Identification of Network Applications based on Machine Learning Techniques

Valentín Carela Español - vcarela@ac.upc.edu
Pere Barlet Ros - pbarlet@ac.upc.edu

UPC Technical Report

Deptartament d'Arqutiectura de Computadors
Universitat Politècnica de Catalunya

Abstract

Recently, traffic classification and, in particular, the identification of network applications has become a new and difficult challenge for both network operators and the network measurement community. A new generation of network applications, such as P2P, has started to consume a large amount of network resources, thus producing several problems to network operators.

Traditional identification techniques that rely on the well-known ports registered by the IANA are no longer valid because of the inaccuracy and incompleteness of its classification results. Moreover, the solution based on looking for characteristics patterns in the payload of the packets is becoming also invalid due to its overhead and privacy-related issues.

This situation has motivated us to study the problem of application identification in the network traffic. In this technical report, we present a supervised machine learning technique based on the well-known C4.5 decision tree to accurately identify the network traffic. Furthermore, we propose an automatic machine learning process that, unlike previous work, does not rely in any human intervention, which significantly simplifies the training phase present in all machine learning-based techniques.

We evaluate our method using an existing passive network monitoring system (SMARTxAC) in a large research and education network achieving an overall accuracy greater than 94%. Furthermore, we also adapt our method to operate with NetFlow data, which is an extended but scarcely investigated scenario, obtaining only an overall slightly lower accuracy, but using only the information that NetFlow provides.

Finally, we study the impact of sampling on the accuracy of our identification method and point out possible solutions to improve its accuracy in the presence of sampling.

Chapter 1

Introduction

Recently, bandwidth-eater applications, such as P2P and Video Streaming, have significantly increased their presence in the Internet. This class of applications is a challenge for network operators who have to deal with the management of the network resources. Network administrators need to know what is going over their networks in order to manage the traffic in accordance with their requirements (e.g., low delay for VoIP applications). Therefore, the identification of network applications is very valuable for the interests of ISPs. In addition, ISPs could use the identification of the traffic for usage-based charging and billing purposes [39].

Traditionally, the identification of network applications has been carried out using the well-known ports technique. This solution identifies the traffic according to the ports registered by the IANA [21]. This method is no longer valid because of the inaccuracy and incompleteness of its classification results. For example, new types of applications use different strategies to camouflage their traffic in order to evade detection. These obfuscation techniques have motivated the study of new methods to identify this type of traffic.

The first alternatives to the well-known ports method used the inspection of the packet payloads to identify the network traffic [2, 24–26, 31, 37]. These methods, usually called deep packet inspection techniques, examine the content of the packets looking for characteristic signatures. Although this solution can achieve high identification accuracy, its high resources requirements and limitations with encrypted traffic make it unpractical in nowadays high-speed networks.

In order to solve these limitations, the network measurement community has recently proposed several Machine Learning (ML) techniques for application identification [3, 6–8, 13–15, 18–20, 23, 32, 36, 38, 41, 48, 49]. Nguyen et al. survey and compare the complete literature in the field of ML techniques for application identification in [34]. These methods study, in an offline phase, different traffic features (e.g., *ports*, *packets*, *TCP flags*) trying to find characteristic patterns of network applications. These features are used to build a classifier, which is later used to

identify the traffic in real-time. Other alternatives to solve these problems include methods based on the host-behavior that can classify the traffic according to information extracted from the interactions of the end-hosts [27, 28, 46].

1.1 Contributions

The first contribution of this technical report is the study of a machine learning-based method to solve the application identification problem on high-speed link scenarios where the only current feasible solution is the (inaccurate) well-known ports method. In order to validate our solution we evaluate our method under the requirements of a real scenario. The *Anella Cientifica* is the Catalan Regional Research and Education Network (RREN) that connects the Catalan universities and research institutions to *RedIRIS*, the Spanish National Research and Education Network (NREN). Nowadays, the *Anella Cientifica* is based on 10Gb/s links. These links are monitored by a tailor-made monitoring system called SMARTxAC [4, 5].

The current load of the *Anella Cientifica* is almost 5Gb/s and is continuously increasing. Monitoring this huge amount of traffic is very problematic due to its large resource requirements. The well-known ports method used by the current SMARTxAC implementation does not consume many resources but is very inaccurate. However, a more accurate technique could result in uncontrolled packet drops due to the high resource requirements of more sophisticated classification methods. In order to avoid running out of resources in worst case scenarios or network attacks, the network operators usually apply sampling to the collected network traffic. For this reason, second contribution of this work is the study of the effects of the traffic sampling on the traffic classification methods.

The third contribution of this technical report is the study of the classification problem using NetFlow data instead of packet-level traces. NetFlow is a widely extended network protocol developed by Cisco to export IP flow information from routers and switches [9]. Unlike in the case where packet-level traces are available, when using NetFlow data the main constraint is the limited amount of existing information available to be used as features of machine learning-based methods. Another important limitation with NetFlow is the low sampling rates typically used by network operators (e.g. 1/1000) in order to prevent possible router saturation. We also address this limitation by studying how sampling affects our classification method with NetFlow data.

The last contribution of this work is the development of an automatic machine learning process. Unlike most of the related work, we present a method that is able to identify network applications without requiring any human intervention in the different phases of the classification process.

Chapter 2

Background and Related Work

This chapter describes the existing literature in the field of traffic classification. The application identification problem has been changing due the efforts of two factors that are in a continuous competition. On the one hand, the applications, and especially those that do not want to be detected (e.g., P2P applications), in order to use the network resources without control. On the other hand, a group of network operators, investigators and even ISPs who need to know the traffic characteristics of their networks to manage the resources or even charge the users depending on their consumption.

2.1 Port-based approach

As mentioned in Chapter 1, the identification of network applications traditionally has been carried out using the well-known ports technique. At the beginning, the initial network applications registered their ports in the Internet Assigned Numbers Authority (IANA) [21]. The solution of the well-known ports identifies the traffic according to the ports registered in the IANA. Table 2.1 shows an example of several ports assigned by the IANA with the corresponding application. For example, web applications use port 80 and email applications use port 25 (SMTP) to send emails and port 110 (POP3) to receive them.

This method is no longer valid because of the inaccuracy and incompleteness of its classification results [24, 25, 31]. It is difficult to set bounds on the current accuracy of this method because it meanly depends on the characteristics of the network being monitored and the system to establish the base-truth used to compare. There are some studies where this technique achieves an accuracy of 50%-70% [31, 32], while others, like the present work, where the accuracy is less than 20%. However, the complete literature agrees that the well-known ports technique is not able to classify the new generation of applications that are the

Assigned Port	Application
20	FTP Data
21	FTP Control
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
110	POP3
123	NTP
161	SNMP
3724	WoW

Table 2.1: IANA assigned port numbers for several well-known applications

ones that consume more bandwidth. This new type of applications, especially P2P applications, use different strategies to camouflage their traffic in order to evade detection. For example, they use dynamic ports in their connections or ports from other well-known applications (e.g., 80 typically used by web applications). These obfuscation techniques have motivated the study of new methods to identify this type of traffic.

2.2 Payload-based approach

The first alternatives to the well-known ports method used the inspection of the packet payloads to identify the network traffic [2, 24–26, 31, 37]. These methods, usually called deep packet inspection techniques, examine the content of the packets looking for characteristic signatures. The work done under this approach aims to identify specially P2P applications because they are the most assiduous to use camouflage strategies and evade the detection of the well-known ports technique. Table 2.2 present an example of patterns used by Karagiannis et al. in [26].

There are some papers [24–26] that presented a hybrid method using the well-known ports method to identify the traditional network applications and a deep packet inspection to classify the new ones. Although this solution could achieve high identification accuracy (>95%), it had some problems with two main factors. First, pattern searching in the payload of every packet produces a high consume of resources. Furthermore, it does not work with encrypted traffic, a new tendency among the P2P applications. Second factor is the privacy issues, Allman and Paxon presented in [1] several problems regarding the legality of the inspection of the network traffic. Therefore, its high resources requirements for the pattern

P2P Protocol	String	Trans. Prot.
eDonkey 2000	0xe319010000	TCP/UDP
	0xe53f010000	
Fasttrack	"Get /.hash"	TCP
	0x2700000002980	UDP
BitTorrent	"0x13Bit"	TCP
Gnutella	"GNUT" "GIV"	TCP
	"GND"	UDP
Ares	"GET hash:"	TCP
	"Get sha1:"	

Table 2.2: Strings at the beginning of the payload of P2P protocols defined by Karagiannis

searching, the limitations with encrypted traffic and the privacy issues make it unpractical in nowadays high-speed networks. However, this family of techniques is still the basic solution to establish the base-truth.

