowadays make the difficulty to determine a network that explains a global view to the entire Internet traffic [60]. Therefore measuring and characterizing such traffic can be challenging for engineers and researchers. There are different traffic characteristics in different networks, in other words, the properties of local point in some network might not be the same at another network. The traffic attributes seem to be not the same at home network, university campus, backbone network, and access network. Moreover, the packets that are passing through the physical layer could be affected by corruption, delay and loss that are not seen in the network layer. Capturing packets at high-speed links is a challenge as such links produce hundreds of megabytes per second that make the data processing, storing and managing very difficult. The suitable case is to capture packets for a short time or summaries these packets in the flow form or capturing process would be exclusive for only packet header. Moreover, traffic collection could contain sensitive information for both users and ISPs as capturing full packets could reveal different user activities such as passwords, visiting websites and emails. ISPs could display information about the network such as customers, interconnection points, network peers, and policies which regard important information for the competitors.

2.5 Statistical modelling

Statistics is a mathematical technique that deals with a numerical data from collection, analysis, interpretation, presentation, and organization. It could be classified into two parts:

1. **Descriptive statistics:** it refers to the initial presentation of the data in a meaningful manner especially with a lot of data with basic calculation such as mean, median, and standard deviation to show information about a group of data. These statistics are a measure of central tendency or variability and as follows:
    - **The measure of central tendency:** This measure summaries statistics for a feature to display how the distribution of the values around the middle. The most frequently measures that are used in determining the central tendency of the data are mean and median. These are very simple arithmetic that calculates the average and midpoint of the data respectively, they are powerful as they are very sensitive to the outliers in the data. The outliers usually have high or low values in the feature that deviate from other values, pre-processing such outliers is very important to avoid overfitting in the classifier.
    - **The measure of variability:** The variability measures the dispersion in a feature value and displays how the distribution of the data is spread out which is an opposite concept for the measure of central tendency. The feature values are more consistent when variability is low, while with high variability, the values are farther from others. The most common measures of the variability are range and stranded deviation, the range is the difference between two

extremist values and become useful when the size of the sample is small. In our work, the range was divided into two separate measures (i.e., maximum and minimum), these two measures were calculated for each feature. While the standard deviation is the difference between each value in the feature and the mean value for that feature, higher standard deviation means the feature values more spread out, while when the data are closer from the mean, the standard deviation is lower.

2. **Inferential statistics:** these techniques deal with a subset of entire data and draw conclusions based on hypothesis testing, estimation of the parameters, and their correlation within data. This type of analysis reveals the hidden information of the relationship between the numerical characteristics that cannot obtain with the machine learning techniques. These statistics can be test them using the following parameters:

- **Hypothesis testing:** this is the procedure of carrying out some statistical tests on a sample of data to draw conclusions about the overall population. There are two hypotheses (null and alternate) which test the validity of our assumption for statistically significant or not.
    1. Null hypothesis: this hypothesis assumes that there is not a difference or a significance in the sample and it is always homogeneous.
    2. Alternative hypothesis: if there is a difference or a significance in the sample, the null hypothesis will be rejected based on P-value.
- **P-value:** after proceeding a hypothesis test in data, the P-value measures the significance of the results.
    - If the P-value less than 0.05, the null hypothesis will reject.

- If the P-value greater than 0.05, the null hypothesis is true.

To summarise the process of hypothesis testing, firstly, the null hypothesis would be considered, secondly, collect the data and compute the test statistics. Finally, the null hypothesis is rejected or accepted based on the P-value.

2.6 Classification

The early sections have presented the amount of traffic that could be generated from the Internet applications which is considered as big data. For a review, a one trillion web pages or more is available on the Internet; every one-second new video is uploaded to the YouTube, and over 20PB of information are processed by Google every day [75][76]. Powerful techniques and algorithms are needed for analysing this data, machine learning techniques have been proposed as an essential tool for this problem. This section introduces the most effective machine learning algorithms that were used in this work.

2.6.1 Decision Tree

A decision tree is a tree that a feature is represented by a node. A decision (rule) is represented by a link and an outcome is represented by a leaf. The classes are split in each level by recursive binary splitting to identify records with the purest class. The problem in the decision tree is the overfitting due to high variance in estimating each single data point, this makes the algorithm unreliable with the presence of noisy data. This problem was solved by using Bagging algorithm, which is an ensemble technique deployed on decision trees. The technique divides the samples into subsample of records and for all features, subsequently, applying decision tree individually for each subsample and later ensemble the results by choosing the ultimate vote. C5.0 and random forest are the most powerful techniques that are used in this filed [77].

- **C5.0:** The decision tree could be generated by several algorithms, but the C5.0 algorithm, an improved version of the earlier C4.5, is more well-known [78]. The source code of this algorithm was made publically available and also incorporated into data analysis tools such as R programming language. Furthermore, the decision trees implemented by C5.0 algorithm are quite robust and are easy to deploy and understand. Supervised C5.0 also performs better than other algorithms such as Neural Network and Support Vector Machine [37]. The advantage and disadvantages of C5.0 could be summarised in Table 2-2. C5.0 is accurate and needs lower time in execution compared with other ML methods. Several techniques have been added to this algorithm such as boosting.

Table 2-2: Pros and Cons of C5.0 Algorithm

Advantages	**Disadvantages**
• Doing well for all purposes • High automatic processing to specify the nominal features • Choose only the most important features • Can be implemented using small training data, and more powerful from other complex algorithms	• Splitting features that having a large number of levels • It is easy to overfit a model • Sensible to changes in the training data

In decision trees, the first challenging task is to recognize which parameters to split data upon. C5.0 uses entropy to measure the segments of data that includes only a single class. The entropy of a sample of data refers to how the class values are mixed. If the entropy is equal to 0, that means the sample of data is completely homogenous, while, if it is 1, that means the segment of the data is non-homogenous. The drawback of decision trees is that these grow continuously when features are splitting and divided into smaller and smaller partitions until the classifier finished or run out of features. This problem could affect the training data. C5.0 algorithm has an attribute of pruning which reduces growth in C5.0.

- **Random forest:** The concept of this algorithm is the same as in bagging algorithm except that the random forest merges multiple tree decisions to obtain more accurate prediction. Moreover, bagging algorithm was developed more using random forest by adding more randomness to the model and searching about the best features within a random subset of features that lead to low bias and low variance [79].

2.6.2 **Boosting**

The boosting refers to algorithms that apply weak classifiers to build a strong classifier by combining the results. The algorithm gives all records the same weight and applies a sequence of iterations of classification; the misclassified records increase their weight, while the weight of the right classified records is reduced. Finally, a strong classifier is created from incorporating the individual ones with the best tuning for the parameters to avoid overfitting [80] [81]. There are many algorithms for boosting such as AdaBoost and Gradient Tree boosting

- **AdaBoost:** in bagging classifier, a bootstrap method is applied to the training data through a parallel process as each sample treats independently. In contrast, boosting does not use the bootstrap sampling as the method works sequentially, each tree depends on the previously treated tree until reach a strong classifier.
- **Gradient Boosting:** gradient Boosting is a machine learning technique that is used for solving problems in regression and classification. The concept of this algorithm is similar to the AdaBoost algorithm that gives higher

weight to the weak learners but it uses gradients in the loss function for best fitting of the miss-classified samples.

2.6.3 Cross-validation

The common approach of modelling is by dividing a data into two parts, one for training a model and the other for evaluating it. The disadvantage of such an approach is missing out some important information of the data leads to low prediction performance. Cross validation is statistical method that divide data into equal folds, one fold used for validation the model, and the others used for training it. Each round, a different fold is used for validation until all folds are cycled through. This technique is used to evaluate the performance of machine learning model by testing the model on unseen data to avoid overfitting and underfitting problems.

2.6.4 Confusion matrix

The confusion matrix is a table that categorizes predictions according to whether they match the actual value in the data. When the predicted value is the same as the actual value, this is a correct classification, correct prediction falls on the diagonal in the confusion matrix [82]. The model's ability depends on its performance to recognize one class from others. The class of interest is known as the positive class, while all others are known as negative. The relation between positive class and negative class predictions can be depicted as a 2*2 confusion matrix in Table 2.3 that tabulates whether the obtained prediction falls into one of four categories:

- True positive (TP): correctly classified as the class of interest
- True negative (TN): correctly classified as not the class of interest
- False positive (FP): incorrectly classified as the class of interest
- False negative (FN): incorrectly classified as not the class of interest

Table 2-3: confusion matrix

		Predicted data	
		No	Yes
Actual data	No	TN	FP
	Yes	FN	TP

Various measures, such as error-rate, accuracy, specificity, sensitivity, and precision, are derived from the confusion matrix.

- **Accuracy:** the accuracy is the proportion of true positive and true negative divided by a total number of predictions. The best accuracy is 1, whereas the worst is 0. With the 2*2 confusion matrix, the formula of prediction accuracy is shown in Eq. 2.1

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \dots\dots\dots\dots \quad (2.1)$$

- **Error rate:** Error rate (ERR) is calculated as the number of all incorrect predictions divided by the total number of the dataset. The best error rate is 0, whereas the worst is 1. With the 2*2 confusion matrix, the formula of the prediction error rate is shown in Eq. 2.2

$$Error\ Rate = \frac{FP + FN}{TP + TN + FP + FN} \quad \dots\dots\dots\dots \quad (2.2)$$

Similarly in Eq. 2.3

$$error\ rate = 1 - accuracy \quad \dots\dots\dots\dots\dots \quad (2.3)$$

- **Sensitivity**: The sensitivity (Recall or True positive rate) measures the proportion of positive examples that correctly classified; its formula is in Eq. 2.4

$$sensitivity = \frac{TP}{TP + FN} \quad \dots\dots\dots\dots\dots \; (2.4)$$

- **Specificity:** measures the proportion of negative examples that correctly classified, and its formula is as in Eq. 2.5

$$specificity = \frac{TN}{TN + FP} \quad \dots\dots\dots\dots\dots \; (2.5)$$

2.6.5 **Entropy, and Bias versus variance**

The entropy could be defined in physics and in communication theory. Generally, it refers to a process in which a randomness increases with the time. Naturally, the universe evolves into a highest entropy, for example, the differences in the thermal lead to disappear. Accordingly, the temperature will be uniform for everything in the universe. In data communication, the entropy means the randomness degree, errors are frequently being signalled with higher entropy. Also, the entropy is a measure of impurity or the randomness in the data being processed, the entropy is zero when the sample is homogeneous. A higher value for the entropy means more heterogeneous in the sample with more difficulty to describe the data, until the value becomes 1, the sample becomes most heterogeneousness.

A bias is a measure that compere between the prediction values of a model and the actual values in order to assess the bias. By repeating the process of the model building more than one time, different predictions will be generated for the model because of the randomness in different data sets. The bias is high when the actual values are far off from the predicted values and it indicates that the model is too simple to deal with the complexity of the data and causes under fitting. For example, linear regression that based on an assumption that the target has linear relationship with features.

A variance measures how the predicted value is scattered from the actual value. High variance means that the model is very flexible for training data points, which gives them a lot of attention but does error rates on testing data; therefore, overfitting is caused by high variance. For example, in supervised learning, when a model try to capture the noise in the data points, overfitting is caused.

A trade-off is important between variance and bias without overfitting or under fitting the data. Under fitting is caused by high bias when a model is too simple and has few features. On the other hand, the overfitting problem in the model caused by high variance when the model becomes more sensitive to any small change in the training data.

2.7 Conclusions

This chapter presented the amount of traffic that is transferred in Internet's infrastructure nowadays that provides a transmission of a huge data due to being accessed by billions of users for different applications. The development of the Internet from single backbone before 1995 to enormous interconnections nowadays make the difficulty to determine a network that gives a general view to the entire Internet traffic. Moreover, the applications need various requirements such as availability of resources and response time due to an enormous number of users access them over the Internet. Characterising such a traffic is essential for monitoring, service discovery, routing control, and resource optimisation. However, capturing packets at high-speed links is a challenge as such links produce hundreds of megabytes per second that make the data processing, storing and managing very difficult. Using flow measurements as an alternative to packet traces for traffic classification have gained momentum due to a dealing with less amount of data and avoid the encryption with preserving user’s privacy. Data collection using flow-based relies on packet headers that summaries traffic characteristic. Next chapter introduces various methods for traffic classification.

3 Application identification - existing methods and limitations

3.1 Introduction

As highlighted by the previous chapter, the proliferation of end users along with the advanced technologies of wireless connectivity, and the growing in the number of web applications have produced a complex Internet topology. Managing such a complex configuration with a huge amount of traffic is more challenging that impact negatively on both QoS and the QoE. This work is focusing on monitoring this traffic and proposed reliable approach of traffic classification that can cope with real time usage. A number of studies [16, 83, 92, 84-91] proposed models for traffic classification, with many thorough surveys [7, 93-97] that determined the most important requirements and recommendations for a successful approach. The proposed alternatives could be categorised into four techniques: port-based, packet payload based, host behavioural, and statistical-based. The traditional way to identifying traffic flows typically focuses on using IANA assigned port numbers and deep packet inspection (DPI) [35, 36]. However, an increasing number of Internet applications nowadays that frequently use dynamic post assignments and tunnelling which renders port-based traffic classification extremely challenging and prone to errors. DPI is useful, but it requires significant computational resources, presenting scalability issues in achieving real-time traffic identification, and cannot cope with the encrypted traffic.[4, 7]. In recent years, two other techniques have been introduced, focusing on host behaviour and statistical methods, to avoid these limitations. The former technique is based on the idea that hosts generate different communication patterns at the transport layer; by extracting these behavioural patterns, activities and applications can be classified. Although the method showed acceptable performance (over 90%) [9] and it can detect the application type, however, it cannot correctly identify the application names, classifying both Yahoo or Gmail as email [98]. Thereby, studies have focused on using statistical features approach for identifying traffic associated with applications based on machine learning algorithms [99]. This method relies on characteristics of IP flows such as a number of packets in a flow, size and duration of a flow which reflect unique patterns for applications. The aforementioned method considered flexible for emerging traffic as it utilizes network level (packet header) with promising results rather than application level (packet contents) [100]. Moreover, this method is less influenced by the DPI when the traffic is encrypted and it does not touch the user's privacy; consequentially, recent efforts have been put in this approach [101]. In the following sections, an extensive study for methods that were used in traffic classification, display advantages and disadvantages of each method; discussion and conclusion end this chapter.

3.2 Port-Based Technique

Historically, the first approach of traffic classification is port-based, using the transport layer port number. Port numbers in the range of (0-1023) are the well-known ports and assigned to popular services by IANA [35] such as port 25 for SMTP and port 80 for HTTP, while the port range numbers from 1024 to 49151 are registered for specific services. On the other hand, the range from 49152 to 65535 contains dynamic or private ports that are unregistered and utilised for private or customized services and temporary communication purposes using dynamic and ephemeral allocation. Port-based classification is simple and yields highly accuracy for certain applications such as SMTP or DNS that use specific (static) port numbers. However, most of the present Internet applications use dynamic port numbers[102]. Some applications also use encryption and tunnel traffic through well-known port numbers such as HTTP or HTTPS. Furthermore, firewalls, address translation, port forwarding and tunnelling make it quite difficult to match a service with a particular port [10]. As such port-based traffic classification is now considered ineffective showing not more than 70% accuracy when tested against other available methods [103]. In spite of providing low classification accuracy, port-based traffic identification is still relevant in Internet backbone due to the scalability of use and relatively minimum computational power required [98]. In brief, port-based classification aids in determining the tendency of overall application trends when combined with additional techniques resulting in hybrid approaches. Many recent studies, therefore, combine port-based classification with machine learning and statistical analysis of network traffic resulting in higher accuracy, discussed later in this chapter. To overcome this limitation, deep packet inspection (DPI) method became the preferred solution.

3.3 Deep Packet Inspection (DPI) Technique

It is argued that the low accuracy associated with the port-based method can be solved using DPI. This approach includes not only the inspection of packet headers but also the packet's payload traversing the network. The evolution of DPI started by recording the signatures of each application or protocol format (manually) using reverse engineering or vendor white papers describing the behavioural of applications. In [104], DPI was used to classify P2P applications; they produced signatures for each P2P application according to the available documentation and analysing packet traces. The recorded signatures were subsequently used in designing filters to identify P2P applications in real-time traffic. The authors chose five P2P applications to test the filters and the results showed that the ratio of false negatives and false positives was less than 5%.

Moreover, the study claimed that the technique could classify P2P applications by examining only a few packets which makes the approach more scalable for high-speed analysis. To avoid the manual efforts, application signatures were extracted from the payload contents of IP traffic using three machine learning algorithms [105]. The proposed method was evaluated by collecting 100GB of data traffic from 500 customers to identify seven applications (i.e., FTP control, SMTP, POP3, IMAP, HTTPS, HTTP, and SSH). The results showed high accuracy up to 99% with the ability to work in real-time environments. Although this method achieved high accuracy, one of the obvious limitations is the requirement of high processing power while dealing with a huge amount of data and requires prior knowledge about application signatures. Moreover, DPI-based approaches cannot identify encrypted traffic or proprietary protocols. Additionally, due to privacy concerns, the analysis of data and information at the application layer may be deemed illegal because it may reveal personal information. The research community, therefore, proposed new techniques regarding traffic classification that are more promising are shown in the next sections.

3.4 Host Behavioral Techniques

This techniques are based on the idea that hosts generate different communication patterns at the transport layer; by extracting these behavioural patterns, activities and applications can be classified according to these patterns. The success of this method relies on parameters that should be collected and analysed from different flows as this method based on end-point activity such as number of connected hosts, time frame and protocol type. In 2004, [106] proposed two heuristics to identify P2P applications (source-destination IP pairs and IP-port pair). They utilised the payload approach for identifying nine P2P applications by doing reverse engineering and analysing these applications. In

2006 [13] put six rules that described precisely the behaviour of 10 P2P applications with high accuracy. The same heuristics used in previous studies were utilized in [14] with only 0.2% of data remained unclassified from large real traffic. At the same time and in 2005, [90] introduced a new technique named BLINK that analysed and identified the connection patterns of host behaviour based on three levels (i.e., social level, network level, and the application level). The proposed features proved high accuracy in classifying different types of traffic by more than 95%. Other studies [11, 12] proposed heuristics to identify whether the hosts use P2P applications or not. Authors in [11] introduced only features such as the ratio of number of ports used to the number of IPs connected to by the host, and a number of failed connections to explore the P2P traffic. Hurley in [12] proposed four semantics (source and destination host, further connections between hosts, and flow activity). They claimed that about 90% of web and P2P flows could be identified with misclassification less than 2.86% of flows for P2P and 0.54% for the web. While authors in [15] studied the effectiveness of correlation information in the multiple flows to classify P2P applications (such as Skype, Thunder, and PPTV). They proposed a novel set of features vector that showed high accuracy to identify the known P2P applications over 90%. Other studies [9] [107] tried to identify one application, for instance, the authors in [9] set three heuristics to describe Bit Torrent application based on any *NetFlow* record that is provided by Cisco routers. They designed a traffic classification model based on the selected features with high accuracy above 92% to discriminate the Bit Torrent from mixed real Internet traffic. Similarly, in [107], the authors studied semantics that describe the application behaviour (Google Hangout) and extracted suitable features set to design a classification model. Naive Base, decision tree and AdaBoost were used to classify data collected and their findings were that the accuracy increased as new classes were added. The authors used recall as a metric for evaluation and the results for the three algorithms were 99.98%, 100%, and 100% respectively. However, the experiments were carried out only to identify Google services. Recently, [10] studied the mechanism of data exchange for two protocols (TCP and UDP) at the end host to classify Internet traffic. The study collected P2P application traffic (eMule, FrostWire, Skype, µ Torrent and Vuze) as well as non-P2P traffic (Web, Dropbox LAN, FTP and SMTP) using the Wireshark tool. The heuristics used included the port number, port pairs, unique IP addresses and TCP to UDP protocol percentage. The results showed that only 0.2% of classified traffic remained unknown. The study assumed that any peer that utilized port 80 was using non-P2P applications. This assumption may lead to misclassification when applied in different network environments due to the fact that most P2P applications masquerade their ports using well-known ports (like port 80) to avoid detection [108]. Although the method showed acceptable performance (over 90%) with low resources compared to payload methods and it can detect the application type, it cannot correctly identify the application name, classifying both Yahoo or Gmail as email [98] [9]. Moreover, this technique as shown from the previous studies is primary used to identify P2P applications with high accuracy as the approach relies on the connection patterns that are generated from the peers.

3.5 Machine-Learning-Based approaches

Whilst the previous methods have limitations in terms of application identifications, recent studies focused on employing a statistical approach that can characterise traffic associated with an application based upon statistics and information theory. This approach does not rely on the contents of the packet and can potentially profile encrypted traffic [99]. Moreover, this method utilises flow measurements which become available in most network devices that provide traffic accounting solution in low cost [28]. The solution assumes that each application has unique statistical characteristics that could be extracted from the collected data. Usually, statistical approaches utilise machine-learning algorithms (MLAs) to identify the patterns in the communication and attempt to link them to specific applications [18, 28, 109, 110]. A huge academic effort has been concentrated on recruiting the MLAs in classifying Internet traffic based on statistical method [97]. High accuracy was achieved (over 95%) by applying these techniques [16, 17, 26]. The advantage of using ML algorithms is that they can be used in a real time environment that provides rapid application detection with high accuracy. Machine learning based techniques could be divided into three categories depending on the type of algorithms used. These techniques include supervised, unsupervised and semi-supervised learning. Each of the classification techniques is discussed as follows.

a) Supervised learning techniques

In supervised learning, a type of the traffic that needs to be classified needs to be labelled to produce a ground truth or training data. This data represents the signatures of the application that is used to build a classification model. This method is powerful, and it has high accuracy, but it depends on the quality of ground truth (training data), however, it cannot identify new applications [111]. A number of prior studies have used supervised learning techniques in tandem with flow records to classify traffic.

Some studies [19, 112] [119] used one algorithm (i.e., support vector machine (SVM)) to classify traffic. These studies utilized a flow of packets that are transferred in each direction as statistical features such as packet size and number of packets. Three data sets were applied to evaluate the SVM classifier such as UNIBS set (private), LBNL and CAIDA (public) with accuracy over 90%. For instance, Hong et. al. in [19] used SVM algorithm to identify Seven classes (Mail, FTP, Database, Multimedia, P2P and WWW) based on statistical information that were extracted from *NetFlow* records. The results showed that 99% of web traffic could be identified correctly; however, each class needed different type of SVM algorithm to identify. For instances, (database, FTP and P2P traffic) could be identified by using SVM-4 rather than other SVMs. Mail traffic could be classified with more precision by using SVM-3 and Multimedia traffic with SVM-5. Although the proposed scheme achieved high accuracy, it can only identify a traffic class. For further accuracy, the authors in [21, 113, 114] suggested a framework that consists of many algorithms. In [21], the authors selected a series of simple linear binary classifiers to characterize a real data traffic that was collected from different ISP locations; the combination showed promising results. Similarly, in [113], the authors applied seven classifiers (i.e., NBTree, PART, J48, Bayes Net, Bayes, kernel, and SVM) to identify different levels of real data traffic from local to the wide area network. They argued that each dataset was classified correctly based on different classifiers as each network has features which could be different from other networks. Therefore, they concluded that a need for a framework that contains many algorithms is essential. However, using more classifiers in traffic classification, enlarge the framework and increases the complexity of the scheme. Therefore, studies such as [115-117] made comparison between different machine learning algorithms. For example, [116] proposed six ML algorithms (i.e., AdaBoost, Support Vector Machine, Naive Bayesian, RIPPER and C4.5) to identify SSH and Skype traffic. The authors used basic attributes such as size of packets in each direction and inter-arrival time. Also, in [117], the authors tried to identify the SSH protocol using 46 statistical features. Three datasets were used to evaluate three algorithms of machine learning (i.e., C4.5, k-means and Multi-Objective Genetic Algorithm (MOGA)). Both studies showed that the results of the C4.5 classifier accomplished the best accuracy.

Based on the success of the C4.5 classifier, recent studies such as [29, 37, 117] utilized C5.0, which is a developed version of C4.5 for traffic classification. [118] identified HTTP traffic from non-HTTP traffic with an accuracy of 94%; the most features that were used by the classifier were payload size and number of PSH flags to the client direction. The same authors in [37] used the same classifier to identify seven applications (i.e., web browser traffic, Skype, torrent, interactive gaming and SSH, FTP, web radio) with high accuracy over 99% and with different statistics of basic attributes. In 2018 [29], the authors used C5.0 to identify modern applications such as Facebook and Google services using the very first packets and they achieved high accuracy reached up to 98%. The selected classifier (i.e. C5.0) outdo other methods such as Naïve Bayes and K-NN. These studies achieved high accuracy as they identified only traffic class such as email and video streaming or protocols such as http and FTP. Other studies, such as [91, 119, 120] proposed a transfer learning as an alternative to a traditional assumption of classical machine learning, which both training and testing data belong to the same source. In [91], they claimed that the data distribution would be changed with different time, location and traffic types. Therefore, they trained different data from different data sources and made a transfer of knowledge from a target model to a source model. They argued that high classification accuracy was accomplished by the proposed method based on the same features by just changing the statistics operations.

A new technique was deployed in recent studies [121-123] that describe the behavioural of an application based on the packets or messages exchanged between client and server. [121] proposed a new approach of classification named as SLFC (session level flow classification) that groups traffic flows into sessions to represent the behavioural of the application. The proposed design consists of two parts, flow classification and flow grouping classification. The first part identified an application based on the packet size distribution (PSD) of each flow and compared individual flows to pre-applications. The second part classifies network flows into groups using port locality; the authors claimed that the operating system generates similar port numbers for the same application within a short time. The method achieved high accurate results about 98%; however, the execution time for the method could be slow as the decision relies on inspecting 300 packets. Therefore, the same authors in [122] proposed a new approach that could be suitable to the real-time, named message size sequence classifier (MSSC) that could make a decision by inspecting only 15 packets. This approach depends on the exchanged packets between client and server that derive a sequence based on the directions and sizes of these packets. The traffic flows were classified by comparing the message size sequences (MSSes) of each flow with pre-labelled applications to determine which application is related to a flow. Similarly, Hajjar *et. al.* in [123], proposed an identification model which depends on using first messages of application-layer by utilizing flow size, direction and position of respective messages in the flows. The study argued that the first messages of each application have sufficiently discriminating control information. Some other studies, such as [124, 125], argued that the message size remains very important in classification traffic flows. However, applications that have the same statistical attributes due to the similarity in their protocols are quite difficult to identify.

b) Unsupervised learning techniques

In unsupervised algorithms, the traffic classes are categorised based on the similarity of the objects. This method does not need prior knowledge of the classes; therefore, it is able to explore new applications without any training data. Well-known methods were used in traffic classification such as Auto Class [126], k-means [127], DBSCAN [84] and fuzzy C-means [128]. For instance, Zander et al. [126] used Auto Class approach (i.e., unsupervised Bayesian classifier) to group traffic flows based on statistical features. These features were mean and variance of packets length, size of each direction, flow duration and mean of inter-arrival time. The authors used a feature selection method based on machine learning to determine the optimal features set. These features were evaluated using datasets collected form traffic traces and from different Internet locations with average accuracy reached up to 86.5%. Erman *et al.* [84] Utilized k-means, DBSCAN and Auto class algorithms to group traffic flow for two data traces. The authors used characteristics based mainly on the previous work Zander et al. [126]. The authors claimed that the accuracy of clustering increased when the number of clusters were more than the number of classes. McGregor [129] proposed using expectation maximizing (EM) algorithm to create clusters for the traffic flows and labelled them manually. New features were added to the proposed system such as the bulk of data transferred and idle time. The authors defined the bulk when more than three successive packets are transferred in one direction, while the idle time was defined when no packets are transferred within 2 seconds. The problem in clustering methods is how to set the number of clusters without any information about the real applications. Moreover, previous work [84, 126, 127, 129-133] showed that using traditional clustering algorithms led to low accuracy cause of the produced clusters usually are not equivalent to the application classes. The flows of specific applications often spread within clusters or the cluster includes different flows of an application. Therefore, other studies [134, 135] used K-means for grouping the unlabelled traffic and utilized payload analysis tool for labelling traffic to avoid using supervised training data.

c) Semi-supervised techniques

Semi-supervised learning algorithms utilize both labelled and unlabelled data and these respective techniques have taken more attention in last decade. In several data collection conditions, labelled data samples are expensive to obtain or limited; however, unlabelled samples are easy to collect making the combination of limited labelled data with unlabelled records for effective classifier learning [136]. The aim of using such approach is to detect zero-day applications, many studies followed this approach such as [26, 137-140]. For instances, Erman's in [137] proposed combining supervised training data set with unsupervised technique by training a few known samples with many unknown samples and they achieved high accuracy greater than 90%. Flows would be labelled based on the nearest of predefined cluster, while other flows identify as unknown. Also, Vlăduţu et al. [26] proposed an automatic scheme to detect zero-day traffic by clustering traffic flows using k-means based on statistical features of unidirectional and bidirectional flows. Secondly, these clusters used to train supervised classier C4.5 to determine the new or unseen flows with accuracy over 90%. The study classified protocols such as HTTP or SSH. These studies used statistical features that described flows as individual (i.e. duration and size of the flow, the total number of packets in flow, size of packet and inter-arrival time).

