

Sudan University of Science and Technology

College of Engineering

School of Electronic Engineering



IP traffic classification using machine learning

تصنيف حركة بيانات الانترنت عن طريق تعلم الآله

A Research Submitted In Partial fulfillment for the Requirements of the
Degree of B.Sc. (Honors) in Electronics Engineering

Prepared by:

- 1-Alaa Mohammed Yousif Ahmed
- 2-Aya FathelrahmanAwad Mohammed
- 3- MarwaEdriss Ali Mohammed
- 4- Noon Salah Mohammed Abdalgader

Supervisor

Dr. AbuaglaBabiker Mohammed Babiker

October 2016

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

{ اقْرَأْ بِاسْمِ رَبِّكَ الَّذِي خَلَقَ (1) خَلَقَ الْإِنْسَانَ مِنْ
عَلَقٍ (2) اقْرَأْ وَرَبُّكَ الْأَكْرَمُ (3) الَّذِي عَلَّمَ
بِالْقَلَمِ (4) عَلَّمَ الْإِنْسَانَ مَا لَمْ يَعْلَمْ (5) }

سورة العلق - الآية { 5 - 1 }

Dedication

To the big heart our dear father

To the fountain of patience and optimism and hope

To each of the following in the presence of God and His

Messenger, our dear mother

To those who have demonstrated to us, what is the most beautiful

of our brothers and sisters life

To the people who paved our way of science and knowledge

To the taste of the most beautiful moments with our friends

I guide this research

Acknowledgement

The work presented in this thesis was made possible by many people. First of all, we would like to thank our supervisor AbuaglaBabiker Mohammed Babiker and our co-supervisors Ahmed Abdallah for being excellent mentors and for their guidance, feedback, and patience throughout our studies. we would like to thank Mr.Haitham Ahmed for the support .we would like to thank colleagues from the Department of Electronic engineering school for being a truly wonderful bunch of people.

Abstract

Network traffic classification is the foundation of many network research works. In recent years, the research on traffic classification based on machine learning method is giving more accuracy. The focus of this research is to classify the traffic based on application type specially (p2p, video, http) .Performance comparison among machine learning classifiers have been done, results shows that C4.5 tree is the suitable classifier based on accuracy and stability. This offline comparison has been done by building training data sets using two methods the first use WIRESHARK for sniff the network traffic the a pre-classification has been done. The second approach is based on SNORT for sniffing and detecting the type of application and then extracted short flow by excel sheet and calculated the feature of flow Evaluate the system by using only 7 features and its response with good accuracy. The prototype system results reach average accuracy 82.7% by first data set and 88.9% by the second data set.

Finally, an idea of an online classifier has been presented based on short flows so as to quick the reaction time regarding the enhancement of the usage of the internet links.

المستخلص

تصنيف حركة بيانات الانترنت اساس معظم الابحاث في مجال الانترنت . في الاونه الاخيريه اصبح تصنيف حركة بيانات الانترنت بإستخدام تعلم الآله يعطي دقه عاليه . هذا البحث يركز علي تصنيف بيانات الانترنت علي حسب نوع التطبيق المولد لها و علي وجه الخصوص تطبيقات (الند للند , الفيديو و التصفح). تم عمل مقارنة من ناحية الجوده بين مصنفات تعلم الآله. النتائج اظهرت ان شجرة سي 4.5 مناسبة بإعتماد الدقة و الاستقرار. كذلك قدمت طريقة مبنيه علي اساس التدفق القصير لتحسين جوده الخوارزمية عن طريق بناء نظام يتم التدخل فيه اثناء التشغيل لنقل البيانات من مرحله لاخري . كذلك تم عمل مجموعه بيانات لتعلم الآله عن طريق استخدام برنامجين الاول ويركشارك و تم عمل تصنيف مبدئي للبيانات المقدمة من هذا البرنامج و الثاني سنورت استخدم للتصنيف وتحديد نوع التصنيف . بعد ذلك تم استخراج البيانات في شكل تدفق قصير بإستخدام برنامج الاكسيل و حساب المتغيرات في كل تدفق . قيم النظام علي اساس سبعة متغيرات وكانت الاستجابة بدقه جيده و كانت نتائج النظام المبدئي متوسط الدقه له 82.7% عن طريق مجموعه البيانات الاولي و 88.9% عن طريق مجموعه البيانات الثانيه.

اخيرا تم اقتراح نظام للتصنيف لا يتم التدخل فيه اثناء التشغيل مبني علي اساس التدفق القصير ليكون سريع الاستجابة ليناسب تحسين استخدام خطوط نقل الانترنت.

List of Contents

CHAPTER	TITLE	PAGE
	Dedication	II
	Acknowledgement.....	III
	Abstract.....	IV
	المستخلص	V
	List of contents.....	VI
	List of Tables.....	VIII
	List of Figures	IX
	List of Abbreviatio.....	XI
	Chapter One: Introduction	1
1.1	Introduction.....	2
1.2	Problem Statement.....	2
1.3	Proposed Solution	3
1.4	Aim and Objectives	3
1.5	State of the art	3
1.6	Methodology	5
1.6.1	Pre-processing	5
1.6.2	Features Selection and Extraction	5
1.6.3	ML logarithm training.....	5
1.6.4	Classification process	6
1.7	Thesis Outlines	6
	Chapter Two: Literature Review.....	7
2.1	Background	8
2.1.1	Quick Overview of P2P Networking	8
2.1.2	Port based classification.....	9
2.1.3	Payload based Classification	10

2.1.4	Host-behaviour-Based Approach.....	10
2.1.5	Statistical Properties Classification.....	10
2.1.6	Background on Machine Learning	10
2.1.6.2	Machine Learning Algorithms.....	12
2.2	Related work	13
Chapter Three: Methodology.....		18
3.1	Pre-processing	19
3.2	Pre-classification	20
3.3	Features Selection and Extraction	20
3.4	Convert to ARFF.....	21
3.5	Select MLA.....	21
3.6	Machine Learning a logarithm training	21
3.7	Classification Process.....	22
3.8	Using snort to collect and pre-classify accurate data set.....	22
3.8.1	Install Snort.....	22
3.8.2	Create Snort directory	24
3.8.3	Configure Snort (edit snort .conf)	25
3.8.4	Create rule.....	26
Chapter Four: Result and discussion		30
4.1	Overview of the prototype system	31
4.1.1	Offline Phase	31
	Training dataset 1	35
4.1.2	Training Data Set 2	38
4.2	Evaluation and Validation.....	39
4.3	Conclusion.....	40
4.4	Online Phase.....	40
Chapter five: Conclusion and Recommendations.....		45
5.1	Conclusion.....	46

5.2 Recommendations.....46
References.....48

List of Tables

TABLE NO.	TITLE	PAGE
4.1	Final result by data set 1.	38
4.2	Final result by data set 2.	39
4.3	Evaluation metrics.	39

List of Figures

FIGURE NO.	TITLE	PAGE
2-1	The structure of peer-to-peer networking.	8
2-2	Traffic Classification Approaches	9
3-1	Flow diagram for Classification Task.	19
3-2	Features used for ip traffic classification.	20
3-3	Weka explorer Classification window GUI.	21
3-4	snapshot of select the interface listening to.	23
3-5	snapshot of packet capture.	24
3-6	snapshot of creating log file.	25
3-7	snapshot of snort.conf edit to specific rules.	26
3-8	snapshot of p2p rules.	27
3-9	snapshot of detecting p2p packet from trace of internet.	28
3-10	snapshot of packet save as CSV.	29
4-1	snapshot of data capture by WIRESHARK.	31
4-2	snapshot of data capture as csv.	32
4-3	snapshot of flow data capture as csv .	33
4-4	snapshot of flow data capture include needed feature as csv.	34
4-5	snapshot of flow data pre-classified saved as CSV.	34
4-6	snapshot of flow data capture include needed feature as ARFF.	35
4-7	snapshot training dataset as AEFf format.	36
4-8	snapshot of the training process doing by WEKA.	37
4-9	snapshot of cross validation process doing by WEKA.	38
4-10	snapshot of select the interface listening to.	41

4-11	snapshot of packet capture.	42
4-12	snapshot of creating flow of packet	43
4-13	snapshot of classifier result.	44

List of Abbreviations

ARFF	Attribute relation file format
CI	conditional independent
CSV	Comma separated values
DAC	directed acyclic graph
DNS	Domain name server
FTP	File transfer protocol
DPI	Deep Packet Inspection
HTTP	Hyper text transfer protocol
IANA	Internet Assigned Numbers Authority
IMAP	Internet message access protocol
IP	Internet protocol
IPTV	Television Internet protocol
ISP	internet service provider
ML	Machine Learning
P2P	peer to peer
QoS	Quality of service
SMTP	Simple mail transfer protocol
SVM	support vector machines

TCP	Transmission control protocol
VoD	Video on Demand
VoIP	voices over ip

Chapter One: Introduction

1.1 Introduction

The volume of network traffic is continuously increasing because of new multimedia applications (like peer to peer (P2P), voice over internet protocol (VoIP) and video) and advancements in Internet technology [1]. In this type of situation, application classification becomes very important for managing quality of service (QoS) in the Network and security monitoring for various internet service providers (ISP) and other governmental and private organizations. Accurate and efficient application classification is the key stone of network monitoring, and on the basis of the classification results network administrators can design various policies to enhance the network security. However, the challenging task to classify the applications based on the traffic characteristics due to the massive data in high-speed networks[2]. Although various methods for traffic identification have been proposed, not a single method identifies all types of Internet traffic. Research community is responded by looking particular at application of Machine Learning (ML) method.

1.2 Problem Statement

Network administrators need to know what is going over the networks in order to manage the traffic in accordance with the requirements. ISP needs to know what real bandwidth consumes over any type of application to enhance the QoS. Moreover, both of them need to manage the bandwidth according to the application used. Nowadays, there exist some of the harmful applications which are designed intentionally to avoid detection. There for an accurate and fast classification approach is highly required.

1.3 Proposed Solution

To design and implement prototype system that classifies the internet traffic based on applications type using machine learning.

1.4 Aim and Objectives

The main aim of this project is accurately and fast classification of the IP traffic using machine learning to the corresponding application type, this step is an essential requirement towards the accurate and fast control of bandwidth. The detailed **objectives** are:

- To propose a lightweight and accurate internet traffic classification scheme.
- To select a suitable measurement approach as well as relative features for the classifiers.
- To propose an online traffic classifier.

1.5 State of the art

Traffic classification techniques can be broadly divided into port and packet payload based classification, behavioral identification techniques, and statistical measurement based approaches [3]. Port based classification techniques are now considered obsolete given the frequent obfuscation techniques and dynamic range of ports used by applications, packet payload inspection methods remain relevant primarily due to their high classification accuracy. Payload based classifiers inspect packet payloads using deep DPI to identify application signatures of packets. Although the resulting classification is highly accurate it also presents significant computational costs as well as being error-prone in dealing with encrypted packets. In comparison, behavioral classification techniques work higher up the networking stack and peruse the total traffic patterns of the end-points

(hosts and servers) such as the number of machines contacted and the protocol used to identify the application being used on the host. Behavioral techniques are highly promising and provide a great deal of classification accuracy with reduced overhead compared to payload inspection methods. However, behavioral techniques focus on end-point activity and require parameters from a number of flows to be collected and analyzed before successful application identification. With increasing ubiquity of flow level network monitoring which presents a low-cost traffic accounting solution, specifically utilizing NetFlow due to scalability and ease of use, statistical classification techniques utilizing flow measurements have gained momentum [4]. Statistical approaches exploit application diversity and inherent traffic footprints (flow parameters) to characterize traffic and subsequently derive classification benchmarks through data mining techniques to identify individual applications. Statistical classification is considered light-weight and highly scalable from an operational point of view especially when real-time or near real-time traffic identification is required. While traffic classification in the network core is increasingly challenging and seldom implemented, application flow identification at the edge or network ingress allows operators in shaping the respective traffic further upstream [3]. Statistical flow based traffic classifications however, due to minimal number of available features in a typical flow record such as NetFlow, report low classification accuracy and increasingly rely on additional packet payload information to produce effective results [5]. The present work picks up from this narrative and solely utilizes NetFlow attributes using two-phased machine learning (ML), incorporating a

combination of unsupervised and supervised to achieve high accuracy in application traffic classification.

1.6 Methodology

The method of Machine Learning traffic classification techniques involves following steps. First traffic is captured with the help 5-tuple (source and destination port, source and destination address and protocol) and traffic flow is created, then flow statistical properties are calculated in term of the each feature, to create the statistics data set. The feature are called attribute. The attributes of network traffic flow are calculated over more number of packets (such as minimum or maximum packet sizes, variance of packet length, flow durations, Direction of packet and inter arrival times of packets), then Machine learning classifier is trained on features that are calculated from flow that have labeled classed, then Machine learning algorithms are applied for classification of unknown network traffic using previously labeled classes[6].

After careful analysis of the system has been identified to have the following modules

1.6.1 Pre-processing

Here the IP packets crossing across a network is collected and used for constructing the flows by examining the header of packets.