2.3 Host-behavior-based approach

In order to overcome the limitations of the payload methods and find an alternative to identify applications without the necessity of packet inspection Karagiannis et al. with BLINC [27, 28] and Xu et al. in [46] developed a classification approach based on the behavior of the hosts. The Karagiannis approach study the behavior of the hosts at three levels:

- At social level, they value the popularity of a host depending on the number of communications it has with different hosts.
- At functional level, they value the role of the host in the network finding out if the host is a provider or consumer of a service, or it participates in collaborative communications. For example, a host which has a lot of connections at the same port is likely to be provider of a service in that port.
- At application level, they capture the transport layer interactions between hosts trying to identify the application of origin. For example, a host with a lot of connections from the same source port and IP to a unique destination IP but different ports would be the behavior of a port scan attack.

According with these three metrics and some refining heuristics BLINC classifies the behavior of the applications. BLINC classifies approximately 80%-90% of

the total number of flows with 95% accuracy. However, BLINC presents some limitations regarding the necessity to work with traces collected in the edge and with bidirectional flows. Furthermore, BLINC classifies the applications based on behavior groups but it is not able to identify exactly the applications. It could suppose a problem with applications that are theoretically from different groups but with similar behavior (e.g., VOIP and P2P).

2.4 Flow features-based approach

The other branch that appears to solve the limitations of the payload-based methods is based on machine learning (ML) techniques. These methods detect, in an offline phase, characteristic patterns of the different applications based on a set of features. More in detail, machine learning methods use a labeled dataset from which is collected a set of features. This information serves as input of the ML technique that extracts and outputs the knowledge in different structures (e.g., decision tree, rules, clusters) depending on the ML technique used. The structure obtained is later used to classify unlabeled instances assuming that the features of the still unknown instances will have the same behavior as the known one.

The wide related work in this field could be structured in two main areas: the supervised and unsupervised learning approaches. Next subsections provide brief descriptions of these two areas. However, Nguyen et al. surveyed the complete literature of the ML techniques in the field of application identification in [34] comparing in detail all the different approaches.

2.4.1 Supervised Learning Approach

Supervised methods, also known as classification methods, extract knowledge structures to classify new instance in pre-defined classes. It is important to note that is called supervised because the output classes are pre-defined. The process of a supervised learning methods start with a training dataset TS defined as, $TS = < x_1, y_1 >, < x_2, y_1 >, ..., < x_N, y_M >$, where x_i is the vector of values of the features corresponding to the i^{th} instance, and y_i is its output class value. It discovers the different relations between the instances and output a structure, usually a decision tree or classification rules, that will classify the instances in a discrete set $y_1, y_2, ..., y_M$.

There is a lot of related work that use supervised techniques [3, 13, 20, 23, 32, 36, 41, 50] with a promising results. Initially, Roughan et al. in [36] proposed a method based on the algorithms nearest neighbours (NN), linear discriminate analysis (LDA) and Quadratic Discriminant Analysis (QDA) to classify applications in three groups: Bulk data (FTP-data), Interactive (Telnet) and Streaming

(RealMedia). Using these groups they achieved errors lower than 3.4%. They also experimented with seven groups of applications getting an error rate between 9,4% and 12.6%.

In 2007, Auld et al. proposed a method based on Bayesian neural network in [3] that achieved an accuracy of 99% with data trained and tested on the same day, and 95% accuracy with data of eight month later.

2.4.2 Unsupervised Learning Approach

Unlike the supervised learning methods, unsupervised methods, also known as clustering methods, do not need a complete labeled dataset. Therefore, the output of the ML training does not classify in a predefined classes. It is the own method who discovers the natural clusters (groups) in the data. Similar to the supervised methods, the clustering methods have been deeply studied [6–8, 14, 15, 17–19, 38, 48, 49]

Basically, there are three main clustering methods: the classic k-means algorithm that forms clusters in numeric domains, partitioning instances into disjoint clusters; the incremental clustering who generates a hierarchical grouping of instances and; the probability-based method who assigns instances to classes probabilistically, not deterministically.

In 2004, Soule et al. presented in [38] a solution based on the unsupervised method Expectation Maximization (EM). However, this work classifies the traffic in classes (i.e., Elephants, Buffallos, Dragonflies and Mices) instead of applications.

Zander et al. proposed in [48, 49] a method based on AutoClass, which is an unsupervised Bayesian classifier, using the EM algorithm to determine the best clusters set from the training data. Zander achieved with this method an accuracy $\geq$80%.

In 2006, Bernaille et al. presented in [6] a technique based on the clustering algorithm Simple K-Means. The main contribution of the Bernaille work was that they developed first in [6] and later in [7, 8] a method who is able to identify the applications only using the first few packet of the traffic flow instead of the complete flow. This technique, also known as "Early Identification", has a tradeoff between the number of packets used to identify the application and the accuracy. For example, it achieved more than 80% accuracy with the first five packets. However, Bernaille et al. were not taking into account the impact of the lost of any of the first packets in the accuracy.

Finally, since 2006, Erman et al. presented in several papers [15, 17–19] an extended work using clustering methods. Their work is set in a core network scenario with only uni-directional flows available. Furthermore of the identification method, they also developed and evaluated an algorithm that could estimate missing statistics from a uni-directional packet trace. Erman et al. using the well-

known K-Means algorithm and tuning with different numbers of clusters achieved a final accuracy of 95% in terms of flows and 79% in terms of bytes.

It is important to note that usually the clustering methods have a tradeoff between the number of clusters of the output and the final accuracy. Having more clusters you could classify better but the training and classification time of the method will also increase.

2.4.3 Comparing Learning Approaches

So far, we present a spread work in the field of machine learning techniques to identify applications. However, it is not clear which one is the best method to identify applications in network traffic. Some investigators address this problem in several papers comparing different machine learning techniques [16,29,40,43–45].

In 2006, Williams et al. made in [45] a perform evaluation between seven supervised algorithms. The used algorithms were C4.5 Decision Tree, Naive Bayes, Nearest Neighbor, Naive Bayes Tree, Multilayer Perception Network, Sequential Minimal Optimization and Bayesian Networks. They concluded that C4.5 is the most accurate supervised technique with and accuracy of 99,4%, followed by Bayes Network 99,32% and Naive Bayes Tree and Naive Bayes with 98,3%. Although the accuracy difference is minimum, Williams et al. show significantly differences regarding the classification time, where C4.5 outperformed the rest of methods being twice faster than Naive Bayes and more than five times faster than Bayes Network.

Also in 2006, Erman et al. presented the study of the comparison of the supervised method Naive Bayes and the unsupervised method AutoClass. They concluded that the unsupervised method achieved a 91% accuracy, 9% more than the supervised method.

At the end of 2008, Kim et al. presented in [29] a comparison of different supervised methods, BLINC and the well-known port method implemented by CoralReef [12]. Comparing the different approach the supervised machine learning techniques achieved better accuracy than BLINC and CoralReef. Among the different supervised methods (i.e.,Naive Bayes, Naive Bayes Kernel Estimation, Bayesian Network, C4.5, k-NN, Neural Networks and Support Vector Machine (SVM)) the paper concluded that SVM outperformed the rest of supervised methods with more than 98% overall accuracy while C4.5, in the same scenario, achieved an overall accuracy of ≈94%. However, C4.5 classified the instances a hundred times faster than SVM.

We can conclude that according to the literature there is not a clear ML technique that outperforms the rest of techniques. However, it seems that all the methods could achieve a very good accuracy.

Chapter 3

Methodology

This section describes the methodology used to study the application identification problem. First, in Section 3.1 we explain the metrics used to evaluate the performance of our method. Next, we present a brief overview of the scenarios where our solution could be implemented. Finally we describe what, from our point of view, are the three most important points of a ML traffic classification method: the base-truth system, the ML process and the evaluation datasets.

3.1 Performance Metrics

In order to evaluate our method, we used three representative metrics: *overall accuracy*, *precision* by group of applications and *recall* by group of applications. In order to define these metrics it is necessary to define *True Positives (X)* as the number of correctly classified flows of the application group X, *False Positives (X)* as the incorrectly classified flows of the application group X and *False Negatives (X)* as the X flows classified as a different application group. We also define a flow as the 5-tuple (*source IP address, destination IP address, protocol, source port, destination port*) with a timeout of 64 seconds [11].

- *Overall accuracy*

$$\frac{TruePositives}{TruePositives + FalsePositives}$$

- *Precision* by application group.

$$\frac{TruePositives(X)}{TruePositives(X) + FalsePositives(X)}$$

- *Recall* by application group.

$$\frac{TruePositives(X)}{TruePositives(X) + FalseNegatives(X)}$$

The three metrics are given by flows, furthermore we also present the overall accuracy by packets and bytes.

3.2 Scenarios

In order to fulfil the challenges presented in section 1.1 we address the problem of application identification in network traffic in two differentiate scenarios.

The first scenario and the precursor of this work is SMARTxAC [4], a tailor-made passive network monitoring system. SMARTxAC platform is used by the Supercomputing Center of Catalonia (CESCA) for continuously monitoring the Catalan Research and Education Network (Scientific Ring). One of the main usages of SMARTxAC is the application identification of the network traffic. In order to address this task SMARTxAC is currently using the well-known port method, based on the IANA ports [21]. This method is no longer valid because of its inaccuracy and the huge amount of *unknown* flows it detects. This situation encourages us to study a new application identification method able to work under the requirements of SMARTxAC. Therefore, we need to define a method able to classify in real-time, without access to the content of the packets and with better accuracy than the actual method.