In contrast, several studies [20, 24, 100, 141-143] used a heuristic of three tuples (destination IP, destination port and protocol) for flows during a certain period of time. They claimed that flows that sharing these tuples belong to the same application. Zhang et al. [20] utilized these tuples with features (i.e., total number of packets within flow, size of flow, and the Min, Max, mean and standard deviation for packet size and inter-arrival time). Many experiments were implemented on two data sets and the results revealed improvement even when the training samples were few. The same authors in [24] used the same features that utilized supervised and unsupervised machine learning algorithms to detect zero-day applications. They mixed the labelled and unlabelled samples and utilized the k-means clustering method to divide the traffic flows into k clusters. Zero-day application flows represented the cluster that not carry any predefined labels, while the other unknown flows were classified by the nearest to the labelled cluster. These flows used to train random forest classifier and extracted the zero-day flows in the test stage. The results showed significant improvement in the accuracy compared with other classifiers. However, using cluster analysis to label flows for generating training data caused error in identification traffic [133]. Although these studies achieved good results in classification traffic and detect new classes, they only classified network protocols such as FTP, HTTP, SSH, and SMTP or P2P applications such as BitTorrent and EDONKEY.

3.6 Hybrid Traffic Classification Techniques

Most recent studies [98, 125, 144] attempted to combine more than one method to obtain superior accuracy of up to 99%.

Park *et. al* in [98] proposed a new technique called functional separation method to classify traffic. The authors collected data from the end-hosts using a traffic collecting agent and the pre-processing stage sanitizes and separates applications from each other. Afterwards, the functional separation method partitioned each application according to their functions. The port-based method is used to group the application functions according to the port number similarity. In the other hand, payload-based and communication patterns were used for each group to check the inter-group application similarity. Finally, flow statistics were used per-group to discriminate the functionality in similar port numbers. The study used applications such as P2P, Web storage, messenger, video/music streaming and games for identification. Similarly, Lu and Xue in [144] utilized two approaches to identify Internet traffic (port and payload). The study used the co-clustering method and basic attributes (source/destination IP and destination port number) to characterize the host behaviour. The proposed technique first divided the flows into TCP and UDP and used the payload to classify all the flows into known and unknown traffic. These flows were later combined and the co-clustering method used to cluster the traffic into host communities using port numbers. Finally, each host community was clustered according to destination IP addresses. The experiment was performed using the data collected from a large scale ISP for two days, and the results showed that the accuracy of identifying applications on the first day was 100%, while the accuracy on the next day was about to 86% due to the similarity between applications. Furthermore, the authors discovered attack flows within known traffic which could be easily identified. The authors used the following features: protocol; the number of the packet; flow size; flow duration; Min & Max packet size; Max, Min & average packet arrival time; Min, Max & average payload size; the size of the 1st, 2nd , 3rd, 4th & 5th packet in the flow.

Yoon et. al [125] used the inter-flow relationships in application traffic to generate new signatures which are called behaviour signatures. The study claimed that this behaviour signature is unique for each application carrying out a particular task. The study included a combination of web-based activities and different network applications (Nateon, DropBox, UTorrent, Skype, Teamviewer, Youtube, Google, Facebook, Yahoo, and Wikipedia). The results showed that method precision was 100%, although the recall was low. The method identified encrypted traffic when it was compared with the payload. However, the inter-flow classification was based on the supposition that the single function generated plural flows. Changing this assumption renders the behaviour signature meaningless. The method can only identify the predefined applications and could not deal with zero-day applications seen for the first time. In addition, the study applied the proposed method in a specific network environment consisting of four hosts and at two different time-frames with implementation in a real-time mode not being evaluated. Nevertheless, these studies suffer from the complexity of analysis of using more than one approach.

3.7 Burstiness Based Approach

Selecting the right features represents a measure of the data quality that should be discriminative, informative and independent for building a robust classifier [92]. Given this classification, the statistical differences between inter-arrival times of packets and flows approach outlined in this work strengthens the behavioural and statistical methods by considering arrival times of packets and flows as discriminating features among applications. The authors in [145] proved that there is a variability (burstiness) in network traffic by using a measure called Index of Variability. The hypothesis of timing can be used to discriminate between applications was also put forward in [146], which claimed that applications generate different behaviour based on statistical features relating to the timing of packets arriving. More details about burstiness were proposed by [147] which defined in two levels. The first level was called a small time scale flight (STF) which means that the inter-arrival times of packets occur within a predefined time T (i.e., constant threshold and in the range of 5-10 milliseconds). The second level is a large time scale flight (LTF) and defined larger inter-arrival times of packets with value 40-1000 milliseconds. A different number of bursts would be generated for each definition based on the value of the threshold. Moreover, a study [148] defined a burstiness as a group of consecutive packets with shorter inter-arrival delays than the packets arriving before or after them. The study proposed that inter-arrival time ta (i.e., subtraction of the arriving time of the first bit of packet 2 from that of the last bit of packet 1) should be in the range (τ -d1, τ +d2) where τ is a predefined inter packet arrival time and (di) is the tolerance to form a burst. The burst is formed if the value of ta in the range (τ -d1, τ +d2), the minimum packets to create the burst are two. While the value of di should not exceed the value of τ where di ϵ (0, τ). Figure 3.1 shows how the group of packets forms a burst based on inter-packet arrival time and inactivity of time between bursts. This burstiness phenomenon could happen within packets or within flows. In this study, the burstiness concept will be defined on two levels, the first level is in the context of packet analysis and the second level is in the context of flow analysis.

The previous studies [145-148] defined the burstiness concept as explained earlier in the section, but they did not implement it. This work applied the burstiness definition using *tcptrace* tool by writing a script within its code (open source code) and expanded the concept to produce novel features.

It can be noticed that the statistical approach is appropriate for traffic classification as it can deal with encrypted traffic, which nowadays becomes the dominant, and it can adapt with real-time traffic. The Most studies in the literature put a heavy load on the MLAs to classify and identify Internet traffic, ignoring adding new features to describe more characteristics for traffic nowadays.

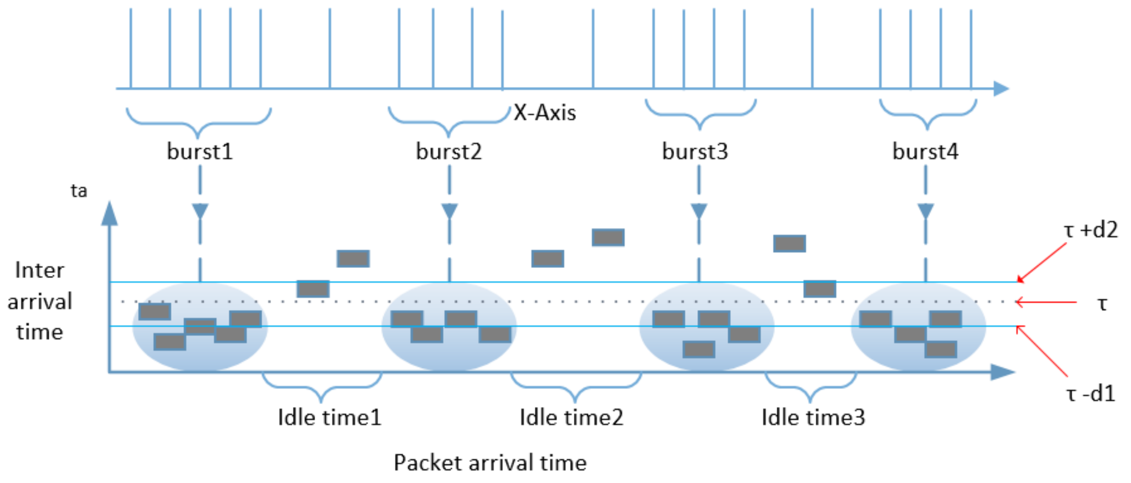


Figure 3-1: Definition of bursts and idle time based on [148]

Moreover, two surveys [96, 136] claimed in their final recommendations that traffic classification needs a multi-classifier model to overcome the limitations in the previous methods. This thesis therefore is seeking to introduce new attributes to give the researchers and administrators a better view to the modern traffic and utilizes the main important classifiers that were used in the literature.

3.8 Splitting Traffic Based On DNS Requests

Internet traffic can also be classified based on DNS inquires and IP address to reveal valuable information. The authors of [149, 150] focused on the volume and variety of DNS queries generated from both clients and servers, aiming to observe the effect of caching mechanisms on the client side. Other studies, such as [151, 152], exploited the DNS information to reveal malware activities. Further, the authors of [153] used DNS queries to classify traffic by matching keywords in the domain names table with the collected flows of traffic. These labelled flows were categorised based on domain name similarity, and the aim was to break down the traffic volume.

Using a similar scenario, [154] argued that traffic could be classified based on the IP address and hostname. Although the results showed that up to 55% of web traffic could be identified based on the proposed method, it also had a high accuracy in identifying applications such as WhatsApp, Twitter, and Dropbox. Based on the long-term monitoring, the authors concluded that the IP addresses of servers associated with each application remain stable for short periods, but they change over long periods. The study recommended updating and checking the IP addresses frequently for the methods that rely on IP address as a key feature. Similarly, the authors in [155] proposed a method to label websites based on server address. Firstly, they collected data from different users working on the same website to ensure that the server addresses belong to the same application, then they built a ground truth of IP addresses for specific applications and used them to classify a mix of traffic flows. The method showed good results when considering DNS queries. Following the same line of research, the authors in [110] used server addresses to group traffic applications to study the user activities. Authors of [156, 157] claimed that the IP address represents an informative feature. Similarly, [158] utilised DNS to tag flows by capturing a first packet of each flow and exploiting domain names which were separated into keywords to form vectors for each application. They claimed that DNS information could be useful to identify more than 30% of traffic. In [159, 160], the authors used DNS to label flows based on the keywords available after resolving IP addresses. Otherwise, the flows would be classified based on selected attributes and with the aid of machine learning to improve accuracy. The previous studies concluded that DNS information and IP address could be effective factors in classifying applications.

3.9 Discussion and Conclusion

The research community suggested four main approaches that have been used for characterizing Internet traffic and giving the administrators, ISPs and engineers a better view of what is happening in computer network. In the early days of the Internet, its applications were identified easily based upon only port number [3]. IANA [35] assigned protocols to well-known transport layer ports in which the identification process was merely based upon matching the port number in the packet header with the table containing the port-applications. Due to the continuous growth of Internet applications, they are no longer used standard ports; instead of, they have been moved towards a web-based front-end or used dynamic ports [4]. Consequently, this method becomes inaccurate in identifying applications and typical performance ranging between 30-70% [5]. A more accurate method is Deep packet inspection (DPI) [6] that relies on the contents of the packets to identify signatures of applications or protocols. This method became inefficient when most applications uses encryption methods; moreover, it breaches the privacy of the users and needs more computational resources [7, 8]. The research community has therefore introduced two techniques, focusing on host behaviour and statistical methods, to avoid these limitations. The former technique is based on how an application behaves depending on a variety of communication patterns at transport layer generated from this application. Despite the high accuracy of this method (over 90%) [9], it is unable to identify application name such as YouTube or Netflix while classifying them as streaming. In contrast, statistical approach outperforms the previous methods with high accuracy (over 95%) and it is widely used by the recent studies [16-18, 26]. This method uses packet header rather than payload information that makes the approach efficient even with encrypted traffic, and does not breach the user's privacy. Although the most studies have been considering that the early methods are inefficient, some recent studies utilized these methods in different scenarios by incorporating them in the most promising approaches as showed in section 3.6 (hybrid approaches).

From the literature, it is noticeable that most papers tried to do coarse classification. In other words, most studies identified either application classes such as streaming and browsing, or protocols such as HTTP and FTP, or P2P applications such as Bit Torrent and skype. Only one paper, [29], tried to do fine-grained classification in 2018 and identified the application type for modern applications such as Facebook and Google services; although the study identified another modern service (i.e. Google services),Google provides multiple services such as Gmail and Google search. It is also found that studies have applied a variety of traffic classification techniques; high accuracy has been performed using the statistical-based with employing machine-learning algorithms MLAs. Different techniques were used in this method from supervised to unsupervised and semi-supervised. In spite of the supervised approach outperforms the other techniques, building robust ground truth data for training a machine-learning model is required. Among the different supervised algorithms that were used by the research community [37, 112, 161-163], decision tree algorithms such as C4.5 and C5.0 were the best in classifying traffic. Recent studies [29, 37, 118] used the developed version of C4.5 (i.e.C5.0) to identify modern applications such as Facebook and Google services with very the first packets with high accuracy.

These studies employed a machine learning approach to classify Internet traffic based on recycling the conventional features. These features normally calculate data that transferres in the network or calculate arrival timing for packets, flows or session such as the total number of packets, number of bytes and inter-arrival time. These features are calculated statistically; as a result, they are subject to change due to continuous changes in the content of web pages. The features introduced in this thesis are based on timing between packets within the flow or between flows within the session based on burstiness and idle time. In other words, they are counting the activities of the user when he/she is browsing internet websites to represent the behaviour of the application. Although the user could have different behaviour each time, the data that are generating from the application would be the same. The next chapter explains in detail the proposed method and the novel features used within it.

4 Application identification based on burstiness

4.1 Introduction

The continuous developments of web applications render the early methods (i.e. port-based and DPI-based) unusable for detection as modern applications use dynamic ports and encrypted methods. On the other hand, the behavioural and statistical methods have been considered the most promising methods because they rely on packet header characteristics in classifying network traffic. Thereby, neither port numbers nor payload signatures would be used for an application identification. The success of these methods depends on using optimal machine learning algorithms and selecting suitable features. Whilst prior art focused upon using different machine-learning algorithms, little attention has been given for proposing innovative and superior features. Proposing new features should be accomplished carefully to sufficiently obtain discriminative features which precisely describe a web application behaviour in order to segregate various user activities. Extracting the most discriminative features, which characterise web applications, is a key to gain higher accuracy without being biased by either users or network circumstances. This chapter investigates novel and superior features that characterize a behaviour of an application based on timing of arrival packets and flows. To this end, the project exploited a concept of burstiness for new features generation, which defines closely spaced data exchanges, and idle periods, which separate longer-term transactions. These concepts are applied in two levels, packet analysis level and flow analysis level. Therefore, the following aims are addressed to be accomplished across the following chapters:

1. Proposing and identifying new features based on inter-arrival timing of packets and flows using burstiness and idle time concept.

2. Determining the ground truth dataset for investigating the proposed innovative features and for labelling a real traffic.
3. Investigating whether burstiness-based features are discriminant for identifying network applications based on the traffic that they exchange.
4. Investigating the efficiency of burstiness-based features versus traditional flow- and volume-based features for identifying network applications.
5. Investigating the unique behaviour of each application based on the proposed new features.
6. Determining the possible correlations (similarity) between input features in order to convert action of many variables with the same correlation to a small number of compound ones.
7. Investigating the minimum set of input features that maximizes the accuracy for output prediction.
8. Demonstrating that different users behaviors do not affect on the application behaviour.

In this chapter, a first aim is addressed by implementing a preliminary study to determine the feasibility of the proposed features.

4.2 Inter arrival timing, burstiness and features

The existing statistical parameters of the footprint generated by web applications such as packet size, flow size and duration, and inter-arrival time of packets are considered by previous studies [18, 28, 109, 110] . These features are calculated statistically; as a result, they are subject to change due to continuous changes in the content of web pages. The leading assumption is that different web applications will have different patterns and different behavior over time [164]. In other words, different applications have different distributions of timing within them due to the inherent behavior of the applications. In addition, the behavior of the human (users) might impact on the application behavior. The work aims to extract features that differentiate between web applications behavior while considering the user behavior using inter-arrival time between packets and flows with session. This work focuses in particular on burstiness that describes objects on the same web page, and idle periods that depicts different objects when a user is moving from one page to another. For instance, streaming a video on Netflix versus E-mail checking or using social media could lead to significantly different packet arrival patterns and hence a slightly different burstiness signature. The following example explains the concept of burstiness and how it may be used to discriminate the behavior of Internet applications. When a user is browsing an application, for instance the BBC news website (bbc.co.uk/news), the session would consist of some pages that the user chooses to visit. Within each page, the browser will be requesting and downloading the objects embedded in the page, some on the same site, some hosted on other sites. From a timing perspective, the download of objects on a page would appear as a burst of connections, followed by a period of inactivity (idle time) while a user reads the page until he/she decides to click on a link and load another page.

4.2.1 **Packet Analysis**

In this level of analysis (packet-based), the bursts and idle times would be formed based on the inter-arrival times for packets during the connection between client and server. This level was defined by [148] as a group of consecutive packets with shorter inter-arrival delays than the packets arriving before or after them. Given one of the two unidirectional data flows within a connection, a **burst_threshold (T)** is defined as a maximum time delay between the arrivals of two consecutive packets that belong to the same burst. In contrast, **idle_threshold (I)** is defined as the distance between groups of packets of inter-arrival time at which could identify the idle time that separates two consecutive data exchanges. In order to provide a meaningful description of the interactions, the analysis must establish the values for **T** and **I**, and whether they should be constant or dynamic. A previous study [147] defined two ranges for T that defined in two levels. The first level is small time scale flight (STF) which means that inter-arrival times of packets occur within a predefined time T (i.e., constant threshold and in the range of 5-10 milliseconds). The second level is large time scale flight (LTF) and defined larger inter-arrival times of packets with a value between (40-1000 milliseconds).

Another study [148] proposed two different scenarios for the value of T; the first one was dynamic which means different values could be for the T, while the second scenario was fixed without proposing any values for T, more details on this study provided in section 3.7. In order to obtain an image of the range of time values for the protocol interaction, Figure 4-1 shows the inter-packet arrival time for five applications. Most distributions of the inter-packet arrival time fall under 1 second, except for YouTube that falls under 0.5 seconds; accordingly, the burst_threshold could be set to 1 second. While the application does indeed exhibit a different signature in terms of packet arrival distribution, user behaviour may also influence this distribution, particularly in relation to long-term activity, as idle times are a factor of user behaviour too. The idle time could be varied according to the behaviour of the user when he/she moves from one page to another. As shown in previous studies, the distribution of timing for user connections may be used as a discriminant for those users [165][110]. However, while users may introduce a level of noise in the distribution, a sufficiently large data sample of users, packets and applications would allow determining the benefits and limitations of the method. It is acknowledged that the number of users in the present study is relatively small to draw statistically-strong conclusions about the efficiency and generalisation of the proposed method. The study also investigated that the users with variable interaction and behaviour may impact on the success of the method. Prior study, such as [166] defined the idle time as a time that there are no packets have been observed; they utilised idle time values typically ranging from 15 seconds to 5 minutes for monitoring flow records. The idle threshold (I) was proposed to be set at 10 seconds, which relates to different actions (interactions) of the same user. A user likely do different actions on an application with some sort of breaks (5 s, 10 s, 20 s). Therefore, 10 s was selected to define a maximum delay for a user to do a new action. The pseudocode in Figure 4-2 summarises the estimation of bursts and idle time between packets and for each flow. For each packet arrival, the inter-arrival time is compared with the two burstiness thresholds to determine whether the packet is part of a new burst or session. As showed in Figure 3.1 in chapter 3, many features could be extracted from each flow and from each direction, such as a total number of bursts in direction a-b/b-a, the total number of packets within bursts for each direction and the total size of bursts in bytes in each direction. The possible features that could be extracted from the pseudocode are described in Table 4-1; each of the inputs in the table is a pair of variables, one for the a-to-b direction and one for the b-to-a direction. The table contains two types of features: the first type of features were generated using *tcptrace* tool. While, the second type that are in green colour represent features that are calculated from the first type such as ratio between two features and average.

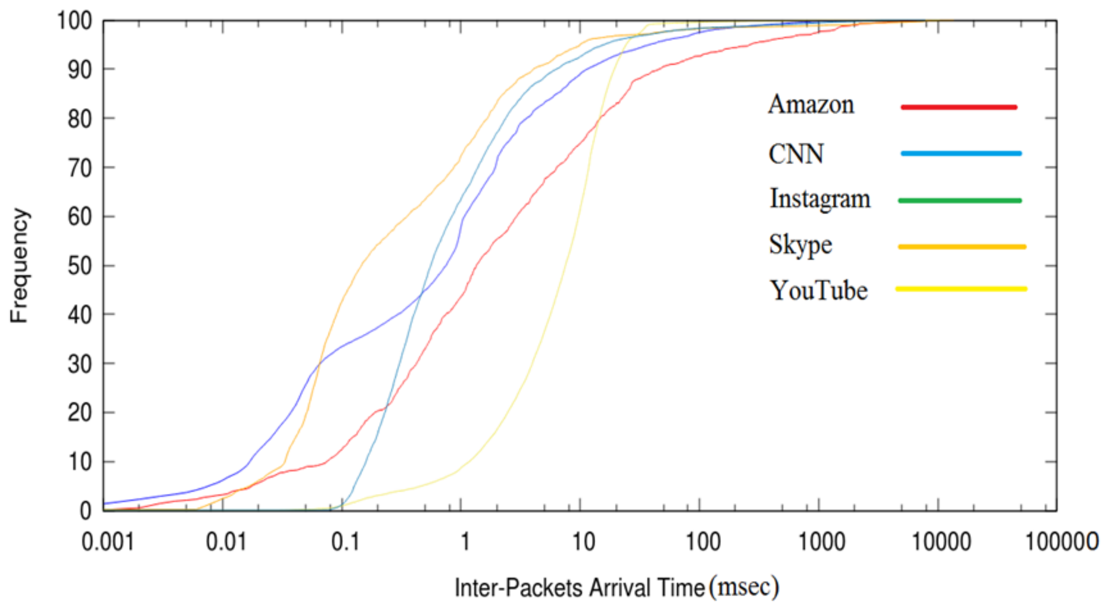


Figure 4-1: Distribution of inter-packet arrival times for five applications

```
burst_threshold = 1s
idle_threshold = 10s
   initialise burst and idle time parameters
   while packets arriving
   do
         calculate interarrival_time
         if interarrival_time < burst_threshold
                current_burst ++
                current_session ++
         else
                burst_counter ++
                current_burst = 1
                if interarrival_time > idle_threshold
                    current_session = 1
                    session_counter ++
                    idle_time += interarrival_time
                fi
         fi
done
```

Figure 4-2: Estimation of packet bursts and idle time

Table 4-1: burstiness & idle time parameters for packet analysis

No.	Features	Features as in *tcptrace*	Definition
1,2	number of bursts	Burst_no_a Burst_no_b	Number of bursts in each flow when the time of successive packets is less than 1 s
3, 4	number of packets in bursts	Pkt_count_a Pkt_count_b	The total number of packets in all bursts in each flow
5	Ratio of b2a	Pkt_count_b / Pkt_count_a	The ratio between the number of packets in bursts in flow b, and the number of packets in bursts in flow a
6, 7	Number of bytes in bursts	burst_size_bytes_a burst_size_bytes_b	The total size of bytes in all bursts in each flow
8	Ratio b2a	Burst_size_bytes_b/ Burst_size_bytes_a	The ratio between the data size of bursts in flow b, and the data size of bursts in flow a
9, 10	Average bytes	Avg_burst_size_bytes_a Avg_burst_size_bytes_b	The total size of bytes in all bursts in each flow divided by the total of data packets within bursts in each flow
11,12	Burst duration	Burst_duration_a Burst_duration_b	The time duration of all bursts in each flow
13,14	Inter-arrival time	Inter_arrival_time_burst_a Inter_arrival_time_burst_b	The time duration of all bursts in each flow divided by the total packets within bursts
15, 16	Idle time	Idle_time_a Idle_time_b	The accumulation of inter arrival packets times when the time being greater than 40s
17, 18	Number of bursts(data)	Burst_data_no_a Burst_data_no_b	Number of bursts in each flow when the time of successive packets is less than 1 s and data size greater than 0
19, 20	Number of data packets in bursts	Pkt_data_count_a Pkt_data_count_b	The number of data packets in all bursts in each flow
21	Ratio b2a	Pkt_data_count_b/ Pkt_data_count_a	The ratio between the number of data packets in all bursts (data) in flow b, and the number of data packets in all bursts(data) in flow a

22, 23	Number of bytes in bursts	Burst_size_bytes_data_a Burst_size_bytes_data_b	The total size of bytes in all bursts(data) in each flow
24	Ratio of data b2a	Burst_size_bytes_data_b/ Burst_size_bytes_data_a	The ratio between the data size of bursts(data) in flow b, and the data size of bursts(data) in flow a
25, 26	Average data bytes	Avg_burst_size_bytes_data_a Avg_burst_size_bytes_data_b	The total size of bytes in all bursts(data) in each flow divided by the total of data packets within bursts(data) in each flow
27, 28	Burst duration	Burst_duration_data_a Burst_duration_data_b	The time duration of all bursts(data) in each flow
29, 30	Inter-arrival time	Burst_duration_data_a Burst_duration_data_b	The time duration of all bursts(data) in each flow divided by the total packets within burst
31, 32	Idle time	Idle_time_data_a Idle_time_data_b	The accumulation of inter arrival packets times when the time being greater than 40s

4.2.2 **Flow Analysis**

The same concept of burstiness and idle time, which was applied in section 4.2.1 was applied to calculate the burst and idle time between flows. The variables included the time differences between the initial times of flows and subsequent flows, which are calculated from the first packet of each flow. The timestamp of a first packet for a first flow is subtracted from the timestamp of a first packet for a second flow. If the time difference is equal or less than 1 second, then the two flows are part of the same burst. Otherwise, the time difference is more than 10 seconds, then the period is considered as an idle time; flows that fall between these periods are ignored. Table 4-2 summarises the burst-based features among flows.

Table 4-2: Burstiness & idle time parameters for flow analysis

No.	Features	Formula	Description
33	Conn_all	No. of connections	Total number of connections
34, 35	Burst-no	no_burst_in_conns_1, no_burst_in_conns_2	Total number of bursts between flows for each session
36, 37	Flows-no	conns_no_in_burst_1, conns_no_in_burst_2	Total number of flows within all bursts for each session
38, 39	Packets-no	packets_no_in_burst_conns_1, packets_no_in_burst_conns_2	Total number of packets within all bursts for each session
40, 41	Packets-data-no	packets_data_no_in_burst_conns_1, packets_data_no_in_burst_conns_2	Total number of data packets within all bursts for each session
42, 43	Burst-size	size_burst_conns_1, size_burst_conns_2	Total size of all bursts for each session
44, 45	Average-burst-size	average_size_burst_conns_1, average_size_burst_conns_2	Average size of bursts for each session
46, 47	Burst-duration	burst_conns_duration_1, burst_conns_duration_2	Total time duration for all bursts
48, 49	Burst-duration/burst-no	inter_arrival_time_burst_conns_1, inter_arrival_time_burst_conns_2	Ratio between burst duration and total number of bursts for each session
50, 51	Burst-idle-time	idle_time_burst_conns_1, idle_time_burst_conns_2	Total inactive time between flows for each session

4.2.3 Conventional Analysis

The previous studies proposed statistical features as showed in chapter three section 3.5; these features are generated using *tcptrace* tool. In this work, the proposed features are compared with features that were suggested by previous studies to show how the effects of the burstiness and idle time method in distinguishing between applications. These features were calculated for each direction of a flow, as shown in Table 4-3 .

4.3 Preliminary Study

A preliminary study was conducted to determine whether the distribution of arrival times does indeed differ when using different applications. The data used for the study included captured the Internet traffic of the activities for six different applications on a machine running Linux, and using Google Chrome as web browser. The data was captured by running the *tcpdump* tool in the background while the user browses the applications. Afterwards, the data was analysed to extract the features, which described in the previous section, and finally C5.0 algorithm was utilized for classification the applications.

4.3.1 Data collection and analysis

For the experiment, the data were captured at University of Plymouth in the CSCAN (Centre for Security, Communications and Network Research) lab from six users, who were full time PhD students and they were only available at time of data collection (i.e., May-July 2017 and 2018). Three computers were used for this experiment with Linux operating system to collect more data samples during short time and also some participants did not install Linux operating system on their PCs. Each user was asked to browse six of most popular web applications (i.e., BBC news, Facebook, Google searching, Skype, Yahoo mail and YouTube)[53]. The reason for selecting these applications as they are considered the most well-known applications [53]. The users accessed separately each application for (30) times and each time was for (2-5) minutes, creating a dataset of 180 sessions per application (i.e., 30 sessions × 6 users). Users were limited to using only a single application in any session and dump files were accordingly labelled with the name of the accessed application.

Table 4-3: Conventional features proposed by previous studies [18, 19, 112-114, 119-125, 20, 141-143, 21, 24, 28, 91, 100, 109, 110]

No.	Features	Formula	Definition
52, 53	total number of packets	Packets_a Packets_b	Total number of packets in each flow
54	ratio of b2a	Packets_b / packets_a	The ratio between the total packets of flow b, and total packets of flow a
55, 56	number of data packets	Data_packets_a Data_packets_b	Number of data packets for each flow
57	ratio of b2a	Data_packets_b/ Data_packets_a	The ratio between the data packets of flow b, and data packets of flow a
58, 59	number of flags packets	Flags_packets_a flags_packets_b	Number of flags packets in each flow
60	ratio of b2a	Flags_packets_b/ Flags_packets_a	The ratio between the flags packets of flow b, and flags packets of flow a
61, 62	ratio of flags and total packets	Flags_packets_a / Packets_a Flags_packets_b/ Packets_b	The ratio between the flags packets and total packets of each flow
63, 64	1st packet size	First_pkt_a First_pkt_b	The size of the first packet in each flow
65, 66	flow duration	Flow_duration_a Flow_duration_b	The time duration of each flow (the time of last packet subtracted by the time of the first packet)
67, 68	Inter arrival time	Inter_arrival_time_a Inter_arrival_time_b	The time duration of each flow divided by the total number of packets

The large and separated dataset were used to build robust classifier model. The data was collected during four months starting in Nov 2016 by running a tcpdump tool [167] in the background, and storing the data in the storage area for preparing to analyse in the next stage. The limitation in such data collection that each file contains only flows about the labelled application, although this approach is useful to build a ground truth dataset for use in the training phase, a classifier trained on this data would not be able to predict traffic from additional applications. Table 4-4 shows more details for the captured data.

