Methods and Algorithms for Network Traffic Classification

Thèse présentée
pour obtenir le grade de docteur
de Télécom Paris Tech

Marcin Pietrzyk

April 2011

Directeurs de Thèse:
Professor Dr. Guillaume Urvoy-Keller, CNRS/UNSA
Dr. Jean-Laurent Costeux, Orange Labs

Abstract

Traffic classification consists of associating network flows with the application that generated them. This subject, of crucial importance for service providers, network managers and companies in general has already received substantial attention in the research community. Despite these efforts, a number of issues still remain unsolved. The present work consists of three parts dealing with various aspects of the challenging task of traffic classification and some of its use cases.

The first part presents an in-depth study of state-of-the-art statistical classification methods which use passive traces collected at the access network of an ISP offering ADSL access to residential users. We critically evaluate the performance, including the portability aspect, which so far has been overlooked in the community. Portability is defined as the ability of the classifier to perform well on sites (networks) different than the one it was trained on. We uncover important, previously unknown issues, and analyze their root cause using numerous statistical tools. A last contribution of the first part is a method, which allows us to mine the unknown traffic (from a deep packet inspection tool perspective).

The second part aims at providing a remedy for some of the problems uncovered in part one, mainly the ones concerning portability. We propose a self-learning hybrid classification method that enables synergy between a priori heterogeneous sources of information (e.g. statistical flow features and the presence of a signature). We first extensively evaluate its performance using the data sets from part one. We then report the main findings for tool deployment in an operational ADSL platform, where a hybrid classifier monitored the network for more than half a year.

The last part presents a practical use case of traffic classification and focuses on profiling the customers of an ISP at the application level. The traffic classifier here is the input tool for the study. We propose to rely on the hierarchical clustering technique to group the users into a limited number of sets. We put emphasis both on heavy hitters and less active users, as the profile of the former (p2p) differs from the one of the second group. Our work complements recent findings of other works reporting an increasing trend of HTTP driven traffic becoming dominant at the expense of p2p traffic.

Acknowledgments

First of all, I would like to express my deep gratitude to my thesis supervisor, Prof. Dr. Guillaume Urvoy-Keller, who suggested that I pursue this research direction and who convinced me it was possible, despite my doubts. You have continuously supported all my efforts, both in the research and in my personal projects, not to mention the invaluable support and countless contributions to this work. Thank you for everything, it was a truly great experience working with you!

I would also like to thank my co-advisor in Orange, Dr. Jean-Laurent Costeux, who was my guide in the jungle of large corporations. Thanks for all your valuable input in this thesis.

Big thanks are also due to all my collaborators, whose experience and knowledge contributed greatly to this work, especially Dr. Taoufik En-Najjary, whose brilliant ideas and incredible theoretical knowledge made many parts of this work possible, it was a great pleasure to have you on my side! Thanks to Louis Plissonneau for sharing the good and the difficult parts of a PhD students life. I would also like to thank Prof. Dr. Rui Valadas, Prof. Dr. Rosario de Oliveira and Denis Collange for their input on the features stability evaluation and the fruitful discussions.

I would like to thank all my colleagues from Orange Labs, especially Frederic and Mike, my office mates, Chantal, Patrick, Olivier, Paul and Seb, as well as all the non-permanent people that I had the chance to meet during the three years spent at the Lab. Last but not least, I would like to thank my interns, especially Michal, for his great input in the development of the monitoring tools.

Finally, I would like to express my deepest gratitude to my parents and girlfriend for their support in all my life choices. Well, I owe you everything!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past few decades we have witnessed the amazing growth of the Internet as a world-wide communication infrastructure. We have observed its evolution from the earliest applications, such as the first e-mail, up to large distributed applications used by nearly two billion people on the planet (as of June 2010). In principle, the Internet is built so that the network only provides a way for information to reach its final destination. The downside of this simplicity is that the network fails to address some of todays needs, especially those related to monitoring and management of the network.

The need we address in this thesis is the ability to classify traffic flowing in the network according to the end-host application that has generated it. This is indeed a challenging task, given that the network does not directly offer such a functionality, and that one needs to rely on limited and often incomplete information to achieve this goal.

Service providers, and more generally network administrators, have expressed their deep interest in being able to identify network traffic for a number of reasons, such as: i) monitoring application trends (e.g. identification of a ï¿œkiller applications), ii) the ability to apply policies depending on the traffic class, for instance, providing better quality of service for voice traffic, iii) knowledge of the applications used helps in understanding end users better and in being a valuable input of many studies, from quality of experience to marketing predictions.

The task is challenging due to the fast emergence of applications and the evolution of existing ones. Furthermore, some applications try to obfuscate traffic to avoid detection, as is the case for peer-to-peer clients. We are thus witnessing a race between the methods of obfuscation and detection similar to those in computer security. So far, despite extensive research in the domain, a number of aspects remain unsolved. In this thesis, through extensive evaluation, we uncover several formerly overlooked issues, as for instance, classifier portability problems. We further propose several techniques to first detect the root causes, then we attempt to address them by designing a machine learning based hybrid classifier.

1.1 Organization of the Thesis

The vast majority of the thesis deals with the different aspects of traffic classification or its use cases. The work is divided into three main parts consisting of short and compact chapters. Part I presents the results of an extensive evaluation of state-of-the-art statistical classification techniques with ADSL traces. In Part II we introduce Hybrid Traffic Identification, a machine learning based technique allowing to combine different classification methods. The last part (III) of the thesis focuses on the methods for profiling residential customer application usage. Each part starts with a short introduction, contributions summary and a list of relevant publications. Chapter 14 concludes the thesis and gives our opinion on how this research could be extended in the future.