Any network component that produces NetFlow data would be our second scenario. NetFlow is a widely extended network protocol developed by Cisco to export IP flow information from routers and switches [10]. There are similar network protocols for other vendors, but the IPFIX group from IETF is trying to set a universal standard in order to export IP flow information from routers based on NetFlow v9 [22]. The main constraints of this scenario are the capacity to work only with information provided by NetFlow, that is, a fix set of features that depends on the NetFlow version, and also the capacity to work with unidirectional flows, a common situation in core networks.

Another possible requirement in both scenarios is the ability to work with sampled data. NetFlow is usually used in core networks to collect information of the traffic. Network operators use to collect Sampled NetFlow [10] data in order to avoid routers to run out of resources in worst case traffic scenarios and network attacks. Furthermore, SMARTxAC is currently running on a 10Gb link and apply sampling could be the only solution to keep monitoring the huge amount of traffic without problems.

Once presented the scenarios and their requirements, we should decide which is the most appropriate type of application identification method. Firstly, we discard the payload-based methods because in both scenarios we had not access to the content of the packets. Furthermore, we need a very light method in order to avoid monitoring systems run out of resources. Between the host-behavior-based methods and the flow features-based methods we discard the first one. Although host-behavior-based methods look suitable for the scenario of SMARTxAC it does not seems very appropriate to work with NetFlow. This is because in core networks the information to study the behavior of the host is less than in border links. Furthermore, we think host-behavior-based methods need to know both direction of the traffic to discover the behaviors of the hosts. Kim et al. in [29] confirm our suppositions about the host-behavior-based methods. After reviewing all the requirements that our scenarios implies we decide to implement a flow feature-based method.

3.3 Base-truth system

We use L7-filter [30] to set the base-truth of our method, which is a known deep packet inspection technique that tries to find characteristic patterns in the packet payloads to label them with the corresponding application.

The establishment of the base-truth is one of the most critical phases of any ML process, because the entire classification process relies on the accuracy of the first labeling. Although L7-filter is probably not as accurate as the manual inspection methods used in other previous works to set the base-truth, it is automatic and does not require human intervention. This is a very import feature given the large traces used in this paper to perform the evaluation.

In order to reduce the inaccuracy of L7-filter we use 3 main rules:

- We apply the patterns in a priority order depending on the degree of over-matching of each pattern (e.g., *skypeout*, *skypetoskype*, *ntp*, *emule* are in the latest positions of the iptables rules).

- We discard labeled packets that do not agree with the rules commented by the pattern creators (e.g., packets detected as *ntp* with a size different than 48 bytes are not labeled).

- We label a flow with the application that has more priority based on the quality of patterns given by L7-filter. If the quality of the patterns is equal, we choose the label with more occurrences.

3.4 Machine Learning Process

Once decided that we are going to use a Machine Learning technique as justified in the section 3.2 we define the ML process, the ML technique implemented in our method and the set features collected.

3.4.1 Machine Learning Technique

We presented in the Section 2.4.3 several papers that compare different ML techniques. According to the related work there is not a clear method that outperforms the rest in every case. Furthermore, there is not a big difference in terms of accuracy between the different ML techniques. Because of this, we based our decision in other important aspects of an application identification method.

We decided to use the supervised C4.5 decision tree method described by Quinlan in [35]. Williams et al. in [43–45] concluded that C4.5 is the most accurate method among Naive Bayes, Nearest Neighbor, Naive Bayes Tree, Multilayer Perception Network, Sequential Minimal Optimization and Bayesian Networks. Recently, Kim et al. in [29] concluded that SVM is the most accurate method, although C4.5 is only 4% less accurate than SVM. However, both papers concluded that C4.5 is the fastest classification algorithm, being even a hundred of time faster than SVM, and its learning time is shorter than most of the methods. Furthermore, a decision tree is much more comprehensible method than other ML techniques. However, it has a drawback regarding the resources used in the training phase because it needs to load in memory all the instances used to create the decision tree.

3.4.2 Machine Learning Features

Besides the definition of the ML algorithm, another important factor of a ML process is the definition of the features used for the classification. Because we have two different scenarios with different requirements we define a different set of features for each scenario.

Due to SMARTxAC receives every packet of the link we are able to collect a very large set of features. Moore et al. presented in [33] a set of 250 possible futures for traffic classification. Although we are able to collect the complete set we want to keep our method as efficient as possible and collecting this huge amount of features for each flow would produce a high load. The first column of the Table 3.1 presents the set of features collected for the SMARTxAC scenario. The features we collect are well-known in the field of traffic monitoring except the features *so_in* and *so_out*. These features estimate the Operating System based on

the increments of the IPIDs between the packets of the same flow as described by Yarochkin in [47].

In order to lighten the feature collecting and perhaps improve the accuracy of the ML methods it could be applied different techniques of feature selection. Feature selection is the process to find a subset of features that will optimize for higher learning accuracy with lower computational complexity. This process eliminates the irrelevant or redundant features keeping or improving the accuracy. Among the possible feature selection algorithms that are described in detail by Nguyen in [34] we use the Correlation-based Filter (CFS) with Best First search that according to Williams et al. in [43, 44] outperforms the other algorithms in terms of classification accuracy and efficiency. Second column of Table 3.1 presents the subset of features chosen by the mentioned algorithm.

Finally, the third column of Table 3.1 presents the set of features for the NetFlow scenario. As mentioned in Section 3.2 NetFlow provides us a fixed set of features according to the NetFlow version used. In our case we use NetFlow version 5 because as far as we know is the most extended. In this scenario we do not apply any feature selection method because, although having less features would simplify the decision tree, in this case the feature collecting phase, that is the most expensive part, is carry out by the NetFlow hardware. Unlike the TCP Flags features for the SMARTxAC scenario in the NetFlow scenario the value of the flags will be 0 if there is no packets in the complete flow with this flag activate or 1 if at least there is one packet with that flag activated.

3.4.3 Machine Learning structure

The ML process has two different phases, the learning phase and the validation phase. First of all, we need a labeled dataset in order to establish the base-truth of our ML method. The establishment of the base-truth is explained in detail in the Section 3.3. Although we are using in this work the payload-based method L7-filter to label the dataset any other automatic method could be used. Unlike most related work, our ML process does not require human intervention that is why we are using an automatic labeling method.

Figure 3.1 presents an overview of our ML process. The first phase, the training phase, receives as input a labeled dataset. In our case we use a labeled trace. We are not able to use more than one trace because the memory constraints of the C4.5 technique. The first process of the training phase is the *Flow Statistics Processing*. In this module is collected all the features per flow and added the corresponding application label. At this point we have a set of instances. An instance consists on a tuple with a vector of feature values and the corresponding application label.

Before introducing the information in the WEKA machine learning software suite [42], we remove the instances labeled as *unknown* from the training set.

Scenario	Features
complete SMARTxAC	*source and destination port, IP protocol, source and destination # packets, source and destination # bytes, source and destination PUSH TCP flag, source and destination ACK TCP flag, source and destination average packet size, source and destination average TCP window size, source and destination initial TCP windows size, source and destination average PID incrementation, source and destination DF IP flag, source and destination Operating System Estimation, fow time and inter-arrival time*
subset SMARTxAC	*destination port, destination # bytes, source TCP flag push, destination and source average packet size and initial source TCP window*
NetFlow	*source and destination port, protocol, ToS, TCP flags, flow time, # packets, # bytes, average packet size and inter-arrival time*

Table 3.1: Set of features for the different scenarios

Among the several ML techniques implemented in WEKA, we used the decision tree method C4.5. justified in Section 3.4.1.

The decision tree obtained in WEKA is later added to our classification software. In the validation phase we calculate the features as in the training phase. Unlike the training phase, in this phase we use the complete labeled dataset (described in detail in Section 3.5) to evaluate the quality of our method. The evaluation results are presented broken down by application groups. These groups of applications correspond to the groups defined in L7-filter documentation and are presented in Table 3.2.