Table 4-4: Summary of the data collection for six applications

Application	**Flows**	**Duration (h)**
BBC news	32,596	15.6
Facebook	5,620	12.9
Google searching	27,640	8.5
Skype	2,632	9.88
Yahoo mail	48,116	10.22
YouTube	11,233	11.3

In the next stage, the collected data was analysed using the *tcptrace* tool [70] with packet trace as input and output flows that have the same five tuples (source IP address, source port number, destination IP address, destination port number and protocol). As part of this study, two levels of features were used – packet-level features (set1) and conventional analysis features (set2) as presented in section 4.2.1 and 4.2.3 respectively. More statistical operations (i.e., maximum, minimum, mean, median, and standard deviation) were applied upon the conventional and packets burst features. The aim of these processes was to summarize the output of all features in one row for each operation. Therefore, the result was five rows per session. Afterwards, the five rows were allocated in one row which represents the signature of the sample (session) that be the input to a classifier.

4.3.2 **The Decision Tree Analysis and Classifier Derivation**

The final dataset that obtained from previous section contained six Internet appliations with more than 1000 sesssions. This dataset included only the features that were introduced in section 4.2.1 & 4.2.3. The evaluation of the proposed features versus the traditional ones was carried out using three feature sets. The first feature set contained the burstiness and idle time features that were proposed by this work as were shown in Table 4.1. The second feature set included the features that were suggested from the previous studies, while the third feature set combined both sets. As highlighted earlier, the research community used the C5.0 algorithm to obtain more accurate results as new features were added to this algorithm (i.e., boosting and pruning). The boosting feature was used in this work, this algorithm gives all records the same weight and applies a sequence of iterations of classification. The iterations could be 10, 20, 50, or 100, and for each iteration the misclassified records are increased their weight, while the weight of the right classified records is reduced. Finally, a strong classifier is created from incorporating the individual ones with the best tuning for the parameters to avoid overfitting. With no boosting, a parallel process is applied as each sample or feature treats independently. In contrast, boosting works sequentially, each tree depends on the previously treated tree until reach a strong classifier. Therefore, this experiment used this algorithm to evaluate the collected data that was split to ratio of 2/1 for training and testing respectively. Table 4-5 presents the accuracy for two feature sets that range between 90-97.96%, the accuracy for the conventional feature set exceeded the accuracy obtained by the proposed features. Combined two sets and using boosting factor (i.e., 10 & 100) slightly enhanced the classifier ability to discriminate the different traffic that were generated from the applications.

The attributes usage (percentage) by the optimal C5.0 in computing the decision tree using feature set 3 is given in Table 4.6. The Table shows the comparison between conventional and proposed features.

Table 4-5: Accuracy of the classifier with feature sets

Feature set	No boost	Boost 10	Boost 100
Set 1 (new approach burstiness)	94.33	96.83	96.83
Set 2 (conventional)	93	97.33	97.5
Set 3 (combined set1 & set2)	90.7	97.96	97.96

The attributes in interval 100 percentage reported maximum usage in segregating among the six applications. Moreover, the attributes in interval between 75-99% percentages showed highly usage by the classifier. These percentages will be explained in chapter 7, page 107 with table 7-3. The burstiness attributes between packets streams were the majority part compared with the conventional ones, which were offered differentiation among applications activities. This is another indicator that the classifier strongly relied on the proposed features as they provided high discrimination between applications. The arrival time of packets and the inter-arrival delay were calculated from the packet traces using a 1-second threshold for the burst size and 10 seconds for delay size.

Figure 4-3 displays a boxplot analysis of the six applications using average burst size per flow feature. The burst size could be defined as a total number of bytes in all bursts in each flow, and this feature was calculated using *tcptrace* tool as showed in Table 4-1 (packet level analysis). This feature is a combination of feature 9 and feature 10 in the table. The boxplot is a plot that displays a data distribution based on a summary of five values (minimum, first quartile, median, third quartile, and maximum). The plot presents the distribution of the data through their quartiles, it can be observed from this descriptive statistics that the distributions of the applications are different. The values of the feature are negative as a normalization technique was applied as part of data pre-processing for machine learning.

Table 4-6: Features used in the classifier model

Conventional features usage	
100%	Mean & median of no. of packets_a; Mean of no. of data_packets_a, Mean_flow_duration_b; Max_flow_duration_b; Median of the first packet_a & the first Packets_b; Standard deviation of inter arrival time_b
75-99%	Mean no. of data Packets_b; Median of no. of flags packets_a / no. of packets_a; sd of the first Packets_b; Max of inter arrival time_a; Mean of the first packet in each direction; Mean of inter arrival time_b; Sd of ratio of no. of packets in both directions; Mean of no. of flags Packets_b; Sd of number of data packets_a; Sd of number of packets_a
Proposed features usage	
100%	Mean of number of data burst_a; Mean of the inter arrival time_data_b ; Max of number of packets in burst_b; max size of the data burst_b; max of data burst duration_b; Max of the average of size of the data burst_b; Median of the duration burst_a; Median of the inter arrival time_data_a; Sd burst_duration_b;Sd_burst_size_bytes_data_a;d_inter_arrival_time_data_b; No. of connections for each session; No. of connections in bursts; Mean of the ratio of size of burst in both direction; Max of the ratio of size of data burst in both direction
75-99%	Max size of burst_b; Max no. of burst_b; Median of ratio of the burst size in both directions; Sd of the no. of packets in burst; Mean of the inter arrival time in the burst; Max no. of the data burst_a; Sd of the average of the size burst_b; Median of ratio of the data burst size in both directions; Max of number of packets in burst_a; Median of Avg. of size of data burst_a; Sd of size burst_a; Sd of Avg of size burst_a; Sd of inter arrival time in burst_a; No. of bursts in connections; Mean of size of data burst_b; Max of inter arrival time in data burst_a; Sd of ratio size burst in both directions; Sd of size of data burst_b; Sd no. of data packets in burst_a

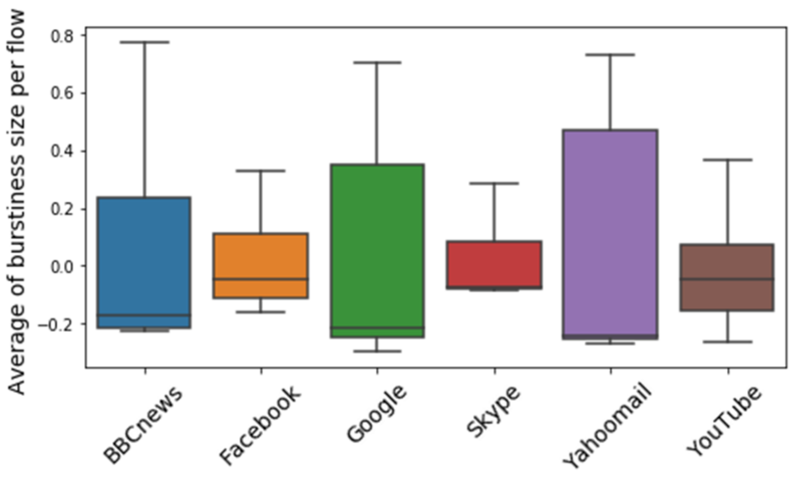


Figure 4-3: various behaviour for six applications

The results signify that the features related to the burstiness and idle time have high efficiency in discriminating the different applications. Combining both sets showed considerable improvement in classification accuracy peaking at (97.96%).

4.4 Conclusion

This chapter presented a novel set of features for applications identification based on inter-arrival times between packets and flows, most specifically burstiness and idle time. The initial assumption was that different applications produce different distributions of data, creating various connections and timing patterns between the generated packets and flows. The features were defined on two levels, the first level was in the context of packet analysis and the second level was in the context of flow analysis. This concept was applied by modifying the *tcptrace* tool to extract the new features by writing a code inside the tool. A preliminary study was established to examine the effectiveness of the proposed features by employing C5.0 classifier with a small data set. Based on the experimental results, the proposed features proved to contain contributory information towards the classification results by providing high discrimination between the applications. In addition, the experimental results showed that the proposed features are the most in the classifier usage than the conventional ones, which were proposed by other studies (this sentence is deleted by the author). Based on the success of the experimental results, the next chapter will focus upon adding more web applications and more features with large data sets (controlled and uncontrolled environments).

5 Methodology and Data Collection

5.1 Introduction

The previous chapter presented a set of novel features based on the inter-arrival time between packets and flows, focusing on the burstiness and idle time in Internet traffic using limited datasets. To test the ability of such features to generalise, two types of larger data sets were collected; the first data set contained 10 users that were browsing 11 of most popular Internet applications. The users were guided/instructed to browse these application in order to build a strong truth data table preparing to use it in labelling a second data set (i.e. uncontrolled data). The second data set was real data that collected from 20 users that were browsing different applications independently. Both data sets were analysed by utilizing *tcptrace* tool to generate the proposed features; different techniques were used for labelling uncontrolled data relied upon DNS and IP addresses.

The remainder of this chapter is organized as follows. Section 5.2 presents a block diagram for the proposed methodology explaining briefly the main steps. Followed by subsections that elaborate data collection for the controlled and uncontrolled data, pre-processing and data analysis. Section 5.3 draws conclusions.

5.2 General Block Diagram for a Proposed System

A high-level architecture of a proposed system is presented in Figure 5.1 that identifies applications based on the concept of the burstiness and idle time that explained in previous chapter. The architecture consists of two main parts, the first part is for applications identification based on data that was collected in a controlled environment (i.e. the users were given instructions sheet of what should they browse). For a second part, a data was collected with an uncontrolled

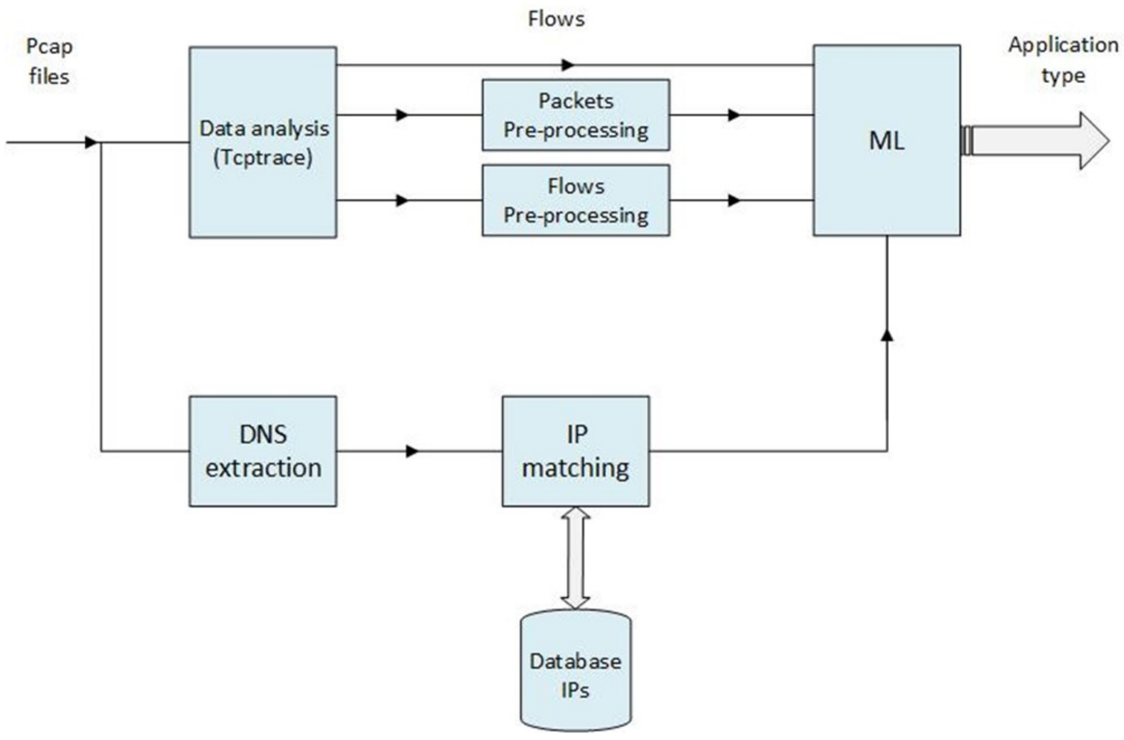


Figure 5-1: Proposed traffic classification methodology

environment (i.e. website browsing was based on user's preference). Highlighting the key components of application identification scheme with a description of principal steps as follows:

1. **Data collection:** Firstly, the data was captured using tcpdump tool from users that were accessing Internet applications. The controlled data was collected per application, stored in files, and labelled according to the application. For uncontrolled data, the data contained different activities based on user's preference; therefore, the data was labelled based on DNS quires, and IP addresses by matching with the controlled data.
2. **Data analysis:** Afterwards, the traffic was analysed by *tcptrace* to extract the features that were explained in previous chapter section 4.2.

Two sets of features were generated from the *tcptrace*, the conventional features and the proposed features for packet analysis. More features were generated from these two sets using a Python script such as ratio between some features and calculate average value for others (see appendix B for the scripts). Additionally, a third set of features which, contains the proposed features for flow analysis, was generated using Python script.

3. **IP matching:** IP addresses for each application in the controlled data were stored in a file and labelled with that application. Therefore, the matching process starts with reading the DNS request in the uncontrolled data to determine the application name and afterwards fetching the specified file of the IP addresses for that application in the controlled data. Secondly, matching the unknown flows (after the DNS request) with the specified IP file until the end of the flow trace and tag them as known flows. Finally, dump known flows in separated files and labelled them according to the application name.
4. **Statistical operations:** Five statistical operations were calculated (Min, Max, mean, median and standard deviation) for only the conventional and packet analysis features. These features were calculated for each feature vertically, for all connections and per session. More details in this step in **subsection 5.2.6**.

Machine Learning/ Decision Tree Analysis: Using machine learning algorithms, different data sets of computed features were utilized to obtain a decision for application classification. Two experiments are implemented in this work; a first experiment is for the controlled data and a second experiment is for the uncontrolled data. In the first experiment, the data was collected using Linux operating system, which was installed via VMware tool under Windows operating system. A NAT (Network Address Translation) network was set up, which translates IP address of virtual machine to the IP address of the host system. In [168], they studied an impact of virtualization on performance of Amazon Elastic Cloud Computing network (EC2). They measured packet delay, packet loss, and TCP/UDP throughput between virtual machines of Amazon EC2 and they found that there are a considerable abnormally delay variations and changing in the throughput. In this work experiment, only one virtual machine was installed per PC to collect data from the network that is not impact on the experiment measures. Aims and details of each experiment are provided in the next subsections.

5.2.1 **Data collection**

Data collection used *tcpdump* but two different type of sessions (i.e. controlled and uncontrolled); given the way data is organized, the uncontrolled data is slightly different from the controlled one. In controlled data, the truth table was derived automatically as the applications that the users accessed were known, while the other one, generating the truth table required additional IP/DNS mapping and this illustrated in the second diagram. More details regarding both types are presented next.

Controlled environment: this data was collected from individual applications under controlled environment for different samples of each application and different users. The authors in [37] claimed that the data collected from a user side when he/she working on known applications leads to accurate results. The data collection was conducted at University of Plymouth in the CSCAN lab. The data collection process spanned between May-July 2017. Eight computers were used for this purpose, six of them belonging to the participants, and two of them belong to the researcher; all these computers were Linux virtual machines. All the users were full-time PhD students that were working in the lab, ten students were chosen to take part in the collection, their ages ranged from (30-43) years old, seven males and three females. Eleven applications were selected to browse, which are the most popular applications that are accessing by users, using Google chrome as Internet explorer [53]. These applications are different in their page contents, they included social websites (i.e. Facebook and Instagram), news websites (i.e. BBC news and CNN), searching engines (i.e. Google search and Bing), and E-mail browser (i.e. Yahoo mail and G-mail), P2P application (Skype), video streaming (YouTube), and shopping website (Amazon). The users were asked to browse these applications separately. Hence, the data was collected per application and dumped in labelled files for analysing. The data collection contained instructions sheet that was given to users and as follows:

1. In the beginning, a user turn on the VMware and run Ubuntu version 15.10.
2. Run command line prompt and start running *tcpdump* tool in the background.
3. Access Google chrome and start browsing one of the eleven applications, for example Facebook, for a period of time between (2-5) minutes. After finishing the browsing, the user closes the explorer and stops tcpdump tool.
4. Label the file, which contains the captured data with user name followed by application name and number of the sample, for example Hussein.facebook1.
5. Repeat the steps from 2 to 4 for the same application and for 30 times.

The total sessions for each user were 30 per application resulting 300 sessions for ten users; the total number of sessions for all applications and for all users were 3300. Table 5-1 summarises the data collection for controlled data.

Table 5-1: summary of the collected data

Traffic class	Application type	Size (GB)	Flows	Duration (h)
News	BBC news	1.72	56394	25
	CNN	7.25	25123	11.2
Social media	Facebook	2.5	9630	21.97
	Instagram	0.469	5641	11.15
Search engine	Google	0.370	45960	13
	Bing	0.578	30953	10.55
Video chat	Skype	0.490	3948	14.88
Email client	Yahoo mail	1.09	76674	15.66
	G-mail	0.6	49720	10.13
Video streaming	YouTube	4.29	18816	17.9
Shopping	Amazon	1.13	51793	12

Uncontrolled environment: A real data was collected for various activities and different users that were accessing websites applications. The raw data traffic was collected in the same lab (CSCAN) at University of Plymouth between May-July 2018. The participants used twenty university computers and they were a mix of laptop and desktop computers. Eighteen computers were installed using Windows operating system and two of them were installed using Windows and Linux operating system and under different virtual machine environments (i.e. VMware and virtual box). The users were PhD students working in the lab, 20 students were chosen to take part in data collection, their ages ranged from (30-43) years old, (14) males and (6) females. Different Internet applications were browsed using Google chrome explorer and based on user's preference and without any instructions given by the researcher. The data was collected using tcpdump tool via a network-based method and it was divided into 24 samples per day. Each sample represents one-hour traffic of pcap format; this division reduces the size and processing time of each sample. In the controlled data, the applications were known as instructions were given the users; while in the uncontrolled data, the applications were unknown as a traffic was a mix of multiple applications and different users. This traffic needs more data processing for labelling; therefore, the next step would be an additional process for only the uncontrolled data as shown in the next subsections.

5.2.2 DNS Queries

The aim of this process is to label flows and this can be accomplished by using DNS queries. The uncollected data was packet-based and contained DNS enquires - thus the contents of the DNS requests were used to identify applications. In each DNS packet request, a keyword refers to requested applications. The procedures of reading the application request is as follows:

1. Read each packet line for port number 53 which represents the DNS enquires.
2. These enquires contain application requests, if the user requests the amazon website, a keyword "www.amazon.com." would be in the DNS enquires. Other keywords for different applications, see the Table 5-2.
3. Create a file named with the requested application, open it and dump all packets in this file for three seconds by setting a timer, this threshold based on assumption that a user needs minimum three seconds to change from a current website to another.
4. Reset the timer after the end of three seconds and store all packets until a next request.
5. Repeat step 1-4 until the end of the data trace.

Table 5-2: Application keywords

Website	Keyword
Amazon	www.amazon.com.
BBC news	www.bbc.co.uk.
Bing	www.bing.com.
CNN	www.cnn.com.
Facebook	www.facebook.com.
Instagram	www.instagram.com.
Yahoo mail	login.yahoo.com.
YouTube	www.youtube.com.
Google engine	www.google.com.
G-mail	accounts.google.com.
University of Plymouth	www.plymouth.ac.uk.

This process partitioned the traffic into many applications considering the specific time stamp of each request preparing for the next stage. Packets after three seconds for each request until the next request remained unknown, in the next stage, the packet trace will be analysed into flows to speed up a matching process. Through monitoring the captured traffic, applications such as CNN and Facebook generate requests for other applications. For example, when a user accesses the CNN website, there are requests for YouTube, Facebook, BBC news or Instagram. This behaviour for some applications causes errors in identifying the real activities based on the previous procedure. Therefore, this study determined the following applications (i.e. YouTube, Facebook, BBC news or Instagram) that could be generated by other websites to check if they are certain requested by users. The study used a resolution of the DNS to read keywords, for example, when a user requests the Instagram, the following keywords are released (‘*instagram-p3-shv-01-lhr3.fbcdn.net.https*’ and ‘*instagram-p3-shv*’). By setting a counter for these keywords, if they exceed 180 within 2 minutes, then the Instagram is certain, otherwise the application is not certain. The same scenario was applied for other applications, Table 5.3 shows these applications with their keywords and counters. The counter was set to 500 as these applications released the keywords in a range (450-550) and within 2 minutes when a user requests any one of these applications.

Table 5-3: DNS enquires

Application	**Keywords**	**counter**
YouTube	*'-in-f14.1e100.net.https'* *'.1e100.net.https'* *'-in-f2.1e100.net.https'*	500
Facebook	*'.facebook.com.https'* *'.fbcdn.net.https'*	500
BBC news	*'bbc'*	500

5.2.3 **Data Analysis**

The collected Internet traffic were analyzed using the *tcptrace* tool that processes pcap files (packet trace) as input and groups them into flows that are sharing the same five tuples (source IP address, source port number, destination IP address, destination port number and protocol). This tool takes pcap files and transfers them into 49 features that were presented in **section 4.2**. Thirteen of them described the packet characteristics and for each direction of a flow such as total number of packets, total number of data packets, total number of flags packets, and size of the first packet. Four of them display the flow duration and inter arrival time. The others show advanced features that were proposed by this study such as total number of bursts, total number of packets in bursts, duration of the burst and idle time. The *tcptrace* tool generated only two levels of the features: packet analysis and conventional analysis, which were described in subsections **4.2.1** and **4.2.3** respectivly. The flow analysis features, which were described in section **4.2.2** and in the table 4.2, were produced from the labelled connections using *Python* script. For control envirnmnent, these features were obtained directly after finishing the data analsis. In contrast, the uncontrolled environmnent, these features would be produced after labelling all traffic (matching process in the next section).

5.2.4 **IP Matching**

From the controlled data, IP addresses for each application were stored in a file labelled with that application. The uncontrolled data was analysed as shown in the previous subsection into flows that contained known flows based on reading the DNS requests plus the three seconds after the requests. Therefore, the matching process firstly started with reading the known flows to determine the application name and afterwards fetching the specified file of the IP addresses for that application. Secondly, matching the unknown flows with the specified file until the end of the flow trace and tag them as known flows. Finally, dumping known flows in separated files and labelling them according to the application name. Based on previous studies, the IP files are subjecting to change continuously by the owners of applications for security reasons. Therefore, updating these addresses are essential, but it must be automatically and during the identification process.

5.2.5 Keywords Matching

Many flows remained unknown after the IP matching process; all were stored in a separated file named as unknown flows. For the controlled data, the IP addresses file for each application were resolved into keywords based on DNS queries and stored in a separated file named as keywords file. The results for these keywords for each application are illustrated in Table 5-4. This process included firstly resolving a server IP address of the unknown flows into keywords. Secondly matching these keywords with the keywords file until the end of the flow trace. Finally, dumping known flows to separated files.

Afterwards, five statistical operations (Min, Max, mean, median and standard deviation) were used for summarizing statistics of each feature vertically. The mean and median measure a central of tendency for feature values to display how the distribution of the values around the middle. They are very sensitive to the outliers in a data; the outliers usually have high or low values that deviate from other values, pre-processing and removing such outliers is very important to avoid overfitting in the classifier. In contrast, the variability measures the dispersion in feature values and displays how a data is spread out. The feature values being more consistent when the variability is low, in opposite, the values being farther from others when the variability is high. The most common measures of the

Table 5-4: DNS enquires

Application	Keywords
Amazon	*'cloudfront.net', 'deploy.static.akamaitechnologies.com.https', 's3-3-w.amazonaws.com.https'*
BBC news	*'an.haven.com.https', '.bbc.co.uk.http', 'www.edigitalsurvey.com.http'*
Bing	*'a-0001.a-msedge.net.http'*
CNN	*'a23-55-58-227.deploy.static.akamaitechnologies.com.https','west-1.compute.amazonaws.com.http','compute-1.amazonaws.com.https', 'akamaitechnologies.com.http','1e100.net.https','fbcdn.net.https', 'pixel.quantserve.com.http'*
Facebook	*'.fbcdn.net.https', '.facebook.com.https', '.fbcdn.net.https'*
Instagram	*'instagram-p3-shv-01-lhr3.fbcdn.net.https','instagram-p3-shv'*
Yahoo mail	*'.ycpi.vip.lob.yahoo.com.https','mpr2.ngd.vip.ir2.yahoo.com.https', 'r1.ycpi.vip.ir2.yahoo.net.https', 'beap3.cbs.vip.ir2.yahoo.com.https', 'ats1.member.vip.ir2.yahoo.com.https', 'pr-bh.pbp.vip.ir2.yahoo.com.https', 'public.comet.vip.bf1.yahoo.com.https','a2.ue.vip.ir2.yahoo.net.https', 'gw.iris.vip.bf1.yahoo.com.https','e1.ycpi.vip.lob.yahoo.com.https','a1.ue.vip .ir2.yahoo.net.https'*
Youtube	*'lhr35s05'*
Google	*'lhr25s','wk-in'*
G-mail	*'lhr35s05'*
University of Plymouth	*'plymouth'*

variability are range and stranded deviation, the range is the difference between two extremist values and become useful when the size of the sample is small. In our work, the range was divided into two separate measures (i.e., maximum and minimum), these two measures were calculated for each feature.

These operations were applied only for conventional and packet analysis features as these features were calculated per flow, therefore, these operations summaries the session statistically. In contrast, the flow features were already summaries the session such as the number of connections per session. The results of these operations to the conventional and packet features were five rows (i.e., one row for each operation) and afterwards these rows were arranged in one row. Therefore, these features were doubled five times, for instance, a feature packet_a, which is a total number of packets in transmitting direction, become min_packet_a, max_packet_a, mean_packet_a, median_packet_a, and standard_deviation_packet_a. These features were combined with flow level features.

The results were 9 applications with details in Table 5-5, as shown in the table that the most application that had been used by the users was the G-mail against very low usage for Yahoo mail.

Table 5-5: Overall results for classification of the observed data

Application	**Flows**	**Duration (h)**	**Number of samples**
BBC news	3,150	1.6	6
Facebook	98,210	33.1	287
Google	59,422	88.5	892
Yahoo mail	6,795	0.8	9
YouTube	66,500	76.5	714
G-mail	1,448,392	143	870
Amazon	23,975	6.6	34
Plymouth.ac.uk	24,225	42.5	286
Bing	10,324	17.2	110

5.3 Conclusion

This chapter presented twofold of data sets for the proposed method and the required processing steps. The first data set was collected within a controlled environment to build a ground truth data for the second data set, which was collected in a real-time environment. Both data sets were analysed into different feature levels using *tcptrace* tool, preparing the data sets for more analysis in the next chapter. The uncontrolled data was labelled based on DNS enquires and matching the connections of traffic with the IP addresses of the applications, which were built from the controlled data. This matching process resulted in unclassified flows due to a changing in IP addresses for the requested applications and these new addresses had not been updated in the database files. As mentioned earlier, the database must be updated regularly, A study [154] investigated the stability of IP addresses and bags of domain over time for popular services (i.e., Facebook, Google, Google video, WhatsApp, Twitter, and

Dropbox). A study collected 12 datasets from residential network over a full year (2015), each dataset contained one month of data. Lists of IP addresses were created that were used by the popular services. The authors noticed that all services showed a change in the lists of IP addresses over the year, but this change is different from one service to another. For instances, Google Video showed relatively stable in the IP list, 15% of changing was for Dropbox. While for Twitter and Google, about 50% of the IP addresses were changed after one month of observation, for Dropbox and Facebook, the lists of IP addresses were completely disappeared after one year of observation. Part of the identification process, this work assumed that the generated traffic after the requesting application until the three seconds could be considered as part of that application. Therefore, this traffic can be used to update the database, however, this small period cannot updated the entire database and the assumption could be inaccurate that might add wrong addresses to the database. In the next chapter, an analysis for the controlled data is applied to reduce the number of features by finding the correlations between these features and visualize the selected features.

6 Analysis

6.1 Introduction

The previous chapter included collecting the data in controlled and uncontrolled environments, analysing the packets trace to flow-based trace, and pre-processing the analysed traffic. This generated the features that were presented in section 4.2 with labelling flows regarding their applications. Due to the large datasets, which contain many features, data reduction approaches were applied through an analysis to find a possible correlations between these features that leads to reduce in the number of them. Selecting the more relevant features and eliminating the irrelevant ones in the initial steps increases the performance of machine learning classifier. Filtering these features manually and trying to find the correlation with the specified target is a difficult task and time-consuming. A clustering analysis is used to explore a similarity between variables; consequently, one of them can represent the variables that have similar correlation. The same technique is used to find the variability between sessions to validate the ability of the proposed features in discriminating between applications. Moreover, principal component analysis (PCA) is used for graph representations as a descriptive analysis.