Part I

Statistical Methods Evaluation and Application

Chapter 2

Introduction

One of the key issues for companies and Internet Service Providers (ISPs) is the ability to precisely identify the applications flowing in their networks. Motivations behind this need are manifold: (i) enforcement of internal or national rules, e.g., banning p2p traffic from an Intranet, (ii) better understanding of actual and emerging applications (iii) assessment of the impact of those applications on peering agreements and/or the return on investment if some p4p initiative was taken [100] or (iv) possibility to offer additional services based on application, for instance, protection of multimedia transfers.

The current state of the art for most companies, including ISPs, is to rely on some proprietary solutions that implement deep packet inspection (DPI) techniques featuring signatures and ad-hoc heuristics to detect current applications. While this approach can be accurate, the trend of obfuscating traffic highlights the need of alternative detection methods. Recently, several solutions based on machine learning techniques and per flow features were proposed in the literature e.g. [62, 5, 4, 58, 67]. The majority of these techniques were tested on academic traces, use different traffic features as inputs to the statistical classification algorithm and define flows and application classes differently. In the first part of the thesis, we are evaluating several aspects of the machine learning based classification (we term such tool a statistical classifier). We adopt the perspective of an ADSL provider and evaluate usefulness of the statistical methods as a complementary tool to deep packet inspection.

We have collected several hour long traces at various ADSL PoPs of a French ISP. Our data set is unique, as those traces form an homogeneous set in the sense that they were captured at about the same period (beginning of 2008) and all PoPs are under the control of the same ISP. Using those traces, we address the following issues:
Can we obtain a high classification performance, and this, for all the applications of interest? Can statistical methods help in mining the traffic that DPI tools failed to classify? Is the statistical model representative of the applications, i.e., can we train the classifier on one site and use it on another one without specific adjustments or re-training? Could we use a statistical tool as an alternative or support for commercial DPI tools?

2.1 Contributions - Part I

Contributions of our study can be categorized into two sets. The first set relates to the use of statistical techniques, with state of the art feature sets, when the statistical classifier is applied on a site **different** from the one on which it was trained. We show that: *Average performance is good* – when considering all flows and applications – but results can greatly deteriorate on an application basis. This means that some applications that are correctly classified when the classifier is applied on the same site where it was trained, become difficult to identify when applied on another site. We demonstrate that many applications can suffer from this problem, including mail, ftp and some p2p applications.

We further demonstrate that similar issues persist for a variety of features sets, including their combinations (obtained with principal component analysis). Finally we show that in depth investigation of those cases allows us to prove that the problem stems on one side, from an *overfitting* of the data, where the classifier learns some site specific characteristics used by local clients/applications. On the other hand, the root cause of the problem lies in the features variations between the sites. To address the latter issue, we use statistical test that aims at verifying the features stability on per application basis. Although, multiple applications are concerned by the cross site problems, we also identify few exceptions which seem to be immune to the issue.

Although we reveal the portability (cross site) problems of the statistical classifier, on the positive side the performance is good if we restrict to a single site. Our second approach presents the case where the classifier is used for **each site independently** of the others. In such a scenario, we demonstrate that: statistical classification can help revealing the traffic left unknown by the ground truth establishment tool. More precisely, we demonstrate that supervised classification techniques can divide by a factor of 2 the amount of bytes previously unidentified by our DPI tool.

2.2 Relevant Publications for Part I

[76] M. Pietrzyk, G. Urvoy-Keller, T. En-Najjary, J-L. Costeux Challenging statistical classification for operational usage : the ADSL case, 9th ACM SIGCOMM Internet Measurement Conference, 2009, Chicago, USA.

[81] M. Pietrzyk, G. Urvoy-Keller, J-L. Costeux, Revealing the unknown ADSL traffic using statistical methods, published in the first Workshop on Traffic Monitoring and Analysis TMA'09, May 2009, Aachen, Germany.

Chapter 3

Background on Traffic Classification

In this chapter we first present formally the problem of traffic classification, followed by a brief survey on the classification methods and the related work. We further introduce two important aspects of the study, that will be referred to throughout the thesis, namely the performance metrics and the issue of the ground truth.

3.1 Definition of the Traffic Classification Problem

Let us formally define the problem of traffic classification. Traffic classification consists in associating each flow to an application, based on the features (e.g. mean size of packets) that have been extracted for this flow. Let X be the n-dimensional random variable corresponding to the flow features. A vector $x = (x_1, \ldots, x_n)$ consisting of the n measured features is associated to each flow. Let us assume that there are c applications. We define a random variable Y that indicates the application that generated a flow. It takes values in the set $\{1, 2, \cdots, c+1\}$. $Y = c+1$ means that the flow is not associated to any class, i.e., it is unknown. The problem of traffic classification is to associate a given flow x with an application y.

3.2 Performance Metrics

In order to benchmark the performance of a classifier performance metrics are needed. We use recall and precision to assess the quality of statistical classifiers. These are popular metrics in classification problems in general. They are built upon the notion of True Positives (TPs), True Negatives (TNs), False Positives (FPs) and False Negatives (FNs). These notions are defined with respect to a specific class. Let us consider such a specific class, say the WEB class. TPs (resp. FNs) are the WEB flows that are labeled (resp. not labeled) as WEB by the statistical classifier. FPs (resp. TNs) are the flows not labeled as WEB by

ODT (ODT is introduced in next section) that are labeled (resp. not labeled) as WEB by the statistical classifier.