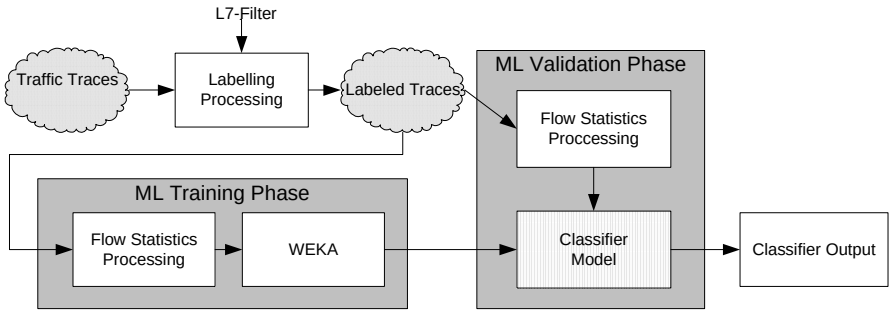

Figure 3.1: Overview of our automatic ML process

Group	Applications
P2P	*Ares, Bittorrent, Directconnect, Edonkey, Fasttrack, Gnutella, Napster and Audiogalaxy*
HTTP	*HTTP*
VoIP	*H323, Skype, Sip, TeamSpeak and Ventrilo*
Network	*BGP, Ciscovpn, DHCP, Netbios, NTP, RDP, SNMP, Socks, SSH, Telnet, VNC and X11*
Streaming	*Live365, ReplayTv, RTP, RTSP, Shoutcast, PPLive, Itunes and QuickTime*
DNS	*DNS*
Others	*CVS, Hddtemp, IPP, LPD, Subversion and TSP*
Chat	*Aim, IRC, Jabber, MSN Messenger and Yahoo Messenger*
Email	*IMAP, POP3 and SMTP*
FTP	*FTP, Gopher, Tftp and UucP*
Games	*Battlefield, Counter-Strike Source, Day of Defeat Source, Doom 3, Half-life 2, MOHAA, Quake and WoW*

Table 3.2: Definition of application groups as L7-filter documentation

3.5 Evaluation Dataset

Our evaluation dataset consists of six full-payload traces collected at the Gigabit access link of the Technical University of Catalonia (UPC), which connects 10 campuses, 25 faculties and 40 departments to the Internet through the Spanish Research and Education network (RedIRIS). The traces were collected in different days and hours trying to make our dataset as representative as possible. The traces are 15 minutes long with approximately 1,8 millions of bidirectional sanitized flows for each trace. Table 3.3 present the details of the collected traces.

Name	Size	Date	Time	Flows
UPC-I	53 Gb	11-12-08	10:00 (15 min.)	1.777.678
UPC-II	63 Gb	11-12-08	12:00 (15 min.)	2.229.850
UPC-III	39 Gb	11-12-08	01:00 (15 min.)	1.278.129
UPC-IV	55 Gb	12-12-08	16:00 (15 min.)	2.179.004
UPC-V	48 Gb	12-12-08	18:30 (15 min.)	1.886.961
UPC-VI	29 Gb	14-12-08	00:00 (15 min.)	1.253.541

Table 3.3: Characteristics of analyzed traces

Figure 3.2 shows the traffic mix of each trace according to the L7-filter labelling. Approximately, the 45% of the flows of the traces is labeled as *unknown*. Although it could seem too much *unknown* flows it is a common situation in the literature.

In order to eliminate these *unknown* flows the related work usually use "human" inspection. Furthermore that one of our contributions is the not necessity of "human" intervention, in our case, manually labelling of these flows is nearly impossible due to the large traces we use. The solution we adopt is to execute the training phase without the *unknown* flows. Similar to the clustering methods, we assume that *unknown* flows will have similar behavior as known ones. Among

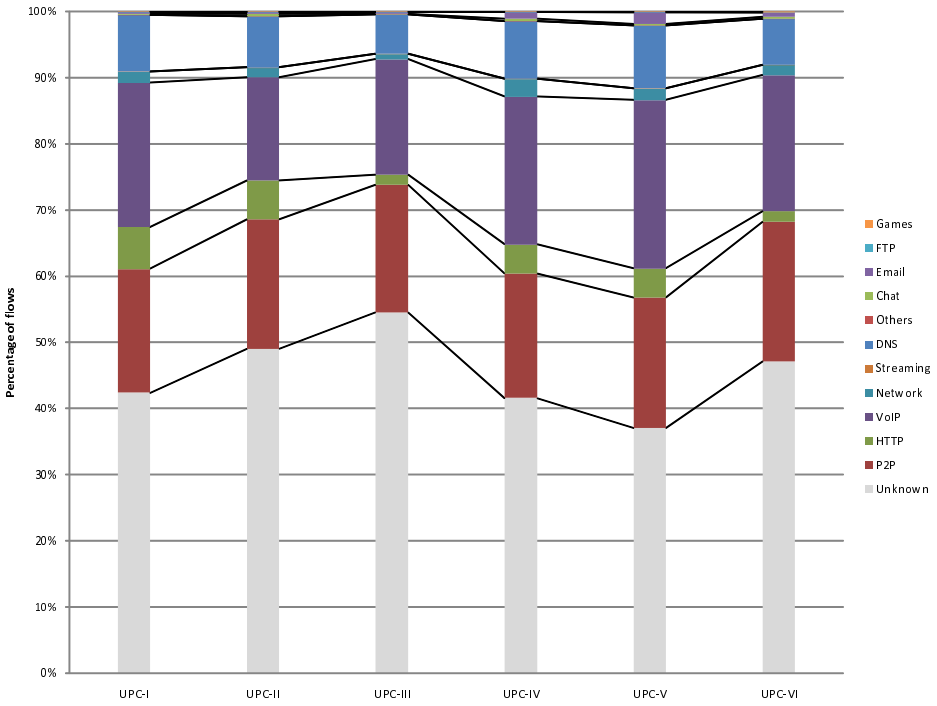

Figure 3.2: Application mix by trace

the six possible traces, we selected one for the training phase (UPC-II). As future work, we plan to use a mix of several instances of various traces in order to build a more complete training set.

Chapter 4

Performance Evaluation

This chapter presents the results of the performance evaluation of our method in the first scenario. As mentioned in Section 3.4.2 for the SMARTxAC scenario we have a set of 25 features and a subset of 6 features selected with the CFS algorithm with Best First search.

4.1 Baseline experiments

In order to compare our method with the related work we present in this section the results of our application identification method with the complete set and the subset of features. We show in Table 4.1 the overall accuracy for each trace, each metric (i.e., flows, packets and bytes) and both feature sets. The first row of results

Features	**Metric**	**Traces**						**Overall**
		UPC-I	**UPC-II**	**UPC-III**	**UPC-IV**	**UPC-V**	**UPC-VI**	**Accuracy**
Complete	Flow	87,03%	91,01%	83,33%	86,71%	86,02%	83,84%	86,33%
	Packet	67,95%	70,02%	57,50%	62,76%	66,92%	59,08%	64,04%
	Bytes	61,00%	60,71%	43,13%	60,02%	61,50%	47,05%	55,57%
SubSet	Flow	92,82%	97,20%	89,90%	96,12%	95,58%	93,24%	94,14%
	Packet	69,64%	87,43%	66,01%	68,27%	73,80%	62,04%	71,20%
	Bytes	63,74%	85,31%	60,38%	67,09%	67,96%	48,41%	65,48%

Table 4.1: Overall accuracy for each trace, metric and set of features in the SMARTxAC scenario

in the Table 4.1 presents the overall accuracy by flow with the complete set of features. We achieve an accuracy of 86,33%. Although we improve the accuracy of the current application identification method on SMARTxAC, we achieve a slightly lower accuracy than the related work. It could seem that our method could need more features in order to improve the accuracy. However, as you can see in the

fourth row of results in the Table 4.1 we significantly improve the accuracy with the subset of features. With the subset of only six features we achieve an overall accuracy by flow of 94,14% that is similar to the related work. Therefore, there are some attributes of the complete set that are irrelevant or redundant and produce errors in the classification.

Regarding the evaluation by packets and bytes the results are lower than by flow. We achieve better accuracy with the subset of features, having 71,20% accuracy by packets and 65,48% accuracy by bytes. These situations are very common with Machine Learning techniques. The reason of this situation is because we are using real traces for the training phase and as commented in several papers (e.g. [38]) the network traffic use to have few flows that carry the most of the traffic (*elephants*) and a lot of flows that carry the less of the traffic (*mices*). Therefore, in the input of our ML process we have a lot of instances of *mice* flows and very few of *elephants* flows. It means that our ML method has less information about the *elephant* flows producing more errors in the classification model of this type of flows. In order to solve this problem we are planning to use long traces and then select more *elephant* flows and discard *mices*. We are not able to use directly longer traces in the training phase because we have a memory constraint.

The results of the Table 4.1 also present the overall accuracy for each trace. As you can see, the overall accuracy in the traces collected at night achieves less accuracy than the rest. This is because the network traffic mix at day is different than at night. In order to improve the accuracy in this case a possible solution is adding flows from traces collected at different times in the training phase.

4.2 Impact of Sampling

Another important results of our study is the effects of sampling to our method. Figure 4.1 depicts the overall accuracy of our method with the two set of features and also comparing with the well-known port method. Applying a sampling rate (p) of *0,1* our method decrease to an overall accuracy of ≈50%. From this sampling rate to the sampling rate where we take one packet out of 10.000 the overall accuracy is more or less similar. However, the results applying these sampling rates are not very representatives because the number of flows significantly decreases.