The chapter addresses the following aims:

- Determining the possible correlations (similarity) between input features in order to reduce in the number of features.
- Investigating the minimum set of input features that maximizes the accuracy for output prediction.
- Demonstrating that different users' behaviors do not effect on the application behavior.

6.2 Cluster analysis

Clustering techniques are utilized to group objects within clusters that are similar to each other, and they have been widely used for solving research problems. These techniques are helpful for displaying these groups in suitable graphs and identifying the correlation between sessions and features. There are two main techniques, hierarchical and k-means clustering, in this framework, a hierarchical clustering is used to group different features and samples of a data set. Only the controlled data is analysed using hierarchical clustering to find the similarity between features by drawing a dendrogram. For the uncontrolled data, a machine learning technique is deployed for features selection in the next chapter.

6.2.1 Controlled data

This data set contained 199 features with 3300 sessions, for better visualization, they are divided into nine data subsets and each subset contains 22 features as average. The hierarchical clustering is applied for each subset individually. Firstly, the technique is applied for the first subset (set1) that contains 24 features. The dendrogram in Figure 6-1 shows different clusters, the x-axes represents the features and the y-axes represents the similarity percentage. Ten clusters are noticed in the figure and each one given different colour. Within individual clusters, which have more than two features, the similarity is varied. For some features, the similarity is above 85%, while for others, the similarity reaches nearly 100%. This similarity shows that the action of these features are the same and

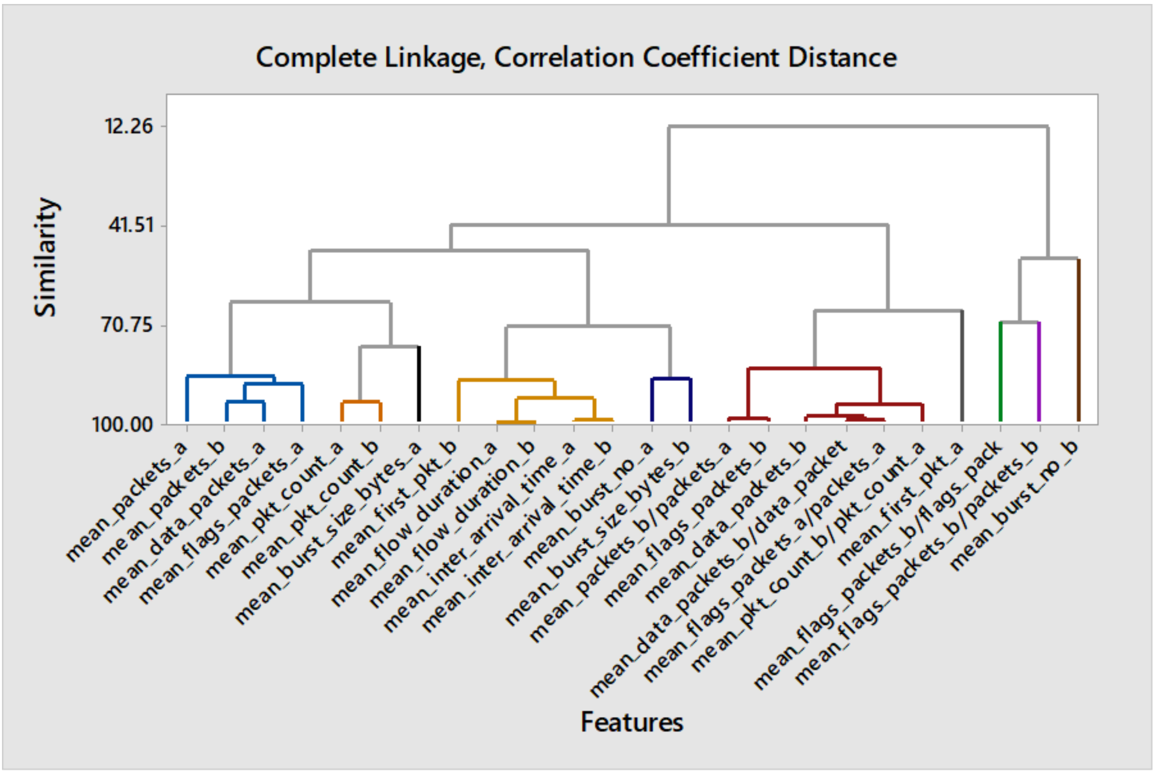


Figure 6-1: Clustering features dendrogram for set1

that leads to reduction in the number of features. Based on the cluster result, the number of features is reduced from 24 features into 10 features (the value 10 represents the cluster number); one feature is chosen from each cluster.

The same procedure was repeated on the remaining data subsets by applying the hierarchical clustering and drawing the dendrogram figures for each subset. The similarity between features is different for each data subset; consequently, the number of clusters are different. Table 6-1 shows the final features reduction of each data subset with 84 features in total for nine data subsets. The table presents also high availability for the proposed features, out of 84 features, there are 60 features belong to the proposed ones, which are in a blue colour in the table. They showed high dissimilarity from the other conventional features, in other words, the proposed features have a unique footprint that could identify various activities.

Table 6-1: Features selection for each data subset

Subsets	Features
Subset1	mean_packets_a,mean_data_packets_b,mean_flags_packets_b/flags_packet_a, mean_flags_packets_b/packets_b,mean_first_pkt_a,mean_burst_no_a, mean_burst_no_b,mean_pkt_count_a, mean_burst_size_bytes_a
Subset2	mean_burst_size_bytes_b/burst_size_bytes_a, mean_AVG_burst_size_bytes_a, mean_AVG_burst_size_bytes_b, mean_burst_duration_a,mean_inter_arrival_time_burst_a, mean_burst_data_no_a,mean_burst_data_no_b, mean_pkt_data_count_b
Subset3	mean_idle_time_data_b, max_packets_a, max_packets_b, max_packets_b/packets_a, max_data_packets_b,max_data_packets_b/data_packets_b, max_first_pkt_a, max_first_pkt_b,max_burst_no_a, max_pkt_count_a, max_pkt_count_b/pkt_count_a
Subset4	max_burst_size_bytes_b,max_burst_size_bytes_b/burst_size_bytes_a, max_AVG_burst_size_bytes_a, max_AVG_burst_size_bytes_b,max_burst_duration_a, max_burst_data_no_a,max_burst_data_no_b
Subset5	max_idle_time_data_a, md_packets_a, md_packets_b,md_packets_b/packets_a, md_data_packets_a, md_flags_packets_a, md_flags_packets_b,md_flags_packets_a/packets_a, md_first_pkt_a,md_burst_no_a, md_pkt_count_b, md_pkt_count_b/pkt_count_a
Subset6	md_burst_size_bytes_a, md_AVG_burst_size_bytes_a, md_AVG_burst_size_bytes_b, md_burst_duration_a, md_idle_time_a, _pkt_data_count_a, md_burst_size_bytes_b/burst_size_bytes_a, md_AVG_burst_size_bytes_data_a,md_burst_size_bytes_data_b/burst_size_bytes_data_a
Subset7	md_inter_arrival_time_data_b,md_idle_time_data_a,sd_packets_a, sd_packets_b/packets_a,sd_data_packets_a, sd_flags_packets_a,sd_flags_packets_b/flags_packet_a, sd_flags_packets_a/packets_a,sd_first_pkt_a, sd_first_pkt_b, sd_burst_no_a,sd_pkt_count_b
Subset8	sd_pkt_count_b/pkt_count_a, sd_burst_size_bytes_a,sd_AVG_burst_size_bytes_a, sd_burst_duration_a, sd_inter_arrival_time_burst_a, sd_burst_data_no_a, sd_burst_data_no_b , sd_pkt_data_count_b, sd_burst_size_bytes_data_b, sd_AVG_burst_size_bytes_data_b,sd_burst_duration_data_a
Subset9	sd_inter_arrival_time_data_a, sd_inter_arrival_time_data_b, sd_idle_time_data_a, sd_idle_time_data_b, No. of connections

For more reduction in the number of features, the same cluster technique is deployed on the final data set with 84 features, and the results are shown in dendrogram Figure 6-2. There are 29 clusters of features with similarity more than 80% in which the number of features is reduced to 29. Due to a high number of features, they are not visible in the figure, but the clustering splits the features into about 29 ones. These features are listed in Table 6-2 for 29 clusters, one feature could represent the cluster behaviour.

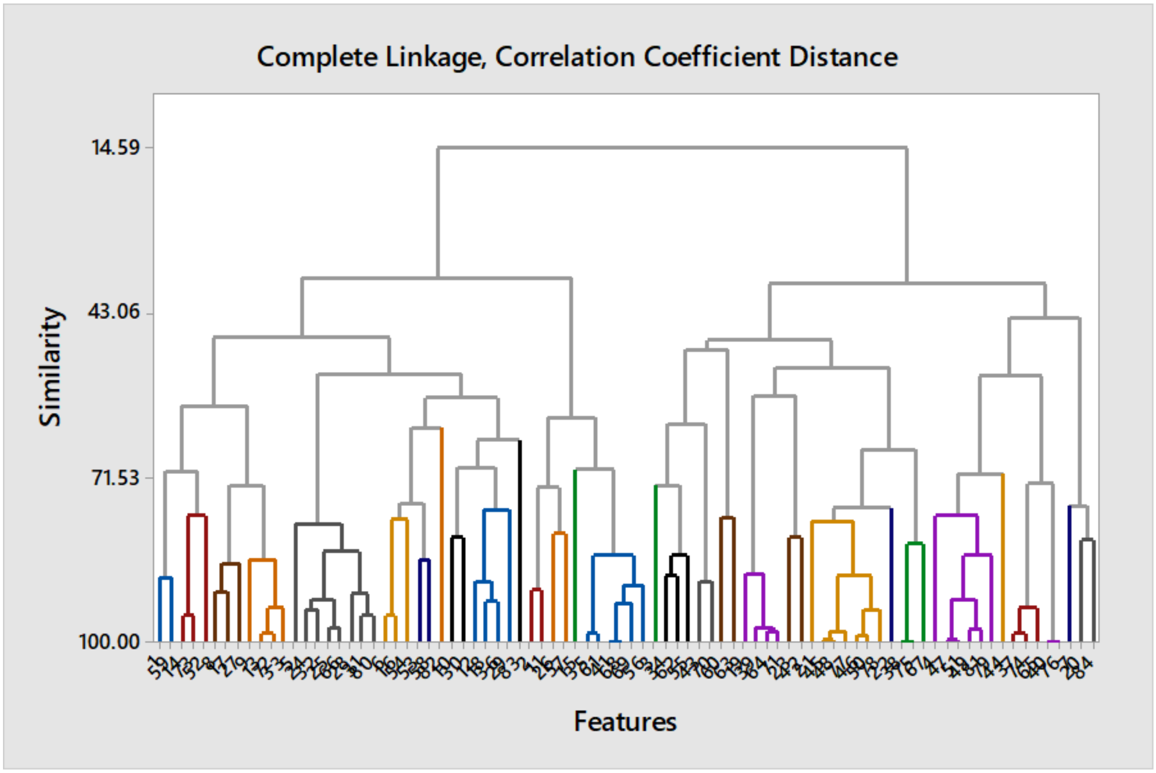


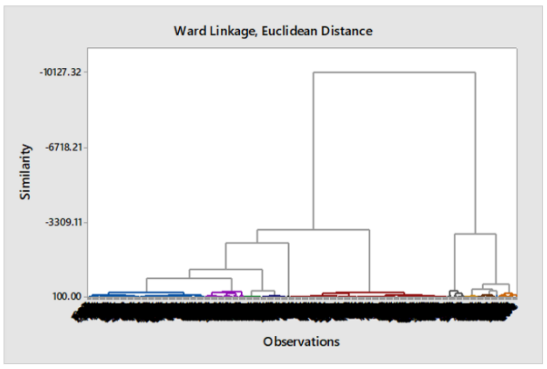
Figure 6-2: Features cluster dendrogram for 84 features

Table 6-2: Features hierarchical clustering

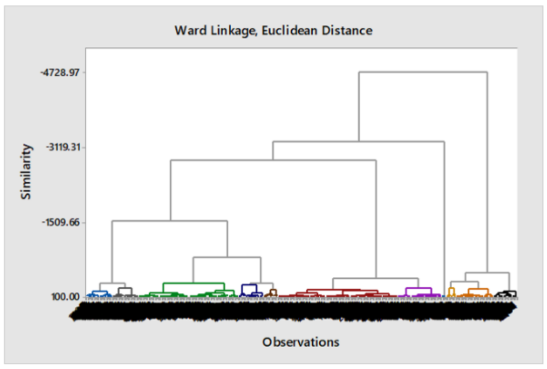
Clusters	Features
Cluster 1	(mean_packets_a); (sd_packets_a)
Cluster 2	(mean_data_packets_b); (mean_AVG_burst_size_bytes_a)
Cluster 3	(mean_flags_packets_b/flags_packet_a)
Cluster 4	(mean_flags_packets_b/packets_b); (md_pkt_count_b/pkt_count _a); (md_burst_size_bytes_b/burst_siz); (md_AVG_burst_size_b

	ytes_b); (sd_burst_duration_data_a); (sd_inter_arrival_time_data_b)
Cluster 5	(mean_first_pkt_a); (max_first_pkt_a); (max_first_pkt_b); (max_pkt_count_b/pkt_count_a); (max_AVG_burst_size_bytes_a); (max_AVG_burst_size_bytes_b); (sd_first_pkt_b); (sd_inter_arrival_time_data_a)
Cluster 6	(mean_burst_no_a); (mean_burst_data_no_b); (md_pkt_data_count_a)
Cluster 7	mean_burst_no_b
Cluster 8	mean_pkt_count_a mean_pkt_data_count_b max_pkt_count_a
Cluster 9	mean_burst_size_bytes_a mean_burst_duration_a max_burst_duration_a sd_burst_duration_a
Cluster 10	mean_burst_size_bytes_b/burst_size_bytes_a max_burst_size_bytes_b/burst_si
Cluster 11	mean_AVG_burst_size_bytes_b mean_idle_time_data_b max_burst_size_bytes_b max_idle_time_data_a
Cluster 12	mean_inter_arrival_time_burst_a md_burst_duration_a sd_inter_arrival_time_burst_a
Cluster 13	mean_burst_data_no_a
Cluster 14	max_packets_a md_packets_b/packets_a sd_flags_packets_a/packets_a sd_AVG_burst_size_bytes_a
Cluster 15	max_packets_b No. of connections
Cluster 16	max_packets_b/packets_a md_burst_no_a md_pkt_count_b md_burst_size_bytes_a md_AVG_burst_size_bytes_a sd_burst_size_bytes_data_b sd_AVG_burst_size_bytes_data_b
Cluster 17	max_data_packets_b
Cluster 18	max_data_packets_b/data_packets_a md_flags_packets_b
Cluster 19	max_burst_no_a md_inter_arrival_time_data_b
Cluster 20	max_burst_data_no_a md_burst_size_bytes_data_b/burs sd_flags_packets_a
Cluster 21	max_burst_data_no_b md_flags_packets_a md_AVG_burst_size_bytes_data_a sd_data_packets_a sd_pkt_count_a sd_pkt_count_b/pkt_count_a
Cluster 22	md_packets_a sd_first_pkt_a sd_burst_data_no_a
Cluster 23	md_packets_b sd_burst_no_a sd_burst_data_no_b
Cluster 24	md_data_packets_a sd_pkt_data_count_b
Cluster 25	md_flags_packets_a/packets_a sd_burst_size_bytes_a
Cluster 26	md_first_pkt_a
Cluster 27	md_idle_time_a md_idle_time_data_a
Cluster 28	sd_packets_b/packets_a sd_flags_packets_b/flags_packet
Cluster 29	sd_idle_time_data_a

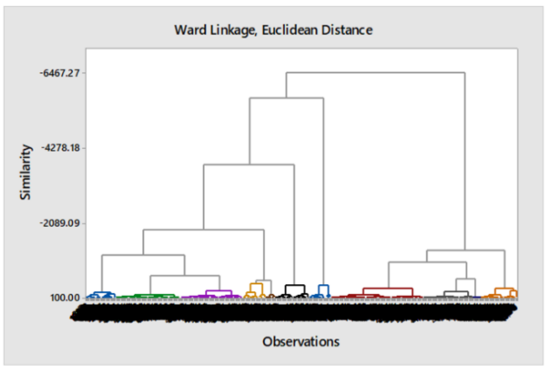
The features were reduced in the previous analysis based on the correlation (similarity) between them from 199 to 84 and finally to 29. To demonstrate the validity of the reduction approach, a clustering was applied to the data set, but between sessions (30,300), which contain the eleven application. Using ward Linkage with Euclidean distance in the Minitab tool for three data sets with different features (199, 84, and 29), a very good improvement in the separations between sessions is achieved for the last data set with 29 features. Figure 6-3 shows three dendograms for three data sets; although the x-axis includes many sessions that cannot recognize them, the separation between sessions is clear for the three figures. For instances, Figure 6-3 (A) shows only seven clusters for dataset that contains 199 features and eleven applications, while Figure 6-3 (B) shows clearly nine clusters for dataset that contains 84 features and the same number of applications. Figure 6-3 (c) shows all the activities of eleven applications for dataset that contains only 29 features. The features are able to discriminate among the application samples based on the statistical differences between inter-arrival times of packets and flows. In other words, the selected applications generate different behaviour based on statistical features relating to the timing of packets arriving. As a result, the proposed features show high ability in identifying the eleven applications.



A (199)



B (84)



C (29)

Figure 6-3: Dentrograms separation of samples into clusters for three data sets

6.3 Principal Component Analysis (PCA)

In the previous section, the cluster analysis was introduced to show the similarity between features as the approach of data reduction. In contrast, a PCA, which examines the variability in the data by generating few features. This technique was applied to the data that consists of **29** features and **2, 200** samples for data visualization and to show how the contributions of the selected features in presenting the data. In other words, it is impossible to visualize data with many features, but PCA can read the variance in the overall data and describe many features in a few components. Figure 6-4 shows the scree plot that describes the variability of the overall data by the contribution of each component based on eigenvalues. The first two components represent most the features, while the remaining components show a decrease in the representation. It is

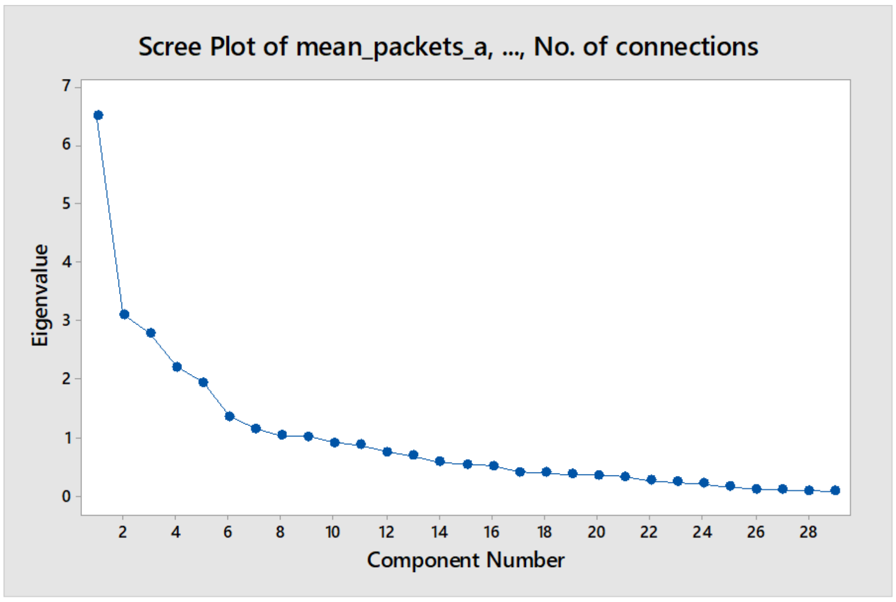


Figure 6-4: Scree plot of data

clearly highlights that most of the variability of the data is presented by the first two component PC1 and PC2. The other significant variability clarified by the components PC3 to PC5, while the remaining componets are illustrated low variance. Although the first component absorbed the largest variability of the data as can be seen from the eigenvalue, the other components also take part in the data variance. From the figure, the features are reduced from 29 to only 5 features, which are represent most of the variability of the data. Figure 6-5 presents a score plot generated from calculating the values of PC1 and PC2 in the x-axis and y-axis respectively. Although the two components are not sufficient to identify clearly the different applications, the figure shows the eight activities of the traffic applications, except for Amazon sessions which do not appear in the graph. Skype and Google are spread within activities, as the second one can be used at each application for searching. Although the plot deals with 11 applications as shown from the class label in the top right of the plot, only 8 applications appear in the figure as the two components (PC1 & PC2) are not able to present all applications. Visualization data in more than two dimensions can give a better understanding of the application behaviour.

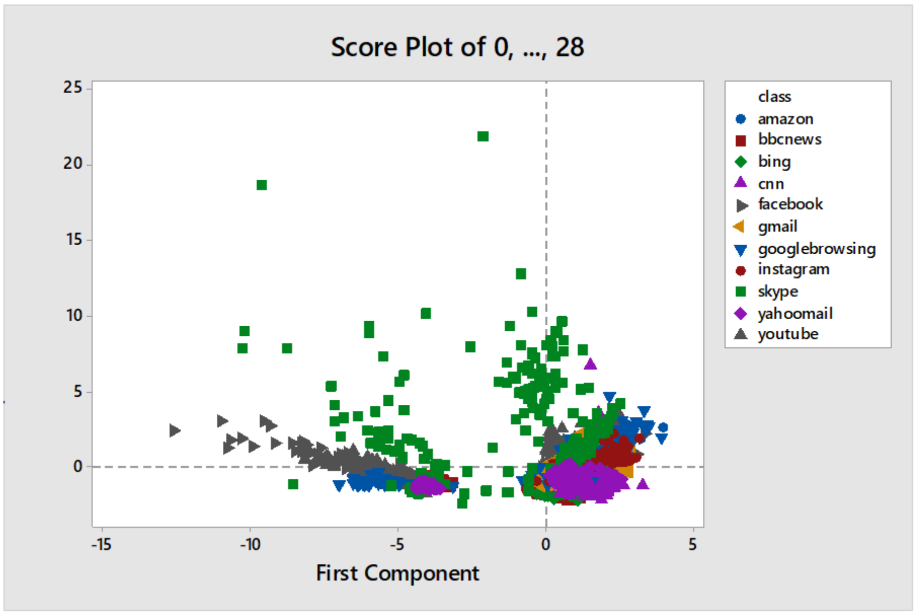


Figure 6-5: Score plot

6.4 T-test

Another categorical variable in the data set is a user; the robust features must be not affected by the user behaviour when the application is browsed by a different user. An analysis is applied based on the same data set that were collected from ten different users that browsing eleven applications. The analysis is based on two-sample t-test that tests whether there is a difference between user1 and remaining nine users (the values are chosen randomly from the 9 users). This test calculated the possible difference of the mean values of feature (sd_burst_data_no_a) between user1 when he/she browsed the Amazon web site versus other users. A P-value was calculated for user1 against other users and the same calculations were applied on user2 against remaining users and so on until user10. The all P-values for ten users present in Table 6-3; all the P-values for the users are greater than 0.005 except for user4 and user5 where the P-values are less than 0.005. Based on a study [169], it claimed that the scientists propose that the default P-value should change from 0.05 to 0.005 for statistical significance. The reason for this change as the traditional threshold (0.05) produces a high false discovery even when there are no errors in the statistical analysis and experimental design. This is obvious from Figure 6-6 that shows the distributions of the feature values for ten users. Similarly, the user4 and user5 show different distributions, while the others show nearly identical distributions. The reason that user4 and user5 do not follow the same pattern of other users could be that these users have different behaviour from the others or these users did not follow the instructions that were set by the researcher (controlled data, see page 83). For instance, the researcher set a period for browsing between (2-5) minutes, changing in the browsing period could effect on the user's behaviour. Moreover, the user's behaviour in this experiment based on only one feature (sd_burst_data_no_a), using different feature or set of features could lead to different results.

Table 6-3: The P-values for ten users

Users	User1	User2	User3	User4	User5	User6	User7	User8	User9	User10
P-Value	0.01	0.69	0.23	2.249e-11	0.0001	0.56	0.98	0.30	0.02	0.53

6.5 Conclusion

This chapter analysed the controlled data set that was presented in section 5.2.1 in order to decrease the number of features. A hierarchical clustering technique is used for features reduction by exploring the correlation between them. The analysis found possible correlations between these features that were reduced from 199 to 84 and to only 29. The reduced features improved a discrimination among the eleven applications rather than the entire features. The analysis also found that the proposed features contributed more in the clustering rather than the conventional ones. Additional analysis was applied on the categorical variable (user) using only one feature (sd_burst_data_no_a). The results showed that this feature can be affected by a user behaviour when different users browse the same application. Using different feature or set of features could lead to different conclusion, therefore, more investigations are needed to prove whether a user's behaviour is affected or not by the proposed features. Next chapter evaluates the collected data using different machine learning algorithms.

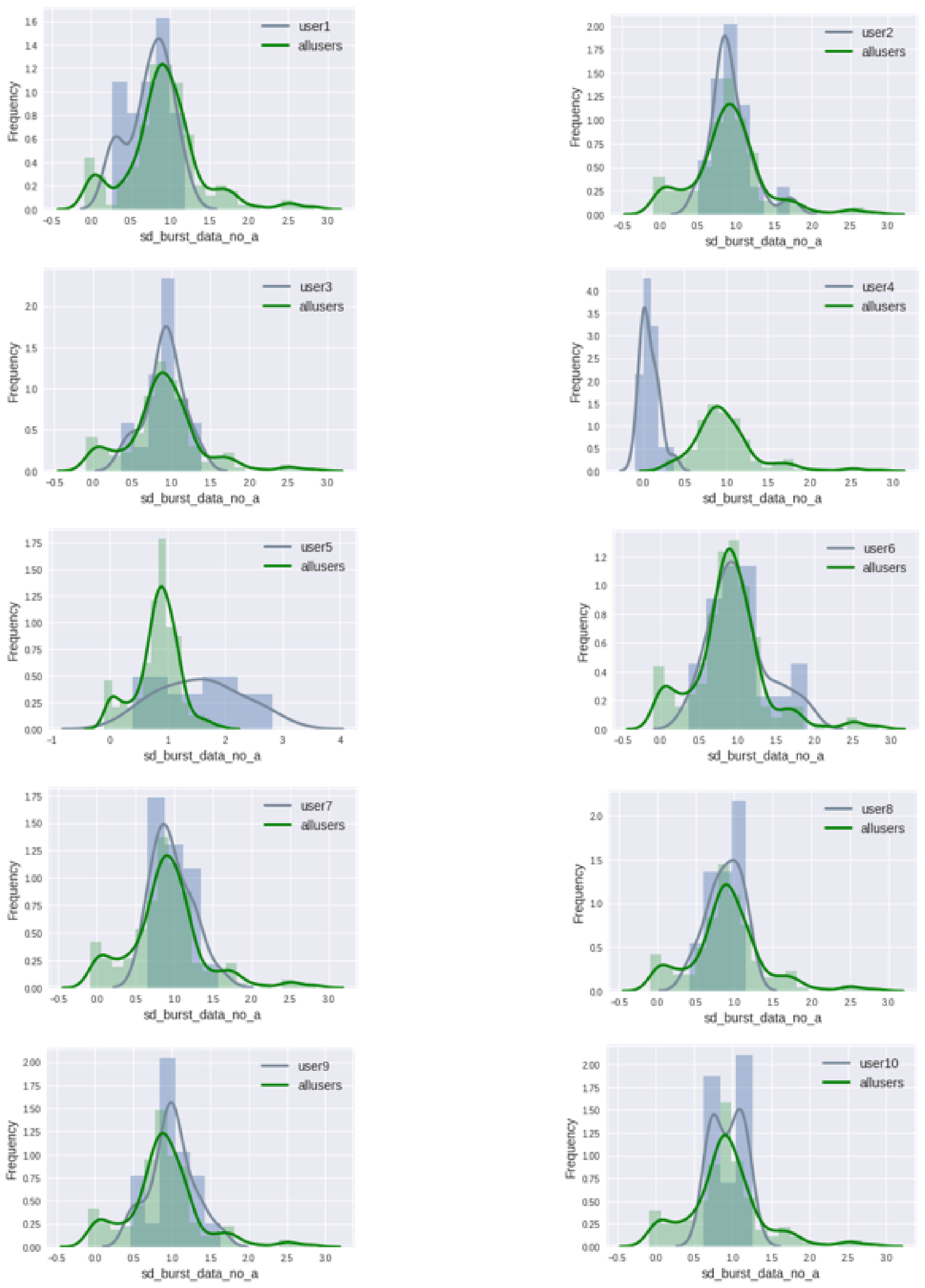


Figure 6-6: Distributions of ten users for Amazon application

7 Evaluation

7.1 Introduction

Building upon the previous chapters that highlighted the features, data collection, and the proposed design, this chapter proceeds to evaluate the proposed method using appropriate classifiers. The chapter investigates whether features associated with packet arrival timing can be used to identify network applications based on their traffic and timing patterns. The evaluation started with the controlled data, which was collected under strict policy, using four machine-learning algorithms. This data represents the truth table for the next step evaluation for the uncontrolled data using C5.0 classifier.

This chapter addresses these aims:

1. Continuing investigating whether burstiness-based features are discriminant for identifying more network applications based on the traffic that they exchange.
2. Investigating the efficiency of burstiness-based features versus traditional flow- and volume-based features for identifying network applications.
3. Investigating the uniqueness behaviour of each application based on the proposed new features.