Recall and precision are defined as follows:

- **Recall**: Fraction of flows/bytes of a specific class correctly classified. It is the ratio of TPs to the sum of TPs and FNs for this class. For example, a recall score of 50% for the WEB class means that only half of the WEB flows/bytes are labeled correctly by the statistical classifier.

$$Recall = \frac{tp}{tp + fn}$$

- **Precision**: For a given class, it is the fraction of TPs in this class. For example, a precision of 100% for the WEB class means that the statistical classifier has put in this class only WEB flows/bytes. This result is satisfactory only if all WEB flows are actually in this class, which is measured by the recall.

$$Precision = \frac{tp}{tp + fp}$$

Ideal classifier needs to maximize both metrics on per class basis.

3.3 Ground Truth

A Crucial component in any classification study is a pre labeled reference set, that we use as a "ground truth" e.g. to compute performance metrics as defined in Section 3.2.

Typically traffic classification studies rely on DPI solutions to establish the ground truth. We are aware that the wording ground truth remains tricky as even DPI tools might fail. We face in this thesis the same issue as former studies in the domain. However, there barely exists any alternative to DPI. Some approaches have been recently proposed to obtain high quality reference data sets. In [35], the authors propose a network driver installed on end hosts (a similar approach was recently proposed in [38]). This application flags flows according to the application generating traffic. However, this solution is not applicable to the case of large ADSL traces as the ones handled in this work.

In this thesis, we rely on an internal tool of Orange, that we term Orange_DPI_Tool, or ODT for short. ODT is in use on several PoPs of Orange in France.

We used two other DPI tools and compare their results on an example ADSL trace (description provided in Table 4.1):

- A tool based on Bro [9] that implements the set of signatures used by Erman at al. in [27], as extracted from the technical report of the same author;
- Tstat v2 [92] that features DPI functions.

The results with Bro turned out to be deceiving, with more than 55% of unknown flows. A more detailed inspection of over thirty of the signatures used, revealed that most of them are outdated.

We thus decided to focus our comparison on ODT and Tstat only. ODT is used for network analysis and dimensioning. It is capable of detecting several tens of applications, including some encrypted ones. It combines several methods of traffic classification, from deep packet inspection to methods as sophisticated as parsing the signaling messages of an application in order to extract endpoint IPs and ports. ODT is constantly updated and tested on several sites in France.

To compare Tstat to ODT we need to devise a set of application classes that both tools detect. We consider the following set: Web, eDonkey, BitTorrent, Ares and Mail.

Tstat 2.0 vs ODT [flows %]			
Class	Tstat Breakdown	ODT Breakdown	Overlap
UNKNOWN	32,67	12	27,92
WEB	58,35	77	81,31
EDONKEY	3,81	4,85	90,72
BITTORENT	0,91	1,06	99,52
ARES	0,09	0,06	99,53
MAIL	3,65	5,06	83,41

Table 3.1: Tstat vs. ODT comparison (main applications only).

Results of the comparison between Tstat and ODT are depicted in Table 3.1. We use for the comparison one of the traces (MSI) that will be introduced in details in later chapter. We report the breakdown of flows obtained using each tool and also the overlap between the two tools taking the union of both sets as a reference for each class. Overlap for class A is defined as $\frac{(T \bigcap O)}{T \cup O}$ where T is set of all the flows classified as A by Tstat, and O is set of all the flows classified as A by ODT. Overlap is expressed in percentages.

For p2p applications, the agreement is very good, in each case higher than 90%. For Mail and Web, we have more significant differences. A closer look at Web traffic revealed that the difference between the two tools is mostly due to ODT identifying more Web transfers than Tstat. This additional set of flows consists of a large fraction of connections to port 443 - Https service - or flows with less than three packets. This most probably explains why Tstat did not classify them as Web. As for the Web flows identified by Tstat only, they appeared to be mostly due to streaming applications over Http, e.g., YouTube video streaming. Tstat labels those flows as Web while ODT labels them as Http Streaming. While there is a limited number of such flows, they carry a significant amount of bytes, which leads to a more pronounced disagreement between Tstat and ODT when focusing on bytes rather than flows. More generally, looking at bytes provides a different picture. For instance, for the case of eDonkey, Tstat and ODT agree for only 50% of the bytes. This is because the Tstat version we use, does not recognize obfuscated eDonkey traffic. We fed Tstat with hand-made obfuscated eDonkey traces to confirm that it does not detect encrypted traffic.

The main lesson we learn from the above study is that even two state-of-the-

art DPI tools can lead to sometimes significantly different reference points. This often overlooked issue, has dramatic impact on the results and conclusions about the performance of the classifier that is evaluated. The issue is studied in more details in recent work by Dusi et al. [22] where authors verify accuracy of DPI tool based on L7 filters [60]. They conclude that, in many cases the ground truth provided is incorrect.

In the remaining of this thesis, we rely on ODT only, due to the lowest fraction of Unknown traffic it offers (on our traces) and the largest variety of the applications that the tool can handle, as compared to Tstat. It offers certainly not perfect, but sufficient approximation of the real traffic classes. Although we decide to relay on the ODT as our reference point, we take several view points that allow us dig in the traffic and better understand the results.

3.4 Methods and Related Work

Many different methods have been introduced to classify traffic. Traditional approaches relied on transport layer *port numbers* [45]. However, early works [54, 66] quantitatively demonstrated the decrease of recall (see section 3.2) of conventional classification methods based on port numbers.