1.6.2 Features Selection and Extraction

Here statistical features are defined and extracted to remove irrelevant and redundant features from the feature set.

1.6.3 ML logarithm training

Here select the ML algorithm and training by using different data set to reach optimizes training.

1.6.4 Classification process

The core of the system is classification process depend on apply ML a logarithm to unknown data set to find the class name of applied dataset.

1.7 Thesis Outlines

The rest of the work has been divided into five chapters; chapter 2 presents the literature review for internet Traffic classification and critically analyze the previous and current techniques used. Chapter 3 describes the designed method; chapter 4 show the implementation of the proposed method, explains the results and gives the corresponding discussions. Moreover, it also evaluates the validated the proposed framework, and carries out performance comparison. Finally in Chapter 5, concluding the work and propose a future research directions.

Chapter Two: Literature Review

2.1 Background

This chapter presents some important aspects related to network traffic classification, which are necessary to understand the content of this thesis.

2.1.1 Quick Overview of P2P Networking

Data and information exchange or content distribution can be in either client-server model or in Peer-to-Peer (P2P) model in addition to hybrid approach. In a P2P network, every work station can play the role of server as well as the client (as shown in figure2-1) below [7].

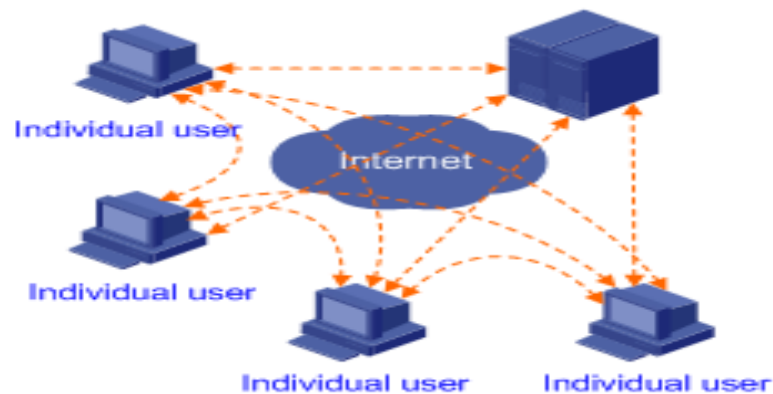


Figure 2-1: The structure of peer-to-peer networking. source:[8]

As seen in figure 2-1 above the movement of data in P2P can be from several peers to one client [9].

Recently many application based on P2P infrastructure, all are considered as bandwidth consume applications such as BitTorrent, IPTV , PPLive , VoD and others. As P2P networks facilitate file transfer and sharing add some disadvantages such as Risks of downloaded content (when a file is

downloaded using the P2P software, it is not possible to know who created the file or whether it is trustworthy), P2P software is vulnerable to bugs [10].

There are many techniques used for internet traffic classification as shown in Figure 2-2 below, described in brief in the next section.

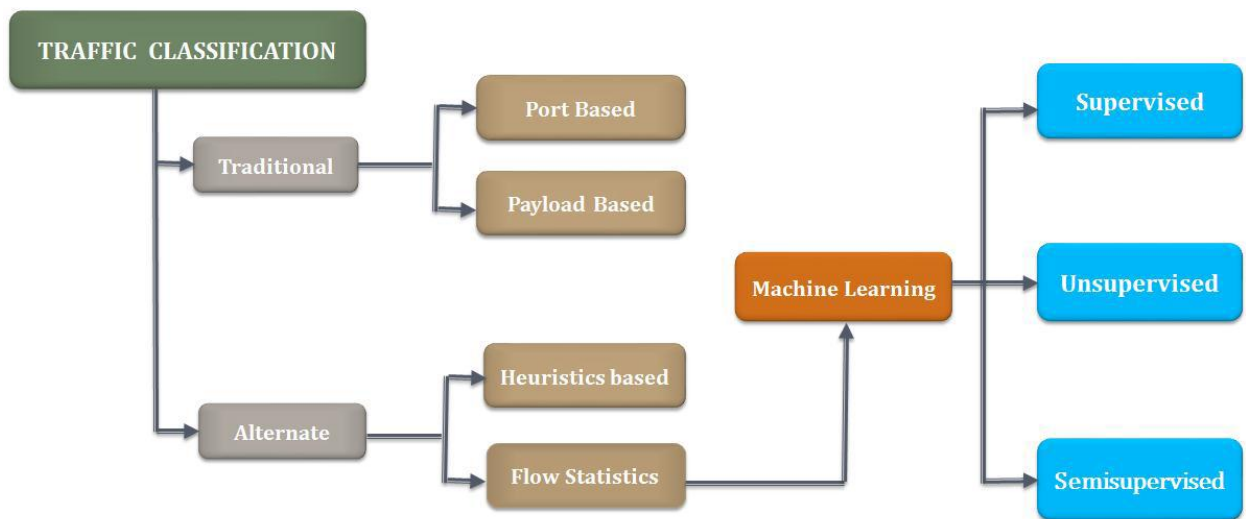


Figure 2-2: Traffic Classification source: [11].

2.1.2 Port based classification

In this method, the application type can be identified using port numbers registered in IANA list. This method is considered simple and easy to implement for classification of applications for online and real time traffic. Nowadays it has shown lower accuracies (50% - 70%) because some applications use dynamically port number (port hopping), some applications use other application ports (tunnelling). For the above reasons, the port number method has become ineffective [11].

2.1.3 Payload based Classification

In this scheme, the application type can be identified by using contents of the packet payloads by extract signature from payload[12]. The result show problem of port based method solved moreover given high accuracy in resent internet traffic, but fail in encryption communication moreover take more time to extract signatures , need high resource in computation and the privacy laws may not allow administrators to inspect the payload [13] .

2.1.4 Host-behaviour-Based Approach

In this method define the type of application by extract connection-level patterns of the hosts observing at transport layer [14]. The main advantage of this method is that there is no need for packet payload, but cannot apply the method in online near real time classification [15].

2.1.5 Statistical Properties Classification

In this scheme the application type define by using the statistical properties like inter-arrival time, packet length and flow duration to classify the traffic. Solve all the problem mentions above and gain high accurate depending of the number of features [16] .

2.1.6 Background on Machine Learning

Machine learning is the subset of algorithms developed in the discipline of Artificial Intelligence and these algorithms use different features to learn a set of rules in order to identify different classes [17]. Machine learning has wide range of applications such as search engines, image screening, marketing, forecasting, medical science, text and hand writing are few among many. The input of a machine learning process is a dataset of instances or examples (training data set) these instances are

statistical parameters in case of networking. Output of such process is the knowledge learnt by the machine[18].

Types of Learning

There are two major types of machine learning in context of network traffic classification. In this research the focus will be on supervised learning, for understanding brief about two types present next.

- **Unsupervised (clustering)**

In this technique the input does not need labeled data. It is finds a way to naturally group the data sets also called clusters, but still these clusters needs to be labeled by an expert. Earliest work done on unsupervised technique uses Expectation maximization algorithm for IP traffic classification based on application[19]. Issues with unsupervised learning are that clusters do not map 1:1to applications, in ideal case number of clusters formed are equal to the number of application to be mapped, but practically that is not the case. The number of clusters is often greater than the number of applications that is one single application might dominate over the number of clusters or application might spread over but do not dominate any of the clusters[20].

- **Supervised (classification)**

It creates knowledge based structures which can help to classify the new instances of different classes. Input instances are provided during the learning processes which are pre-classified into classes and output of such a process depends on these generalized instances[21].The dataset provided for training is labeled and at this stage of process the time does not matter that is how long it takes to process the sampled flows. More the numbers of attributes or feature more will be the time to process them

and better will be the accuracy of classifier, different algorithm have different set of rules developed from the provided dataset and their performance[22]. The most common machine learning algorithms applied to traffic classification are C4.5 Decision Tree, Naïve Bayes, Naïve Bayes Kernel Estimation, Bayesian Network K-NN, Neural network and SVM (support vector machines). Most successful results have been obtained from C4.5 and SVM[23].

2.1.6.2 Machine Learning Algorithms

Here for understanding let us have a look on couple of most used algorithms.

- **C 4.5 Decision Tree**

It is an algorithm developed by Ross Quinlan in 1992 as an extension of ID3.

C4.5 algorithm uses and generate tree based structure which can be used for Classification that is why it is also called statistical algorithm. It uses concept of entropy theory for classification for example we have data set $S = \{s_1, s_2, \dots, s_x\}$ where s_1, s_2, \dots, s_x Represents the training samples of the data set which are characterized by different features, let say $\{X_1, X_2, \dots\}$ are the corresponding features consisting target class. Now C4.5 selects particular feature of the data set on each node, which is used to split these samples into different classes. The idea of selecting the feature depends on the normalized gain information from the samples, feature with the highest normalized gain is selected and the decision is made[24]. Some **advantages** of using decision trees are:

- Self Explanatory and easy to follow
- Can handle both numeric and nominal input attributes

- Can handle a data set with many errors including missing values

However, most decision trees require the target variable to only have discrete values; they tend to perform well with non complex attributes. Furthermore, they are very sensitive to the training data sets any corrupt values close to the root node can change the whole structure of the tree [25].

- **SVM (Support vector Machine)**

SVM are powerful algorithms used to solve classification and regression problems. In order to classify the algorithm transform the input data to a high dimensional hyper plane, where it becomes more separable compared to the original form[26].This is done by using non linear kernel functions, and then linear classifiers are used to construct maximum margin hyper planes to separate the different classes in training data. Two hyper planes are constructed both sides of the hyper plane separating the data which tends to maximize the space between two parallel hyper planes. The assumption made is larger the distance between parallel hyper planes the better the generalization errors of the classifier will be. SVM's learns through historic cases in the form of data point which contribute to very accurate classification, another **advantage** of these algorithms is they can handle missing values and noise effectively. However, these are complex and demands high memory requirements [27].

2.2 Related work

Jayeeta Datta, Neha Katariaand and Neminath Hubballi in (2015) propose a novel technique based on application behavior. Used application semantics to identify a set of features which are subsequently

used by a classification algorithm to identify the class to which a packet belongs. Experimented with a dataset collected for Google Hangout and reported its detection performance using few classification algorithms (Naive Base, j48 and AdaBoost) classification algorithms to assess the detection performance[28].

Babak Rahbarinia , Roberto Perdisci , Andrea Lanzi and Kang Li in (2014) present PeerRush, a novel system for the identification of unwanted P2P traffic PeerRush goes beyond P2P traffic detection, and can accurately categorize the detected P2P traffic and attribute it to specific P2P applications, including malicious applications such as P2P botnets. PeerRush achieves these results without the need of deep packet inspection, and can accurately identify applications that use encrypted P2P traffic. Implemented a prototype version of PeerRush and performed an extensive evaluation of the system over a variety of P2P traffic datasets. The results show that PeerRush can detect all the considered types of P2P traffic with up to 99.5% true positives and 0.1% false positives. Furthermore, PeerRush can attribute the P2P traffic to a specific P2P application with a misclassification rate of 0.68% or less [29].

Valentín Carela-Español,, Pere Barlet-Ros, and Josep Solé-Pareta present classification method based on machine learning process without any human intervention using Net flow V5 to provide the features and C4.5 method as decision tree. The evaluation dataset consists of six full-payload traces collected at the Gigabit access link of the Technical University of Catalonia. The traces are 15 minutes long with approximately 3 millions of unidirectional sanitized flows for each trace.

The average overall accuracy for the complete dataset is about 90%. The maximum overall accuracy is 93,6%, which corresponds to the trace used in the training phase. The minimum overall accuracies are 86% and 88,5%. These lower results belong to the traces collected at night, when the mix of traffic is more different than the trace used in the training phase [30].

Hardeep Singh evaluated two different unsupervised machine learning algorithm K-means and Expectation maximization for different types of network traffic application. Results show that K-means have higher accuracy then EM for all class of application. K-means have the ability to produce cluster that have single traffic class and most of the connection are placed in few clusters. EM also produced good quality of clusters for various traffic applications. Each algorithm tries to make cluster according to single application type. This approach gives good accuracy and overcome the drawback of port based classification. Moreover algorithm does not require pre classified set of data as a training. These techniques can also be applied to real time classification such as VoIP applications and streaming multimedia type of applications that require special Quality of Service[6].

AmjadHajjar, JawadKhalife and JesúsDíaz-Verdejo propose a novel blind, quintuple centric approach by exploring traffic attributes at the application level without inspecting the payloads. The identification model is based on the analysis of the first application-layer messages in a flow (quintuple), based on their sizes, directions and positions in the flow. That using the proposed method it is possible to correctly classify up to around98% of the TCP flows and 99% of the UDP sessions by only

analyzing the first 3 to 5 messages. The classifier uses the sizes of the initial messages exchanged between the hosts involved in the communication as inputs. This work focuses on the messages, not the packets [31].