Comparing our method with the well-known ports method, C4.5 outperforms the well-known port method with all the sampling rates applied. It is important to note that, unlike other related work, the method of well-known ports only classifies correctly less than 20% of the flows in our dataset. The overall accuracy of the well-known ports method increases slightly with lower sampling rates. However, we think this improvement in the overall accuracy is related to the already mentioned decrease in the total number of flows. Figure 4.2 depicts the percentage of precision

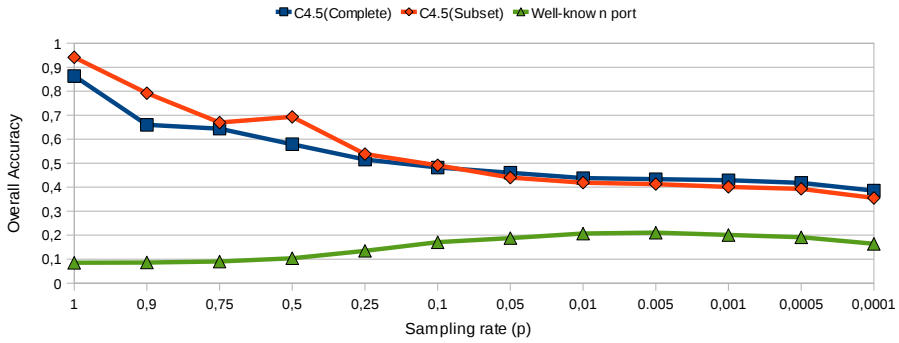

Figure 4.1: Overall accuracy by sampling rate (p) by the complete set and the subset of features in the SMARTxAC scenario

by application group at different sampling rates and also compares our method with the well-known ports technique. With the complete data, we achieve a overall precision of 91,41%. However, we detected low precision for Streaming (67,37%) and Chat (80,27%) application groups. This low precision does not significantly affect the overall accuracy of our results because the base-truth system detects very few flows of these applications. We believe that the lower detection accuracy for these applications is due the lack of flows of these applications in the training phase. Figure 4.3 presents similar results for the recall metric by application group.

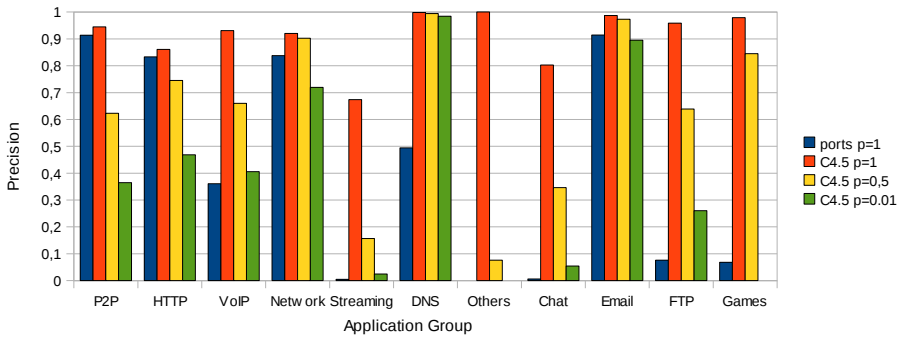

Figure 4.2: Precision by application group in SMARTxAC scenario with the subset of features.

Comparing these results with the well-known ports method, we can observe that, although both methods show similar results with *email* and *http* groups, for the rest of application groups our method correctly identifies much more flows than the well-known ports method also in the presence of sampling.

It is important to note that, although the precision of the well-known port with the P2P group is 91,35% its recall is only 1,41%. Therefore, the P2P flows identified by the well-known port technique are usually correctly identified but this method misclassifies a lot of P2P flows. This situation happens with all the groups except

the *email* and the *http* groups. Given the current scenario, a possible solution to

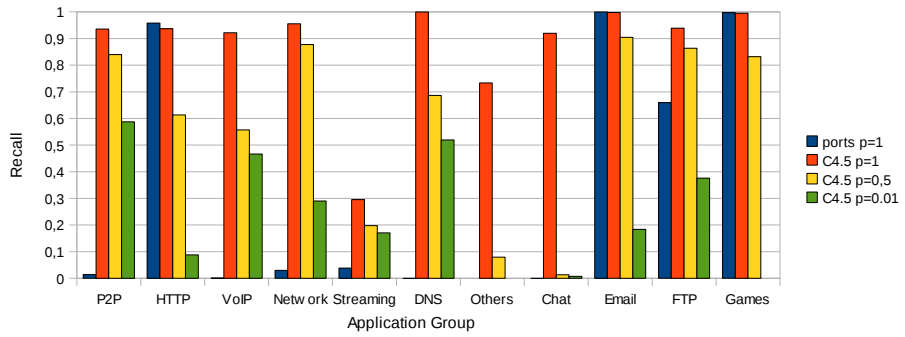

Figure 4.3: Recall by application group in SMARTxAC scenario with the subset of features.

improve the accuracy of those application groups that contain few flows, consists of generating artificial flows of these applications and add them in the training set. Another possible solution is to use larger traces for the training phase and then use the same number of instances per application.

Chapter 5

Performance Evaluation with NetFlow data

This chapter presents the results of the performance evaluation of our method in the second scenario. As mentioned in Section 3.4.2 for the NetFlow scenario we have a fix set of features determinate by NetFlow v5.

5.1 Baseline experiments

In order to set up a comparison base, we evaluate the classification tree over the traces without sampling. Table 5.1 presents the different overall accuracies of the experiments for each trace. The average overall accuracy for the complete dataset is about 90%. The maximum overall accuracy is 93,67%, which corresponds to the trace used in the training phase. The minimum overall accuracies are 86,05% and 88,56%. 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.

Comparing our results with the related work we are obtaining only slightly lower accuracy than the previous packet-based machine learning techniques, despite the fact that we are only using the features that NetFlow v5 provides. Furthermore, the addition of flows from the traces collected at night in the training phase will help to further improve the overall accuracy of our method.

Metric	**Traces**						**Overall**
	UPC-I	**UPC-II**	**UPC-III**	**UPC-IV**	**UPC-V**	**UPC-VI**	**Accuracy**
Flows	89,17%	93,67%	86,05%	90,77%	91,12%	88,56%	89,90%
Packets	66,37%	82,03%	65,87%	67,78%	72,58%	60,60%	69,21%
Bytes	56,53%	77,97%	57,24%	61,80%	63,69%	45,18%	60,40%

Table 5.1: Overall accuracy for each trace and metric for the NetFlow scenario

5.2 Impact of Sampling

So far, we have showed that the C4.5 classification tree can achieve high accuracy with full NetFlow data (without sampling). Next, we present several results applying different sampling rates in NetFlow. Figure 5.1 shows how our method responds to the different sampling rates. The accuracy decreases significantly, as expected, with the sampling rate, arriving at 36,38% when we randomly select one packet out of 10.000. However, the number of flows decreases substantially with the lowest sampling rates resulting in less representative results. As mentioned in Section 4.2 we plan to use longer traces in the future in order to have a meaningful number of flows also for the lowest sampling rates. Figure 5.2 depicts the

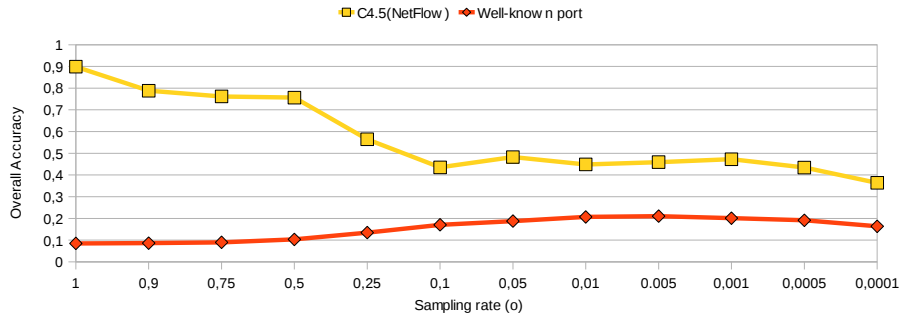

Figure 5.1: Overall accuracy by sampling rate (p) in NetFlow scenario

percentage of precision by application group at different sampling rates and also compares our method with the well-known ports technique. With the complete data, we achieve a precision around 90% for most application groups. However, we detected low precision for Streaming (50,67%) and Chat (67,40%) application groups. This low precision does not significantly affect the overall accuracy of our results because as commented in Section 4.2 the base-truth system detects very few flows of these applications. We believe that the lower detection accuracy for these applications is due the lack of flows of these applications in the training phase. Figure 5.3 presents similar results for the recall metric by application group as in the Section 4.2 for the SMARTxAC scenario. Applying different sampling rates the precision decreases proportionally for the most application groups. Similarly to the previous case, the group of applications with less flows (i.e., *Streaming, Others, Chat, FTP and Games* is losing its recall faster than the popular ones except for the FTP group. That probably is because FTP flows are large in terms of time and usually use the IANA assigned port. As mentioned in Section 4.2 a possible solution to improve the accuracy of those application groups, consists of generating artificial flows of these applications and add them in the training set. Another possible solution is to use larger traces for the training phase and then use the same number of instances per application.