It triggered the emergence of *deep packet inspection* (DPI) solutions that identify the application based on signatures found in packet payloads or connection patterns [92, 60, 9, 85, 88, 80]. Authors in [75] proposed a method for automatically generating application-level signatures, using samples of traffic. DPI based solutions are already commercialized by several vendors, for instance [83, 1]. However, the increasing use of encryption and obfuscation of packet content, the need of constant updates of application signatures and governments regulations, might undermine the ability to inspect packet content.

To address these problems, recent studies relied on *statistical classification* techniques to probabilistically map flows to applications [58, 62, 5, 4, 67, 70, 27, 6, 24, 37, 36, 47]. Hereafter, we cite a representative sample of traffic classification works. For a much more complete survey, we refer to the work by Nguyen et al. [71].

Another argument raised in favor of statistical classification is low DPI scalability to high bandwidth. However, a recent study undermines this belief [13]. The authors compared SVM-based classifier with DPI using several traces. They conclude that an processing cost of both approaches is comparable and highly depends on the type of traffic mixture analyzed. In [12] the authors propose several simple optimization techniques, such as limitation of classification attempts, for DPI based tools. Applying those techniques can further decrease the DPI computational costs.

Moore et al. [67] presented a statistical approach to classify the traffic into different types of services based on a combination of flow features. A naive Bayesian classifier combined with kernel estimation and a correlation-based filtering algorithm was used to classify TCP traffic. They used 10 flow features and obtained recall score (see Section 3.2) between 93% and 96%. However, their

data set contains mainly Web traffic.

Bernaille et al. [5] presented an approach to identify applications using start-of-flow information. The authors used the size and direction of the first 4 data packets and port numbers in each flow as features on which they trained a Gaussian mixture model and a Hidden Markov model. These models featured recall scores over 98%. The authors further extended their work to the identification of encrypted traffic in [4].

Karagiannis et al. [55] addressed traffic identification by analyzing interactions between hosts, protocol usage and per-flow average packet size. Their techniques were able to classify 80%-90% of the traffic with a 95% recall. In their recent work [56], they used those techniques to profile users' activity, and to analyze the dynamics of host behaviors. Similar strategy was applied in recent work by Iliofotou et al. [46]. The authors propose to leverage IP-to-IP communication graph to infer application communities. Profiling few members within a cluster can reveal the whole community. The reported recall is on average around 90% assuming knowledge about 1% of hosts.

Finamore et al. [33] proposed a method for classification of UDP traffic, mainly P2P streaming applications. The method leverages statistical signatures derived from packet payload by means of chi-square test. The average True Positive percentage reported is 99.6%. The method is confronted with another behavioral approach for P2P-TV in [34]. The authors conclude that the behavioral classifier can be as accurate as the payload-based classifier with also a substantial gain in terms of computational cost, although it can deal only with a very specific type of traffic.

There also exists a lot of work focusing on specific applications. For example, Bonfiglio et al. [6] proposed an approach specifically intended to identify Skype traffic by recognizing specific characteristics of Skype. A number of papers have been published focusing on the identification of p2p traffic [54, 53, 19].

A number of works tackled traffic classification by trying to combine existing methods in several ways. Authors of [89] ran several independent classification modules including port based, heuristic based and signature based modules. The authors rely on prioritization to reconcile diverging modules' outputs. A similar idea was proposed in [52]. The authors in [15] design an hybrid classifier for peer-to-peer traffic only, combining several methods used as input of a Flexible Neural Tree model. The classifier is then implemented on a network processor and evaluated in a small and isolated test network.

Nechay et al. [68] proposed a method to provide performance guarantees for particular classes of interest. Their technique is based on Neyman-Pearson classification and one based on the Learning Satisfiability (LSAT) framework that can provide class-specific performance guarantees on the false alarm and false discovery rates, respectively. The method has been validated using traffic data provided by an ISP.

The research community proposed a large number of methods and algorithms to solve the issue of traffic classification. Due to variety of the traces and experiment setups it is difficult to compare those methods. In particular, domain suffers from lack of shared reference sets available. Dainotti et al. [21] proposed a tool

that aims at solving this problem. It is a classification platform that implements several techniques and allows direct comparison between them using the same setup and metrics.

Chapter 4

Methods Evaluation - Background

This chapter introduces the methodology for the evaluation of the statistical methods. We detail the traffic data, the classification algorithms and the features sets we use. Evaluation results will be presented in the next chapters.

4.1 Traffic Data

In this section, we present our dataset, how we establish the reference point (ground truth) that is used as benchmark for our statistical classifier, the definition of our traffic classes and the traffic breakdown.

4.1.1 Dataset

Our dataset consists of four recent packet traces collected at three different ADSL PoPs in France from the same ISP. All traces were collected using passive probes located behind a Broadband Access Server (BAS), which routes traffic to and from the digital subscriber line access multiplexers (DSLAM) and the Internet. Captures, which include full packet payloads, were performed without any sampling or loss and contain over four million TCP flows. Each trace contains at least one hour of full bidirectional traffic, with a similar number of active local users varying between 1380 and 2100. For details, see Table 4.1.

Traces have some important *spatial* and *temporal* features: traces MSI and RIII were captured at the same time at two different locations which helps assess spatial stability of the method[1]. Traces RII and RIII were captured at the same location with an offset of seventeen days between them.

We use the traces presented in this section throughout the thesis (Part I, Part II) for a number of various experiments. In the Part III we will use different data traces which allow also users tracking despite the dynamic IP allocation.