Jeffrey Erman and Martin Arlitt and Anirban Mahanti considers two unsupervised clustering algorithms, namely K-Means and DBSCAN, that have previously not been used for network traffic classification evaluate these two algorithms and compare them to the previously used Auto-Class algorithm, using empirical Internet traces. The experimental results show that both K-Means and DBSCAN work very well and much more quickly than Auto-Class. Our results indicate that although DBSCAN has lower accuracy compared to K-Means and Auto-Class, DBSCAN produces better clusters [32].

Yang Hong¹, Changcheng Huang¹, Biswajit Nandy² and Nabil Seddigh (2015) propose a novel iterative-tuning scheme to increase the training speed of the classification algorithm using Support Vector Machine (SVM) learning .Performance evaluation demonstrates that the proposed iterative-tuning SVM exhibits a training speed that is two to ten times faster than eight other previously proposed SVM techniques found in the literature, while maintaining comparable classification accuracy [33].

Chapter Three: Methodology

THE DESIGN METHOD

This chapter will discuss the design methodology for classification task; furthermore it will highlight the approaches used to achieve the aim by using two phase online and offline (more details about the two phases in chapter 4). Step by step flow chart diagram describe traffic Classification method (as shown in figure3-1).

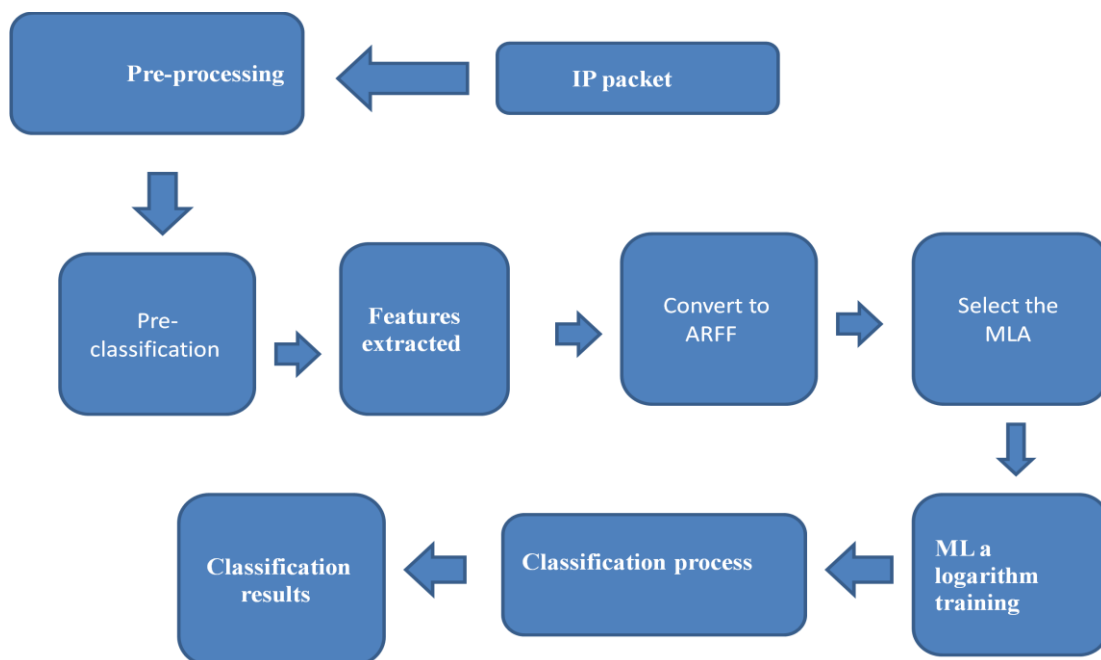


Figure 3-1: Flow diagram for Classification Task.

3.1 Pre-processing

Data collection which is usually called pre-processing, Here the IP packets crossing network is collected and used for constructing the flows by examining the header of packets. A flow can be define as successive IP packets having the same 5-tuple: source IP, source port, destination IP, destination port, and transport layer protocol.

3.2 Pre-classification

Is a process of adding a class feature to the extracted data, the last column in data part is class and represents the values for each row. There could be more than two values for class attribute but it depends on the type of data or classification used. In reference to this study wanted to classify the traffic as http, p2p and video. For this purpose it has three class nominal values “p2p”, “http” and “video”.

3.3 Features Selection and Extraction

Feature selection process is to remove the irrelevant or redundant feature from candidate feature set. Select features which have unique properties among different applications. It is important to know the aim wants to achieve, for example classification between different applications requires different features, to be extracted. The main task here is to select the attributes which have different values of p2p, http and video as shown in figure 3-2 below.

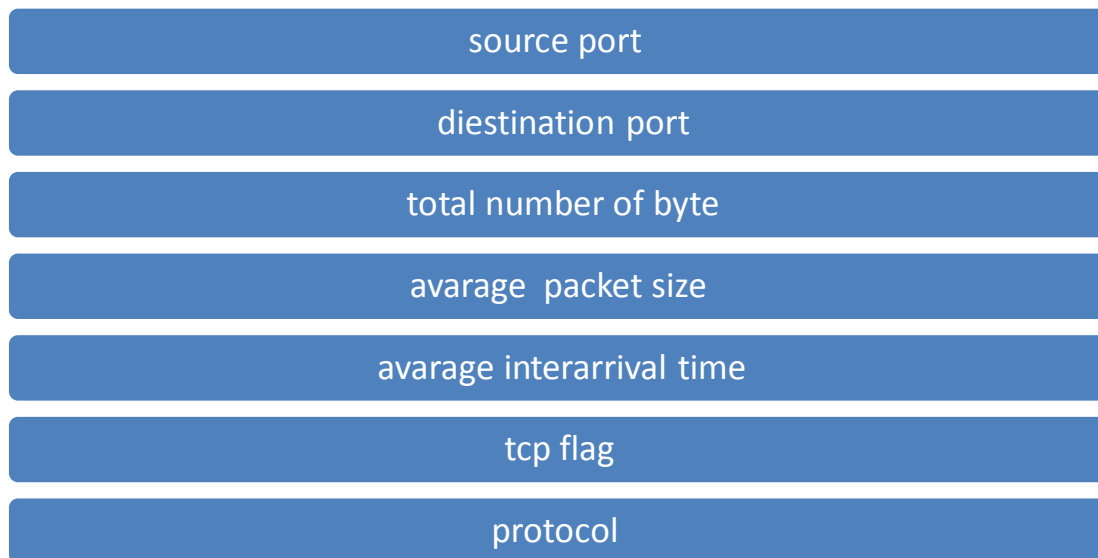


Figure 3-2: Features used for ip traffic classification.

3.4 Convert to ARFF

Here convert the flow from excel accepted format (CSV) to WEKA accepted format (ARFF).

3.5 Select MLA

Here applying the training data set with selected feature to many MLA in WEKA and compare between them by the time of building the model and over all accuracy to find the stable one with the station.

3.6 Machine Learning a logarithm training

Apply data set to select MLA. There are three different ways provided by WEKA GUI to train a particular classifier to understand this process figure 3-3 has been provided below.

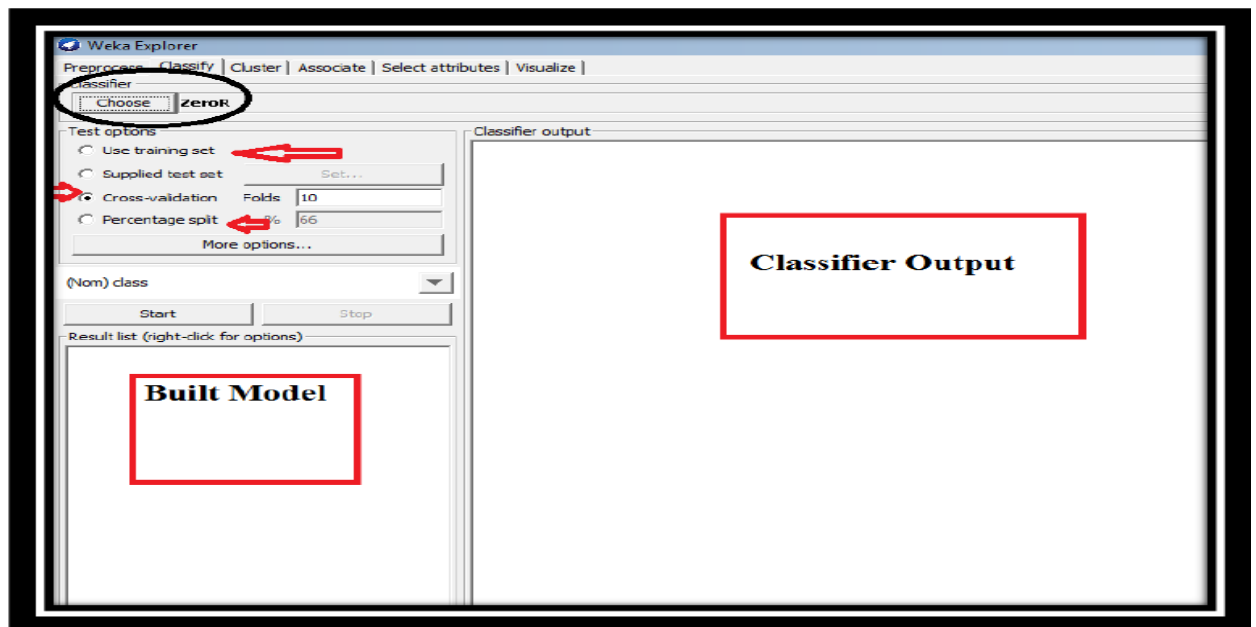


Figure 3-3: Weka explorer Classification window GUI.

The above figure is a screen shot of Weka explorer GUI, after choosing the classifier there are training and testing options. In order to

train the classifier one can provide a separate training set which will build a trained model. Then one can provide a test set to test the already built model, this is one way of training and testing.

Another way of doing the same procedure is using cross validation, by default the value of cross validation is 10 it means that WEKA will split the training data in 10 equal parts and use 9/10 for training and 1/10 for testing the classifier, the process repeats 10 times until all the training sets has been used to train and test the classifier and output of the process it the best average model. The third option is the percentage split, the specified value will split the data accordingly for example if the percentage value is selected 66 percent then WEKA will use 66 percent of data for training the classifier and 34 percent for testing the performance of trained model.

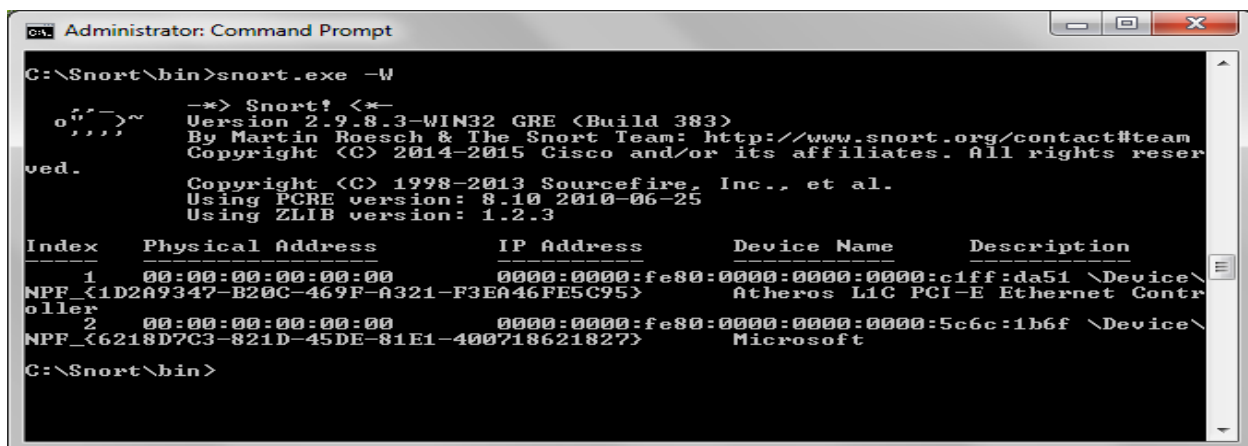
3.7 Classification Process

In this phase the data is un known class when the data become over the MLA and MLA doing the processing can decide the class of the data and send to the control module.

3.8 Using snort to collect and pre-classify accurate data set

3.8.1 Install Snort

To verify that Snort is installed and running correctly we run a couple of commands from the Command Prompt. Open a command prompt as Administrator, switch to the "C:\Snort\Bin" directory and run "snort.exe -W" to see a list of interfaces available to Snort. The following figure 3-4 is output from the command on Windows 7.



```

Administrator: Command Prompt
C:\Snort\bin>snort.exe -W
-*> Snort! <*-
Version 2.9.8.3-WIN32 GRE <Build 383>
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3

Index   Physical Address      IP Address             Device Name            Description
-----
1       00:00:00:00:00:00      0000:0000:fe80:0000:0000:c1ff:da51 \Device\NPF_{1D2A9347-B20C-469F-A321-F3EA46FE5C95} Atheros L1C PCI-E Ethernet Controller
2       00:00:00:00:00:00      0000:0000:fe80:0000:0000:5c6c:1b6f \Device\NPF_{6218D7C3-821D-45DE-81E1-400718621827} Microsoft
C:\Snort\bin>

```

Figure 3-4: snapshot of select the interface listening to.