7.2 Controlled environment evaluation

A first experiment is conducted by using the data that was collected under control environment as showed in the section 5.2.1. Four supervised machine-learning algorithms (i.e., Gradient Boosting (GB), Random forest, SVM and the C5.0) were applied on the controlled data using four different feature sets. The first features set (set1) included 29 features, which were obtained from the analysis in the previous chapter. The second features set (set2) contained the features that were suggested from the previous studies, which were presented in **section 4.2.3**, table 4.3 (17 features multiply by 4 the four statistical operations, the total 68). The third features set (set3) consisted of the burstiness and idle time features, which were introduced in **section 4.2.1**, table 4.1(32 features multiply by 4, plus the first three flow features in table 4.2); while the forth features set (set4) combined the last two feature sets (set2 & set3). Cross validation technique was used in these classifiers for training and testing the model with five folds as ratio 4/1 respectively. Moreover, a grid search technique was utilized for hyper parameter tuning by evaluating the model through the best combination of these parameters. The best combinations of parameters for the GB, Random forest, and SVM were (learning rate = 0.1, Maximum depth = 5, Max leaf nodes = 40, Number of estimators = 100); (Maximum depth = 8, Max leaf nodes = 40, Number of estimators = 100); (C = 1, Kernel = rbf) respectivily. The accuracies for the models with execution times are illustrated in Table 7-1. High performance was recorded for the gradient boosting in classifying different applications for four feature sets compared with low accuracy for the SVM classifier. However, the time consuming for using the gradient boosting is much more from the other classifiers. The best efficiency regarding accuracy and time was for Random forest. The gradient boosting classifier shows a similarity in accurses for the first three feature sets. In other words, the features that were proposed by the prior studies achieved similar accuracy compared with the proposed features by this work. Combing the last two feature sets as in set4 slightly improved accuracy, but with more processing time in traffic classification. Overall, increasing the number of features leads to marginal increase in accuracies for all classifiers to identify 11 applications. The best choice that compromise between accuracy and time is for set3 and random forest classifier, which is achieved 94.51 within 6.2 s.

Table 7-1: Accuracies for the first three classifiers

	Features	Gradient Boosting	Time	SVM	Time	Random forest	Time
Set1	29	94.06	45.2s	87.42	3.3s	92.21	3.5s
Set2	68	94.75	1.5min	87.42	3.3s	92.60	5.3s
Set3	131	94.54	2.3min	86.93	10.7s	**94.51**	6.2s
Set4	199	95.69	3.5min	91.72	14.5s	94.51	8.2s

In contrast, higher accuracy was achieved using the C5.0 model with different boosting values (i.e., 0, 10, 15, 20, 50, and 100) that improved the performance of the classifier. The results of the classifier are presented in Table 7-2. Set1, which included only 29 of the selected features, resulted in a maximum accuracy 95.82% with a maximum allowed boosting factor of 100, but with longest time 10.6 s. The reasonable result regarding the accuracy and time was when the boosting factor is 15 with high accuracy 95.55%. Set2, which contained features of prior studies, resulting in slightly increase in accuracy and processing time when compared to set1 with boosting factor 10 with accuracy 96.18. Set3, which included the proposed features, and resulted in a considerably improved accuracy of 96.91% and at the same boosting factor. Finally, set4 incorporating (set2 & set3) led to a maximum accuracy of 97.36 % with 10 times boost. the results signified that the features related to the burstiness and idle time have high efficiency in discriminating the different applications. Combining both sets showed considerable improvement in classification accuracy peaking at (97.4%). The proposed features showed the ability to better description for the applications than the other parameters, which enhance the classifier capability. Similarly, the top features showed high accuracy and very reasonable execution time. Setting the boosting value at 10 achieved high improvement in accuracy for all data sets, while the following boosting values (i.e., 15, 20, 50, and 100) showed slightly improvement in accuracy for all data sets.

Table 7-2: Accuracy for the C5.0 classifier

	No boost	**Time (sec)**	**Boost 10**	**Time (sec)**	**Boost 15**	**Time (sec)**	**Boost 20**	**Time (sec)**	**Boost 50**	**Time (sec)**	**Boost 100**	**Time (sec)**
Set1	89.45	0.1	94.82	1.1	**95.55**	1.9	95.27	2.3	95.36	5.7	95.82	10.6
Set2	90	0.4	95.45	3.7	**96.18**	5.1	96.18	6	96.55	16.7	96.73	39.6
Set3	88.55	0.8	96.36	6.1	**96.91**	9.1	96.73	10.5	96.73	27.7	96.64	58.1
Set4	89.82	1.3	**97.36**	10.4	96.82	13.3	97.09	16.5	97.45	41	97.36	102.9

C5.0 has an advantage that displays the percentage usage of each attribute that used in building the classifier in training stage. Table 7-3 shows the most attributes as percentage that contribute in the classifier using set4 and at boosting factor 10. In decision tree, the most frequently attribute used is at the root (i.e. high percentage), while the less used when an attribute is further down the tree (less percentage). The table displays a strong availability usage by the classifier for the proposed features compared in low usage for the features that were proposed by prior studies.

Table 7-3: Percentage attributes usage in C 5.0 classifier

Percentage usage	**Proposed attributes**
100%	mean_burst_data_no_a; max_burst_size_bytes_data_b; max_AVG_burst_size_bytes_data_b; md_burst_duration_a; No.of.connections
98.36%	sd_pkt_count_a
97.86%	md_AVG_burst_size_bytes_data_a
92.14%	max_inter_arrival_time_data_a
89.05%	No.of.conns.in.bursts
72.73%	max_idle_time_data_b
70.50%	max_idle_time_data_a
63.36%	mean_burst_size_bytes_data_b.burst_size_bytes_data_a
Percentage usage	**Prior studies attributes**
94.23%	md_first_pkt_b
64.27%	mean_flow_duration_b
62.95%	mean_data_packets_a

The accuracy represents only the ratio of correctly classified instances versus all instances. For further investigation in the performance of the classifier across all applications, Table 7-4 presents the confusion matrix table to describe the performance of the classifier for each class. The row shows the instances in the predicted class while column shows the instances in the actual class. The diagonal of the matrix represents the number of samples that are correctly classified as interest class and called True Positive (TP). The rest of the values in the row of each application are misclassified False Positives (FP), and the rest of the values in the column of each application are misclassified False Negatives (FN). The overall performance of the classifier is considerably high for all applications except for the Bing application. Out of the total tested samples, it was observed that Amazon had the least number of false negatives and zero for Gmail and Skype. The reason for having these applications high classification accuracy could be attributed due to that they have unique behaviour from the others. The applications performing the lowest in terms of classification were Bing and Google. For application Bing, a significant number of samples were misclassified as CNN. In addition, for application, Google was mismatched as Bing, Gmail, Yahoo mail and YouTube. This was due to that the Google application could be as a background search engine for many applications. Other applications also performed rather well, only having two samples classified as false negatives. Overall, the accuracy of all applications was satisfactorily high.

Table 7-4: Confusion matrix for all features

Apps	Amazon	BBC	Bing	CNN		Facebook	G-mail	Google	Instagram	Skype	Y-mail	YouTube
Amazon	99	0	2	0		0	0	0	0	0	0	0
BBC	0	98	2	2		0	0	0	0	0	0	0
Bing	1	0	90	0		0	0	1	0	0	1	0
CNN	0	1	5	98		0	0	0	0	0	0	0
Facebook	0	0	0	0		98	0	0	2	0	1	0
G-mail	0	0	0	0		0	100	2	0	0	0	1
Google	0	0	1	0		1	0	95	0	0	0	0
Instagram	0	0	0	0		0	0	0	98	0	0	0
Skype	0	0	0	0		0	0	0	0	100	0	0
Y-mail	0	1	0	0		0	0	1	0	0	98	2
YouTube	0	0	0	0		1	0	1	0	0	0	97

7.3 Uncontrolled data

The final dataset that obtained from chapter 5 in **section 5.2.6** contained nine Internet appliations (i.e. BBC news, Facebook, Google, Yahoo mail, You tube, G-mail, Amazon, University of Plymouth website, and Bing) with 3200 sesssions. This dataset included all the features that were introduced in chapter 4 , to do comparison between previous studies and current study, this dataset was divided into three subsets features. The first subset (set1) contained the features that were suggested from the previous studies, which were presented in **section 4.2.3**, table 4.3 (17 features multiply by 5 the five statistical operations, the total 85 features). The second subset (set2) consisted of the burstiness and idle time features between packets, which were introduced in **section 4.2.1**, table 4.1(32 features multiply by five, 160 features, plus 18 flow features, which were presented in **section 4.2.2** in table 4.2, the total 178); while the third subset (set3) combined the both.

7.3.1 **Feature selection**

Feature selection approach was applied using random forest on three subsets before classification stage. This approach ranks features from the most significant ones that mostly contribute in building the classifier to least significant ones that have low impact. Therefore, only features that have high importance in discriminating different applications were used to build a C5.0 classifier in training stage. The importance measure of the random forest based on a given feature is being biased significantly into correlated predictor variables [170]. Random forest algorithm was implemented using Python script on the three subsets for feature selection; for all sets, the features were ranked from most important features to low significant ones, (see appendix A, script 4 page 189 for more details about implementation of Python script for Feature selection using Random Forest algorithm).The top 15 features are illustrated in Figure 7-1 for set3, which contained the entire features. The figure shows how significant the impact of the proposed attributes in building the model, 13 of the top 15 features are selected from the proposed features, while only 2 features belong to the conventional features. Figure 7-2 (A) shows the top-ranked attribute, which is the Min (burst-duration-data-a/Packet-data-a). The figure depicts different distribution for the applications for the examined feature. Although, there are some similarities in the variability across the YouTube, Google, and G-mail, they belong to the same company (i.e. Google) as a colocation of servers within the same IP network. In contrast, Figure 7-2 (B) illustrates the lowest ranked feature, which is the burst-duration-data-b. Although this feature contributes the lowest in the random forest, it shows a little variability only for the four applications that were mentioned recently.

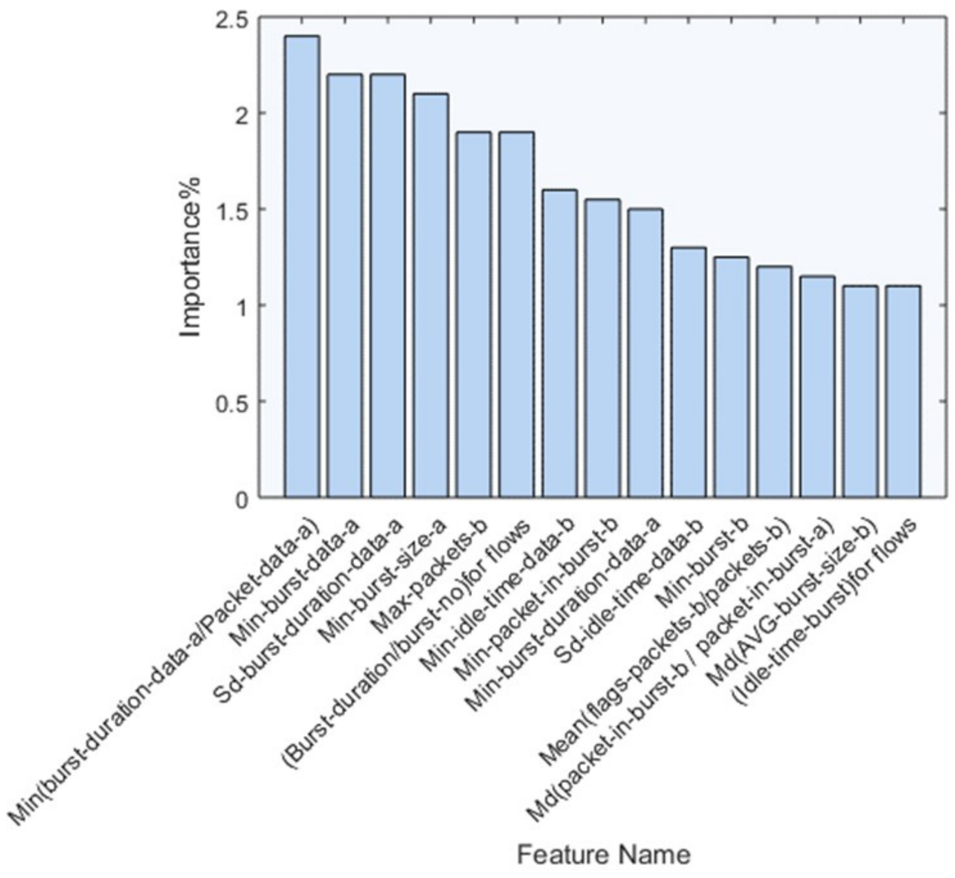


Figure 7-1: Top 15 attributes ranked in Random Forest classifier

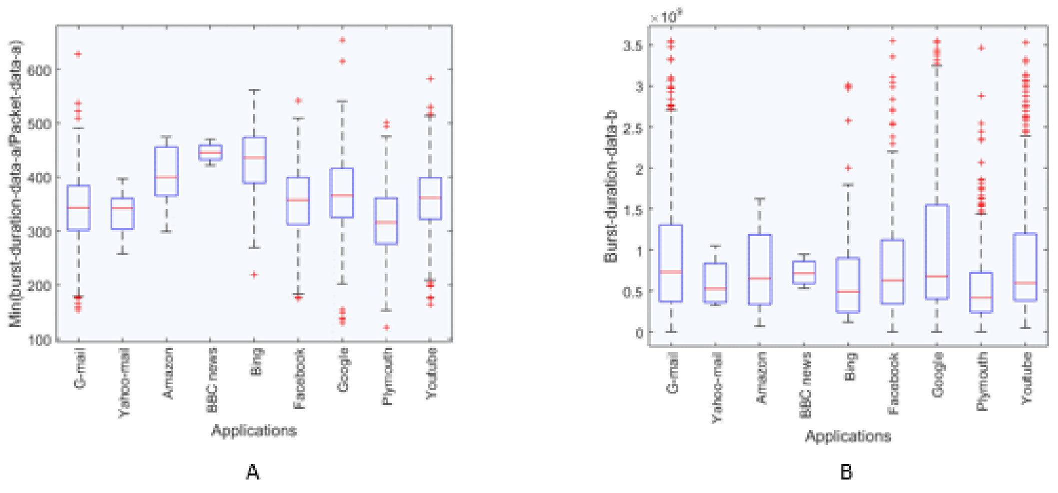


Figure 7-2: Behaviour of eleven applications for most significant feature and the lowest one

7.3.2 C5.0 decision tree classifier

The effectiveness of the suitable features in classifying traffic activities was examined using a set of preliminarily experiments. In the previous section, for all data sets, the features were ranked from most significant feature to the lowest one. Different ranges of the top-ranked features for each data set were taken to explore the performance of these features using C5.0 algorithm. Table 7-5 shows the resulting predication accuracy for training data sets and for different ranges of the top-ranked features. As shown from the figure that the overall accuracy increases when the number of features increases. However, for set2 with the range of features from 70-100 and 120-178, the accuracy decreases from 46.49% to 43.01% and from 48.55% to 45.78% respectively. Therefore, these ranges of features were removed from the set2 that reduced the features from 178 to 90 and achieves high accuracy. Apart from this, increasing the top-ranked features that were obtained from the random forest technique (i.e. set1 and set3) does increase very slightly in terms of C5.0 accuracy.

Table 7-5: Accuracies for different feature sets using C5.0

Set1	**Features**	20	40	60	85			
	Accuracy	45.84	45.93	46.36	47.77			
Set2	**Features**	35	70	100	120	178	90	
	Accuracy	44.83	46.49	43.01	48.55	45.78	49.30	
Set3	**Features**	47	86	122	162	193	263	
	Accuracy	46.94	47.12	48.55	47.21	50.16	52.55	

After applying the features selection for the three data sets, cross validation technique was used in C5.0 classifier for training and testing the model with three folds as ratio 2/1 respectively. This technique partitioned the data into three equal parts; the model was trained on two parts of the data and tested on the remaining part. The process was repeated three times on different parts and the error was calculated by taking the average of all errors. This ratio is different from the ratio for the controlled data, which was 4/1, as the number of samples for some applications were very few. For example, the number of samples for Yahoo mail and BBC web site were 9 and 6 respectively. The classification algorithm was applied to all three feature sets with six different boosting values (0, 10, 15, 20, 50 and 100). The results in Table 7-6 indicate low accuracy for the set1 compared to set2 as the burstiness features increase the efficiency of the classifier in discriminating the different applications. Combining both sets showed considerable improvement in classification accuracy raising up to (79.68% at boost 10). The proposed features showed the ability for better discriminating among the applications in comparison with the other features, which enhances the classifier capability. Table 7-7 compares the number of basic and burstiness features that were used by C5.0 classifier. The burstiness attributes reported superiority in segregating the applications. This is another indicator showing that the classifier strongly relied on the proposed features (i.e. burstiness features) because they provide high discrimination amongst applications.

Table 7-6: Average accuracies with different feature sets using cross validation

Boosting	0	10	15	20	50	100
Set1	47.77	56.56	58.05	58.54	60.30	**60.31**
Set2	49.30	58.75	60.21	61.11	64.23	**65.51**
Set3	52.55	**79.73**	73.99	67.78	68.10	67.13

Table 7-7: Attributes usage in C 5.0 classifier

Basic features usage (75-100)%
data_packets$_{[min, max]}$, flow_duration$_{[mean, min]}$, flags_packets$_{[mean, min, max]}$, inter_arrival_time_data$_{[sd, min]}$
Burstiness features usage (75-100)%
burst_size_bytes$_{[md, min, mean]}$, burst_no$_{[sd, min]}$, idle_time_data$_{[mean, min, sd]}$, pkt_data_count$_{[min, mean]}$, pkt_count$_{[min, sd]}$, inter_arrival_time_burst_conns$_{[min, sd]}$, inter_arrival_time_burst$_{[mean, max]}$, burst_size_bytes_data$_{[max, min, mean]}$, burst_duration$_{[sd, mean]}$, burst_data_no$_{[min]}$

7.3.3 Confusion Matrix

The accuracy, as presented in the previous section, represents only the ratio of correctly classified samples versus all samples. For further analysis in the performance of the classifier, Table 7-8 presents the confusion matrix, with the predicted class on the rows and the actual class on the columns. The overall performance of the classifier is high for all applications except for the Google applications (i.e. Gmail, YouTube, and Google search engine). Out of the total tested samples, it was observed that lowest rate of false negatives was for the University of Plymouth website, out from 70 samples for this application, only five samples classified as G-mail. While all samples (3, 11, 2) for Yahoo mail, Amazon and BBC news respectively were classified correctly. On the other hand, the Google applications (Gmail, YouTube, and Google) performed the worst in terms of classification, as they belong to the same company and they were misclassified as each other.

Table 7-8: Confusion Matrix results for optimal classifier

Applications	Gmail	Y-mail	Amazon	BBC	Bing	Facebook	Google	UoP site	YouTube
Gmail	198	0	0	0	3	6	14	5	13
Ymail	0	4	0	0	0	0	0	0	0
Amazon	0	0	9	0	0	0	0	0	0
BBC	0	0	0	2	0	0	0	0	0
Bing	4	0	0	0	20	2	16	0	7
Facebook	14	0	0	0	5	82	0	0	8
Google	32	0	2	0	2	2	247	0	12
UoP site	11	0	0	0	0	3	0	90	1
YouTube	30	0	0	0	4	0	20	0	198

As showed from the accuracies in tables (7.2 for controlled data & 7.6 for uncontrolled data) that the proposed features have the same impact of conventional features in classifying traffic activities. Merging both features showed significant improvement in the results that majority of them contributed strongly in building the classifiers as showed in tables (Table 7-3 for controlled data, Table 7-7 for uncontrolled data). However, some applications resulted in low accuracies such as Google, G-mail and YouTube as they belong to the same owner. This is one of the limitations of this work that relied on IP addresses and DNS in labelling Internet applications. In other words, the existence of CDN technology in hosting different applications leads to inaccurate results when using IP addresses and DNS in identification[44, 154]. Moreover, the data traffic was collected at the University of Plymouth and from managed-computers owned by the university. They run web-based services in the background that add noise to captured traffic and this case do not happen using Linux operating system. A large dataset with different types of applications would be better to investigate more in performance of the proposed work. Moreover, finding a more accurate method for labelling traffic that enhances the accuracy and leads to clear analysis in different applications.

7.4 Conclusion

This chapter evaluated two types of data (controlled and uncontrolled) using four ML algorithms to investigate that the proposed features able to identify network applications based on their traffic and timing patterns. The study compared the proposed features with the features of prior studies, the results showed very high accuracy for the proposed features in segregating the different traffic activities regarding the controlled data. C5.0 classifier recorded higher accuracy compared with the others classifiers used reached to more than 97%. In addition, the proposed features contributed in the classifier usage more than the prior studies features. For the uncontrolled data, overall accuracy was more than 79%; however, some applications resulted in low accuracies such as Google, G-mail and YouTube as they belong to the same owner. One of the limitations of this work was that the constructing of the truth table for application membership of flows relied on IP addresses and DNS. Unfortunately, due to the underlying CDN hosting of different applications, this classification led to inaccurate results. Moreover, the data traffic was collected at the University of Plymouth and from managed-computers owned by the University and included many web-based services that introduced noise in the collected data. On the other hand, by comparing with results that were achieved by previous studies and obtained high accuracy, most these studies classified traffic according to network protocols such as FTP, IMAP and HTTP or according to application class such as Email, P2P and streaming. These types of traffic are easy to identify and can obtain high accuracy. Based on reviewing the literature, few studies such as [29] that classified modern applications (i.e. Facebook and Google services). However, these studies relied on DPI method for labelling traffic that they used supervised approach for traffic classification. DPI had been considered trustworthy by such studies [116, 132] until in 2009 a study [171] claimed that libraries of DPI are unreliable. Nowadays, current applications are web-based and encrypted; therefore, DPI method cannot cope with modern services as it based on matching payload patterns, IP addresses and port numbers [29].

8 An architecture for application-based management of traffic using SDN

8.1 Introduction

The previous chapters presented novel features for Internet traffic classification. The work proposed a methodology that firstly started the collection of known applications individually in order to examine the validity of the proposed features and to create a database for the next step. Secondly, a real traffic for different Internet applications was collected and labelled based on the IP address and DNS queries. This methodology aimed to build a database that contained flows mapping to their applications. This chapter proposed traffic classification architecture using SDN and this architecture was provided by large database that contained labelled applications. The database required no additional modification or complex hardware to the SDN framework that made the architecture applicable in real time traffic. The architecture exploits software-defined network (SDN) that is capable to route traffic intelligently based on a set of quality of service requirements. SDN does not know which flows belong to which application, therefore, this project provides the correct input to the SDN, which means the correct identification for traffic (i.e. applications). Classifying traffic based on inaccurate method leads to poor identification for the applications; as a result, more resources would be granted to application that does not need them and exclude the suitable resources from the right application. Technically, it might be inaccurate due to misclassification of flows; however, knowing a percentage of traffic flows must be improving the provision for quality of service.

The remainder of this chapter is organized as follows. Section 8.2 presents design requirements, section 8.3 introduces SDN architecture explaining briefly the main steps, and followed by subsections that elaborates the system in more details. Section 8.4 discusses the strengths and weaknesses of the architecture and section 8.5 draws conclusions.

8.2 Design requirements

Inventing new network applications and services such as cloud and virtual machine usage enables the users to access web applications even by using smart phones and iPads that burdens network resources. Internet equipment perform tasks efficiently and independently, however, the network devices become more complex with the growth of Internet. SDN has introduced a solution that simplifies the design of the network devices in which decouples a control plane from a data plane [172]. Also, this approach provides a central management to these devices rather than using traditional tools such as SNMP and CLI [173]. The control plane configures the data plane and programs paths to route flows. In other words, the data flows are forwarding at the data plane based on the information at the control plane. As a results, the traffic classification would be effectively applicable with the presence of SDN. However, applying traffic classification in an enterprise network that contains various venders' devices with their implementations requires sophisticated framework that could be time consuming. The framework should be responsive and efficient with different types of flows. The authors in [174] claimed that there are requirements for traffic classification system which are explained as follows:

1. A framework requires to be consistent for specified types of traffic.
2. Traffic classification must be adaptive for unexpected traffic flows that generated in the network.
3. QoS and traffic engineering need a framework that able to classify traffic before the end of the flow.

Although the SDN platform could fulfil these requirements in terms of managing traffic load efficiently, and lowering a network complexity [175], the application of traffic classification (TC) requires a sophisticated modification in the SDN environment. The authors in [176] acknowledged that building an application identification within SDN can affect either the TC efficacy or can forward performance due to the complexity that being added to the architecture. In addition, the study [174] claimed that adopting TC in the SDN platform could reveal incompatibility issues in protocols or networking devices. Therefore, providing the SDN with known applications flows enables the SDN to prioritize flows according the predefined parameters without adding any complexity in the platform. The following section presented the architecture for application-based management of traffic using SDN.

8.3 Traffic Identification Architecture

Figure 8-1 shows the architecture of the traffic classification with the aid of SDN platform that explains the mechanism of identification; a description of the main steps as follows:

1. User application: the user application refers to a device that initiates a connection with an application server such as YouTube or Facebook.
2. Network devices (data plane): this plane contains devices such as switches and routers that are responsible for forwarding data. These devices contain flow tables and that configured by a controller through OpenFlow protocol.
3. Controller (control plane): this plane configures and updates the flows table to provide best routing paths between server and client based on an application type and predefined requirements.

4. Network application (application plane): this plane contains applications that responsible for performing a modification in network aspects such as polices and behaviour of the network. The database of labelled application flows are resided in the network application, for each freshly established flow, the controller queries the database for possible matching to determine the application type, check policy requirements, prioritize flow in the flows table, and route this flow accordingly. The following subsections present the components and the mechanism in more details.

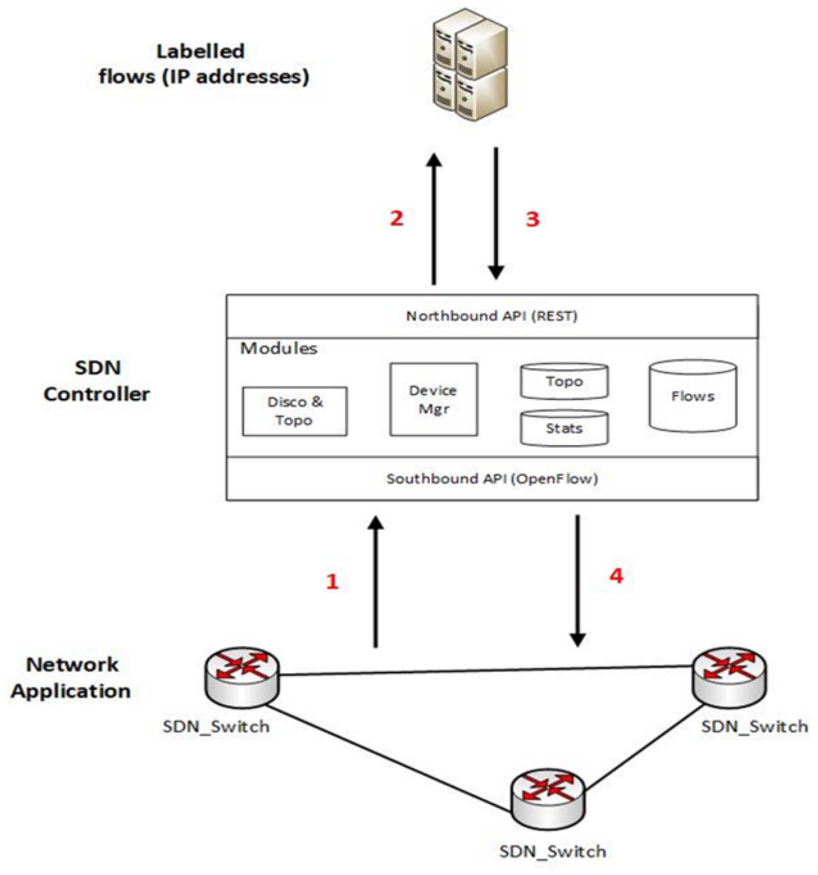


Figure 8-1: SDN architecture with traffic classification

8.3.1 Network devices

This network includes SDN switches and routers that consists of flows table, which deals with any flow entries to instruct the switch to process the flow to best path. In addition, this network contains a secure channel that connects the controller with the switch to transfer packets and commands through OpenFlow protocol. The controller had previously built the flows table through setting suitable rules that downloaded to the device through OpenFlow protocol.

The OpenFlow is a general-purpose protocol that determines the communications (messages and message formats) and exchanges between the controllers (control plane) and switches (data plane) [177]. Furthermore, it defines how the switches to react and respond to commands from the controller. After initiating a connection from a user to request an application, the first packet arrives to an SDN switch that matches this packet with the flows table to execute the appropriate action and forward this packet to a destination. If the packet does not matched with the flows table, it would be forwarded to the controller using southbound API (OpenFlow protocol).

8.3.2 Controller

The controller observes the whole network, executes the policy rules, controls the network devices, and provides two interfaces. The first interface is the southbound that connects the controller with the devices through OpenFlow protocol; the second one is the northbound that connects the controller with the application through REST API. Both controller and network application are participating in implementing policy rules such as forwarding, routing, load balance, and redirection. The controller receives the unmatched flow that can add, delete or update the flows table. The controller uses the network application in order to determine an action regarding this flow; therefore, it sends a flow to the database, which included the labelled application flows, to query the unknown flow through the northbound interface.