[1]We also term this problem as the "cross-site" issue.

Set	Date	Start	Dur	Size [GB]	Flows [M]	TCP Flows [%]	TCP Bytes [%]	Local users	Distant IPs
MS-I	2008-02-04	14:45	1h	26	0.99	63	90.0	1380	73.4 K
R-II	2008-01-17	17:05	1h 10m	55	1.8	53	90.0	1820	200 K
R-III	2008-02-04	14:45	1h	36	1.3	54	91.9	2100	295 K
T-I	2006-12-04	12:54	1h 48m	60	4.1	48	94.7	1450	561 K

Table 4.1: Trace summary.

[flows%/bytes%]				
Class	**MSI**	**RII**	**RIII**	**TI**
WEB	67/49	40/25	26/21	16/21
EDONKEY	4/6	15/27	16/28	33/40
MAIL	4/5	2/1	1/0.83	3/1
CHAT	1/0.51	2/1	0.79/0.25	0.42/1.44
HTTP-STR	1/12	0.83/13	0.61/9	0.27/3
OTHERS	8/2	15/0.16	33/0.36	18/1
DB	1/3	3/0.01	3/0.01	0.49/0.02
BITTORRENT	0.94/3	2/8	7/2	3/7
FTP	0.46/0.11	0.11/0.1	0.16/0.5	0.17/0.67
GAMES	0.08/6	0.12/0.41	0.11/0.09	0.27/0.4
STREAMING	0.05/0.054	0.13/1	0.13/1	0.12/1
GNUTELLA	0.09/1	1/4	0.76/3	1/2
UNKNOWN	9/7	14/15	13/24	21/18

Table 4.2: Application breakdown in the data sets (using ODT).

4.1.2 Reference Point

In order to benchmark the performance of any classification method, a dataset with pre-labeled classes of traffic is needed. We term such a dataset our reference point (a.k.a ground truth). Establishing a reference point is fundamental when evaluating traffic classification mechanisms to provide trust-worthy results. As a human-labeled dataset is almost impossible to have, we rely on DPI tools. We use ODT, which was introduced in the section 3.3

4.1.3 Traffic Breakdown

Classes used in Part I are summarized in Table 4.3. This choice of classes can be considered as a typical one for an ISP that monitors its network. It calls for a few remarks. First, HTTP traffic is broken into several classes depending on the application implemented on top: Webmail is categorized as mail, HTTP streaming as streaming, HTTP file transfers as FTP, etc. Second, popular p2p applications have their own class. Less popular p2p applications are merged into the P2P-REST class. The OTHERS class aggregates less popular applications that ODT recognized (See Table 4.3).

Table 4.2 shows classification results obtained by ODT, in flows and bytes, for our four traces. On PoPs where ODT is used continuously, we checked that the application breakdown is typical of the traffic observed on longer periods of time (day or week). Among the p2p applications, most bytes and flows are due to

Class	Application/protocol
WEB	HTTP and HTTPs browsing
EDONKEY	eDonkey, eMule obfuscated
MAIL	SMTP, POP3, IMAP, IMAPs
	POP3s, HTTP Mail
CHAT	MSN, IRC, Jabber
	Yahoo Msn, HTTP Chat
HTTP-STR	HTTP Streaming
OTHERS	NBS, Ms-ds, Epmap, Attacks
DB	LDAP, Microsoft SQL, Oracle SQL, mySQL
BITTORRENT	Bittorrent
FTP	Ftp data, Ftp control, HTTP file transfer
GAMES	NFS3, Blizzard Battlenet, Quake II/III
	Counter Strike, HTTP Games
STREAMING	MS Media Server, Real Player
	iTunes, Quick Time
GNUTELLA	Gnutella
ARES	Ares
TRIBALL	Triball
P2P-REST	Kazaa, SoulSeek, Filetopia, Others
NEWS	Nntp
UNKNOWN	-

Table 4.3: Application classes.

eDonkey (more precisely eMule client [23]) followed by Bittorrent and Gnutella. Concerning eDonkey, we observed that obfuscated traffic accounts typically for half of the bytes in the EDONKEY class. Less popular file sharing applications (including the P2P-REST class) generated a negligible amount of flows and bytes. We exclude them from our subsequent analysis. We also exclude the NEWS class for similar reasons.

The vast majority of traffic in the HTTP-STR class is due to Dailymotion [20] and Youtube [101], which account for 80% of the bytes. P2P streaming applications, that fall into the STREAMING class, are active during short time periods, e.g., popular sport events, which probably explains why we do not observe such traffic in our data. The OTHERS class contains mostly unidirectional flows to ports 135, 445 and 139. Those Windows services are targeted by a large family of self-propagating malware (see for instance [61]).

Overall, ODT provides fractions of UNKNOWN bytes that range between 8% and 24% depending on the trace. In Sections 5.1 and 5.2 of chapter 5, we consider only traffic known to ODT, keeping unclassified flows aside. We focus on the UNKNOWN class in chapter 8.

4.2 Methodology

This section describes our classification methodology to build our statistical classifier, including the classification algorithms, the flow features and the data cleaning process (flow definition).