As you can see, the computer has many interfaces. If want to use Snort as a sniffer and watch all traffic on first interface, could issue the command "snort.exe -i 1 -v". This command would run Snort in verbose mode (-v) and have it listen on interface 1 (-i 1). It would also dump the header of each packet to the screen. Note: You can use CTRL-C to interrupt the running program as shown in figure 3-5.

```

Administrator: Command Prompt
C:\Snort\bin>snort.exe -i2 -v
Running in packet dump mode

---- Initializing Snort ----
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{6218D7C3-821D-45DE-81E1-400718621827}"
Decoding Ethernet

---- Initialization Complete ----

-*> Snort! <*-
o''~
',,'~
Version 2.9.8.3-WIN32 GRE (Build 383)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3

Commencing packet processing (pid=2748)
*** Caught Int-Signal
=====
Run time for packet processing was 15.210000 seconds
Snort processed 0 packets.
Snort ran for 0 days 0 hours 0 minutes 15 seconds
Pkts/sec: 0
=====
Packet I/O Totals:
Received: 0
Analyzed: 0 < 0.000%>
Dropped: 0 < 0.000%>
Filtered: 0 < 0.000%>
Outstanding: 0 < 0.000%>
Injected: 0
=====
Breakdown by protocol (includes rebuilt packets):
Eth: 0 < 0.000%>
ULAN: 0 < 0.000%>
IP4: 0 < 0.000%>
Frag: 0 < 0.000%>
ICMP: 0 < 0.000%>
UDP: 0 < 0.000%>
TCP: 0 < 0.000%>
IP6: 0 < 0.000%>
IP6 Ext: 0 < 0.000%>
IP6 Opts: 0 < 0.000%>
Frag6: 0 < 0.000%>
ICMP6: 0 < 0.000%>
UDP6: 0 < 0.000%>

```

Figure 3-5 snapshot of packet capture.

Let's start Snort as a sniffer to display packet headers and contents. The command want to enter at command prompt is "snort.exe -i 1 -vd". To stop sniffing packets, break out of the program by pressing Ctrl-C.

3.8.2 Create Snort directory

To create directory can use the next command and the output show in figure 3-6.

```
C:\>snort -vde -l c:\Snort\log -i2
```

```

C:\WINDOWS\system32\cmd.exe - snort -vde -l c:\Snort\log -i2
C:\>snort -vde -l c:\Snort\log -i2
Running in packet logging mode

--== Initializing Snort ==--
Initializing Output Plugins!
Log directory = c:\Snort\log
Initializing Network Interface \Device\NPF_{4B319C62-9381-45ED-A725-4
Decoding Ethernet on interface \Device\NPF_{4B319C62-9381-45ED-A725-4

--== Initialization Complete ==--

-*> Snort! <*-
Version 2.8.6-ODBC-MySQL-FlexRESP-WIN32 IPv6 GRE <Build 38
By Martin Roesch & The Snort Team: http://www.snort.org/sn
Copyright (C) 1998-2010 Sourcefire, Inc., et al.
Using PCRE version: 7.4 2007-09-21
Using ZLIB version: 1.2.3

Not Using PCAP_FRAMES

```

Figure 3-6 snapshot of creating log file.

3.8.3 Configure Snort (edit snort .conf)

Configuration snort to use In packet logger mode, the program will log packets to the disk by using below command And then Save changes to snort.conf as shown in figure 3-7.

- □ \snort\etc\snort .conf
- □ Line #45- **ipvar HOME_NET any** – make this match your internal network;
- □ Line #48 **ipvar EXTERNAL_NET !\$HOME_NET**
- □ Line #104 **var RULE_PATH rules**
- □ Line #109 **var WHITE_LIST_PATH rules**
- □ Line #110 **var BLACK_LIST_PATH rules**
- □ Line #528-output log_tcpdump: tcpdump.log
- □ Line#529output alert_csv: marwa.csv default
- □ Line #543 - delete or comment out all of the “include \$RULE_PATH” lines except:

- `include $RULE_PATH/local.rules`
- `include $RULE_PATH/snort.rules` – add after local.rules

```

542 # site specific rules
543 include $RULE_PATH/local.rules
544
545 # include $RULE_PATH/app-detect.rules
546 #include $RULE_PATH/attack-responses.rules
547 #include $RULE_PATH/backdoor.rules
548 #include $RULE_PATH/bad-traffic.rules
549 # include $RULE_PATH/blacklist.rules
550 #include $RULE_PATH/botnet-cnc.rules
551 include $RULE_PATH/browser-chrome.rules
552 include $RULE_PATH/browser-firefox.rules
553 # include $RULE_PATH/browser-ie.rules
554 include $RULE_PATH/browser-other.rules
555 # include $RULE_PATH/browser-plugins.rules
556 # include $RULE_PATH/browser-webkit.rules
557 #include $RULE_PATH/chat.rules
558 #include $RULE_PATH/content-replace.rules
559 #include $RULE_PATH/ddos.rules
560 #include $RULE_PATH/dns.rules
561 #include $RULE_PATH/dos.rules
562 #include $RULE_PATH/experimental.rules
563 # include $RULE_PATH/exploit-kit.rules
564 # include $RULE_PATH/exploit.rules
565 # include $RULE_PATH/file-executable.rules
566 # include $RULE_PATH/file-flash.rules
567 # include $RULE_PATH/file-identify.rules
568 # include $RULE_PATH/file-image.rules
569 # include $RULE_PATH/file-multimedia.rules
570 # include $RULE_PATH/file-office.rules
571 # include $RULE_PATH/file-other.rules
572 # include $RULE_PATH/file-pdf.rules
573 #include $RULE_PATH/finger.rules
574 #include $RULE_PATH/ftp.rules

```

Figure 3-7 snapshot of snort.conf edit to specific rules.

3.8.4 Create rule

Default Snort installation doesn't contain any rules/signatures. Snort rules can be created by the user.

A rule composed of two distinct parts: the rule header, and the rule options. The rule header contains the rules action, protocol, source and destination IP Addresses and net masks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. Here is a sample rule as shown in figure 3-8.

P2p rule

#by Christopher Campesi

```

alert tcp $HOME_NET 1024: -> $EXTERNAL_NET 1024: (msg:"ET
P2P Ares Server Connection"; flow:established,to_server; dsize:<70;
content:"r|be|bloop|00|dV"; content:"Ares|00 0a|"; distance:16;
reference:url,aresgalaxy.sourceforge.net;
reference:url,doc.emergingthreats.net/bin/view/Main/2008591; class type
:policy-violation; sid:2008591; rev:3;) .

```

```

2 #
3 # This file contains (i) proprietary rules that were created, tested and certified by
4 # Sourcefire, Inc. (the "VRT Certified Rules") that are distributed under the VRT
5 # Certified Rules License Agreement (v 2.0), and (ii) rules that were created by
6 # Sourcefire and other third parties (the "GPL Rules") that are distributed under the
7 # GNU General Public License (GPL), v2.
8 #
9 # The VRT Certified Rules are owned by Sourcefire, Inc. The GPL Rules were created
10 # by Sourcefire and other third parties. The GPL Rules created by Sourcefire are
11 # owned by Sourcefire, Inc., and the GPL Rules not created by Sourcefire are owned by
12 # their respective creators. Please see http://www.snort.org/snort/snort-team/ for a
13 # list of third party owners and their respective copyrights.
14 #
15 # In order to determine what rules are VRT Certified Rules or GPL Rules, please refer
16 # to the VRT Certified Rules License Agreement (v2.0).
17 #
18 #-----
19 # P2P RULES
20 #-----
21 #alert tcp $HOME_NET any -> $EXTERNAL_NET 1214 (msg:"Kazaa port in use"; flow:to_server;established; sid:10503; rev:1;)
22 #alert tcp $HOME_NET any -> $EXTERNAL_NET 1214 (msg:"Kazaa client activity"; flow:from_client,established; content:"GET"; content:"KazaaClient"; classtype:policy
23 #alert tcp $HOME_NET any -> any $HTTP_PORTS (msg:"Bit Torrent Client download"; uricontent:"BitTorrent"; uricontent:".exe"; classtype: bad-unknown; sid:10559; rev
24 #alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Bit Torrent client usage"; content:"|00 00 40 09 07 00 00 00|"; offset:0; depth:4; classtype: policy-violator
25 #Alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"p2p Gnutella client request"; flow:to_server,established; content:"GNUTELLA OK"; depth:40; classtype:policy-vi
26 #Alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"p2p Gnutella client file connection"; flow:from_client,established; content:"X-Gnutel"; classtype:policy-violo
27 alert tcp any any -> any any (msg:"testing p2p!"; sid: 1000002;)
28 #alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET P2P BitTorrent P2P Client User-Agent (Bittorrent/5.x.x)"; flow:to_server,established; content:"User
29 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET P2P BitTorrent Traffic"; flow: established; content:"|0000400907000000|"; depth:8; reference:url,bitconjure
30 #alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET P2P BearShare P2P Gnutella Client HTTP Request "; flow:to_server,established; content:"/gnutella/";
31 #alert tcp any any -> any 21 (msg:"Telnet NOP"; content:"|0000400907000000|"; rawbytes;)
32 #alert tcp any any -> any 80 (content:"cgi-bin/phf"; offset:4; depth:8;)
33 #alert tcp any any -> any any (content:"ABC"; content:"EFG"; within:10;)

```

Figure 3-8 snapshot of p2p rules.

And then extracting data set as shown in figure 3-9 And save the ruslt in directory in CSV format. as shown in figure 3-10.


```

Administrator: Command Prompt
ARP: 3117 < 7.391%
IPX: 0 < 0.000%
Eth Loop: 0 < 0.000%
Eth Disc: 0 < 0.000%
IP4 Disc: 8610 < 20.417%
IP6 Disc: 0 < 0.000%
TCP Disc: 0 < 0.000%
UDP Disc: 0 < 0.000%
ICMP Disc: 0 < 0.000%
All Discard: 8610 < 20.417%
Other: 377 < 0.894%
Bad Chk Sum: 0 < 0.000%
Bad TTL: 0 < 0.000%
S5 G 1: 525 < 1.245%
S5 G 2: 691 < 1.639%
Total: 42171
=====
Action Stats:
Alerts: 22780 < 54.018%
Logged: 22780 < 54.018%
Passed: 0 < 0.000%
Limits:
Match: 0
Queue: 0
Log: 0
Event: 0
Alert: 2111
Verdicts:
Allow: 39163 < 95.629%
Block: 0 < 0.000%
Replace: 0 < 0.000%
Whitelist: 1790 < 4.371%
Blacklist: 0 < 0.000%
Ignore: 0 < 0.000%
<null>: 0 < 0.000%
=====
Frag3 statistics:
Total Fragments: 7
Frag Reassembled: 2
Discards: 0
Memory Faults: 0
Timeouts: 0
Overlaps: 0
Anomalies: 0
Alerts: 0
Drops: 0
FragTrackers Added: 4
FragTrackers Dumped: 4
FragTrackers Auto Freed: 0
Frag Nodes Inserted: 7
Frag Nodes Deleted: 7
=====