8.3.3 Network application

The SDN controller connects with SDN applications via northbound API, these applications execute the above controller. The main tasks of these applications are to configure a best route for a flow between network points, balancing traffic load between different paths, responding to any changing in the network behaviour such as adding new devices or dropping a failure one. The network application contains two databases; the first database includes the applications flows for nine applications that labelled in the previous chapter. The second database keeps the policy requirements for these applications, which defined by the owners of the applications. The controller uses a destination IP address of a received flow to match with the IP addresses in the database; thereby, each packet arrives to the controller queries the database. Consequently, the database server replies with the appropriate application such as YouTube, afterwards, the controller checks appropriate policy requirements for this application. Accordingly, the controller adds new entry in a flow table of a switch with a prioritization order. Hence, the switch can forward the flows that have high priority and to the best route based on application type and its requirements to obtain optimal performance regarding quality of service.

8.4 Strengths and weaknesses of the architecture

The more likely benefits of applying SDN architecture is to route traffic intelligently and with high efficiency. Moreover, it provides a programmable environment for engineers and administrators to configure, manage and prioritize network traffic via API. Thus, labelling application flows in advance and providing them with the optimal set of quality of service parameters, distributing network resources effectively based on applications requirements. In other words, the SDN is configured to prioritize flows according to the applications in addition to their requirements without adding any complexity. The SDN can tag flows by providing it with a database, which contained IP addresses for application flows of nine applications. This database was labelled by applying traffic classification that was explained in the previous chapters. The database is used by the controller to match with a destination IP address of a received flow; therefore, the controller queries the database for each packet entry. Afterwards, the controller checks an appropriate policy requirements, accordingly, the controller adds new entry in a flow table of a switch with a prioritization order. Hence, the switch can forward the flows that have high priority and to a best route based on application type and its requirements to obtain optimal performance regarding quality of service.

In contrast, mapping the IP addresses to applications may be changed over time that leads to incorrect classification; therefore, updating the IP addresses is important. A flow that is based on correct information will be allocated in correct requirements. Otherwise, if the proposed method determined that the IP address of flow has changed assignment from application a to application b, the database will be updated and the next traffic should be provided with the correct set of quality of service provision.

Also, another issue that when the IP address of the flow checks with the database, a conflict more likely to happen for different applications. For instance, the Google services share the same IP address for G-mail and Google search as they are belong to the same company. In this case, a counter can be set to predict the flow correctly. This counter determines the amount of traffic for each application. For example, if the flows are classified within particular time into three applications based on matching process, then the controller deduces that the correct application has the majority of flows.

8.5 Conclusions

The proposed method of traffic classification is applied using SDN framework due to its efficacy and simplicity in managing and routing traffic flows. The method provides the SDN with a large database that contains on nine applications mapping to their IP addresses. Although the accuracy of this mapping is not completely accurate, the identification approach supplies the network with a reasonable portion of labelled flows which improve the quality of service. The controller matches the incoming flows with the IP addresses in the database that make the process of flow identification is suitable in the real time. However, relying on IP addresses in traffic classification could result in misclassified for large flows throughout time as the IP addresses are changing their assignment dynamically to the application. Thereby, updating them frequently is vital for correcting the database that leads to accurate results.

9 Conclusion and Future Work

This chapter summaries the thesis by outlining the main achievements of the research, followed by discussing the limitations of the project. The chapter, also, highlighted the future research directions within Internet traffic classification filed.

9.1 Achievements of the research

Overall, the research has accomplished all the objectives originally stated in Chapter 1, with a series of experimental and analysis undertaken towards the development of characterizing Internet traffic mechanism. The key contributions and achievements of this research are listed as follows:

1. Presenting a review of Internet traffic classification techniques (chapter 3). Many techniques in characterizing and classifying traffic were discussed in a more detail, attention was given for those which are applicable to providing accurate results and require low resources in traffic classification. Statistical and behavioural approaches (section 3.4 and section 3.5 respectively) are the most promising methods within the research community that describe a better view of Internet traffic nowadays. By utilizing these methods, characterizing flows to which they belong can take place rapidly and with high accuracy as the traditional ones are no longer applicable.

2. Proposing a novel feature set that effectively described the application and user behaviour as seen through the generated network traffic (Chapter 4). The project presented parameters such as the on/off data transfer, defining characteristics for a number of typical applications

considering timing and patterns for user events as part of a network application session. This set of features used to discriminate between network applications, based on the statistical differences between inter-arrival times of packets and flows. A concentration has been given to burstiness, which defines closely-spaced data exchanges, such as objects on the same page. Additionally, idle periods, which separate longer-term transactions, such as moving from one page to another when the user is browsing a website. These novel features have been derived based on different distributions of packet size, duration, the distribution of the bursts, and the idle time parameters, which are obtained from various applications. Therefore, this would be generating different amounts of data, creating various connection and timing patterns between the generated packets and flows, beyond the generic distribution of connections for overall traffic.

3. Collecting two types of data, the first data was collected within control environment for eleven applications and for ten users. This type was collected under strict instructions by the researcher in order to build ground truth data for flows mapping to the correct applications. This data became the basis for traffic classification as it drew the behavior of the web applications according to the proposed features. The second type was collected from real traffic network for nine applications and 20 users. The users were browsing the applications based on their preferences and without any restrictions.

4. An experimental investigation and evaluation of the feasibility of the new traffic featues that defined application and user traffic profiling. A series of experiments were carried out on both controlled and real data

for different applications that were accessed by many users. Firstly, a preliminarily experiment was conducted on six applications and six users (chapter 4) that showed high accuracy for the proposed features. The second experiment evaluated more data that contained eleven applications and ten users (chapter 7). Also, the results reflected high ability of the new features to classify these applications. The third experiment was performed on real data that collected from (CSCAN) lab at Plymouth University for two months for nine applications and 20 users (chapter 7). The results exhibited good accuracy and usage for the proposed method.

5. Deep analysis for the proposed features (chapter 6) to determine whether the proposed features have positive impact in discriminating Internet applications. Data analysis aims to explore a correlation and variability amongst the features that led to data reduction in these features. Firstly, the data was initially decreased based on basic calculations (i.e., mean, median, max, min and standard deviation). Afterwards, hierarchical clustering was used to group different features and samples in the data set. The data was analyzed using hierarchical clustering to find the similarity amongst features by drawing a dendrogram. This technique reduced the features from 199 to 29. This technique was applied to the data that consist of 29 features and 2,200 samples for data visualization and to show how the contributions of the selected features in presenting the data.

6. Traffic classification architecture was proposed using SDN, the architecture was provided by large database that contained labelled

applications. This database was contributed to this work through experimentation and analysis that achieved in this thesis. The database required no additional modification or complex hardware to the SDN framework that made the architecture applicable in real time traffic.

A number of papers related to the research project have been presented and published in refereed journal and conferences (provided in Appendix A). As a result, the research is deemed having made positive contributions to the field of Internet traffic, and specifically in classifying Internet applications.

9.2 Limitations of the Research

Although the objectives of the project have been met, a number of limitations associated with the project can be identified. The key limitations of the research are summarised as follows:

1. The data was browsed only through the Google Chrome web browser, using different explorers could effect on the proposed features and consequently on classification accuracy. Moreover, the first data, which was collected under controlled environment, captured under Linux operating system, while the second data, which was collected under uncontrolled environment, captured under Windows operating system, which owned by the University of Plymouth. The two datasets were collecting under different operating systems as the datasets were captured using the University's computers. It was observed that when a data is captured under windows (university) systems, there is still background traffic even a user is not accessing the Internet due to web programs/updates/network broadcasts, etc. continuing in the background

and adding noise to captured traffic. In the case of researcher installed Linux operating systems, the background traffic can be better managed and even stopped due to administrative privilege, improving the ground truth data capturing of individual applications. Therefore, the controlled data was collected under researcher installed Linux OS, and used to build ground truth dataset.

2. Both types of data were collected from the same environment, which was University of Plymouth. Due to the fact that the application flows of real data was labelled based on the IP addresses of the first data, changing the data collection environment could impact on the classification results.
3. The real data was collected at the CSCAN lab that included limited users (20) that affected the collected data as it produced limited browsing sessions for some applications such as BBC news, Yahoo mail, and Amazon included 6, 9, and 34 sessions respectively.
4. The data traffic was collected at the University of Plymouth and from managed-computers owned by the university that added many web-based services that introduced noise in the collected data.
5. Labelling the real data was based on mapping the application flows to the IP addresses and DNS queries and due to the underlying CDN hosting of different applications. This hosting led to inaccurate results in traffic classification.
6. The traffic features generation has been accomplished using fixed thresholds values (i.e. burst_threshold and idle_threshold). However, exploiting dynamic threshold values can possibly change the classification accuracy.

9.3 Scope for Future Work

The research program has enhanced the domain of Internet traffic classification. However, there are a number of areas of future work that could be carried out to further advance the findings of this research. The validation results are promising, but ultimately, there is a room for improving. The details of the suggestion are listed below:

1. Considering larger dataset with different types of applications and more end users in order to fully investigate the performance of the proposed work. Moreover, future work will also focus on recognizing new applications that emerge over time by applying the proposed method.
2. A superior approach for labelling the network traffic can also be incorporated to ensure the robustness of the method.
3. Investigating more in the implications of using the proposed method in traffic prioritization architecture.
4. With introducing new trend applications in internet environment such as web 2.0 and mobile applications, it is important to identify these applications to build standard ground data that contains main objects and classes.

Completing these identified topics of future work would make the classification of network traffic more accurate (low error rates), which would adapt with the continuous changing of networks and applications to manage in precis the future networks.

References

[1] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, and A. Nucci, “DNS to the Rescue: Discerning Content and Services in a Tangled Web,” *Proc. 2012 ACM SIGCOMM Internet Meas. Conf. (IMC ’12)*, pp. 413–426, 2012.

[2] Cisco, “Cisco Visual Networking Index: Forecast and Trends, 2017–2022,” 2019.

[3] N. Al Khater and R. E. Overill, “Network traffic classification techniques and challenges,” *10th Int. Conf. Digit. Inf. Manag. ICDIM 2015*, no. Icdim, pp. 43–48, 2016.

[4] A. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” *Passiv. Act. Netw. Meas.*, 2005.

[5] R. Zou, T. Xu, and H. Hou, “An enhanced Netflow data collection system,” *Proc. 2012 2nd Int. Conf. Instrum. Meas. Comput. Commun. Control. IMCCC 2012*, pp. 508–511, 2012.

[6] A. Boukhtouta, S. A. Mokhov, N. E. Lakhdari, M. Debbabi, and J. Paquet, “Network malware classification comparison using DPI and flow packet headers,” *J. Comput. Virol. Hacking Tech.*, vol. 12, no. 2, pp. 69–100, May 2016.

[7] M. Finsterbusch, C. Richter, E. Rocha, J. A. Müller, and K. Hänßgen, “A survey of payload-based traffic classification approaches,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2014.

[8] B.-R. P. Bujlow T, Carela-Español V, “Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification,” 2014.

[9] A. Bashir, C. Huang, B. Nandy, and N. Seddigh, “Classifying P2P activity in Netflow records: A case study on BitTorrent,” *IEEE Int. Conf. Commun.*, pp. 3018–3023, 2013.

[10] J. M. Reddy and C. Hota, “Heuristic-Based Real-Time P2P Traffic Identification,” *2015 Int. Conf. Emerg. Inf. Technol. Eng. Solut.*, pp. 38–43,

2015.

[11] J. Yan, Z. Wu, H. Luo, and S. Zhang, “P2P traffic identification based on host and flow behaviour characteristics,” *Cybern. Inf. Technol.*, vol. 13, no. 3, pp. 64–76, 2013.

[12] J. Hurley, E. Garcia-Palacios, and S. Sezer, “Host-Based P2P Flow Identification and Use in Real-Time,” *ACM Trans. Web*, vol. 5, no. 2, pp. 1–27, 2011.

[13] M. Perényi, T. D. Dang, A. Gefferth, and S. Molnár, “Identification and Analysis of Peer-to-Peer Traffic,” *J. Commun.*, vol. 1, no. 7, pp. 36–46, 2006.

[14] W. John and S. Tafvelin, “Heuristics to classify Internet backbone traffic based on connection patterns,” *2008 Int. Conf. Inf. Networking, ICOIN*, no. April, 2008.

[15] D. Wang, L. Zhang, Y. Xue, and Y. Dong, “Characterizing Application Behaviors for classifying P2P traffic,” *2014 Int. Conf. Comput. Netw. Commun.*, pp. 21–25, 2014.

[16] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, “Traffic classification through simple statistical fingerprinting,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, p. 5, 2007.

[17] R. Alshammari and A. N. Zincir-Heywood, “How Robust Can a Machine Learning Approach Be for Classifying Encrypted VoIP?,” *J. Netw. Syst. Manag.*, vol. 23, no. 4, pp. 830–869, 2015.

[18] A. Ulliac and B. V Ghita, “Non-Intrusive Identification of Peer-to-Peer Traffic,” in *2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*, pp. 175–183.

[19] Y. Hong, C. Huang, B. Nandy, and N. Seddigh, “Iterative-Tuning Support Vector Machine For Network Traffic Classification,” *Integr. Netw. Manag. (IM), 2015 IFIP/IEEE Int. Symp. (pp. 458-466). IEEE*, pp. 458–466, 2015.

[20] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, “Network traffic classification using correlation information,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 104–117, 2013.

[21] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, “A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks,” *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 1–34, 2012.

[22] G. Piraisoody, C. Huang, B. Nandy, and N. Seddigh, “Classification of applications in HTTP tunnels,” *Proc. 2013 IEEE 2nd Int. Conf. Cloud Networking, CloudNet 2013*, pp. 67–74, 2013.

[23] H. Alizadeh, “Traffic Classification and Verification using Unsupervised Learning of Gaussian Mixture Models,” *Meas. Netw. (M&N), 2015 IEEE Int. Work. (pp. 1-6). IEEE*, 2015.

[24] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, “Robust Network Traffic Classification,” *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1257–1270, 2015.

[25] R. Lin, O. Li, Q. Li, and Y. Liu, “Unknown network protocol classification method based on semi-supervised learning,” *Comput. Commun.*, pp. 300–308, 2015.

[26] A. Vlăduţu, D. Comăneci, and C. Dobre, “Internet traffic classification based on flows’ statistical properties with machine learning,” *Int. J. Netw. Manag.*, vol. 27, no. 3, p. e1929, May 2017.

[27] F. R. Taylor, “Evaluation of Supervised Machine Learning for Classifying Video Traffic,” Nova Southeastern University. Retrieved from NSUWorks, College of Engineering and Computing, 2016.

[28] T. Bakhshi and B. Ghita, “On Internet Traffic Classification: A Two-Phased Machine Learning Approach,” *J. Comput. Networks Commun.*, vol. 2016, no. May, 2016.

[29] Z. Aouini, A. Kortebi, Y. Ghamri-Doudane, and I. L. Cherif, “Early classification of residential networks traffic using C5.0 machine learning algorithm,” *IFIP Wirel. Days*, vol. 2018-April, pp. 46–53, 2018.

[30] Cisco, “Cisco IOS NetFlow Version 9 Flow-Record Format,” 2011.

[31] Ben Popper, “Internet traffic jams are widespread in the US, and are probably about to get a lot worse - The Verge,” 2014. [Online]. Available:

https://www.theverge.com/2014/10/31/7138449/m-lab-netflix-comcast-verizon-isp-business-dispute-congestion-traffic-interconnection. [Accessed: 26-Jun-2019].

[32] Incapsula, “Half of all web traffic ‘not human’- study - Digital Intelligence daily digital marketing research,” 2012. [Online]. Available: http://www.digitalstrategyconsulting.com/intelligence/2012/03/half_of_all_web_traffic_not_hu.php. [Accessed: 18-Dec-2016].

[33] Cisco, *WAN and Application Optimization Solution Guide*. San Jose, USA, 2008.

[34] “ntopng – ntop.” [Online]. Available: https://www.ntop.org/products/traffic-analysis/ntop/. [Accessed: 12-Feb-2019].

[35] M. S. Joe Touch; Eliot Lear, Allison Mankin, Markku Kojo, Kumiko Ono and and A. Z. Lars Eggert, Alexey Melnikov, Wes Eddy, “IANA.” [Online]. Available: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. [Accessed: 04-Mar-2016].

[36] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, “BlindBox: Deep Packet Inspection over Encrypted Traffic,” in *In SIGCOMM*, 2015.

[37] T. Bujlow, T. Riaz, and J. M. Pedersen, “A method for classification of network traffic based on C5.0 machine learning algorithm,” in *Proceedings of ICNC*, 2012.

[38] V. C. Español, “Network traffic classification: from theory to practice,” PhD thesis, Barcelona University, 2014.

[39] R. K. Kurose J, *Cmputer Networking A top-Down Approach*. England: Pearson Education Limited, 2013.

[40] Nader F.Mir, *Computer and Communication Networks*. Westford: Pearson Education Limited, 2015.

[41] M. Satyanarayanan, “The Emergence of Edge Computing,” IEEE, 2015.

[42] J. Moura and D. Hutchison, “Review and analysis of networking challenges in cloud computing,” *J. Netw. Comput. Appl.*, vol. 60, pp. 113–129, 2016.

[43] N. L. S. da F. ; R. Boutaba, *Cloud services, networking, and management*.

Wiley-IEEE Press., 2015.

[44] Kate Gerwig, “What is CDN technology, and who are the current CDN providers?,” 2017. [Online]. Available: https://searchnetworking.techtarget.com/tutorial/What-is-CDN-technology-and-who-are-the-current-CDN-providers. [Accessed: 12-Feb-2019].

[45] Incapsula, “What is a CDN? How does a CDN Work?” [Online]. Available: https://www.incapsula.com/cdn-guide/what-is-cdn-how-it-works.html. [Accessed: 12-Feb-2019].

[46] “Number of Internet Users (2016) - Internet Live Stats.” [Online]. Available: http://www.internetlivestats.com/internet-users/. [Accessed: 12-Feb-2019].

[47] Akamai, “CDN Technology.” [Online]. Available: https://www.akamai.com/uk/en/resources/cdn-technology.jsp. [Accessed: 12-Feb-2019].

[48] FBI, “2014 Internet Crime Report,” *Fed. Bur. Investig. Internet Crime Complain. Cent.*, pp. 1–48, 2014.

[49] Verizon, “2015 Verizon Data Breach Report,” 2015. [Online]. Available: http://higherlogicdownload.s3.amazonaws.com/GOVERNANCEPROFESSIONALS/a8892c7c-6297-4149-b9fc-378577d0b150/UploadedImages/Landing Page Documents/DBIR Executive Summary vv 4-10-15.pdf.

[50] T. A. Mattei, “Privacy, Confidentiality, and Security of Health Care Information: Lessons from the Recent WannaCry Cyberattack,” *World Neurosurg.*, vol. 104, pp. 972–974, 2017.

[51] R. K. Kurose James, *Computer Networking A top-Down Approach*. England: Pearson Education Limited, 2013.

[52] R. K. Medhi D., *Network Routing, Algorithms, Protocols, and Architectures*. San Francisco: Morgan Kaufmann, 2007.

[53] Alexa, “Top sites in United Kingdom,” 2016. [Online]. Available: https://www.alexa.com/topsites. [Accessed: 12-Feb-2019].

[54] M. Zink, K. Suh, Y. Gu, and J. Kurose, “Characteristics of YouTube network

traffic at a campus network - Measurements, models, and implications," *Comput. Networks*, vol. 53, no. 4, pp. 501–514, 2009.

[55] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network: measurements and implications," *Computer Science Department Faculty Publication Series. 177.*, 2008.

[56] Alex Hern, "Netflix in row over net neutrality support | Technology | The Guardian," 2015. [Online]. Available: https://www.theguardian.com/technology/2015/mar/05/netflix-row-net-neutrality-support-australia. [Accessed: 12-Feb-2019].

[57] S. E. Middleton and S. Modafferi, "Scalable classification of QoS for real-time interactive applications from IP traffic measurements," *Comput. Networks*, vol. 107, no. Part 1, pp. 121–132, 2016.

[58] P. Wang, S. C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," *Proc. - 2016 IEEE Int. Conf. Serv. Comput. SCC 2016*, pp. 760–765, 2016.

[59] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-Performance Internet Applications," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, p. 2, Aug. 2010.

[60] K. B. Crovella Mark, *Internet Measurement Infrastructure, traffic, & applications*. West Sussex: John Wiley & Sons, 2006.

[61] R. Jain and S. Routhier, "Packet Trains--Measurements and a New Model for Computer Network Traffic," *IEEE J. Sel. Areas Commun.*, 2004.

[62] K. C. Claffy, H. W. Braun, and G. C. Polyzos, "A Parameterizable Methodology for Internet Traffic Flow Profiling," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1481–1494, 1995.

[63] J. Mogul, *efficient use of workstations for passive monitoring of local area network*. ACM Press, 1990.

[64] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, 2010, p. 267.

[65] M. James D, *Network Analysis, Architecture, and Design*. Burlington: Morgan Kaufmann, 2007.

[66] Cisco, "Cisco 10000 Series Router Quality of Service Configuration Guide," 2013.

[67] M. B. Stephen, *Network Management MIBs and MPLS*. Pearson Education Limited, 2003.

[68] K. B. crovella M., *Internet Measurement Infrastructure, traffic, & applications*. West Sussex: John Wiley & Sons, 2006.

[69] Cisco, "Introduction to Cisco IOS® NetFlow," 2012. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.pdf.

[70] Shawn Ostermann, "tcptrace - Official Homepage." [Online]. Available: http://www.tcptrace.org/.

[71] L. Hu and L. Zhang, "Real-time internet traffic identification based on decision tree," *World Autom. Congr.*, pp. 1–3, 2012.

[72] S. S. Lopes Pereira, J. E. Bessa Maia, and J. L. de Castroe Silva, "ITCM: A Real Time Internet Traffic Classifier Monitor," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 6, pp. 23–38, 2014.

[73] H. Zhang, G. Lu, M. T. Qassrawi, Y. Zhang, and X. Yu, "Feature selection for optimizing traffic classification," *Comput. Commun.*, vol. 35, no. 12, pp. 1457–1471, 2012.

[74] D. Z. Moore, Andrew, *Discriminators for use in flow-based classification*. Queen Mary and Westfield College, Department of Computer Science, 2005.

[75] D. S. Valentino Zocca, Gianmario Spacagna, *"Python Deep Learning."* Packt Publishing, Limited. Ebook, 2017.

[76] K. P. Murphy, *"Machine Learning : A Probabilistic Perspective."* MIT Press. Ebook, 2012.

[77] Pratap Dangeti, *"Statistics for Machine Learning."* BIRMINGHAM - MUMBAI: Packt Publishing Ltd, 2017.

[78] Rulequest, “Rulequest research data mining tools,” 2006. [Online]. Available: http://www.rulequest.com/. [Accessed: 12-Jan-2017].

[79] N. Donges, “The Random Forest Algorithm,” 2018. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. [Accessed: 12-Jan-2019].

[80] R. E. Schapire, “The boosting approach to machine learning: an overview,” *Nonlinear Estim. Classif.*, vol. 171, pp. 149–171, 2003.

[81] Y. Freund and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” *J. Comput. Syst. Sci.*, vol. 139, pp. 23–37, 1995.

[82] A. P. Bradley, “The Use of the Area under the Roc Curve in the Evaluation of Machine Learning Algorithms,” *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.

[83] X. Bo, C. Ming, L. Fei, and W. Na, “P2P flows identification method based on listening port,” *Proc. 2009 2nd IEEE Int. Conf. Broadband Netw. Multimed. Technol. IEEE IC-BNMT2009*, pp. 296–300, 2009.

[84] J. Erman and M. Arlitt, “Traffic classification using clustering algorithms,” *2006 SIGCOMM Work.*, pp. 281–286, 2006.

[85] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, “Unsupervised traffic classification using flow statistical properties and IP packet payload,” *J. Comput. Syst. Sci.*, vol. 79, no. 5, pp. 573–585, 2013.

[86] N. F. Huang, G. Y. Jai, and Chao, “Application traffic classification at the early stage by characterizing application rounds,” *Inf. Sci. (Ny).*, vol. 232, no. 22, pp. 130–142, 2013.

[87] A. Fahad, Z. Tari, and Khalil, “An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion,” *Futur. Gener. Comput. Syst.*, vol. 36, pp. 156–169, 2014.

[88] M. Iliofotou, K. Hyun-chul, and Faloutsos, “Graph-based P2P traffic classification at the internet backbone,” *Proc. - IEEE INFOCOM*, 2009.

[89] D. Rossi and S. Valenti, “Fine-grained traffic classification with netflow

data," *Proc. 6th Int. Wirel. Commun. Mob. Comput. Conf. ZZZ - IWCMC '10*, p. 479, 2010.

[90] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 229–240, 2005.

[91] G. Sun, L. Liang, and Chen, "Network traffic classification based on transfer learning," *Comput. Electr. Eng.*, vol. 69, pp. 920–927, 2018.

[92] A. Hajjar, J. Khalife, and J. Díaz-Verdejo, "Network traffic application identification based on message size analysis," *J. Netw. Comput. Appl.*, vol. 58, pp. 130–143, 2015.

[93] C. So-In, "A Survey of Network Traffic Monitoring and Analysis Tools," *Cse 576-06 Comput. Syst. Anal. Proj.*, pp. 1–24, 2009.

[94] A. C. Hall, "A Survey of Network Traffic Classification Techniques," *IEEE Commun. Surv. TUTORIALS, VOL. 10, NO. 4*, pp. 56–76, 2008.

[95] Y. Aun, S. Manickam, and S. Karuppayah, "A review on features' robustness in high diversity mobile traffic classifications," *Int. J. Commun. Networks Inf. Secur.*, vol. 9, no. 2, pp. 294–304, 2017.

[96] J. Khalife, A. Hajjar, and J. Diaz-Verdejo, "A multilevel taxonomy and requirements for an optimal traffic-classification model," *Int. J. Netw. Manag.*, vol. 24, no. 2, pp. 101–120, Mar. 2014.

[97] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surv. Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[98] B. Park, Y. Won, J. Chung, M. Kim, and J. W.-K. Hong, "Fine-grained traffic classification based on functional separation," *Int. J. Netw. Manag.*, vol. 23, no. 5, pp. 350–381, 2013.

[99] S. Valenti, D. Rossi, A. Dainotti, A. Pescap, A. Finamore, and M. Mellia, "Reviewing traffic classification," *TMA*, 2013.

[100] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang, "Internet traffic classification using constrained clustering," *IEEE Trans. Parallel*

Distrib. Syst., vol. 25, no. 11, pp. 2932–2943, 2014.

[101] K. Yogo, R. Shinkuma, T. Konishi, S. Itaya, and S. Doi, "A Survey of Methods for Encrypted Traffic Classification and Analysis," *Int. J. Netw. Manag.*, vol. 00, no. 22, pp. 1–11, 2012.

[102] R. Raveendran, "A Novel Aggregated Statistical Feature Based Accurate Classification For Internet Traffic," *Data Min. Adv. Comput. (SAPIENCE), Int. Conf. (pp. 225-232). IEEE*, 2016.

[103] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "NDPI: Open-source high-speed deep packet inspection," in *IWCMC 2014 - 10th International Wireless Communications and Mobile Computing Conference*, 2014, pp. 617–622.

[104] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," *Proc. 13th Int. Conf. World Wide Web*, p. 521, 2004.

[105] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: Automated Construction of Application Signatures," *Proceeding 2005 ACM SIGCOMM Work. Min. Netw. data - MineNet '05*, no. May, p. 197, 2005.

[106] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," *IMC '04 Proc. 4th ACM SIGCOMM Conf. Internet Meas.*, pp. 121–134, 2004.

[107] J. Datta, N. Kataria, and N. Hubballi, "Network Traffic Classification in Encrypted Environment : A Case Study of Google Hangout," in *Communications (NCC), 2015 Twenty First National Conference*, pp. 1–6.

[108] Yiming Gong, "Identifying P2P users using traffic analysis | Symantec Connect." [Online]. Available: http://www.symantec.com/connect/articles/identifying-p2p-users-using-traffic-analysis. [Accessed: 07-Apr-2016].

[109] A. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surv. Tutorials*, vol. PP, no. 99, p. 1, 2015.

[110] T. Bakhshi and B. Ghita, "Traffic profiling: Evaluating stability in multi-

device user environments,” *Proc. - IEEE 30th Int. Conf. Adv. Inf. Netw. Appl. Work. WAINA 2016*, pp. 731–736, 2016.

[111] P. Pinky and S. E. V. Ewards, “A Survey on IP Traffic Classification Using Machine Learning,” *Int. J. Eng. Res. Appl.*, vol. 3, no. 1, pp. 2099–2104, 2013.

[112] A. Este, F. Gringoli, and L. Salgarelli, “Support Vector Machines For Tcp Traffic Classification,” *J. Comput. Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.

[113] A. Callado, J. Kelner, D. Sadok, C. A. Kamienski, and S. Fernandes, “Better network traffic identification through the independent combination of techniques,” *J. Netw. Comput. Appl.*, vol. 33, no. 4, pp. 433–446, 2010.

[114] J. R. Chandrakant and D. L. Shashikant, “Machine learning based internet traffic recognition with statistical approach,” *India Conf. (INDICON), 2013 Annu. IEEE*, pp. 1–6.