4.2.1 Classification Algorithms

In part one of the thesis, we rely on machine learning algorithms provided in the Weka suite [98], that is widely used in the context of traffic classification [62, 58, 99]. Specifically, we evaluated the following supervised learning algorithms [62, 58]:

Naive Bayes with Kernel Estimation: This algorithm is a generalization of the Naive Bayes one. Bayesian statistical conclusions about the class c_j of an unobserved flow x are based on probability conditional on observing the flow x. This is called the posterior probability and is denoted by $p(c_j|x)$. The Bayes rule gives a way of calculating this value:

$$p(c_j|x) = \frac{p(c_j)f(x|c_j)}{\sum_{c_j} p(c_j)f(x|c_j)} \tag{4.1}$$

$p(c_j)$ denotes the probability of obtaining class c_j independently of the observed data (prior distribution), $f(x|c_j)$ is the distribution function (or the probability of x given c_j) and the denominator is a normalizing constant.

The goal of the supervised Bayes classification problem is to estimate $f(x|c_j)$ given some training set. Naive Bayes makes certain strong assumptions about the data, that are partially solved by the kernel estimation. For further details we refer the reader to the work by Moore et. al [67].

Bayesian Network: this algorithm makes use of a model that represents a set of features (or categories) as its nodes, and their probabilistic relationship as edges. In some cases, Bayesian Network may outperform Naive Bayes.

C4.5 Decision Tree: The C4.5 algorithm constructs a model based on a tree structure, in which each internal node represents a test on features, each branch representing an outcome of the test, and each leaf node representing a class label. While training, the algorithm iteratively looks for the best feature to partition the data at a given node, relying on the relative information gain ratio (equation 4.2). The division continues until the node becomes a leaf node. Information gain ratio measures the correlation between two random variables: a feature and a class label in our case. In the general case, given two discrete random variables X and Y, the gain ratio is defined as:

$$GAINRATIO(X|Y) = \frac{H(X) - H(X|Y)}{H(X)} \tag{4.2}$$

where:

$$H(X) = -\sum_{x_i} p(x_i) \log p(x_i) \tag{4.3}$$

and:

$$H(X|Y) = -\sum_{j} p(y_i) \sum_{i} p(x_i|y_j) \log p(x_i|y_i) \tag{4.4}$$

The version we use [98] incorporates a number of improvements such as pruning that aims at reducing data over-fitting. More details about the algorithm can be found in [51]. C4.5 has been widely used for traffic classification, [62, 58, 99]. For all the scenarios we investigated in this part, C4.5 offered the best performance in terms of recall and precision (see Section 3.2 for definitions). *Unless stated otherwise*, results presented in this part were obtained with the C4.5 decision tree algorithm. We will elaborate on the results of the other algorithms in Section 5.4.

4.2.2 Features

Two broad families of features have been used for classification in the literature. The first one relies on packet-level information like packet sizes [5, 4]. The second family of features consists of flow-level statistics like duration or fraction of push flags [62]. Accordingly, we use two feature sets, one from each family. The first one, that we designate as **set A**, was proposed in [5]. It consists of the size and direction of the first few data packets of a transfer. The second one, **set B**, consists of per flow features inspired by [62]. The full list of features we use is given in Table 4.4[2]. We test separately both sets. To extract packet sizes we used the tool released by authors of [5]. For set B, we used ad-hoc tools.

Abbreviation	**Description**
Push_pkt_down	Count of packets with Push flag downstream
Push_pkt_up	Count of packets with Push flag upstream
Avg_seg_size_down	Data bytes divided by # of packets downstream
Min_seg_size_down	Minimum segment size down
Data_pkt_down	Packets with payload downstream
Pkt_size_median_up	Packet size median upstream
Local_port	Local TCP port
Distant_port	Distant TCP port

Table 4.4: Set B - per flow features.

[2]The features were computed over the whole flow in contrast to [62] where the first five packets of each transfer was used.

4.2.3 Training Set

With supervised machine learning algorithm, one generally trains the classifier on a fraction of the dataset and tests its performance by applying the (trained) classifier on the remaining of the dataset. Classically, one relies on the 10-fold cross validation technique: for each trace, the algorithm is trained on one tenth of the data and then applied on the remaining flows for each possible slice comprising 10% of the data. Reported results are averages of those ten experiments.

A problem faced in traffic classification is that the number of samples per class is highly varying. This might lead the most prevalent classes to bias the training phase of the classifier. As an alternative, one can use a training set with the same number of flows per class. This approach was advocated in [5]. With our dataset and classes definition, we must limit the number of flows per class to a few hundreds if one wants to apply this approach.

In order to evaluate the impact of different learning scenarios, we trained our classifier using two training sets: (i) 200 flows for each class, (ii) 10,000 flows for the applications with enough flows, and the maximum number of available flows for the less popular applications.

In both cases we obtained similar results with our datasets: less popular classes (e.g. HTTP-STREAMING, GAMES, DB) obtained higher accuracies as compared to the legacy 10-fold cross validation technique, but we observe a decrease of recall for the dominant classes, e.g., it drops from 97% to 53% for the WEB class in trace R-III. A closer look at the confusion matrix[3] reveals that by balancing the number of training flows, we are favoring less popular applications causing popular classes to be misclassified. More generaly, we can conclude that in case of unbalanced data sets like ours, there apparently exists a tradeoff between the overall recall and the recall of less popular traffic classes.

Given the above observations, we decided to use 10-fold cross validation in Section 5.1 where training and testing are performed on the same trace. On the contrary, when training and testing are performed on difference traces – Section 5.2 – we use the whole dataset to build the model.