```

Figure 3-9 snapshot of detecting p2p packet from trace of internet.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.5.85	54042	74.125.135	80	00:16:EO:C00:00:5E:0	0x598	***AP***	0xDAEE87	0xADE9DCEB	0xF696	63	0	130			
2	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.4.156	49657	5.144.132	80	00:16:EO:C00:00:5E:0	0x46	***A****	0xF73828E	0xD6A92DEA	0x4125	127	0	252			
3	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	161.139.14	49179	10.19.8.5	554	00:23:89:A00:16:EO:C	0x40	***A****	0x8D3022	0x270CF316	0x8000	124	0	312			
4	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.5.85	54083	74.125.135	80	00:16:EO:C00:00:5E:0	0x1F5	***AP***	0x9133D8f	0xB1C5BA25	0xF707	63	0	130			
5	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.5.85	54192	173.194.22	80	00:16:EO:C00:00:5E:0	0x42	*****S	0xEDAD6C	0x0	0x2000	63	0	130			
6	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.19.8.5	554	161.139.14	49179	00:16:EO:C00:00:5E:0	0x53A	***AP***	0x270CF31	0x8D302275	0xFFFF	63	0	445			
7	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	5.144.132	80	10.8.4.156	50995	00:23:89:A00:16:EO:C	0x562	***AP***	0xEEFE97f	0x25E281CC	0x7B	33	0	198			
8	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	5.144.132	80	10.8.4.156	50995	00:23:89:A00:16:EO:C	0x562	***A****	0xEEFEBBf	0x25E281CC	0x7B	33	0	198			
9	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.4.156	50995	5.144.132	80	00:16:EO:C00:00:5E:0	0x46	***A****	0x25E281C	0xEEFEBBB7	0x4125	127	0	252			
10	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.4.156	50995	5.144.132	80	00:16:EO:C00:00:5E:0	0x46	***A****	0x25E281C	0xEEFEC0D3	0x4125	127	0	252			
11	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.7.45	56372	69.171.235	80	00:16:EO:C00:00:5E:0	0x40	***A****	0xE10F082	0xA7520FB9	0x3FBC	127	0	41			
12	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.9.186	56352	108.161.16	80	00:16:EO:C00:00:5E:0	0x46	***A****	0x37D4FBf	0xF8B23F0D	0x4029	127	0	315			
13	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	2.16.219.8	443	10.8.3.42	51903	00:23:89:A00:16:EO:C	0x406	***AP***	0x8A3749f	0xAD959531	0x1E9C	42	0	321			
14	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.5.148	64861	125.56.215	443	00:16:EO:C00:00:5E:0	0x5A	***A****	0xA24D9E	0x168CF594	0x410C	127	0	208			
15	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	10.8.9.118	50760	54.239.155	80	00:16:EO:C00:00:5E:0	0x40	***A****	0x260DDC	0x9C9BF51	0x4029	127	0	11			
16	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	103.21.81	80	10.8.6.121	63887	00:23:89:A00:16:EO:C	0x486	***AP***	0x130195E	0x3CA78EE7	0x36	40	0	84			
17	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	23.23.138	443	10.8.4.26	43595	00:23:89:A00:16:EO:C	0x46	***A****	0xDAAC9E	0xCBDF67FC	0x250	34	0	645			
18	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	23.23.138	443	10.8.4.26	43595	00:23:89:A00:16:EO:C	0x46	***A****	0xDAAC9E	0xCBDF734C	0x250	34	0	645			
19	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	23.23.138	443	10.8.4.26	43595	00:23:89:A00:16:EO:C	0x46	***A****	0xDAAC9E	0xCBDF6DA4	0x250	34	0	645			
20	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	23.59.191	443	10.8.9.148	52195	00:23:89:A00:16:EO:C	0x1E1	***AP***	0x53E0C6f	0xD33ED6E0	0x22CC	41	0	124			
21	05/13-02:42:05.812139	1	1000002	0	testing p2 TCP	208.117.24	80	10.8.6.73	52809	00:23:89:A00:16:EO:C	0x5EA	***A****	0x14A49E	0x64C5E402	0x112	46	0	90			

Figure 3-10 snapshot of packet save as CSV.

Then create a flow of packet by excel sheet and extracted the needed feature and remove unneeded one by the same step of offline phase in chapter four.

Chapter Four: Result and discussion

4.1 Overview of the prototype system

Building two prototype of the system one for offline used to determine the feature needed and ML algorithm used with accepted accuracy then build the online one according with needed details.

4.1.1 Offline Phase

Build prototype of the system using WIRESHARK to capture the traffic as shown in figure 4-1.

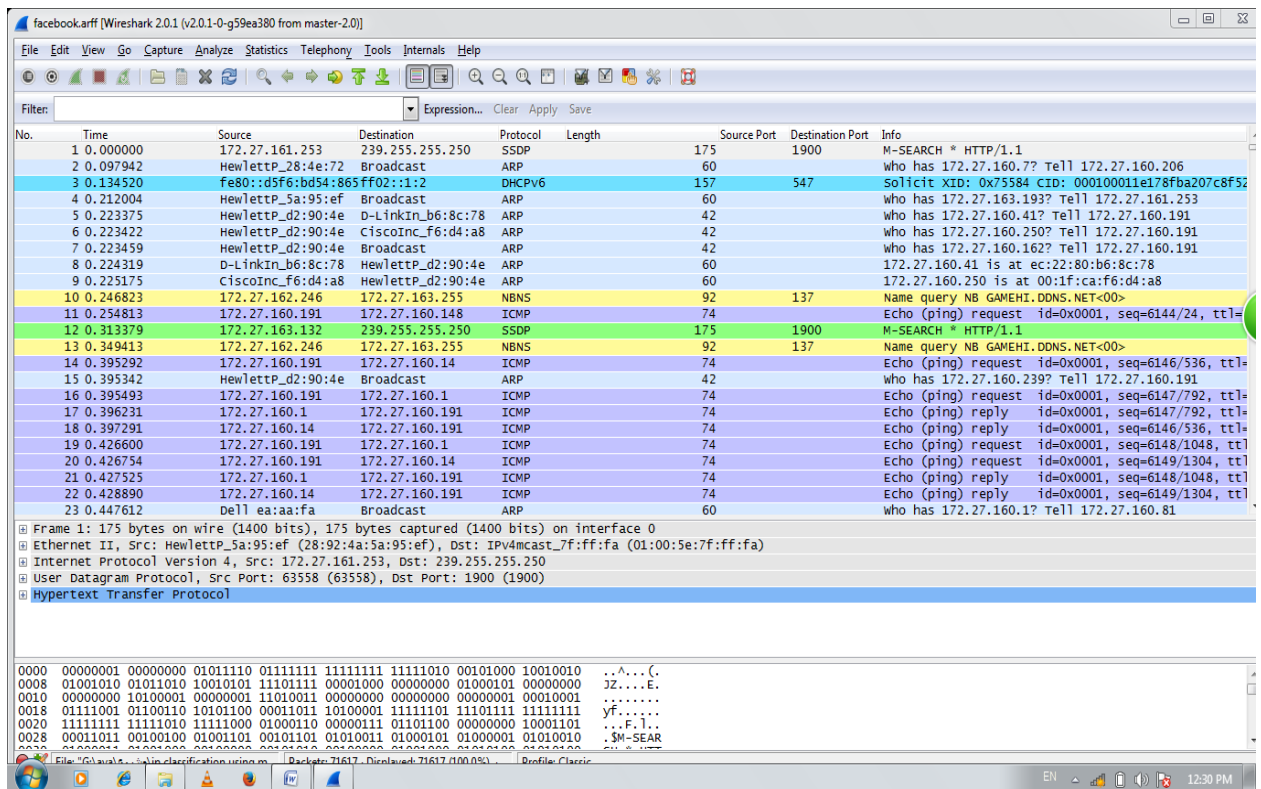


Figure 4-1 snapshot of data capture by WIRESHARK.

And then save the data capture as CSV (to become readable by excel sheet) as shown in figure 4-2.

Source Po	Destination	Source Po	Protocol	Total Leng	Arrival Tim	Differential	Protocol	Frame len	Protocol	Destination	Source Po	Length	Protocol	Destination	Source	Time	No.
80					Oct 17, 2016 14:17:20.05813600			86			80	86	TCP	2c0f.fec8:82a00.1450		0	1
57590			TCP	48	Oct 17, 20 0x00		TCP	62	TCP		57590	62	TCP	173.192.22.172.27.163	0.085157		2
443			TCP	71	Oct 17, 20 0x00		TCP	85	TCP		443	85	TLSv1.2	172.27.163:173.241.24	0.230738		3
443			TCP	40	Oct 17, 20 0x00		TCP	60	TCP		443	60	TCP	172.27.163:173.241.24	0.230886		4
57458			TCP	40	Oct 17, 20 0x00		TCP	54	TCP		57458	54	TCP	173.241.24:172.27.163	0.230987		5
57458			TCP	40	Oct 17, 20 0x00		TCP	54	TCP		57458	54	TCP	173.241.24:172.27.163	0.231524		6
57445			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57445	55	TCP	62.67.193. 172.27.163	0.235136		7
					Oct 17, 2016 14:17:20.36438800			60				60	ARP	Broadcast Shenzhen	0.306252		8
57577			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57577	55	TCP	198.47.121:172.27.163	0.307177		9
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:198.47.127	0.308098		10
443			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		443	66	TCP	172.27.163:62.67.193	0.329385		11
57473			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57473	55	TCP	104.122.22:172.27.163	0.343218		12
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:104.122.22	0.344121		13
57547			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57547	55	TCP	104.103.22:172.27.163	0.425227		14
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:104.103.22	0.426175		15
443			TCP	40	Oct 17, 20 0x00		TCP	60	TCP		443	60	TCP	172.27.163:173.241.24	0.440678		16
57463			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57463	55	TCP	104.122.22:172.27.163	0.455196		17
57571			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57571	55	TCP	66.117.28. 172.27.163	0.495182		18
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:66.117.28	0.496078		19
57573			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57573	55	TCP	66.117.28. 172.27.163	0.517176		20
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:66.117.28	0.518092		21
443			TCP	52	Oct 17, 20 0x28		TCP	66	TCP		443	66	TCP	172.27.163:104.122.22	0.576418		22
57568			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57568	55	TCP	104.103.22:172.27.163	0.633104		23
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:104.103.22	0.633974		24
57565			TCP	41	Oct 17, 20 0x00		TCP	55	TCP		57565	55	TCP	66.117.28. 172.27.163	0.645149		25
80			TCP	52	Oct 17, 20 0x00		TCP	66	TCP		80	66	TCP	172.27.163:66.117.28	0.646061		26

Figure 4-2 snapshot of data capture as csv.

Doing pre-processing of capture packet by excel sheet to convert it to flow (by open the CSV file and sort with the 5-tuple (source port ,destination port , source IP ,destination IP and protocol) then calculate the average of the size , the total number of byte and duration of the same 5-tuple and save as CSV again) as shown in figure 4-3.

Destination Port	Source Port	total number of byte	Protocol	Destination	Source
1900		408	SSDP	239.255.255.250	172.27.160.184
	80	313110	TCP	172.27.163.172	95.101.34.51
137		276	NBNS	172.27.163.255	172.27.160.70
	443	300	TLSv1.2	172.27.163.172	216.58.208.226
138		243	BROWSER	172.27.163.255	172.27.161.158
68		357	DHCP	255.255.255.255	172.27.160.1
	8080	3016	HTTP	172.27.163.172	41.67.53.144
		240	IGMPv3	224.0.0.22	169.254.139.130
	443	594	TCP	172.27.163.172	185.29.133.208
4008		730	tcp	172.27.163.172	103.235.46.9
	80	1428	tcp	172.27.163.172	104.103.237.114
	80	2416	http	172.27.163.172	151.101.61.7
	80	642	tcp	172.20.163.172	156.154.200.36
	443	2822	TLSv1.2	172.27.163.172	172.217.20.34
	51084	492	TCP	172.27.163.172	172.27.161.119
	80	516	Tcp	172.27.163.172	185.84.60.25
	80	2173	TCP	127.27.163.172	192.132.33.31
	80	11932	TCP	172.27.163.172	2.22.62.114
	49445	3294	DNS	172.27.163.172	41.67.16.2
	80	12790	TCP	172.27.163.172	104.103.236.189
	80	23150	TCP	172.27.163.172	104.116.245.27
	443	640	TLSv1.2	172.27.163.172	108.160.172.206
	443	462	TCP	172.27.163.172	104.122.200.225
	443	246	TCP	172.27.163.172	128.30.52.100
	443	1057	TCP	172.27.163.172	13.107.21.200
	443	450	TCP	172.27.163.172	136.243.131.40

Figure 4-3 snapshot of flow data capture as CSV.

Then select the needed feature (source port, destination port, protocol, average size, average inter-arrival time and total number of byte), deleting other feature as shown in figure 4-4.

avareage-size	Flags	Destination Port	Source Port	total number of byte	Protocol	duration
136	0x00	1900		408	SSDP	43.23442
1379.339	0x00		80	313110	TCP	50.54809
92	0x00	137		276	NBNS	24.46076
100	0x00		443	300	TLSv1.2	48.61669
357	0x00	138		243	BROWSER	43.06447
228.5	0x00	68		357	DHCP	10.2762
1508	0x02		8080	3016	HTTP	66.51545
60	0x00			240	IGMPv3	9.320488
66	0x02		443	594	TCP	47.93612
73	0x02		4008	730	tcp	94.10976
64.90909	0x02		80	1428	tcp	83.79798
1208	0x02		80	2416	http	86.23.949
64.2	0x02		80	642	tcp	51.87892
470.3333	0x00		443	2822	TLSv1.2	95.12286
61.5	0x02		51084	492	TCP	24.64125
64.5	0x02		80	516	Tcp	41.35414
197.545	0x02		80	2173	TCP	99.38959
947.0909	0x02		80	11932	TCP	61.96517
253.3846	0x00		49445	3294	DNS	66.75184
737.6471	0X02		80	12790	TCP	58.59948
756.667	0X02		80	23150	TCP	52.40262
199.3333	0X02		443	640	TLSv1.2	31.96651
52	0X02		443	462	TCP	35.1392
43	0X02		443	246	TCP	12.4659
90.5	0X02		443	1057	TCP	30.31535
48.57143	0X02		443	450	TCP	31.68326

Figure 4-4 snapshot of flow data capture include needed feature as CSV.