[115] A. Dainotti, A. Pescapé, and C. Sansone, “Early classification of network traffic through multi-classification,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6613 LNCS, pp. 122–135, 2011.

[116] R. Alshammari and a. N. Zincir-Heywood, “Machine learning based encrypted traffic classification: Identifying SSH and Skype,” *IEEE Symp. Comput. Intell. Secur. Def. Appl. CISDA 2009*, 2009.

[117] D. J. Arndt and A. N. Zincir-Heywood, “A Comparison of three machine learning techniques for encrypted network traffic analysis,” *IEEE SSCI 2011 - Symp. Ser. Comput. Intell. - CISDA 2011 2011 IEEE Symp. Comput. Intell. Secur. Def. Appl.*, pp. 107–114, 2011.

[118] T. Bujlow, T. Riaz, and J. Pedersen, “Classification of HTTP traffic based on C5. 0 Machine Learning Algorithm,” *Proc. Fourth IEEE Int. Work. Perform. Eval. Commun. Distrib. Syst. Web-based Serv. Archit. (PEDISWESA 2012)*, pp. 882–887, 2012.

[119] Y. Yao and G. Doretto, “Boosting for transfer learning with multiple sources,” *2010 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*,

no. December, pp. 1855–1862, 2010.

[120] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*. 2010.

[121] C. N. Lu, Y. D. Lin, C. Y. Huang, and Y. C. Lai, “Session level flow classification by packet size distribution and session grouping,” *Proc. - 26th IEEE Int. Conf. Adv. Inf. Netw. Appl. Work. WAINA 2012*, vol. 56, no. 1, pp. 221–226, 2012.

[122] C. N. Lu, C. Y. Huang, Y. D. Lin, and Y. C. Lai, “High performance traffic classification based on message size sequence and distribution,” *J. Netw. Comput. Appl.*, vol. 76, no. April, pp. 60–74, 2016.

[123] A. Hajjar, J. Khalife, and J. Díaz-verdejo, “Network traffic application identification based on message size analysis,” *Elsevier JNCA*, vol. 58, pp. 130–143, 2015.

[124] A. Iacovazzi and A. Baiocchi, “Internet traffic privacy enhancement with masking: Optimization and tradeoffs,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 353–362, 2014.

[125] S. Yoon, J. Park, and M. Kim, “Behavior Signature for Fine-grained Traffic Identification,” *Appl. Math, 9(2L), pp.523-534*, vol. 534, no. 2, pp. 523–534, 2015.

[126] S. Zander, T. Nguyen, and G. Armitage, “Automated traffic classification and application identification using machine learning,” *IEEE Conf. Local Comput. Networks 30th Anniv. (LCN’05)l*, 2005.

[127] J. Erman, A. Mahanti, and M. Arlitt, “Internet traffic identification using machine learning,” *Glob. Telecommun. Conf. GLOBECOM’06. IEEE*, pp. 1–6, 2006.

[128] D. Liu and C.-H. Lung, “P2P traffic identification and optimization using fuzzy c-means clustering,” *IEEE Int. Conf. Fuzzy Syst.*, pp. 2245–2252, 2011.

[129] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, “Flow clustering using machine learning techniques,” *Int. Work. Passiv. Act. Netw. Meas.*, pp. 205–214, 2004.

[130] F. Hernández-Campos, “Statistical clustering of internet communication patterns,” *Proc. 35th Symp. Interface Comput. Sci. Stat. Comput. Sci. Stat.*, vol. 35, 2003.

[131] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, p. 23, 2006.

[132] L. Bernaille, R. Teixeira, L. Bernaille, R. Teixeira, E. Recognition, A. Conference, L. Bernaille, and R. Teixeira, “Early Recognition of Encrypted Applications,” in *in Proc. 8th International Conference, Passive and Active Measurement Conference, Louvain-la-Neuve, Belgium*, 2007.

[133] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Semi-supervised network traffic classification,” *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, p. 370, 2007.

[134] L. Bernaille, R. Teixeira, I. Akodjenou, and Soule, “Traffic Classification On The Fly,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, 2006.

[135] M. M. Alessandro Finamore, “Mining Unclassified Traffic Using Automatic Clustering Techniques,” in *Proc. Third Int’l Traffic Monitoring and Analysis (TMA),* 2011, pp. 150–163.

[136] S. Sun, “A survey of multi-view machine learning,” *Neural Comput. Appl.*, vol. 23, no. 7–8, pp. 2031–2038, 2013.

[137] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Perform. Eval.*, vol. 64, no. 9–12, pp. 1194–1213, 2007.

[138] D. M. Divakaran, L. Su, Y. S. Liau, and V. L. Vrizlynn, “SLIC: Self-Learning Intelligent Classifier for network traffic,” *Comput. Networks*, vol. 91, pp. 283–297, 2015.

[139] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, “Unexpected means of protocol inference,” *Proc. 6th ACM SIGCOMM Internet Meas. - IMC ’06*, p. 313, 2006.

[140] P. Ducange, G. Mannara, F. Marcelloni, R. Pecori, and M. Vecchio, “A novel approach for internet traffic classification based on multi-objective

evolutionary fuzzy classifiers," *IEEE Int. Conf. Fuzzy Syst.*, no. July, pp. 1–6, 2017.

[141] M. Canini, M. Zadnik, and A. W. Moore, "Experience with High-Speed Automated Application-Identi fi cation for Network-Management," *Proc. Fifth ACM/IEEE Symp. Archit. Netw. Comm. Syst.*, 2009.

[142] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, and B. Xie, "Internet traffic clustering with side information," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 1021–1036, 2014.

[143] R. Zhang, "EDW-Voting : Robust Realtime Traffic Classification Combined with Flow Side Information," *2018 Tenth Int. Conf. Adv. Comput. Intell.*, pp. 438–442, 2018.

[144] W. Lu and L. Xue, "A Heuristic-Based Co-clustering Algorithm for the Internet Traffic Classification," *2014 28th Int. Conf. Adv. Inf. Netw. Appl. Work.*, no. 5, pp. 49–54, 2014.

[145] G. Y. Lazarou, J. Baca, V. S. Frost, and J. B. Evans, "Describing Network Traffic Using the Index of Variability," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1672–1683, 2009.

[146] M. Roughan and S. Sen, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," *ACM SIGCOMM Internet Meas. Work. 2004, Taormina, Sicily, Italy*, pp. 135–148, 2004.

[147] S. Shakkottai, N. Brownlee, and kc claffy, "A Study of Burstiness in TCP Flows BT - Passive and Active Network Measurement," *Passiv. Act. Netw. Meas.*, pp. 13–26, 2005.

[148] R. Krzanowski, "Burst (of packets) and burstiness," *66th IETF Meet.*, 2006.

[149] R. Liston, S. Srinivasan, and E. Zegura, "Diversity in DNS performance measures," *Proc. Second ACM SIGCOMM Work. Internet Meas. Work. - IMW '02*, p. 19, 2002.

[150] J. Jung, E. Sit, H. Balakrishnan, and R. M. Dns, "Performance and Effectiveness of Caching," *SIGCOMM Internet Meas. Work. San Fr. CA, Nov*, vol. 10, no. 5, pp. 589–603, 2001.

[151] D. Wessels, “Is your caching resolver polluting the internet?,” *Proc. ACM SIGCOMM Work. Netw. Troubl. Res. theory Oper. Pract. meet malfunctioning Real.*, pp. 271–276, 2004.

[152] D. Whyte, E. Kranakis, and P. Van Oorschot, “DNS-based detection of scanning worms in an enterprise network,” *Netw. Distrib. Syst. Symp.*, no. 1, pp. 1–17, 2005.

[153] D. Plonka and P. Barford, “Flexible Traffic and Host Profiling via DNS Rendezvous,” in *in Proceedings of the SATIN*, 2011.

[154] M. Trevisan, I. Drago, M. Mellia, and M. M. Munafo, “Towards web service classification using addresses and DNS,” *2016 Int. Wirel. Commun. Mob. Comput. Conf. IWCMC 2016*, pp. 38–43, 2016.

[155] L. M. Torres, E. Magana, M. Izal, and D. Morato, “A popularity-aware method for discovering server IP addresses related to websites,” *Glob. Inf. Infrastruct. Symp. GIIS 2013*, 2013.

[156] A. N. Mahmood and Leckie, “An Efficient Clustering Scheme to Exploit Hierarchical Data in Network Traffic Analysis,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 752–767, 2008.

[157] D. R. A. P. Davide Tammaro, Silvio Valenti1, “Exploiting packet sampling measurements for traffic characterization and classification,” *Int. J. Netw. Manag.*, vol. 22, no. 6, pp. 451–476, 2012.

[158] and M. P. P. Foremski, C. Callegari, “DNS-Class Immediate classification of IP flows using DNS,” *Int. J. Netw. Manag*, vol. 24, pp. 272–288, 2014.

[159] N. F. Huang, C. C. Li, C. H. Li, C. C. Chen, C. H. Chen, and I. H. Hsu, “Application identification system for SDN QoS based on machine learning and DNS responses,” *19th Asia-Pacific Netw. Oper. Manag. Symp. Manag. a World Things, APNOMS 2017*, pp. 407–410, 2017.

[160] G. Mamidisetti and G. T. Varma, “Performance Issues of Internet Protocol Versions,” *Int. J. Soft Comput. Eng.*, vol. 3, no. 6, pp. 30–32, 2014.

[161] T. Auld, A. W. Moore, and S. F. Gull, “Bayesian neural networks for internet traffic classification.,” *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 223–239, 2007.

[162] J. Li, S. Zhang, Y. Lu, and J. Yan, “Real-time P2P traffic identification,” *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pp. 2474–2478, 2008.

[163] Y. Zhang, H. Wang, and S. Cheng, “A method for real-time peer-to-peer traffic classification based on C4.5,” *Int. Conf. Commun. Technol. Proceedings, ICCT*, pp. 1192–1195, 2010.

[164] B. V. Ghita, S. M. Furnell, B. M. Lines, and E. C. Ifeachor, “Endpoint study of Internet paths and Web pages transfers,” *Campus-Wide Inf. Syst.*, vol. 20, no. 3, pp. 90–97, 2003.

[165] T. Bakhshi and B. Ghita, “User Traffic Profiling In a Software Defined Networking Context,” *2015 Internet Technol. Appl. ITA 2015 - Proc. 6th Int. Conf.*, no. September, pp. 91–97, 2015.

[166] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, “Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. c, pp. 1–1, 2014.

[167] tcpdump, “Tcpdump/Libpcap public repository,” 2016. [Online]. Available: http://www.tcpdump.org/#. [Accessed: 27-Dec-2016].

[168] G. Wang and T. S. E. Ng, “The impact of virtualization on network performance of Amazon EC2 Data Center,” *Proc. - IEEE INFOCOM*, 2010.

[169] J. Daniel, O. James, A. Brian, A. Kenneth, D. Christopher, D. Thomas, D. Boeck, I. Edward, P. Donald, G. Anthony, D. Jarrod, V. Larry, H. Ho, J. Daniel, P. A. John, J. Holland, E. Scott, A. Don, L. Stephen, H. Timothy, D. Felix, and V. Zandt, “Redefine Statistical Significance,” *Nat. Hum. Behav.*, vol. 2, no. 1, p. 6, 2018.

[170] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, “Conditional variable importance for random forests.,” *BMC Bioinformatics*, vol. 9, p. 307, 2008.

[171] G. Maier, A. Feldmann, V. Paxson, and M. Allman, “On dominant characteristics of residential broadband internet traffic,” *IMC’09, Novemb. 4–6*, p. 90, 2009.

[172] S. Shenker, M. Casado, T. Koponen, and N. McKeown, “The Future of

Networking , and the Past of Protocols Software-Defined Networking," *Open Netw. Summit*, 2011.

[173] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," *Commun. Mag. IEEE, vol. 51, no. 2,* pp. 114–119, 2013.

[174] B. Ng, M. Hayes, and W. K. G. Seah, "Developing a traffic classification platform for enterprise networks with SDN: Experiences & lessons learned," *' Proc. IFIP Netw. Conf. (IFIP Networking), Toulouse, Fr.*, vol. 5, pp. 1–9, 2015.

[175] K. Guerra Perez, X. Yang, S. Scott-Hayward, and S. Sezer, "A configurable packet classification architecture for Software-Defined Networking," *Int. Syst. Chip Conf.*, pp. 353–358, 2014.

[176] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable Architecture for SDN Traffic Classification," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3203–3214, 2018.

[177] T. Bakhshi and B. Ghita, "OpenFlow-enabled user traffic profiling in campus software defined networks," *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, 2016.

APPENDIX- A

Experimental Analysis Scripts for Traffic Classification

1. Tcptrace analysis script to analyse captured traffic and calculate burstiness, idle time and some of conventional features.
2. R-script to calculate flow and remaining conventional features.
3. Python scripts for analysing uncontrolled dataset.
4. Python script for Feature selection using Random Forest for only uncontrolled dataset.
5. Machine Learning Techniques for controlled environment using Gradient Boosting, SVM, and Random Forest.
6. C5.0 classifier for both controlled and uncontrolled (R-script).

1. Tcptrace analysis script to analyse captured traffic and calculate burstiness, idle time and some of conventional features (chapter 4, section 4.2.1, section 4.2.3)

```
#
=============================================================
==============
# This Script is writtin in c langauge by modifying the Tcptrace tool, which is an
open source tool.
# This tool takes packet trace as input and output flows with the proposed features.
# This tool is applied for both controlled and uncontrolled environment
#
=============================================================
==============
# This script was written in C script, which calculates the burstiness and idle time
features
//calculate the first time of each connection added by hussien
first_time_connection = ((ptp_save->first_time.tv_sec * 1000) + (ptp_save-
>first_time.tv_usec));
//fprintf(stdout,"\tfirst packet: %s", ts2ascii(&ptp_save->first_time));
inter_connection_time = first_time_connection - last_time_connection;
if (inter_connection_time < 1000) {
++cur_conn_burst;
++count_conn;
}else if (count_conn >=2){
++burst_conn;
conn_no+=cur_conn_burst;

cur_conn_burst=0;

count_conn=0;

}else {

cur_conn_burst=0;

count_conn=0;

}

fprintf(stdout,"No.of conns: %d",num_tcp_pairs+1);

fprintf(stdout," No.of bursts in activity: %d",burst_conn);

fprintf(stdout," No.of conns in burts: %d\n",conn_no);

//fprintf(stdout," %ld\n", inter_connection_time);

last_time_connection = ((ptp_save->first_time.tv_sec * 1000) + (ptp_save-
>first_time.tv_usec));

}

ptp_save->last_time = current_time;

//the code

if (dir == A2B) {

thisdir = &ptp_save->a2b;

otherdir = &ptp_save->b2a;

//Determine the first time of this direction

if (thisdir-> count_a2b == 0){

thisdir->first_time=current_time;

++thisdir-> count_a2b;}


} else {

thisdir = &ptp_save->b2a;
```

```
otherdir = &ptp_save->a2b;
//Determine the first time of this direction
if (thisdir-> count_b2a == 0){
thisdir->first_time=current_time;
++thisdir-> count_b2a;}
}
/* meta connection stats */
if (SYN_SET(ptcp))
++thisdir->syn_count;
if (RESET_SET(ptcp))
++thisdir->reset_count;
if (FIN_SET(ptcp))
++thisdir->fin_count;
/* calculate data length added by hussein */
tcp_length = getpayloadlength(pip, plast);
tcp_data_length = tcp_length - (4 * TH_OFF(ptcp));
//burst calculation
thisdir->current = ((current_time.tv_sec * 1000) + (current_time.tv_usec));
thisdir->inter_time = thisdir->current- thisdir->last;
//burst calculation for only data packets > 0
if (thisdir->inter_time < 1000){
if (tcp_data_length > 0) {
++thisdir->crt_burst_data;
++thisdir->count_data;
thisdir->burst_size_bytes_data_tmp+=tcp_data_length;
if (thisdir->count_data ==1){
```

```
//Start time of each burst
thisdir->first_time_burst_data=thisdir->current;}
if (thisdir->count_data >1){
//End time of each packet in burst
thisdir->last_time_burst_data=thisdir->current;
thisdir->burst_duration_data_tmp=thisdir->last_time_burst_data-thisdir-
>first_time_burst_data;}
}
} else if (thisdir->count_data >= 2)
{
//Burst number
++thisdir->burst_data_no;
//Number of packets in burst
thisdir->pkt_data_count+= thisdir->crt_burst_data;
//Data size
thisdir->burst_size_bytes_data+=thisdir->burst_size_bytes_data_tmp;
thisdir->burst_duration_data+= thisdir->burst_duration_data_tmp;

//Idle time between bursts for each direction
if (thisdir->inter_time > 10000){
thisdir->idle_time_data+= thisdir->inter_time;}
//Initials the parameters again
thisdir->count_data=0;
thisdir->crt_burst_data=0;
thisdir->burst_size_bytes_data_tmp=0;
thisdir->first_time_burst_data=0;
```

```
thisdir->last_time_burst_data=0;
}else {thisdir->count_data=0;thisdir->crt_burst_data=0;thisdir-
>burst_size_bytes_data_tmp=0;thisdir->
first_time_burst_data=0;thisdir->last_time_burst_data=0;}
//burst calculation for all packets
if (thisdir->inter_time < 1000 )
{
//No.of packets for each burst and set a counter
++thisdir->crt_burst;
++thisdir->count;
//No.of bytes in Bursts in each direction
thisdir->burst_size_bytes_tmp+=tcp_data_length;
if (thisdir->count ==1){
//Start time of each burst
thisdir->first_time_burst=thisdir->current;}
}else if (thisdir->count >=2){
//End time of each burst
thisdir->last_time_burst=thisdir->last;
//No.of bursts for each direction
++thisdir->burst_no;
//Burst duration for each direction
thisdir->burst_duration+= thisdir->last_time_burst - thisdir->first_time_burst;
//No.of packets in bursts for each direction
thisdir->pkt_count+= thisdir->crt_burst;
//Data size
thisdir->burst_size_bytes+=thisdir->burst_size_bytes_tmp;
```

```
//Idle time between bursts for each direction

if (thisdir->inter_time > 1000){

thisdir->idle_time+= thisdir->inter_time;}

3

thisdir->crt_burst=0;

thisdir->count=0;

thisdir->burst_size_bytes_tmp=0;

thisdir->first_time_burst=0;

thisdir->last_time_burst=0;

}else {thisdir->count=0;thisdir->crt_burst=0;thisdir->burst_size_bytes_tmp=0;

thisdir->first_time_burst=0;

thisdir->last_time_burst=0;}

thisdir->last=((current_time.tv_sec * 1000) + (current_time.tv_usec));
```

2. R-script to calculate flow and remaining conventional features (chapter 4, sections 4.2.2 and 4.2.3).
    i. R-script to calculate remaining conventional features

```
#
===========================================================================
===============
# This script calculates additional packet level features
#
===========================================================================
===============
#this script for one activity
#set the directory
#setwd("f:/csv")
path="C:/Users/hjoudah/Dropbox/hussein/python/real data results/samples files/
group_comp_withoutrunR/
echotrace-2018-07-17_09.02.14.pcap/burst_packets_features.csv"
setwd("C:/Users/hjoudah/Dropbox/hussein/python/real data results/samples files/
group_comp_withoutrunR/
echotrace-2018-07-17_09.02.14.pcap/R")
ldf <- list() # creates a list
listcsv <- dir(pattern = "*.csv") # creates the list of all the csv files in the
directory
#loop to read each file
for (k in 1:length(listcsv)){
```

```
#read file

ldf[[k]] <- read.csv(listcsv[k])

print (listcsv[k])

h=ldf[[k]]

h$x <- NULL

h$y <- NULL

#

#complete the claculation of some field

h$packets_b.packets_a<-h$packets_b/h$packets_a

h$data_packets_b.data_packets_a<-h$data_packets_b/h$data_packets_a

h$flags_packets_b.flags_packets_a<-h$flags_packets_b/h$flags_packets_a

h$flags_packets_a.packets_a<-h$flags_packets_a/h$packets_a

h$flags_packets_b.packets_b<-h$flags_packets_b/h$packets_b

h$flow_size_bytes_b.flow_size_bytes_a<-h$flow_size_bytes_b/h$flow_size_bytes_a

h$Avg_flow_size_bytes_a<-h$flow_size_bytes_a/h$data_packets_a

h$Avg_flow_size_bytes_b<-h$flow_size_bytes_b/h$data_packets_b

h$pkt_count_b.pkt_count_a<-h$pkt_count_b/h$pkt_count_a

h$burst_size_bytes_b.burst_size_bytes_a<-h$burst_size_bytes_b/h$burst_size_bytes_a

h$AVG_burst_size_bytes_a<-h$burst_size_bytes_a/h$pkt_data_count_a

h$AVG_burst_size_bytes_b<-h$burst_size_bytes_b/h$pkt_data_count_b

h$inter_arrival_time_burst_a<-h$burst_duration_a/h$pkt_count_a

h$inter_arrival_time_burst_b<-h$burst_duration_b/h$pkt_count_b

h$pkt_data_count_b.pkt_data_count_a<-h$pkt_data_count_b/h$pkt_data_count_a

h$burst_size_bytes_data_b.burst_size_bytes_data_a<-h$burst_size_bytes_data_b/h
$burst_size_bytes_data_a

h$AVG_burst_size_bytes_data_a<-h$burst_size_bytes_data_a/h$pkt_data_count_a
```

```
h$AVG_burst_size_bytes_data_b<-h$burst_size_bytes_data_b/h$pkt_data_count_b
h$inter_arrival_time_data_a<-h$burst_size_bytes_data_a/h$pkt_data_count_a
h$inter_arrival_time_data_b<-h$burst_size_bytes_data_b/h$pkt_data_count_b
#Replace each NAN or infinite with 0
h=rapply( h, f=function(x) ifelse(is.nan(x),0,x), how="replace" )
h=rapply( h, f=function(x) ifelse(is.infinite(x),0,x), how="replace" )
#calculate some statistical features
m<-colMeans(h)
mn<-apply(h,2,min)
mx<-apply(h,2,max)
md<-apply(h,2,median)

sd<-apply(h,2,sd)
#put the varibles in one dataframe
total <- rbind(m,mn,mx,md,sd)
ldf[[k]]=total
#create matrix
ldf[[k]]=as.matrix(sapply(ldf[[k]], as.numeric ))
#this is to convert the rows to one row
ldf[[k]] <- c(t(ldf[[k]]))
names(ldf[[k]]) <- c(outer(colnames(df), rownames(df), paste, sep="."))
#convert the colum to row
ldf[[k]]= t(ldf[[k]])
}
#bind all file and put them in one file
if (k == 30){total <-
```

```
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]],ldf[[8]],
ldf[[9]],ldf[[10]],ldf[[11]],ldf[[12]],ldf[[13]],ldf[[14]],ldf[[15]],ldf[[16]],ldf
[[17]],ldf[[18]],
ldf[[19]],ldf[[20]],ldf[[21]],ldf[[22]],ldf[[23]],ldf[[24]],ldf[[25]],ldf[[26]],ld
f[[27]],ldf[[28]],
ldf[[29]],ldf[[30]])
} else if (k==10) {total <-
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]],ldf[[8]],
ldf[[9]],ldf[[10]])
} else if (k==9) { total <-
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]],ldf[[8]],
ldf[[9]])
} else if (k==8) { total <-
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]],ldf[[8]])
} else if (k==7) { total <-
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]])
} else if (k==6) { total <-
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]])
} else if (k==5) { total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]])
} else if (k==4) { total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]])
} else if (k==3) { total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]])
} else if (k==2) { total <- rbind(ldf[[1]],ldf[[2]])
} else if (k==1) { total <- (ldf[[1]])
}
#total <-
rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]],ldf[[8]])
```

```
#total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]],ldf[[7]])


#total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]],ldf[[6]])
#total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]],ldf[[5]])
#total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]],ldf[[4]])
#total <- rbind(ldf[[1]],ldf[[2]],ldf[[3]])
#total <- rbind(ldf[[1]],ldf[[2]])
#total <- ldf[[1]]
#set new path of directory
#setwd("C:/Users/hjoudah/Dropbox/test")
#write to file
#setwd("C:/Users/hjoudah/Dropbox/hussein/python/real data results/samples files/
sample_21/")
write.table(total,path,sep=",",row.names = FALSE,col.names = FALSE)
```

ii. R-script to calculate flow features

```
#
==========================================================================
===============
# This script to read the flow trace
#
==========================================================================
===============
path = directory_path+folder+'/matched_flows/'
pathr= directory_path+folder+'/R/'
path2=directory_path+folder+'/burst_conns_features.csv'
files = [x for x in os.listdir(path) if x[-3:] == 'csv']
for app in files:
print app
# Define variables
inter_conns_time =[]
cur_conn_burst =0
count_conn = 0
no_burst_in_conns_1 = 0
conns_no_in_burst_1 = 0
all_features =0
packets_no_in_burst_conns_1 =0
packets_data_no_in_burst_conns_1=0
size_burst_conns_1=0
average_size_burst_conns_1=0
```

```
burst_conns_duration_1=0

idle_time_burst_conns_1=0

packets_no_in_burst_conns_1_tmp=0

packets_data_no_in_burst_conns_1_tmp=0

size_burst_conns_1_tmp=0

# Read the file

pd.options.display.float_format = '{:,.6f}'.format

start_time = pd.read_csv(path+app, header = None, sep =',')

#add the second part with micrisecond part

start_time[0] = start_time[0]+start_time[1]/1000000

#drop the microsecond column

start_time.drop([1], axis = 1,inplace = True)

# Calculte burstiness

for index, single_time in enumerate(start_time.iterrows()):

if index != len(start_time) -1:

time = start_time[0][index+1] - start_time[0][index]

if time > 10:

idle_time_burst_conns_1+=time

if time < 1:

cur_conn_burst = cur_conn_burst +1

count_conn = count_conn + 1

packets_no_in_burst_conns_1_tmp+= start_time[2][index]+start_time[3]

[index]

packets_data_no_in_burst_conns_1_tmp += start_time[5][index]

+start_time[6][index]

size_burst_conns_1_tmp+= start_time[13][index]+start_time[14][index]
```

```
#caculate the burst duration
if count_conn == 1:
first_time_conn = start_time[0][index]
else:
last_time_conn = start_time[0][index]
burst_conn_duration_tmp =last_time_conn-first_time_conn
elif count_conn >= 2:
no_burst_in_conns_1 = no_burst_in_conns_1 +1
conns_no_in_burst_1+=cur_conn_burst
packets_no_in_burst_conns_1+=packets_no_in_burst_conns_1_tmp
packets_data_no_in_burst_conns_1+=packets_data_no_in_burst_conn
s_1_tmp
size_burst_conns_1+=size_burst_conns_1_tmp
burst_conns_duration_1+= burst_conn_duration_tmp
cur_conn_burst=0
count_conn=0
packets_no_in_burst_conns_1_tmp=0
packets_data_no_in_burst_conns_1_tmp=0
size_burst_conns_1_tmp=0
else:
cur_conn_burst=0
count_conn=0
packets_no_in_burst_conns_1_tmp=0
packets_data_no_in_burst_conns_1_tmp=0
size_burst_conns_1_tmp=0
inter_conns_time.append(time)
```

```
#Write inter connections time on file
file = open('inter_time_1.txt', 'w')
for line in inter_conns_time:
file.write('%s' % line +'\n')
file.close()
#calculate other features from the above features
try:
average_size_burst_conns_1 = size_burst_conns_1 /
packets_data_no_in_burst_conns_1
except ZeroDivisionError:
os.remove(path+app)
os.remove(pathr+app)
print('this file is
delected:***************************************************************
*************
**********************************************************************
***************
********************************************************************',pa
th+app)
continue
inter_arrival_time_burst_conns_1 = burst_conns_duration_1/
packets_no_in_burst_conns_1
#the second part of program caculate the burstiness and idle time based on
last and first time
#define varibles
inter_conns_time_2 =[]
```

```
cur_conn_burst =0
count_conn = 0
no_burst_in_conns_2 = 0
conns_no_in_burst_2 = 0
all_features =0
packets_no_in_burst_conns_2 =0
packets_data_no_in_burst_conns_2=0
size_burst_conns_2=0
average_size_burst_conns_2=0
burst_conns_duration_2=0
idle_time_burst_conns_2=0
packets_no_in_burst_conns_2_tmp=0
packets_data_no_in_burst_conns_2_tmp=0
size_burst_conns_2_tmp=0
# Read the file
pd.options.display.float_format = '{:,.6f}'.format
start_time = pd.read_csv(path+app, header = None, sep =',')
start_time.drop([60], axis = 1,inplace = True)#for dropping the blank field
#start_time = start_time.loc[:,6:]
start_time[0] = start_time[0]+start_time[1]/1000000
start_time[61] = start_time[61]+start_time[62]/1000000
start_time.drop([1], axis = 1,inplace = True)
start_time.drop([62], axis = 1,inplace = True)
# Calculte burstiness
for index, single_time in enumerate(start_time.iterrows()):
if index != len(start_time) -1:
```

```
if start_time[61][index] > start_time[0][index+1]:
cur_conn_burst = cur_conn_burst +1
count_conn = count_conn + 1
packets_no_in_burst_conns_2_tmp+=start_time[2][index]
+start_time[3][index]
packets_data_no_in_burst_conns_2_tmp+=start_time[5][index]
+start_time[6][index]
size_burst_conns_2_tmp+=start_time[13][index]+start_time[14]
[index]
#caculate the burst duration
if count_conn == 1:
first_time_conn = start_time[0][index]
else:
last_time_conn = start_time[61][index+1]
burst_conn_duration_tmp =last_time_conn-first_time_conn
elif count_conn >= 2:
no_burst_in_conns_2 = no_burst_in_conns_2 +1
conns_no_in_burst_2+=cur_conn_burst
packets_no_in_burst_conns_2+=packets_no_in_burst_conns_2_tmp
packets_data_no_in_burst_conns_2+=packets_data_no_in_burst_conn
s_2_tmp
size_burst_conns_2+=size_burst_conns_2_tmp
burst_conns_duration_2+= burst_conn_duration_tmp
idle_time_burst_conns_2+=start_time[0][index+1] -
start_time[61][index]
cur_conn_burst=0
```

```
count_conn=0
packets_data_no_in_burst_conns_2_tmp=0
packets_no_in_burst_conns_2_tmp=0
size_burst_conns_2_tmp=0
else:
cur_conn_burst=0
count_conn=0
packets_data_no_in_burst_conns_2_tmp=0
packets_no_in_burst_conns_2_tmp=0
size_burst_conns_2_tmp=0
inter_conns_time_2.append(start_time[61][index] - start_time[0][index
+1])
#Write inter connections time on fileon file
file = open('inter_time_2.txt', 'w')
for line in inter_conns_time_2:
file.write('%s' % line +'\n')
file.close()
#calculate other features from the above features
print app
try:#this action is to remove file that is devision by zero
average_size_burst_conns_2 = size_burst_conns_2 /
packets_data_no_in_burst_conns_2
except ZeroDivisionError:
os.remove(path+app)#remove the file if it is devsion by zero from
matchflows
os.remove(pathr+app)#remove the file if it is devsion by zero from R
```

```
print('this file is
delected:*************************************************************
*************
**********************************************************************
***************
',path+app)
continue
inter_arrival_time_burst_conns_2 = burst_conns_duration_2/
packets_no_in_burst_conns_2
#write features on file
features = [no_burst_in_conns_1, conns_no_in_burst_1,
packets_no_in_burst_conns_1, packets_data_no_in_burst_conns_1,
size_burst_conns_1, average_size_burst_conns_1,
burst_conns_duration_1, inter_arrival_time_burst_conns_1,
idle_time_burst_conns_1, no_burst_in_conns_2,
conns_no_in_burst_2, packets_no_in_burst_conns_2,
packets_data_no_in_burst_conns_2, size_burst_conns_2,
average_size_burst_conns_2, burst_conns_duration_2,
inter_arrival_time_burst_conns_2, idle_time_burst_conns_2]
file = open('burst_conns_features_tmp.csv', 'a')
for line in features:
file.write('%s ' % line+',')
file.write(app)
file.write('\n')
file.close()
#put the header and features values togather in file
```

```
headers = ['no_burst_in_conns_1', 'conns_no_in_burst_1',
'packets_no_in_burst_conns_1', 'packets_data_no_in_burst_conns_1',
' size_burst_conns_1' , 'average_size_burst_conns_1 ',
'burst_conns_duration_1', 'inter_arrival_time_burst_conns_1',
'idle_time_burst_conns_1', 'no_burst_in_conns_2',
'conns_no_in_burst_2', 'packets_no_in_burst_conns_2',
'packets_data_no_in_burst_conns_2', ' size_burst_conns_2' ,
'average_size_burst_conns_2 ', 'burst_conns_duration_2',
'inter_arrival_time_burst_conns_2', 'idle_time_burst_conns_2' ]
#burst_conns_features_2 = pd.read_csv('burst_conns_features_2.csv', header =
None, sep =',')
burst_conns_features = open(path2, 'w')
orig = open('burst_conns_features_tmp.csv', 'r')
burst_conns_features.write(','.join(headers) + '\n')
for line in orig.readlines():
burst_conns_features.write(line)
orig.close()
burst_conns_features.close()
os.remove('burst_conns_features_tmp.csv')
##############################################################################
########
```