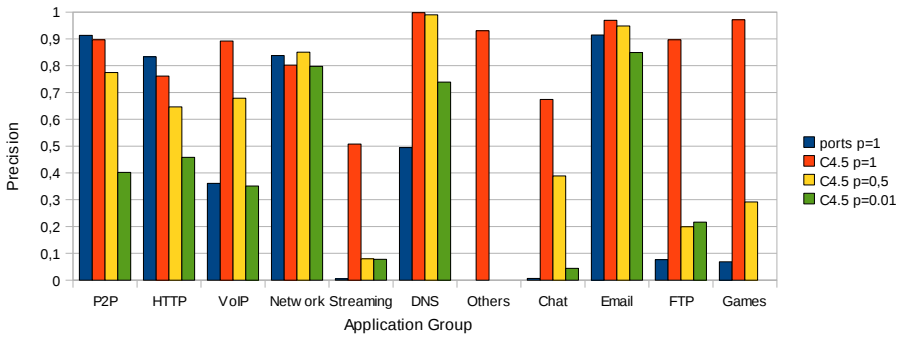

Figure 5.2: Precision by application group in NetFlow scenario.

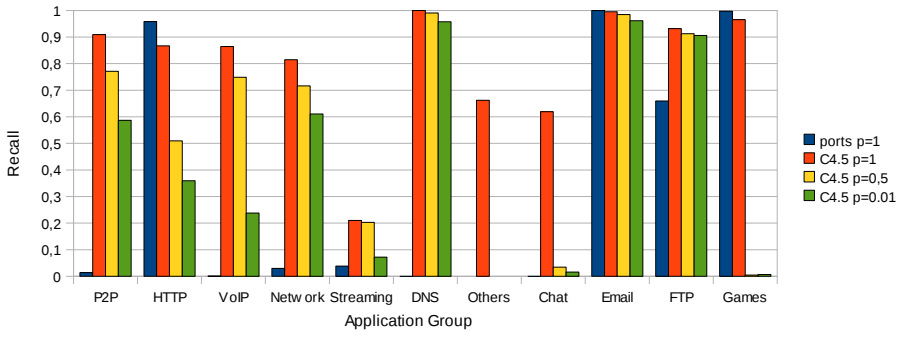

Figure 5.3: Recall by application group in NetFlow scenario.

Chapter 6

Discussion

In this chapter we discuss different aspects of our method and the application identification problem in general. Section 6.1 comments the advantage and drawbacks of our base-truth system. The quality of our method regarding the two presented scenarios is discussed in Section 6.2. Section 6.3 presents the results of an important output of the application identification, the traffic mix. We also present in Section 6.4 the impact of sampling in our method and a possible solution to improve the accuracy in sampled scenarios is discussed in Section 6.5

6.1 The base-truth system

As commented in the Section 3.3 we used L7-filter to establish our base-truth dataset. The main reason of this decision was L7-filter is, as far as we know, the best automatic open-source method. However, we find out that L7-filter is not as accurate as we thought. We have detected an overmatching of VoIP and Network applications.

Although L7-filter does not seem the best system to establish the base-truth, L7-filter is available for everybody and this would solve one of the problems in the literature. We are referring the problem of comparing different papers, because each paper use its own labeling method, usually with an own "human" inspection technique. Because of this, comparing results of different papers is very subjective. Using L7-filter and the same policies to label the instances would solve this problem.

There is another problem of comparing different papers. It is the different datasets used in each paper. However, as commented in [1], there are a lot of constraints to publish network traffic traces. In order to solve this problem we are planning to publish our traces without the content and the corresponding labels by L7-filter.

6.2 Our method in SMARTxAC and NetFlow scenarios

Reviewing the results presented in the Chapters 4 and 5 we can affirm that our method is a very good solution to identify application in network traffic. Regarding the SMARTxAC scenario we achieve a very good accuracy, similar to the related work, with an efficient C4.5 decision tree of only 6 features.

We also present in this technical report a method that is able to identify applications in network traffic using only the information provided by NetFlow version 5 and achieving a similar accuracy than the related work.

Regarding the requirements of our scenario we believe that C4.5 is the best method because in our case, it is more valuable having a method ten times faster than the majority, achieving only slightly low accuracy than the best method. However, we think that we have to work in our method in order to improve the accuracy by bytes and packets.

Among the three possible methods that we implemented, the best one is the C4.5 with the subset of features. As you can see in the Figures 6.1 and 6.2 the technique that use the 6 features slightly outperforms the rest of methods almost in all the cases.

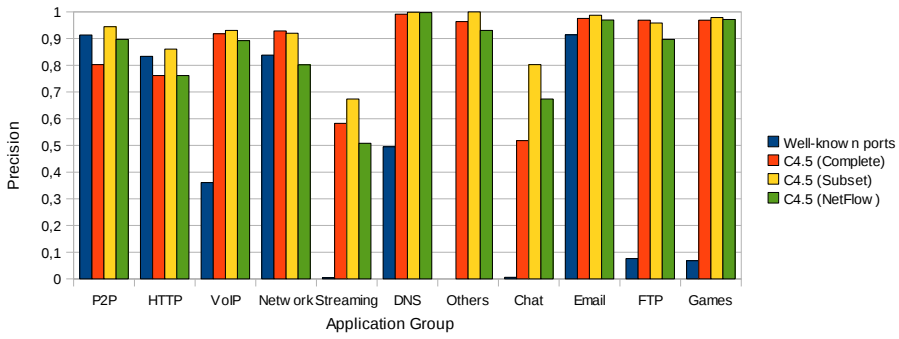

Figure 6.1: Precision by application group using different classification methods.

6.3 Traffic Mix

An important objective of the traffic classification is showing operators the mix of traffic of their networks. Figure 6.3 shows the traffic mix based on the five methods: the deep packet inspection method L7-filter that is our base-truth, the well-known ports method and the three version of our method. The number of Unknowns flows with L7-filter is nearly 42%, on the other hand the well-known ports method only classifies the 16% of the traffic. As commented in the Section 3.5 we eliminate

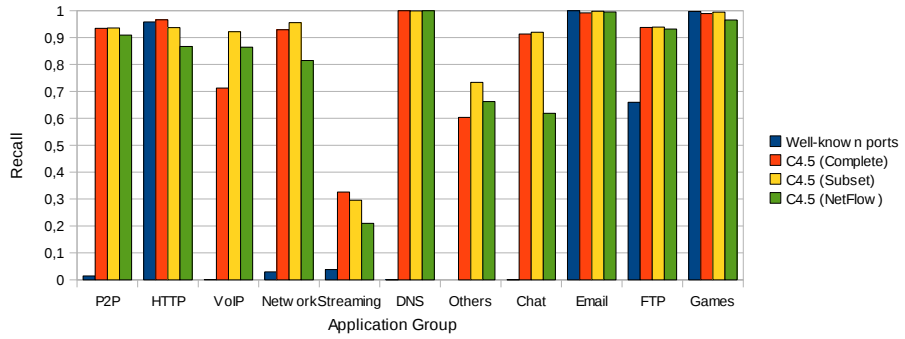

Figure 6.2: Recall by application group using different classification methods.

the *unknowns* flows in the training phase assuming that the *unknowns* flows for L7-filter have a similar behaviour as the rest of flows. It is noteworthy the huge amount of VoIP traffic detected with our method (≈40%). We know this is not the actual percentage of VoIP traffic but it is directly related to our system to establish the base-truth. The L7-filter pattern of Skype, the most extended VoIP application, is very inaccurate and produces a lot of false positives. However we are pretty sure that better patterns would solve this problem.

6.4 Impact of sampling

The results presented in the Sections 4.2 and 5.2 show that applying sampling rate on the network traffic have a significantly impact on the identification of applications. Unlike the results presented by Jian et al. in [23], only applying a sampling rate of *0,1* our method decreases significantly its accuracy.

Figure 6.4 shows that all the variations of our method respond similarly to the different sampling rates. However, we have detected that the C4.5 decision tree with the NetFlow features is a bit more resilient to the sampling than the rest of variations.

6.5 Dealing with sampling

We arrive at this point with a machine learning method capable of classify traffic in real-time with an overall accuracy greater than 94% with only six features. This result is quite encouraging but when we apply packet sampling, a common action in the high-speed networks, the overall accuracy and the precision and recall by application groups significantly decrease.

In order to solve this problem we are studying new methods to improve the accuracy under sampling conditions. One of our first attempts in this study is using

sampled data in the training phase. Unlike the literature, that uses complete flows in the training phase, we propose a new method that consists of applying sampling to the labeled traces that are going to be used as a dataset for the training set. Regarding the sampling rate we propose to use the same sampling rate that is applied in the scenario that we are going to monitor.