Add class name to the tanning data set as shown in figure4-5 below.

class	avareage-size	Flags	Destination Port	Source Port	total number of byte	Protocol	duration
vedio	136	0x00	1900		408	SSDP	43.23442
vedio	1379.339	0x02		80	313110	TCP	50.54809
vedio	92	0x00	137		276	NBNS	24.46076
vedio	100	0x00		443	300	TLSv1.2	48.61669
vedio	357	0x00	138		243	BROWSER	43.06447
vedio	228.5	0x00	68		357	DHCP	10.2762
vedio	1508	0x02		8080	3016	HTTP	66.51545
vedio	60	0x00			240	IGMPv3	9.320488
vedio	66	0x02		443	594	TCP	47.93612
p2p	73	0x02		4008	730	tcp	94.10976
p2p	64.90909	0x02		80	1428	tcp	83.79798
p2p	1208	0x02		80	2416	http	86.23.949
p2p	64.2	0x02		80	642	tcp	51.87892
p2p	470.3333	0x00		443	2822	TLSv1.2	95.12286
p2p	61.5	0x02		51084	492	TCP	24.64125
p2p	64.5	0x02		80	516	Tcp	41.35414
p2p	197.545	0x02		80	2173	TCP	99.38959
p2p	947.0909	0x02		80	11932	TCP	61.96517
p2p	253.3846	0x00		49445	3294	DNS	66.75184
HTTP	737.6471	0X02		80	12790	TCP	58.59948
HTTP	756.667	0X02		80	23150	TCP	52.40262
HTTP	199.3333	0X02		443	640	TLSv1.2	31.96651
HTTP	52	0X02		443	462	TCP	35.1392
HTTP	43	0X02		443	246	TCP	12.4659
HTTP	90.5	0X02		443	1057	TCP	30.31535
HTTP	48.57143	0X02		443	450	TCP	31.68326

Figure 4-5 snapshot of flow data pre-classified saved as CSV.

By using of ARFF viewer convert excel sheet to ARFF format accepted by WEKA platform as shown in figure 4-6.

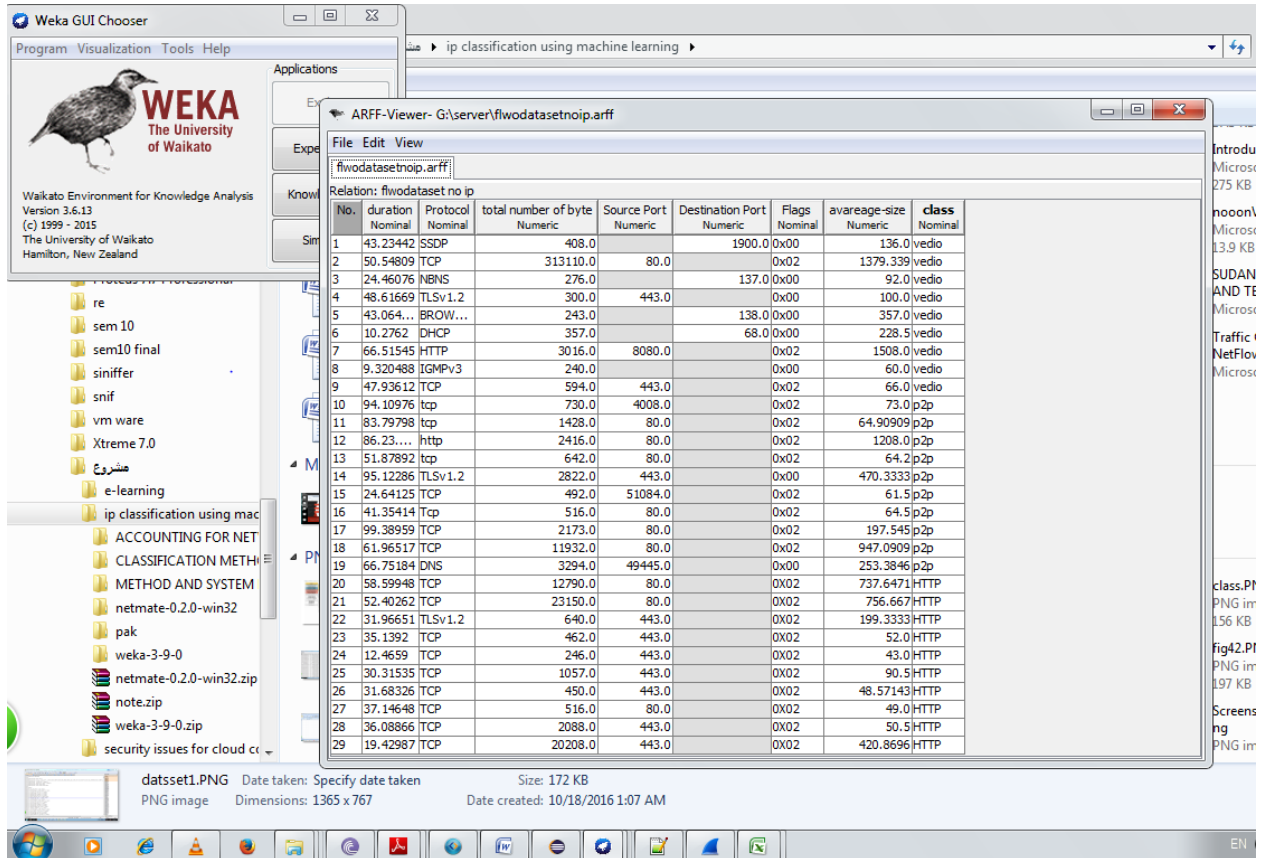


Figure 4-6 snapshot of flow data capture include needed feature as ARFF.

By using above steps now ready to extract the training and testing data set below.

Training dataset 1

The data set conotation of 10 flows for every application type and contain of 7 features as shown in figure 4-7.

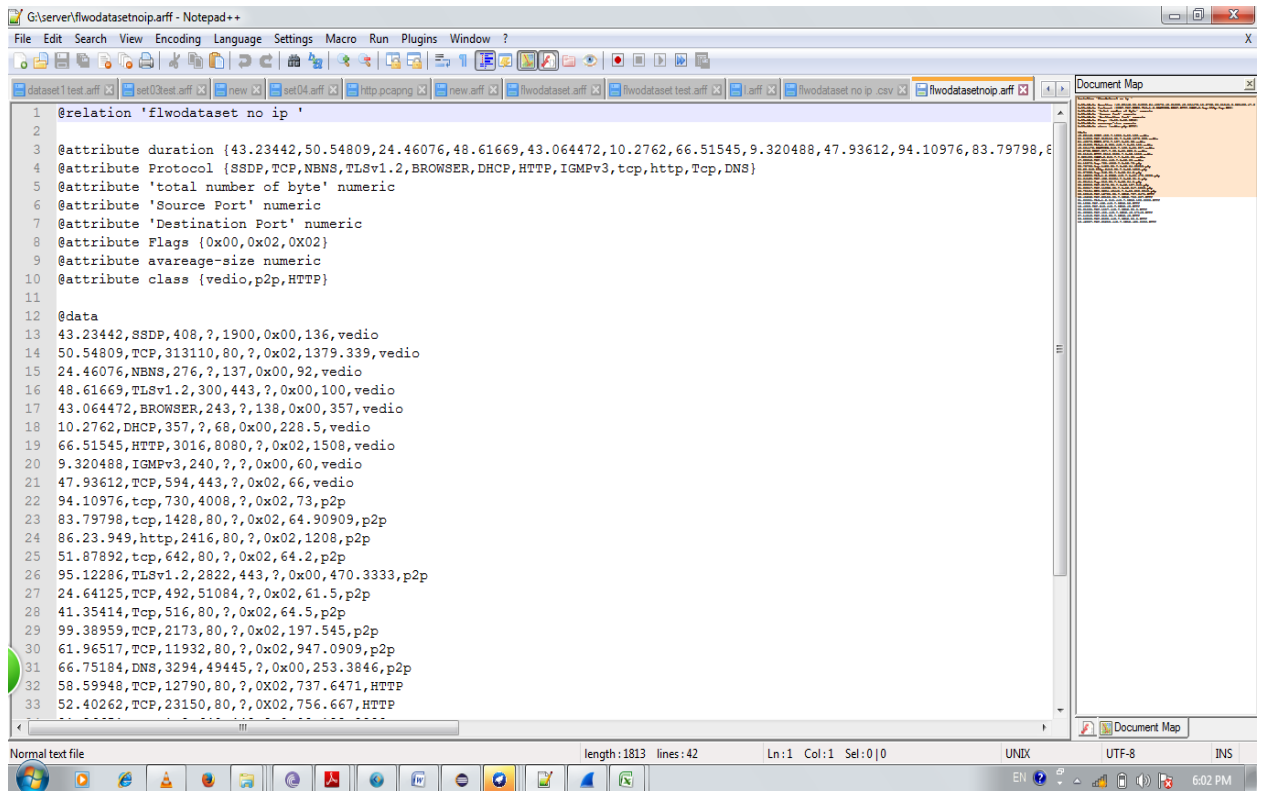


Figure 4-7 snapshot training dataset as AEF format.

Then open WEKA and enter the data set of training to train MLA (after trying many algorithm like J48 tree,bayes navie and other find J48 is more stable one)as shown in figure 4-8 below.

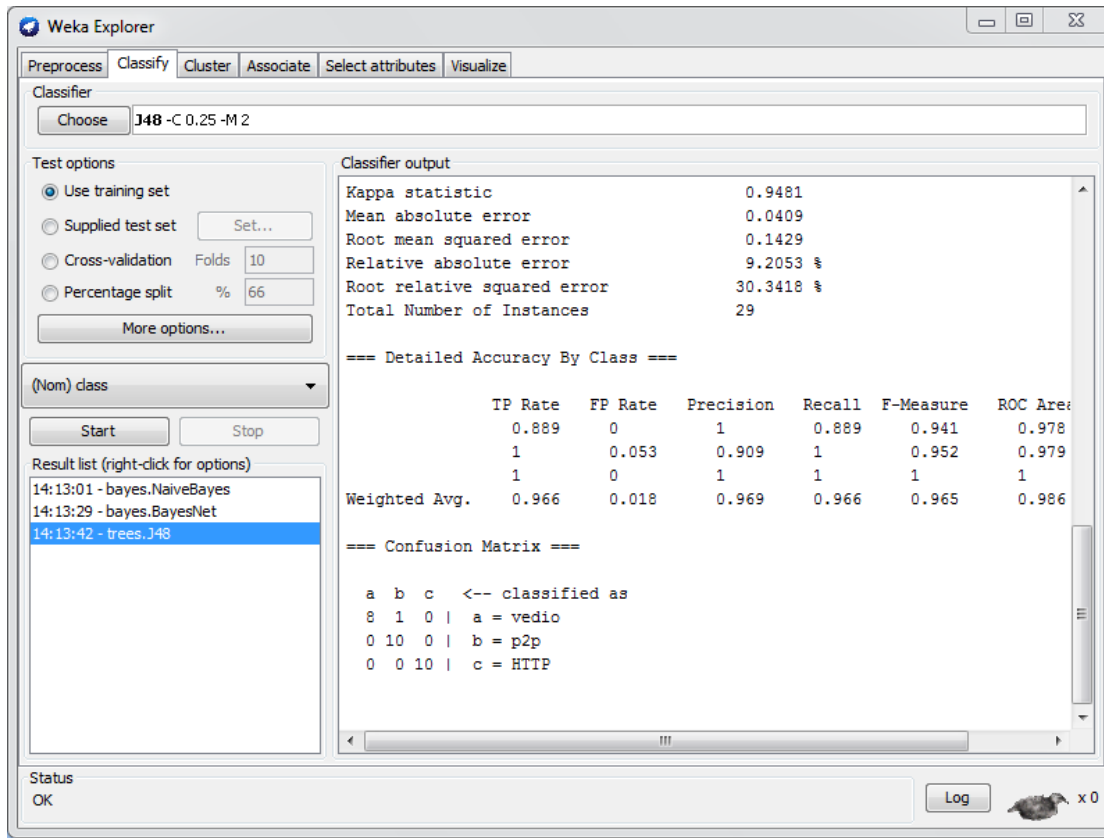


Figure 4-8: snapshot of the training process doing by WEKA.

Then select the cross validation mode as show in figure 4-9.