4.2.4 Flow Definition - Data Cleaning

We seek to classify bidirectional TCP flows. We use the definition of a flow based on its 5-tuple {*source IP address, destination IP address, source port, destination port, protocol*}. We restrict our attention to TCP flows as they carry the vast majority of bytes in our traces. We are still left with the issue of defining the set of flows to be analyzed. We might restrict ourselves to flows for which a three-way handshake is observed. We can be even more restrictive by imposing observation of a FIN or RST flag at the end of the transfer. The latter option is advocated by the authors in [62], as they observed that for their (academic) traces, imposing this additional constraint does not significantly reduce the fraction of flows and bytes to be analyzed. This is not the case with our traces as we will see below.

[3]Confusion matrix shows all the missclassifications. It allows to understand which classes are confused whith each other.

Some restrictions might also be imposed by the classification method itself. For instance, when using as features the size of the first 4 data packets (the choice of 4 is justified in Section 5.1.1), we implicitly exclude all flows with less than 4 data packets. Note that padding small flows with zeros would fool the classifier, and thus it is not an option.

To gain a clear view of the impact of the various filtering options, we applied successively the three following flow definitions to the flows in our traces:

- **S/S:** Only flows with a three way handshake.
- **S/S+4D:** Only flows with a three way handshake and at least four data packets. We used the tool publicly released after the work in [5].
- **S/S+F/R:** Only flows with a three way handshake and with a FIN or RST flag at the end of the data transfer.

Results are depicted in Table 4.5 for the case of the MS-I trace (other traces offer similar results), with one line per application and the last line presenting average results. Clearly, imposing constraints on the termination of the flow appears extremely restrictive as about 50% of the bytes are exluded from the analysis. On a per application case, the issue can be even more pronounced.

Even imposing the observation of a three way handshake can heavily impact some applications. This is the case for STREAMING, GAMES, DB, and OTHERS. The latter case (OTHERS) results from the nature of traffic carried (presumably attacks), as explained in Section 4.1.2. For the other classes, this decrease in bytes can be due to flows for which we do not observe the beginning.

Observing the beginning of a transfer is however crucial for traffic classification in general, as it carries application level specific information (while the rest of the transfer might be user data for instance). We thus analyzed only those flows for which we observed a proper three-way handshake. Note that even though the amount of bytes is reduced for some classes, the remaining number of flows per class is large enough (at least several hundreds) to justify further statistical analysis.

Our first set of features (packet sizes) imposes that we have at least 4 data packets per transfer. As we can see from Table 4.5, this further reduces the number of flows per application but has little impact on the number of bytes due to the heavy-tailed nature of the Internet traffic.

4.3 Summary

In this chapter we introduced background information for the traffic classification experiment described further in the next chapter. We described the data traces and the breakdown of traffic, followed by the description of the machine learning algorithms and the features set. Last but not least we evaluated several common strategies of data cleaning that affect the set of flows under analysis.

Class	MS-I [flows%/bytes%]		
	S/S+4D	**S/S**	**S/S+F/R**
WEB	32%/73%	89%/83%	80%/64%
EDONKEY	88%/91%	97%/98%	86%/51%
MAIL	78%/79%	86%/80%	57%/55%
CHAT	81%/80%	87%/80%	80%/60%
HTTP-STR	85%/98%	92%/99%	81%/79%
OTHERS	11%/35%	22%/42%	16%/24%
DB	27%/11%	33%/12%	15%/9%
BITTORRENT	31%/83%	90%/90%	80%/38%
FTP	29%/65%	76%/67%	71%/64%
GAMES	33%/7%	53%/7%	44%/5%
STREAMING	44%/25%	67%/32%	60%/18%
GNUTELLA	12%/90%	96%/95%	91%/46%
UNKNOWN	19%/19%	39%/21%	34%/14%
OVERALL	34%/69%	77%/75%	68%/55%

Table 4.5: Remaining flows/bytes depending on the flow definition.

Chapter 5

Method Evaluation

This chapter presents the results of the traffic classification evaluation study, while using supervised machine learning algorithms and state of the art feature sets. The experiment is a first large scale evaluation of such methods using quality ADSL traces and addressing portability of the classifier (cross site evaluation).

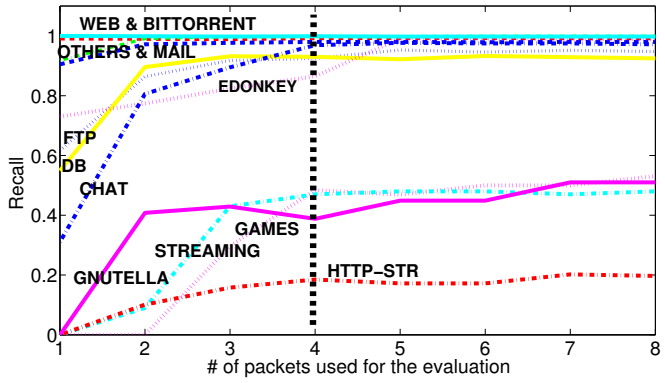
We use the methodology and the data sets presented in chapter 4, namely the supervised machine learning algorithms (e.g. C4.5, Naive Bayes) and two popular flow feature sets. We first demonstrate the outcome of the static and cross site evaluations. It is followed by the discussion on the impact of the exact algorithm and some insight about the instability problems we encountered. The last part will be further developed in chapter 7 which is dedicated to finding the root causes of the cross site portability problems described in the current chapter.