Next, we present the results of a preliminary experiment set in a fictitious scenario where is applied a sampling rate of 1/10. We repeat the training phase and the validation phase with the same traces described in Section 3.5 but using random sampled instances for the training phase. It is important to note that applying sampling in the traces produces much less instances having less information for the training phase. In this experiment we did the training phase with four times less instances than the normal case.

The results of this test are very encouraging because the overall accuracy of the method with a sampling rate of 1/10 increase from an overall accuracy of 54,65% to 80,34%. Furthermore, in the Figure 6.5 and 6.6 is showed how the precision and recall by group of applications also significantly increase. However, there are some points that have to be improved like the mysterious decreasing of the FTP recall that decrease from 94,32% to 4,54%.

Given that the sampled instances are created randomly depending of a sampling rate we can create as much instance as we want. We believe this also will improve the accuracy of the application identification method. Regarding the low accuracy of the groups *Streaming*, *Others* and *Chat* we believe that a new update of the L7-filter patterns and a different structures of the groups could help to improve accuracy in the classification

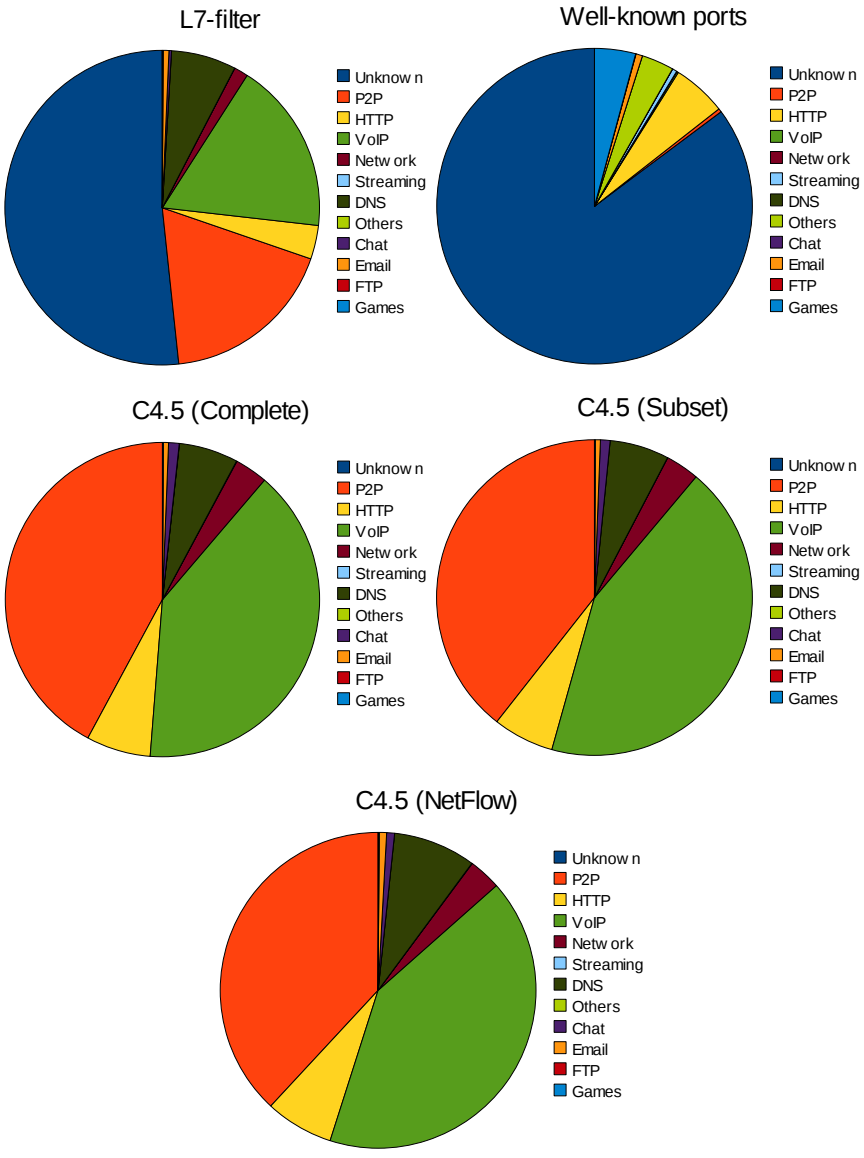

Figure 6.3: Traffic mix using different classification methods

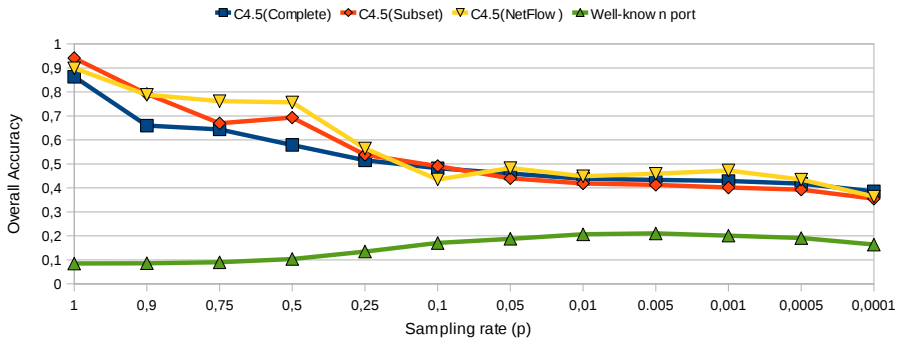

Figure 6.4: Overall accuracy by sampling rate (p) using different classification methods

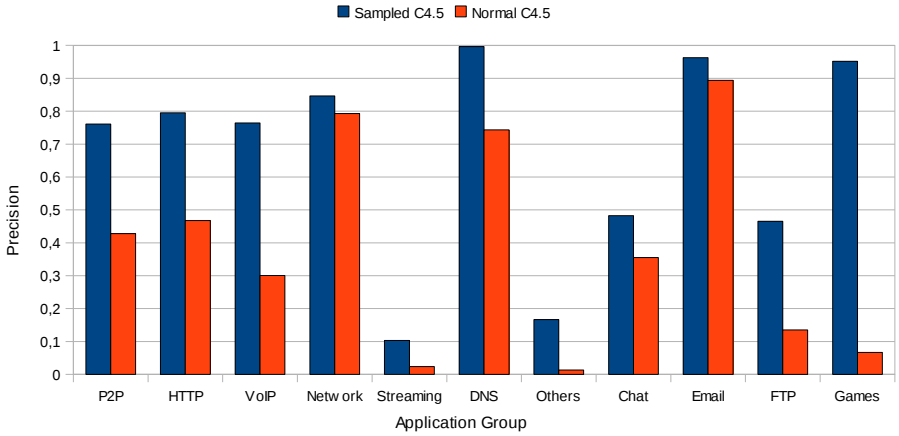

Figure 6.5: Precision by application group with normal and sampled training

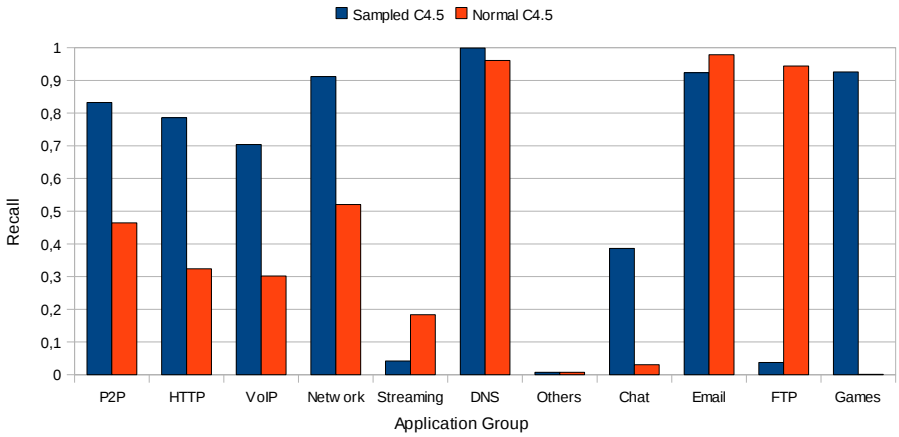

Figure 6.6: Recall by application group with normal and sampled training

Chapter 7

Conclusions and Future Work

This final chapter summarizes the work done in this technical report. An overview of the conclusions resulting from this technical report and its contributions are presented in Section 7.1. Finally, related open research problems and our future work are discussed in Section 7.2.

7.1 Conclusions

This technical report proposed and evaluated a supervised machine learning method for application identification in network traffic using only flow statistics. Our method, based on the C4.5 decision tree algorithm [35], achieves an overall accuracy per flow of 94,14% using only a subset of six traffic features. Our method, according to the previous work [29, 43–45], classifies faster than the rest of the related work, while keeping very high accuracy. This makes our method very suitable for our first goal, which was the application identification using the SMARTxAC system.

In this technical report, we also tested the feasibility of identifying network applications with Sampled NetFlow data, an extended scenario but still scarcely investigated. In particular, our results show only a slightly lower overall accuracy compared to the related work, despite the fact that we are only using the features that NetFlow v5 can provide.