3. Python scripts for analysing uncontrolled dataset (chapter 5, sections (5.2.2 – 5.2.5)

```
#
==============================================================
===============
# These are gruop of scripts for analysing datasets in uncontrolled environment
#
==============================================================
===============
# This script was writtin in Python to label mixed traffic based on IP address and
DNS queries
## Note:
# We do not need to read both colums for IP, we only read the forth colum for IP
server because the second colum alawys 192.168
# as it is data collected from one client
#the program is run twice, the first run is for determining the requests and the
second one for collect flows until the next request
#************************************************************
# Import some libraraies
import string
import re
import pandas as pd
import os
directory_path ='C:/Users/hjoudah/Dropbox/hussein/python/real data results/samples
```

```
files/group_1/'
#directory_path ='C:/Users/hjoudah/Dropbox/pcaplinux/'
for folder in os.listdir(directory_path):#read the directory files
print 'the name of folder is here ******************************' , folder
#
============================================================
===============
# This script is to read the packet trace
#
============================================================
===============
################ for run only change the folder
#########################################
path1=directory_path+folder+'/matched_flows/'
path2=directory_path+folder+'/R/'
path3=directory_path+folder+'/input_files/'
pd.options.display.float_format = '{:,.6f}'.format#to display the float value
untial six value
# Read the text files, the first one with IP address, while the second one with
domain name(input files)
# note the time stamp of both files should be seconds
text1 = path3+'packet.dat' # contain only IPs address
text2 = path3+'packetn.dat' # contain the domain names
netflow_csv = pd.read_csv(path3+'netflow.csv', header = None, sep =' ')
#netflowdomain_csv = pd.read_csv('netflown.csv', engine='python', header = None)
netflow_csv['known_tag'] = 0
```

```
#
=================================================================
===============
# This script is for DNS queries (section 5.2.2)
#
=================================================================
===============
first_req_detected = 0
request = '.53: ' #the request
netflow_csv[6] = netflow_csv[6]+netflow_csv[7]/1000000 # to combine the seconds
part with the microseconds part
netflow_csv.drop([7], axis = 1,inplace = True)# remove the microsececonds part
from the file
requests =[]
#print(netflowdomain_csv.head())
# The applications keywords, this keywords come with the request of the
application
amazonKeyword = ' www.amazon.com. '
bbcKeyword = ' www.bbc.co.uk. '
bingKeyword = ' www.bing.com. '
cnnKeyword = ' www.cnn.com. '
facebookKeyword = ' www.facebook.com. '
instagramKeyword = ' www.instagram.com. '
yahoomailKeyword = ' login.yahoo.com. '
youtubeKeyword = ' www.youtube.com. '
googleKeyword = ' www.google.com. '
```

```
gmailKeyword = ' accounts.google.com. '
plymouthkeyword = 'www.plymouth.ac.uk.'
#put all the keywords in list to easy to read
keywords = [amazonKeyword, bbcKeyword, bingKeyword, cnnKeyword,
instagramKeyword, yahoomailKeyword, youtubeKeyword, facebookKeyword,
googleKeyword, gmailKeyword, plymouthkeyword]
#Create a file for each application to put the traffic that belong to it
new_ip_dict = {'www_amazon_com':[], 'www_bbc_co_uk':[], 'www_bing_com':[],
'www_cnn_com': [], 'www_instagram_com': [],
'login_yahoo_com': [], 'www_youtube_com': [], 'www_facebook_com':
[], 'www_google_com': [], 'accounts_google_com': [], 'www_plymouth_ac_uk': []}
matched_server_ip_dict = {'www_amazon_com':[], 'www_bbc_co_uk':[],
'www_bing_com':[], 'www_cnn_com': [], 'www_instagram_com': [],
'login_yahoo_com': [], 'www_youtube_com': [],
'www_facebook_com': [], 'www_google_com': [], 'accounts_google_com': [],
'www_plymouth_ac_uk': []}
keyword_application = {'www_amazon_com':['cloudfront.net',
'deploy.static.akamaitechnologies.com.https', 's3-3-w.amazonaws.com.https'],
'www_bbc_co_uk':['an.haven.com.https', '.bbc.co.uk.http',
'www.edigitalsurvey.com.http'],'www_bing_com':['a-0001.a-msedge.net.http'],
'www_cnn_com':
['a23-55-58-227.deploy.static.akamaitechnologies.com.https','west-
1.compute.amazonaws
.com.http','
compute-1.amazonaws.com.https',
'akamaitechnologies.com.http','1e100.net.https','fbcdn.net.https','pixel.quantserve.c
om.http'],
'www_facebook_com':['.fbcdn.net.https',
'.facebook.com.https', '.fbcdn.net.https'],
'www_instagram_com':['instagram-p3-shv-01-
lhr3.fbcdn.net.https','instagram-p3-shv'],
'login_yahoo_com':
['.ycpi.vip.lob.yahoo.com.https','mpr2.ngd.vip.ir2.yahoo.com.https','r1.ycpi.vip.ir2.
yahoo.net.https',
'beap3.cbs.vip.ir2.yahoo.com.https',
'ats1.member.vip.ir2.yahoo.com.https','pr-bh.pbp.vip.ir2.yahoo.com.https',
'public.comet.vip.bf1.yahoo.com.https',
'a2.ue.vip.ir2.yahoo.net.https','gw.iris.vip.bf1.yahoo.com.https',
'e1.ycpi.vip.lob.yahoo.com.https','a1.u
e.vip.ir2.yahoo.net.https'],
'www_youtube_com':['lhr35s05'], 'www_google_com':
['lhr25s','wk-in'], 'accounts_google_com':['lhr35s05'], 'www_plymouth_ac_uk' :
['plymouth']}
#2/0/0 CNAME clients.l.google.com.
#'www_youtube_com':['-in-f14.1e100.net.https','.1e100.net.https','-inf2.1e100.
net.https']
stepIndex = 0
#open the first file
with open(text1, 'rb') as f:
lines = f.read().splitlines()
for index, row in enumerate(lines[stepIndex:]):
if any(keyword in row for keyword in keywords) and request in row:#
```

```
check the keyword and request in the line
# print (index).......................for error check
try:
key = string.replace(row.split(" A? ", 1)[1][:-6], '.', '_')#
the key is the request after add _ to be equel
#to defination of www_applicationname_com
except IndexError:
key = string.replace(row.split(" AAAA? ", 1)[1][:-6], '.',
'_')# the key is the request after add _ to be equel
#to defination of www_applicationname_com
if len(matched_server_ip_dict[key]) > 0:# to check if the file is
empty (i.e., we read a request for one time)
pass
else:
print 'Keyword is found in line :', index, key # printing the
line number and request
with open(text2) as file2:# read the second text file because
it contains domain names
linefile2 = file2.read().splitlines()
if key == 'www_youtube_com':# this is just to check that this
application is youtube
# these words are generated when the Youtube is requested
youtubeServer_1 = '-in-f14.1e100.net.https'
youtubeServer_2 = '.1e100.net.https'
youtubeServer_3 = '-in-f2.1e100.net.https'
youtubeServerCounter = 0
```

```
for lane in linefile2[index:]:#read the text file from the
request
if youtubeServer_1 in lane or youtubeServer_2 in lane
or youtubeServer_3 in lane:# check the words in
#the line
if float(lane[0:10]) - float(row[0:10]) > 120:
break
youtubeServerCounter += 1#count the number of words
if youtubeServerCounter < 500:
continue
elif key == 'www_facebook_com':# this is just to check that
this app is facebook (the same proceture as in YouTube)
facebookServer_1 = '.facebook.com.https'
facebookServer_2 = '.fbcdn.net.https'
facebookServerCounter = 0
for index2,lane in enumerate(linefile2[index:]):
if facebookServer_1 in lane or facebookServer_2 in
lane:
if float(lane[0:10]) - float(row[0:10]) > 120:
break
facebookServerCounter += 1
#print facebookServerCounter
if facebookServerCounter < 500:
continue
elif key == 'www_bbc_co_uk': # this is just to check that
this application is bbc
```

```
#(the same proceture as in YouTube)
bbcServer_1 = 'bbc'
bbcServerCounter = 0
for index2, lane in enumerate(linefile2[index:]):
if bbcServer_1 in lane :
if float(lane[0:10]) - float(row[0:10]) > 120:
break
bbcServerCounter += 1
# print facebookServerCounter
if bbcServerCounter < 500:
continue
elif key == 'www_instagram_com':# this is just to check that
this app is instagram (the same proceture as in YouTube)
instagramServer_1 = 'instagram-p3-shv-01-
lhr3.fbcdn.net.https'
instagramServer_2 = 'instagram-p3-shv'
instagramServerCounter = 0
for index2,lane in enumerate(linefile2[index:]):
if instagramServer_1 in lane or instagramServer_2 in
lane:
if float(lane[0:10]) - float(row[0:10]) > 180:
break
instagramServerCounter += 1
if instagramServerCounter < 1000:
continue
with open(key + '.txt', 'rb') as file:# open the IPs file#and
```

```
this is the start of taking the real key
IPsfile = file.read().splitlines()
#
==============================================================
===============
# IP matching script section 5.2.4
#
==============================================================
===============
# this is to extract each IP from the line and compare it with the IPs_list and
within three seconds----------------
for internalIndex, internalLines in
enumerate(lines[index:]):#this loop to matach the IP in the line (text file)
#with the IP in the IPs file
if internalLines.split()[2][:8] != '192.168.': # take
only the line that is not start by
#192.168(read the first part of the line)
IP = internalLines.split()[2] # read the colum two
IP_only = re.findall(r'[0-9]+(?:\.[0-9]+){3}', IP)
# extract the IP without port
if not IP_only:#this is for that the extracted ip
form make an error
pass
else:#if the ip extracted without error
if IP_only[0] not in IPsfile and
(float(internalLines[0:10]) - float(row[0:10]) < 3):
```

```
new_ip_dict[key].append(IP_only[0])# append
to the file of target application
elif float(internalLines[0:10]) -
float(row[0:10]) > 3:
#start check flows
break
elif internalLines.split()[4][:8] != '192.168.': # take
only the line that is not start by
#192.168(read the second part of the line)
IP = internalLines.split()[4] # read the colum two
IP_only = re.findall(r'[0-9]+(?:\.[0-9]+){3}', IP)
# extract the IP without port and final
if not IP_only:
pass
else:
if IP_only[0] not in IPsfile and
(float(internalLines[0:10]) - float(row[0:10]) < 3):#[0:10] to take
#all the time digits
new_ip_dict[key].append(IP_only[0]) #
append ip to the file of target application
#(update the ips file)
elif float(internalLines[0:10]) -
float(row[0:10]) > 3:# after three seconds the update the ips file
#will stop
break
else:
```

```
pass
stepIndex = index+internalIndex# this is to update the pointer
of the index
#---------------------------------------------------------------------------
-----------------------
#this is to update the ips files from the dictionary ( write the dictionary on
the ips files----------------------
# for Key, value in new_ip_dict.iteritems():#to inter in each
dictionary and make a loop
# if len(new_ip_dict[Key]) > 0:
# with open(key + '.txt', 'a') as IPs_file: # open
the IPs file
# new_ip_dict = {a: list(set(b)) for a, b in
new_ip_dict.items()}##remove duplicate IPs from dict
# for line in new_ip_dict[Key]:
# IPs_file.write('\n' + line)
#----------------------------------------
#--------------- this is to filter the flows from the first reqest, check the
first flow time and compare it if it is greater
#than the first request
if first_req_detected == 0:
netflow_csv = netflow_csv[netflow_csv[6] >=
float(row[0:10])]
first_req_detected = 1
#Match each server IP in the flow file with the IP in the IPs_file and append to
the application file--------------
```

```
#this is for filter the fllows that are greater than request
selected_flows_within_time = netflow_csv[netflow_csv[6] >=
float(row[0:10])]
#netflow_csv.to_csv('matched_flows/netflowwithtime.csv',
header=False)# write the filter flows
#on file netflowwithtime
# this is for matching the filter flows with IPs file
with open(key + '.txt', 'rb') as file: # open the IPs file
IPsfile = file.read().splitlines()
counter_reqest =0
for index, flow in
enumerate(selected_flows_within_time.iterrows()): # this loop to matach the IP in
the line
#(netflow file) with the IP in the IPs file
counter_reqest = counter_reqest+1
if counter_reqest ==1:requests.append(flow[0]+1) #
IPsfile_subnet = ['.'.join(ip.split('.')[:3]) for ip in
IPsfile]# this to split the IP and remove the last
#one and then join only the remaining three parts with
dot.
current_index = flow[0] # read the index
if "192.168" not in flow[1][1]:
#if flow.split()[2][:8] != '192.168.':
server_IP = flow[1][1] # read the colum two(ip
server part)
server_IP = '.'.join(server_IP.split('.')[:3])# this
```

```
to split the IP and remove the last one and then
#join only the remaining three parts with dot.
if (server_IP in IPsfile_subnet) :# to check the ip
(in netflow file) in the ips file
flow = str(flow[1][9]) + ' ' + str(flow[1][10])
+ ' ' + str(flow[1][11]) + ' ' + str(flow[1][12])
+ ' ' + str(flow[1][13])
#flow = str(flow[1][0])+' : '+str(flow[1][1])+'
'+str(flow[1][2])+' : '+str(flow[1][3]+'
'+str(flow[1][4])+' : '+str(flow[1][5]))# build
the flow that we need
matched_server_ip_dict[key].append(flow) #
append to the file of target application
#print flow
selected_flows_within_time =
selected_flows_within_time.loc[selected_flows_within_time.index
!= current_index]#select the flows that are unknown
netflow_csv.set_value(current_index,
'known_tag', 1)#tag each flow with 1 to be known
#netflowdomain_csv.set_value(current_index,
'known_tag', 1)
else:#to check the row in (netflow file) that contain
the keywords in the dictionary of the key
#if key == 'accounts_google_com':
# continue
#print netflow_csv.loc[current_index][4]
```

```
################## this script to check that
the remaining flows match with the keywords section 5.2.5
for app_domain in keyword_application[key]:
#this if below for real data
#if app_domain in
netflow_csv.loc[current_index][4] or app_domain in
#netflow_csv.loc[current_index][8]:
if app_domain in
netflow_csv.loc[current_index][4]:
flow = str(flow[1][9]) + ' ' +
str(flow[1][10]) + ' ' + str(flow[1][11])
+ ' ' + str(flow[1][12]) + ' ' +
str(flow[1][13])
#flow = str(flow[1][0]) + ' : ' +
str(flow[1][1]) + ' ' + str(flow[1][2]) + ' :
' + str(flow[1][3] + ' ' +
str(flow[1][4]) + ' : ' + str(flow[1][5])) # build the flow that we need
matched_server_ip_dict[key].append(fl
ow) # append to the file of target application
selected_flows_within_time =
selected_flows_within_time.loc[selected_flows_within_time.index
!= current_index]# select the flows that are known
netflow_csv.set_value(current_index,
'known_tag',1) # tag each flow with 1 to be known
#netflowdomain_csv.set_value(current_
index, 'known_tag', 1)
```

```
break
elif "192.168" not in flow[1][3]:
server_IP = flow[1][3] # read the colum two(ip
server part)
server_IP = '.'.join(server_IP.split('.')[:3])# this
to split the IP and remove the last one and then join
#only the remaining three parts with dot.
if (server_IP in IPsfile_subnet) :# to check the ip
(in netflow file) in the ips file
flow = str(flow[1][9]) + ' ' + str(flow[1][10])
+ ' ' + str(flow[1][11]) + ' ' + str(flow[1][12])
+ ' ' + str(flow[1][13])
#flow = str(flow[1][0])+' : '+str(flow[1][1])+'
'+str(flow[1][2])+' : '+str(flow[1][3]+' '+str(flow[1][4])+'
: '+str(flow[1][5]))# build the flow that we
need
matched_server_ip_dict[key].append(flow) #
append to the file of target application
#print flow
selected_flows_within_time =
selected_flows_within_time.loc[selected_flows_within_time.index
!= current_index]#select the flows that are unknown
netflow_csv.set_value(current_index,
'known_tag', 1)#tag each flow with 1 to be known
#netflowdomain_csv.set_value(current_index,
'known_tag', 1)
```

```
else:#to check the row in (netflow file) that contain
the keywords in the dictionary of the key
#if key == 'accounts_google_com':
# continue
#print netflow_csv.loc[current_index][4]
for app_domain in keyword_application[key]:
#this if below for real data
#if app_domain in
netflow_csv.loc[current_index][4] or app_domain in netflow_csv.loc[current_index][8]:
if app_domain in
netflow_csv.loc[current_index][4]:
flow = str(flow[1][9]) + ' ' +
str(flow[1][10]) + ' ' + str(flow[1][11]) + ' ' + str(flow[1][12])
+ ' ' + str(flow[1][13])
#flow = str(flow[1][0]) + ' : ' +
str(flow[1][1]) + ' ' + str(flow[1][2]) + ' : ' + str(flow[1][3] +
' ' + str(flow[1][4]) + ' : ' +
str(flow[1][5])) # build the flow that we need
matched_server_ip_dict[key].append(fl
ow) # append to the file of target application
selected_flows_within_time =
selected_flows_within_time.loc[selected_flows_within_time.index
!= current_index] # select the
#flows that are known
netflow_csv.set_value(current_index,
'known_tag',1) # tag each flow with 1 to be known
```

```
#netflowdomain_csv.set_value(current_
index, 'known_tag', 1)
break
#this to write flows in the files for applications
for Key, value in matched_server_ip_dict.iteritems():#to inter in each dictionary
and make a loop
if len(matched_server_ip_dict[Key]) > 0:
#with open('matched_flows/'+Key+'_flows.txt', 'w') as f:#write the
contents of each dictionary to file (this the path and the name)
#for line in matched_server_ip_dict[Key]:
#f.write(line+'\n')
print Key,':', len(matched_server_ip_dict[Key])
# The unknown flows in IPs form
netflow_csv_unknown = netflow_csv[netflow_csv['known_tag'] == 0]
#netflow_csv_unknown.to_csv('matched_flows/unknown_flows.txt', header=False,
index=False)
requests.append(10000000000000)
requests.append(10000000000000)
path=directory_path+folder+'/R/'
files = [x for x in os.listdir(path) if x[-3:] == 'csv']#read the files
for app in files:
netflow_csv = pd.read_csv(path+app, header = None, sep =',')
netflow_csv.drop([0,1,60,61,62], axis = 1,inplace = True)# remove the time
parts from the file and blank field
netflow_csv_header = pd.read_csv('C:/Users/hjoudah/Dropbox/hussein/python/
real data results/samples files/facebook.csv', sep =',').columns #read the columns
```

```
only (header)

netflow_csv.columns = netflow_csv_header# put the header to the file header

netflow_csv.to_csv(path+app, index=False)# write the file

########################################################################
```

4. Python script for Feature selection using Random Forest for only uncontrolled dataset (chapter 7, section 7.3.1)

```
===========Feature selextion rbased on RF==============
Number of important features
num_selected_feature = 100
Display the features that belong to the number
'sd_inter_arrival_time_data_b'
dataset.columns[242]
[94, 82, 239, 71, 99, 261, 97, 69, 92, 244, 67, 10, 168, 173, 253, 101, 81, 96, 210,
# Split the dataset in two equal parts
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0, s
X_train, selected_features, importances = feature_imp_RandomForest(X_train,
y_train)
top_rakned_features_list = selected_features.values[:num_selected_feature].tolist()
print(top_rakned_features_list)
X_test = X_test.values
X_test = X_test[:,top_rakned_features_list]
indices = np.argsort(importances)[::-1][:15]
# Plot the feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(15), importances[indices],
def feature_imp_RandomForest(X_train, y_train):
```

```
rf = RandomForestClassifier(n_estimators=300, max_depth=8,
min_samples_leaf=4, max_fea
rf.fit(X_train, y_train)
importances = rf.feature_importances_
####################################
#for showing the features ranks
features_rank = list(zip([x for x in range(0, X_train.shape[1])], rf.feature_importan
features_importance_df = pd.DataFrame(features_rank, columns=['features', 'rank'])
features_importance_df = features_importance_df.sort_values(by=['rank'],
ascending=Fal
global top_features
selected_features = features_importance_df['features']
####################################
X_train = X_train.values
X_train = X_train[:,rf.feature_importances_.argsort()[::-1][:num_selected_feature]]
#joblib.dump(rf.feature_importances_.argsort()[::-1][:num_selected_feature],'pre-
train
return X_train, selected_features, importances
color="b", align="center")
plt.xticks(range(15), indices)
plt.xlabel('Feature Number')
plt.ylabel('Importance %')
plt.show()
================ End of feature selction ===============
```

5. Machine Learning Techniques for controlled environment using Gradient Boosting, SVM, and Random Forest (Python script)(chapter 7, section 7.2)

```
Import libraraies
# Standard useful data processing imports
import random
from math import sqrt
import numpy as np
import pandas as pd
# Visualisation imports
import matplotlib.pyplot as plt
import seaborn as sns
# Scikit learn for preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold,StratifiedKFold
#from sklearn.cross_validation import cross_val_score, cross_val_predict
from sklearn.model_selection import cross_validate,cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from keras.utils.np_utils import to_categorical
%matplotlib inline
from google.colab import files
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBRegressor
from xgboost import plot_importance
from sklearn.datasets import make_regression
import multiprocessing as mp
!pip install -U seaborn
import xlrd
Loading data
dataset = pd.read_csv("/content/controlled_data_both.csv")
user mean_packets_a mean_packets_b mean_packets_b/packets_a
mean_data_packets
0 user1 23.719298 6.0 107.0 14
1 user1 24.441558 7.0 173.0 1
2 user1 19.333333 6.0 242.0 1
3 user1 17.423729 7.0 112.0 1
4 user1 18.031250 6.0 154.0
5 rows × 201 columns
dataset.head()
dataset['class'] = dataset['class'].str.strip(to_strip=None)
CategoricalDtype(categories=['amazon', 'bbcnews', 'bing', 'cnn', 'facebook',
'gmail',
'googlebrowsing', 'instagram', 'skype', 'yahoomail',
'youtube'],
ordered=False)
```

```
dataset['class'] = dataset['class'].astype('category')

dataset['class'].dtypes

dataset['class'] = dataset['class'].cat.codes

mean_packets_a mean_packets_b mean_packets_b/packets_a

mean_data_packets_a

count 2200.000000 2200.000000 2200.000000 2200.000000

mean 73.781318 38.166867 856.327523 27.535815

std 214.902133 187.058186 2587.607310 128.671790

min 7.048951 1.000000 0.381103 1.658363

25% 14.152352 4.000000 76.500000 7.000000

50% 22.148756 6.000000 179.000000 9.000000

75% 36.758152 7.000000 427.750000 11.500000

max 2232.142857 2737.571429 25351.000000 1836.454545

8 rows × 200 columns

dataset.describe()

dataset.groupby(['class']).count()

mean_packets_a mean_packets_b mean_packets_b/packets_a

mean_data_packets_a

class

0 200 200 200 200

1 200 200 200 200

2 200 200 200 200

3 200 200 200 200

4 200 200 200 200

5 200 200 200 200

6 200 200 200 200
```

```
7 200 200 200 200

8 200 200 200 200

9 200 200 200 200

10 200 200 200 200

11 rows × 199 columns
```

```
dataset.drop('user', inplace=True, axis=1)

X, y = dataset.loc[:, dataset.iloc[:,:].columns != 'class'], dataset.loc[:,

dataset.iloc[:,:].co
```

Grid Search

```
grid_param = {

'learning_rate':[.3,.2,.1,.09,.07],

'max_depth': [3,4,5,7,9],

'max_leaf_nodes': [20,30,40,50],

'n_estimators':[80,100,150,200,250],

}

grid_param = {

'learning_rate':[.1],

'max_depth': [5],

'max_leaf_nodes': [40],

'n_estimators':[100],

}
```

GradientBoostingClassifier

```
model = GradientBoostingClassifier()

model = GridSearchCV(model, grid_param, cv=StratifiedKFold(5),verbose=1)

p=mp.Pool(4)

model = model.fit(X, y.values.ravel())
```

```
model.best_params_
y_pred = model.predict(X)
```

SVM

```
grid_param = {
'C':[1.0],
'kernel': ['rbf']
}
from sklearn.svm import SVC
model = SVC()
model = GridSearchCV(model, grid_param, cv=StratifiedKFold(5),verbose=1)
model = model.fit(X, y)
model.best_params_
model.best_score_
0.09090909090909091
```

Random Forest

```
grid_param = {
'max_depth': [8],
'max_leaf_nodes': [40],
'n_estimators':[100],
}
model = RandomForestClassifier()
model = GridSearchCV(model, grid_param, cv=StratifiedKFold(5),verbose=1)
model = model.fit(X, y.values.ravel())
model.best_params_
0.9040909090909091
```

6. C5.0 classifier for both controlled and uncontrolled (R-script) (chapter 7, section 7.2 and chapter 7, section 7.3.2 respectively)

```
#
==================================================================
====================
# This script to classift traffic using machine learning C5.0
#
==================================================================
====================
# Import libraries
library(caret)
library(e1071)
#set the directory
setwd("C:/Users/hjoudah/Dropbox/hussein/")
#read the file
applications <-read.csv("all.csv")
#View (applications)
X <- applications [,1:58]
Y <- applications [,59]
trainx <- X[1:34630,]
trainy <- Y[1:34630]
testx <- X[34631:34919,]
```

```
testy <- Y[34631:34919]

treeModel <- C50::C5.0(trainx, trainy)

summary (treeModel)

p <- predict (treeModel, testx, type="class")

sum (p==testy)/length(p)

confusionMatrix (p, testy)
```