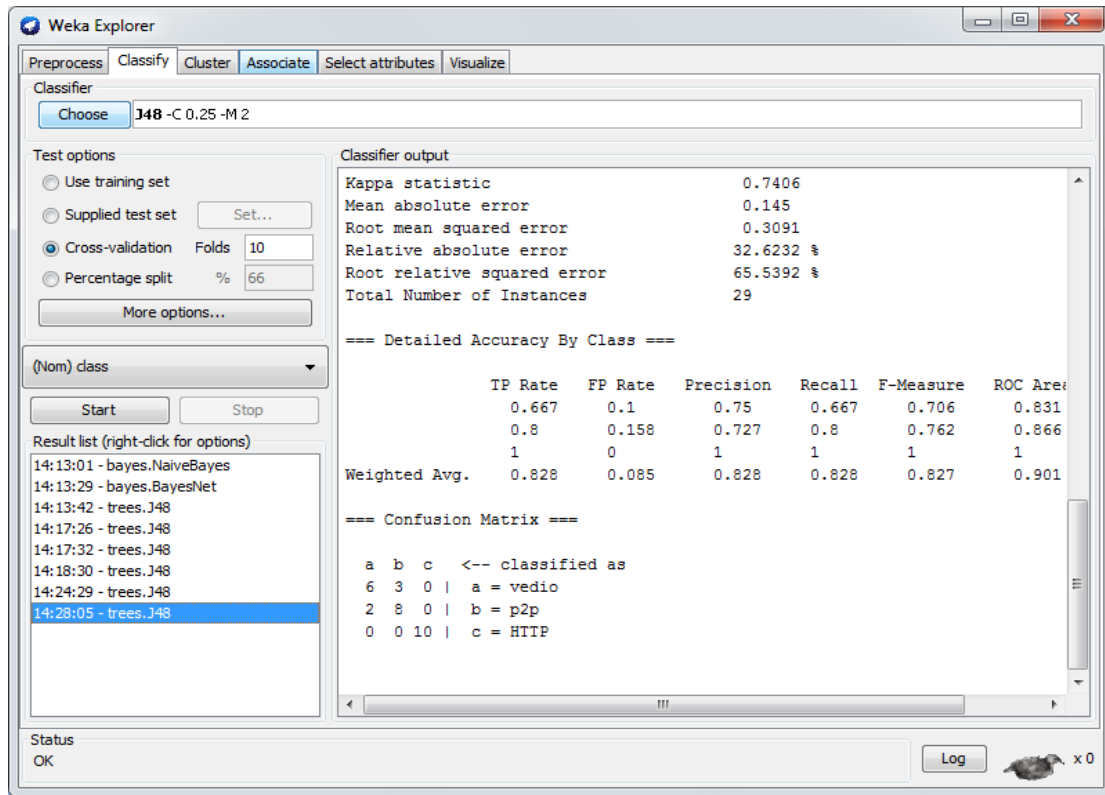


Figure 4-9: snapshot of cross validation process doing by WEKA.

Table 4.1 Final result of data set 1.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.828	0.085	0.828	0.828	0.827	0.901	Average
0.667	0.1	0.75	0.667	0.706	0.831	Vedio
0.8	0.158	0.727	0.8	0.762	0.866	p2p
1	0	1	1	1	1	HTTP

4.1.2 Training Data Set 2

The data captured and detected by snort convert it to flow by excel sheet then remove non needed feature then forwarded to ARFF format.

The data set conation of 10 flows and contain of 7 features. By apply the above data set to MLA selecting by above step in weka platform reach the following result as shown in below table4.2.

Table 4.2 Final result of data set 2.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0.2	0.8	1	0.889	0.9	P2P
1	0	1	1	1	1	HTT P
0	0	0	0	0	0.75	video
0.889	0.089	0.8	0.889	0.84	0.928	Average

4.2 Evaluation and Validation

High accuracy in field of network classification means low (positive false, False negative rate) and high (True positive, True negative) the table 4-3below explain.

Table 4.3 Evaluation metrics.

Belongs to →	X	Y
X	True positive	False negative
Y	False positive	True negative

If it is a class “X” in which we are interested then the accuracy with these parameters is measured as:

False positive → percentage of members belonging to Y but classified as X.

True positive → percentage of members of class X correctly classified as X.

False negative → percentage of members belonging to X incorrectly classified as Y.

True Negative → percentage of members not belonging to X and correctly classified as Y.

There are two more metrics which are often used as Machine learning evaluation metrics:

Recall: percentage of members belonging to X and correctly classified.

Precision: percentage of member classified as X truly belongs to X.

$$\text{Accuracy} = \frac{\text{true positive}}{\text{true positives} + \text{false negative}}$$

Accuracy from data set 1 = 82.8%

Accuracy from data set 2 = 88.9%

4.3 Conclusion

As shown in Table 4.1 and Table 4.2 reach different Accuracy by the same system and same feature that meaning the system is stable and can be reach high Accuracy depending on training data set. More accurate data set depending on efficiencies analysis of the data and the number of instance in data set.

4.4 Online Phase

In this phase try to build system that capture the network traffic and then save the capture packet and then convert it to short flow and then

extract the needed feature from flow and then use the ML a logarithm to classify the flow to specific application type.

Select java language because the power of java runs anywhere JVM fond.

First of all create java project and import JNETPCAP library and WEKA library in form of JAR file and then write simple code to do the job.

Capture the traffic first select the interface listening to as show in the below figure 4-10.

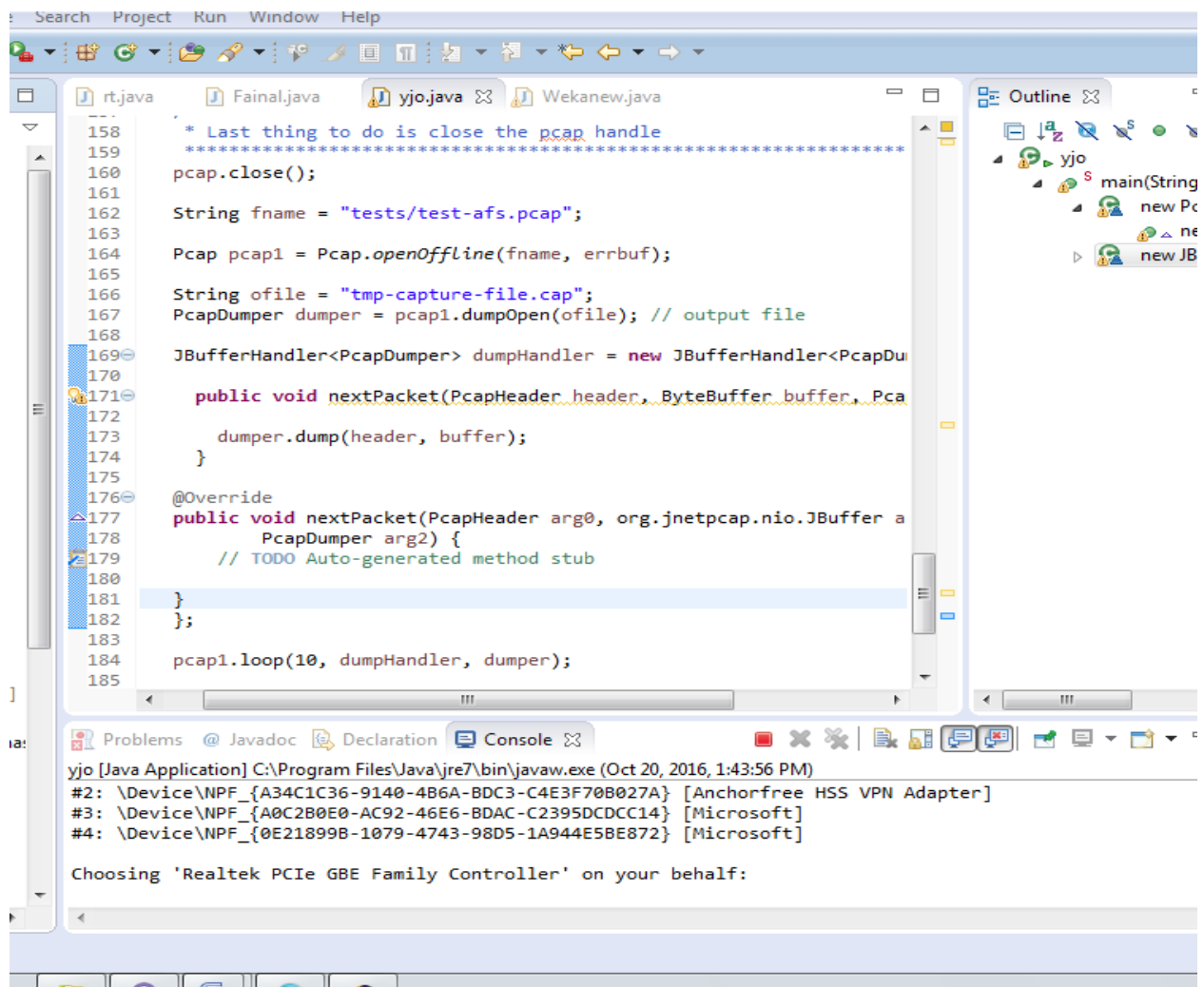


Figure 4-10 snapshot of select the interface listening to.

Then capture the traffic by 10 packet every time and extract the below feature from every packet (Source IP , Destination IP , source port , destination port, protocol, total length , size and inter arrive time) and save the feature as array (as show in the below figure 4-11).

The screenshot shows the Eclipse IDE with a Java file named 'yjo.java' open. The code defines a 'Tcp' class and a 'main' method that captures network traffic. The code extracts source and destination IP addresses, source and destination ports, and the total length of the packet. It also checks for the HTTP protocol. The console output shows the results of the capture, listing source and destination IP addresses, source and destination ports, and total lengths for several packets.

```

110     Tcp tcp = new Tcp();
111
112     Ip4 ip = new Ip4();
113     byte[] sIP = new byte[4];
114     byte[] dIP = new byte[4];
115     byte g = packet.getBytes(o);
116     //System.out.println(g);
117     //int flag=tcp.flags();
118
119     //
120     String sourceIP="";
121     String destIP="";
122     if(packet.getHeader(ip)&&packet.getHeader(tcp)){
123         sIP = packet.getHeader(ip).source();
124         sourceIP = org.jnetpcap.packet.format.FormatUtil.
125         dIP = packet.getHeader(ip).destination();
126         destIP = org.jnetpcap.packet.format.FormatUtils.
127         // System.out.println(""+sourceIP+"*"+destIP);
128         System.out.print("Source IP "+ sourceIP+"Destin
129         // System.out.println("Destination IP "+ destIP);
130         // System.out.println("source port "+tcp.source(
131         // System.out.println("destination port "+tcp.de
132         // System.out.println("size "+s);
133         // System.out.println("total length"+g);
134         if(tcp.source()==80){
135             System.out.println("HTTP protocol");

```

```

<terminated> yjo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 20, 2016, 2:20:50 PM)
Source IP 172.27.163.172Destination IP 103.235.46.9source port 49394destination port 4007size141total
Source IP 103.235.46.9Destination IP 172.27.163.172source port 4007destination port 49394size60total
Source IP 103.235.46.9Destination IP 172.27.163.172source port 4007destination port 49394size120total
Source IP 103.235.46.9Destination IP 172.27.163.172source port 4007destination port 49394size60total
Source IP 172.27.163.172Destination IP 103.235.46.9source port 49394destination port 4007size54total
Source IP 172.27.163.172Destination IP 103.235.46.9source port 49394destination port 4007size54total
Source IP 103.235.46.9Destination IP 172.27.163.172source port 4007destination port 49394size60total
Source IP 172.27.163.15Destination IP 172.27.163.172source port 49593destination port 2869size60total

```

Figure 4-11 snapshot of packet capture.

Convert packet to flow for every 10 packet look at Source IP , Destination IP , source port , destination port, protocol and sort the array of packet based on the 5-tuple and save the same packet with 5-tuple as

one flow (as show in the below figure4-12).

The screenshot shows an IDE with a Java file named `Fainal.java` open. The code is a snippet of a network flow analysis program. It uses `System.out.printf` to print TCP and HTTP header information and frame numbers. The console output at the bottom shows several network flows with their respective IP addresses, ports, protocols, and packet statistics.

```

69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

System.out.printf("tcp.src_port=%d\n", tcp.source
System.out.printf("tcp.ack=%x\n", tcp.ack());
}
if (packet.hasHeader(tcp)) {
    System.out.printf("tcp header::%s\n", tcp.toStri
}
if (packet.hasHeader(tcp) && packet.hasHeader(http))

    System.out.printf("http header::%s\n", http);
}
System.out.printf("frame #d\n", packet.getFrameNumb
}
}, errbuf);