We also presented an automatic machine learning process that does not require human inspection. This is an initial step that will help to solve the problems regarding the comparisons between different works because each system uses its own method to establish the base-truth.

Finally, we presented several interesting results regarding the effects of traffic sampling on the accuracy of our classification method, which is a common practice among network operators, but barely investigated in the field of application

identification. Our results show that sampling significantly affects the application identification, decreasing the overall accuracy from 94,14% to 40,11% when applying a sampling rate of 1/1000, which is commonly used in core networks. In order to solve this limitation, we propose a variant of our method that consists of using sampling instances in the training phase. Preliminary experiments with this technique show very encouraging results, improving the overall accuracy from 54,65% to 80,34%, when applying a sampling rate of 1/10.

7.2 Future Work

The research in the field of traffic classification is still very open. Next paragraphs present possible continuations of this work.

First, we plan to study better labeling methods to establish the base-truth. As mentioned in Section 6.1, we detected that our base-truth system based on L7-filter presents several problems with VoIP patterns. We believe that a more accurate automatic base-truth system would help to significantly improve the accuracy of our classification method. In particular, we are currently studying the base-truth system used by Kim et al. in [29]

Second, we are currently studying the behaviour of other machine learning techniques (e.g., Support Vector Machines) in the scenarios described in Section 3.2 as well as novel methods to improve their accuracy when traffic sampling is applied.

Third, in order to solve the problems described in Section 3.5, we are planning to perform more experiments using longer traces and using instances from different traces in our training phase.

Fourth, we also plan to study new features more resilient to sampling that can improve the accuracy of our current application identification method. A study of the quality of the 250 features presented by Moore et al in [33] would help the community of application identification.

Finally, Bernaille et al. presented in [6–8] an "early" classification method based on clustering. We are also interested in developing a supervised C4.5 decision tree method that can classify the flows before they finish.

Bibliography

[1] M. Allman and V. Paxson. Issues and Etiquette Concerning Use of Shared Measurement Data. In *Proc. IMC*, 2007.

[2] D. Antoniades, M. Polychronakis, S. Antonatos, E. Markatos, S. Ubik, and A. Øslebø. Appmon: An Application for Accurate per Application Network Traffic Characterisation. *submitted for Broadband Europe, December*, 2006.

[3] T. Auld, A. Moore, and S. Gull. Bayesian Neural Networks for Internet Traffic Classification. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 18(1):223, 2007.

[4] P. Barlet, J. Pareta, J. Barrantes, E. Codina, and J. Domingo. SMARTxAC: A Passive Monitoring and Analysis System for High-Speed Networks. In *Proc. Terena Networking Conference. May*, 2006.

[5] P. Barlet-Ros, J. Sole-Pareta, J. Barrantes, E. Codina, and J. Domingo-Pascual. SMARTxAC: a passive monitoring and analysis system for high-speed networks. *Campus-Wide Information Systems*, 23(4):283–296, 2006.

[6] L. Bernaille, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006.

[7] L. Bernaille and R. Teixeira. Early Recognition of Encrypted Applications. *LECTURE NOTES IN COMPUTER SCIENCE*, 4427:165, 2007.

[8] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference*. ACM New York, NY, USA, 2006.

[9] Cisco IOS. NetFlow white papers. http://www.cisco.com/en/US/products/ps6601/prod_white_papers_list.html.

[10] Cisco System: Sampled NetFlow. http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html.

[11] K. Claffy, H. Braun, G. Polyzos, S. Center, G. Atomics, and C. San Diego. A parameterizable methodology for Internet traffic flow profiling. *Selected Areas in Communications, IEEE Journal on*, 13(8):1481–1494, 1995.

[12] CoralReef. http://www.caida.org/tools/measurement/coralree/.

[13] M. Crotti and F. Gringoli. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37(1):5–16, 2007.

[14] A. Dainotti, W. de Donato, A. Pescape, and P. Rossi. Classification of Network Traffic via Packet-Level Hidden Markov Models. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5, 2008.

[15] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286. ACM New York, NY, USA, 2006.

[16] J. Erman, A. Mahanti, and M. Arlitt. Internet Traffic Identification using Machine Learning. In *Proc. IEEE Globecom*, pages 1–6, 2006.

[17] J. Erman, A. Mahanti, and M. Arlitt. Byte me: a case for byte accuracy in traffic classification. In *Proceedings of the 3rd annual ACM workshop on Mining network data*, pages 35–38. ACM Press New York, NY, USA, 2007.

[18] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation*, 64(9-12):1194–1213, 2007.

[19] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *Proceedings of the 16th international conference on World Wide Web*, pages 883–892. ACM Press New York, NY, USA, 2007.

[20] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: automated construction of application signatures. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 197–202. ACM New York, NY, USA, 2005.

[21] Internet Assigned Numbers Authority (IANA). http://www.iana.org/assignments/port-numbers, as of August 12, 2008.

[22] IPFIX: IP Flow Information Export. http://www.ietf.org/html.charters/ipfix-charter.html.

[23] H. Jiang, A. Moore, Z. Ge, S. Jin, and J. Wang. Lightweight application classification for network management. In *Proceedings of the 2007 SIGCOMM workshop on Internet network management*, pages 299–304. ACM New York, NY, USA, 2007.

[24] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. File-sharing in the Internet: A characterization of P2P traffic in the backbone. *University of California, Riverside, USA, Tech. Rep*, 2003.

[25] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is P2P dying or just hiding. In *IEEE Globecom*, 2004.

[26] T. Karagiannis, A. Broido, and M. Faloutsos. Transport layer identification of P2P traffic. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134. ACM New York, NY, USA, 2004.

[27] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 229–240. ACM New York, NY, USA, 2005.

[28] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos. Profiling the End Host. *LECTURE NOTES IN COMPUTER SCIENCE*, 4427:186, 2007.

[29] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices. In *Proceedings of the 2008 ACM CoNEXT conference*, 2008.

[30] L7-filter. Application Layer Packet Classifier. http://l7-filter.sourceforge.net/.

[31] A. Moore and K. Papagiannaki. Toward the Accurate Identification of Network Applications. In *Passive & Active Measurement Workshop*. Springer, 2005.

[32] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. *ACM SIGMETRICS Performance Evaluation Review*, 33(1):50–60, 2005.

[33] A. Moore, D. Zuev, and M. Crogan. Discriminators for use in flow-based classification. *Cambridge, Intel Research*, 2005.

[34] T. Nguyen and G. Armitage. A Survey of Techniques for Internet Traffic Classification using Machine Learning. *IEEE Communications Surveys and Tutorials*, 2008.

[35] J. Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[36] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification. 2004.

[37] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*, pages 512–521. ACM New York, NY, USA, 2004.

[38] A. Soule, K. Salamatia, N. Taft, R. Emilion, and K. Papagiannaki. Flow classification by histograms: or how to go on safari in the internet. *ACM SIGMETRICS Performance Evaluation Review*, 32(1):49–60, 2004.

[39] B. Stiller, P. Barlet-Ros, J. Cushnie, J. Domingo-Pascual, D. Hutchison, R. Lopes, A. Mauthe, M. Popa, J. Roberts, J. Solé-Pareta, et al. Pricing and QoS. Quality of Future Internet Services: COST Action 263 Final Report 2856, 2003.

[40] G. Szabo, D. Orincsay, S. Malomsoky, and I. Szabo. On the Validation of Traffic Classification Algorithms. *LECTURE NOTES IN COMPUTER SCIENCE*, 4979:72, 2008.

[41] G. Szabo, I. Szabo, and D. Orincsay. Accurate Traffic Classification. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–8, 2007.

[42] WEKA: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/.

[43] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Computer Communication Review*, 36(5):5–16, 2006.

[44] N. Williams, S. Zander, and G. Armitage. Evaluating machine learning algorithms for automated network application identification. *CAIA Technical Report*, 060410B, 10 04 2006.

[45] N. Williams, S. Zander, and G. Armitage. Evaluating machine learning methods for online game traffic identification. *CAIA Technical Report*, 060410C, 10 04 2006.

[46] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 169–180. ACM New York, NY, USA, 2005.

[47] F. Yarochkin. Remote OS detection via TCP/IP Stack FingerPrinting. *Available http://insecure. org*, 12:14–24, 1999.

[48] S. Zander, T. Nguyen, and G. Armitage. Automated Traffic Classification and Application Identification Using Machine Learning. In *CONFERENCE ON LOCAL COMPUTER NETWORKS*, volume 30, page 250. IEEE, 2005.

[49] S. Zander, T. Nguyen, and G. Armitage. Self-learning IP Traffic Classification based on Statistical Flow Characteristics. In *Proc. of the 6th Passive and Active Network Measurement Workshop, March*. Springer, 2005.

[50] D. Zuev and A. Moore. Traffic classification using a statistical approach. In *Passive and Active Measurement*. Springer, 2005.