```

```

<terminated> Fainal [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 20, 2016, 2:08:24 PM)
flow[186] 31.13.75.8:443 -> 192.168.101.11:47045 Tcp fw/rev/tot pkts=[60/0/60],
flow[187] 31.13.75.8:443 -> 192.168.101.11:58456 Tcp fw/rev/tot pkts=[32/0/32],
flow[188] 31.13.75.8:443 -> 192.168.101.11:58457 Tcp fw/rev/tot pkts=[36/0/36],
flow[189] 192.168.101.11 -> 192.168.101.255:17 Ip4 tot pkts=[1],
flow[190] 216.58.210.162:443 -> 192.168.101.11:53162 Tcp fw/rev/tot pkts=[60/0/60],
flow[191] 203.205.151.159:8080 -> 192.168.101.13:55263 Tcp fw/rev/tot pkts=[20/0/20],
flow[192] 192.168.101.11 -> 192.168.101.1:17 Ip4 tot pkts=[8],
flow[193] 31.13.75.8:443 -> 192.168.101.11:58447 Tcp fw/rev/tot pkts=[4/0/4],
flow[194] 192.168.101.11 -> 192.168.101.255:17 Ip4 tot pkts=[1],

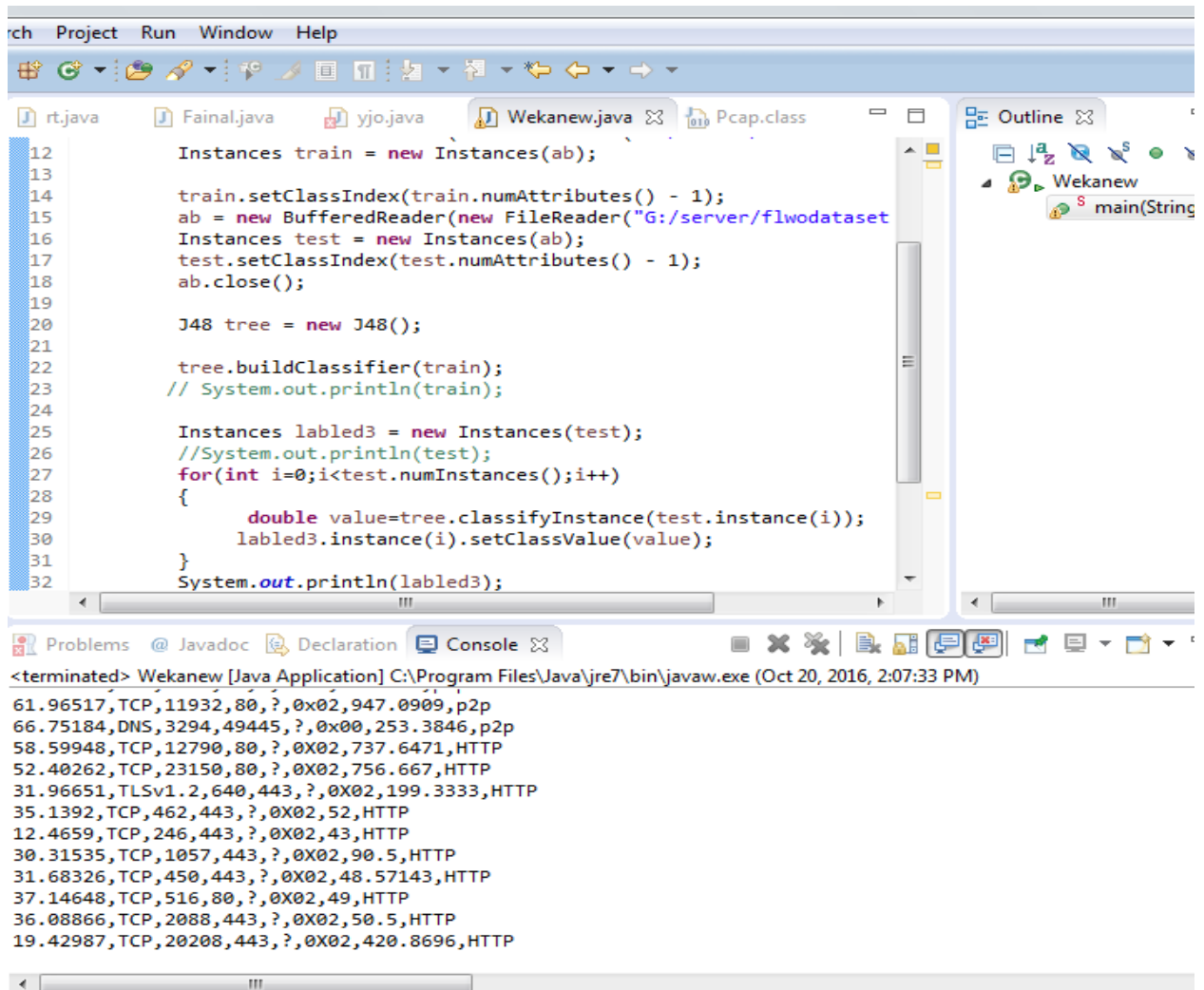
```

Figure 4-12 snapshot of creating flow of packet.

Feature extract for every flow calculate the average size (by average the size of all packet on flow),total number of byte (by sum the total length of all packet on flow) and duration(by subtract every two packet inter arrive time for every packet on flow) then save the Source IP , Destination IP , source port , destination port, protocol, total number of byte , average size and inter arrive time for every flow as array and save array on text file .

Convert the text file to ARFF very sad to notate that this function is non work very well. But to show the next step working use offline converter.

Classify the flow to specifics application build an object from J48 tree to use as classification a logarithm. Read the file of training data set and train the tree by the train data and then read the test data set from the ARRF file save in before step and then display the class of each flow(as show in the below figure4-13).



```

12     Instances train = new Instances(ab);
13
14     train.setClassIndex(train.numAttributes() - 1);
15     ab = new BufferedReader(new FileReader("G:/server/flwodataset
16     Instances test = new Instances(ab);
17     test.setClassIndex(test.numAttributes() - 1);
18     ab.close();
19
20     J48 tree = new J48();
21
22     tree.buildClassifier(train);
23     // System.out.println(train);
24
25     Instances labled3 = new Instances(test);
26     //System.out.println(test);
27     for(int i=0;i<test.numInstances();i++)
28     {
29         double value=tree.classifyInstance(test.instance(i));
30         labled3.instance(i).setClassValue(value);
31     }
32     System.out.println(labled3);

```

```

<terminated> Wekanew [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 20, 2016, 2:07:33 PM)
61.96517, TCP, 11932, 80, ?, 0x02, 947.0909, p2p
66.75184, DNS, 3294, 49445, ?, 0x00, 253.3846, p2p
58.59948, TCP, 12790, 80, ?, 0X02, 737.6471, HTTP
52.40262, TCP, 23150, 80, ?, 0X02, 756.667, HTTP
31.96651, TLSv1.2, 640, 443, ?, 0X02, 199.3333, HTTP
35.1392, TCP, 462, 443, ?, 0X02, 52, HTTP
12.4659, TCP, 246, 443, ?, 0X02, 43, HTTP
30.31535, TCP, 1057, 443, ?, 0X02, 90.5, HTTP
31.68326, TCP, 450, 443, ?, 0X02, 48.57143, HTTP
37.14648, TCP, 516, 80, ?, 0X02, 49, HTTP
36.08866, TCP, 2088, 443, ?, 0X02, 50.5, HTTP
19.42987, TCP, 20208, 443, ?, 0X02, 420.8696, HTTP

```

Figure 4-13 snapshot of classifier result.

Chapter five: Conclusion and Recommendations

5.1 Conclusion

In this thesis demonstrated the application of supervised Machine-Learning to classify network traffic by application. We illustrated the performance both in terms of accuracy and trust in the resulting classification of traffic. Several machine learning classifiers have been evaluated, results showed that a C4.5 tree is most stable supervised Machine-Learning for classifier is able to provide 82.7% accuracy from training data set capture by WIRESHARK and 88.9% from data set capture by snort. Furthermore, this research also starts building an online classification module which based on the idea of short flows so as to enable the concept of real time reaction or control; however a remaining work is still there to complete the module in its final version.

5.2 Recommendations

- ✓ Although, an offline and online phases of the classifiers have been posed in this research, however In future researchers must plan to examine the machine learning classifiers with different data sets in addition to different number of features. That is to choose the optimum algorithm in different situations. Moreover, it is also to fine tuning the selected algorithm according to the suitable training datasets with acceptable number of features so as to reach the goal of an accurate classifier with minimum testing time. Thus, to deal with the quick response so as to give fastest recovery.
- ✓ Update the dataset so as to include several application types
- ✓ Although a prototype of the online classification has been presented in this thesis, however an additional effort is still required to complete the online phase and apply a system on a real network based on this

framework; moreover an evaluation and validation of the online is required so as to determine the capabilities and limitation of the online classifiers among different scenarios according to the network environment.

- ✓ The outcome of the previous point (online phase) should be integrated with a control phase to enhance the performance according to administrative needed.

Reference

- [1] S.-H. Yoon, K.-S. Shim, S.-K. Lee, and M.-S. Kim, "Framework for multi-level application traffic identification." pp. 424-427.
- [2] T. Qin, L. Wang, Z. Liu, and X. Guan, "Robust application identification methods for P2P and VoIP traffic classification in backbone networks," *Knowledge-Based Systems*, vol. 82, pp. 152-162, 2015.
- [3] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification," *Data Traffic Monitoring and Analysis*, pp. 123-147: Springer, 2013.
- [4] 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 Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 4, 2012.
- [5] P. Bermolen, M. Mellia, M. Meo, D. Rossi, and S. Valenti, "Abacus: Accurate behavioral classification of P2P-TV traffic," *Computer Networks*, vol. 55, no. 6, pp. 1394-1411, 2011.
- [6] H. Singh, "Performance Analysis of Unsupervised Machine Learning Techniques for Network Traffic Classification." pp. 401-404.
- [7] "<http://searchnetworking.techtarget.com/definition/peer-to-peer> accessed on 1/3/2016."
- [8] "<http://www2.connectseward.org/shs/students/students15/Q3/daltongoke/p2p/page2.html> accessed on 1/3/2016."
- [9] B. Anton, and A. Norbert, "PEER TO PEER SYSTEM DEPLOYMENT," *Acta Electrotechnica et Informatica*, vol. 16, no. 1, pp. 11-14, 2016.
- [10] D. S. Touceda, J. M. S. Cámara, and J. T. Isaac, "Privacy in Peer-to-Peer Networks," *Privacy in a Digital, Networked World*, pp. 111-139: Springer, 2015.
- [11] R. Raveendran, and R. Menon, "An Efficient Method for Internet Traffic Classification and Identification using Statistical Features."
- [12] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection classifier for udp traffic," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1505-1515, 2010.
- [13] J. Kim, J. Hwang, and K. Kim, "High-Performance Internet Traffic Classification Using a Markov Model and Kullback-Leibler Divergence," *Mobile Information Systems*, vol. 2016, 2016.
- [14] A. Dainotti, A. Pescapé, and C. Sansone, "Early classification of network traffic through multi-classification." pp. 122-135.

- [15] S.-H. Yoon, J.-S. Park, and M.-S. Kim, "Behavior Signature for Fine-grained Traffic Identification," *Appl. Math*, vol. 9, no. 2L, pp. 523-534, 2015.
- [16] L. Peng, B. Yang, Y. Chen, and Z. Chen, "Effectiveness of Statistical Features for Early Stage Internet Traffic Identification," *International Journal of Parallel Programming*, vol. 44, no. 1, pp. 181-197, 2016.
- [17] O. Mula-Valls, "A practical retraining mechanism for network traffic classification in operational environments," Master Thesis in Computer Architecture, Networks and Systems, Universitat Politecnica de Catalunya, 2011.
- [18] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*: Springer Science & Business Media, 2013.
- [19] N. Namdev, S. Agrawal, and S. Silkari, "Recent Advancement in Machine Learning Based Internet Traffic Classification," *Procedia Computer Science*, vol. 60, pp. 784-791, 2015.
- [20] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, "Unsupervised traffic classification using flow statistical properties and IP packet payload," *Journal of Computer and System Sciences*, vol. 79, no. 5, pp. 573-585, 2013.
- [21] F. R. Taylor, "Evaluation of Supervised Machine Learning for Classifying Video Traffic," 2016.
- [22] B. Hu, and Y. Shen, "Machine learning based network traffic classification: a survey," *Journal of Information and Computational science*, vol. 9, no. 11, pp. 3161-3170, 2012.
- [23] N.-F. Huang, G.-Y. Jai, H.-C. Chao, Y.-J. Tzang, and H.-Y. Chang, "Application traffic classification at the early stage by characterizing application rounds," *Information Sciences*, vol. 232, pp. 130-142, 2013.
- [24] A. Zhu, "A P2P Network Traffic Classification Method Based on C4. 5 Decision Tree Algorithm." pp. 373-379.
- [25] R. Alshammari, and A. N. Zincir-Heywood, "Identification of VoIP encrypted traffic using a machine learning approach," *Journal of King Saud University-Computer and Information Sciences*, vol. 27, no. 1, pp. 77-92, 2015.
- [26] V. D'Alessandro, B. Park, L. Romano, and C. Fetzer, "Scalable network traffic classification using distributed support vector machines." pp. 1008-1012.
- [27] P. Pinky, and S. V. Edwards, "A Survey on IP Traffic Classification Using Machine Learning."

- [28] J. Datta, N. Kataria, and N. Hubballi, "Network traffic classification in encrypted environment: A case study of Google Hangout." pp. 1-6.
- [29] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194-208, 2014.
- [30] V. Carela-Espanol, P. Barlet-Ros, and J. Solé-Pareta, "Traffic classification with sampled netflow," *traffic*, vol. 33, pp. 34, 2009.
- [31] A. Hajjar, J. Khalife, and J. Díaz-Verdejo, "Network traffic application identification based on message size analysis," *Journal of Network and Computer Applications*, vol. 58, pp. 130-143, 2015.
- [32] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms." pp. 281-286.
- [33] Y. Hong, C. Huang, B. Nandy, and N. Seddigh, "Iterative-tuning support vector machine for network traffic classification." pp. 458-466.