5.1 Classification - Static Case

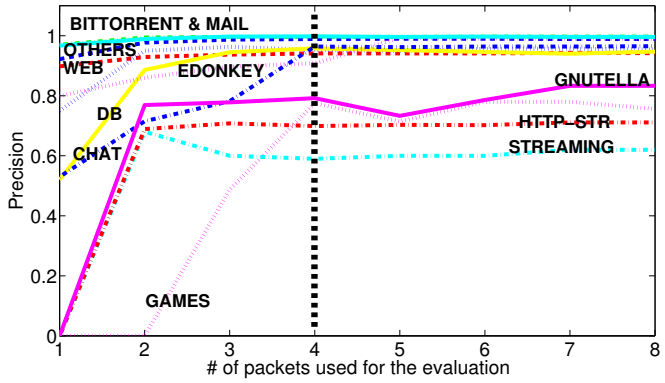
In this section we investigate the performance of statistical classification on each site, independently of the others. We term "static case" this issue, as compared to the cross-site case that we will detail in Section 5.2.

5.1.1 Number of Packets - Set A

When using the sizes of the first data packets of a transfer as classification features (set A described in chapter 4), we must choose the actual number of packets to be considered. We denote this number as k. We choose the k value that offers the best trade off between recall and precision per application. In Figures 5.1(a) and 5.1(b), we depict the evolution of recall and precision for increasing k values. Results presented were obtained using trace MS-I, as they are similar with other traces. Based on those results, we set k to four packets for the rest of this chapter. Note that this value is in line with the ones recommended in [5].



(a) Recall



(b) Precision

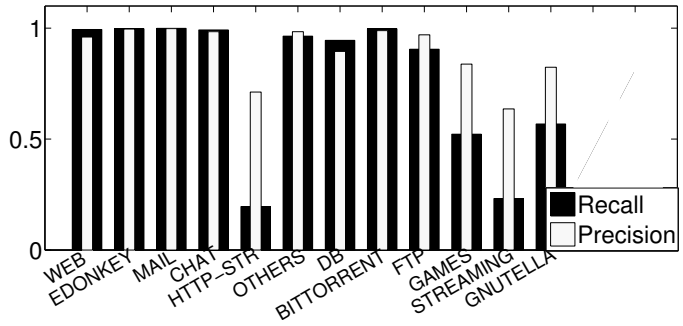
Figure 5.1: Per-class recall and precision vs. number of packets used.

5.1.2 Static Results

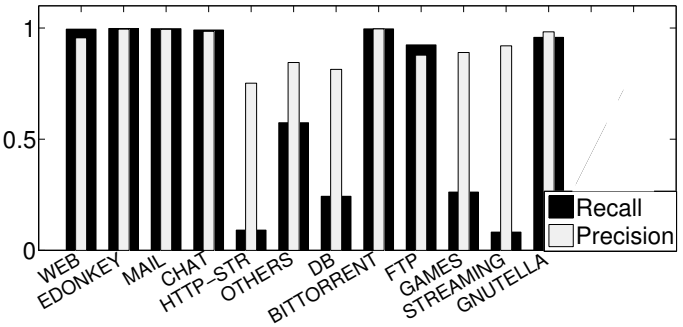
When the classifier is run on the trace on which it was trained, we obtained overall recall (over all classes) that are consistently high, above 90% for both sets A and B. The reason behind this result is that the dominant classes in each traces (WEB and EDONKEY) are always very well classified by the statistical classifier. Results on a per application basis are however much more contrasted. Per application recall and precision are presented in Figures 5.2 and 5.3 for set A and B respectively (results for R-III are omitted as they are similar to the ones of R-II).

The main observation we make is that there exist two broad families of classes. The first family features both a high recall and precision for all traces. It contains the following classes: WEB, EDONKEY, BITTORRENT, GNUTELLA, CHAT, FTP, MAIL and OTHERS (GNUTELLA and OTHERS classes have lower recall for some traces but the results are still reasonably good).

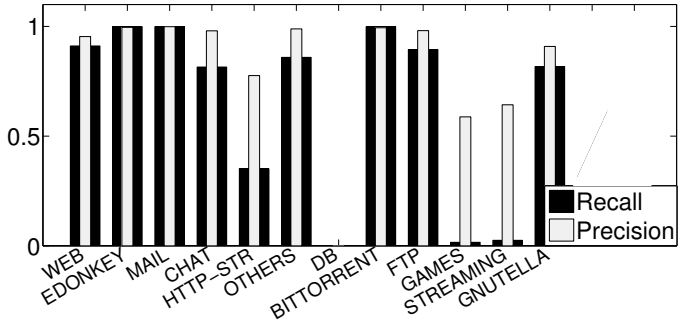
The second family of classes is characterized by a high precision but a low recall. This means that in such a class, one finds mostly correctly classified flows, but a large fraction of the flows that should be in this class, have been classified elsewhere. This is the case for GAMES, STREAMING and HTTP-STREAMING. In order to better understand the problem of those poorly performing classes, we use the confusion matrix (see Figure 5.4 obtained for set A).



(a) MS-I



(b) R-II

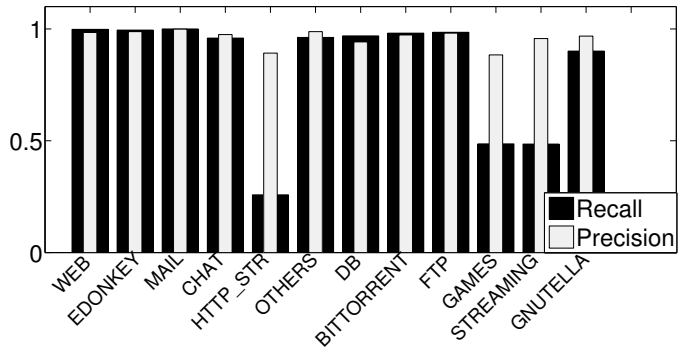


(c) T-I

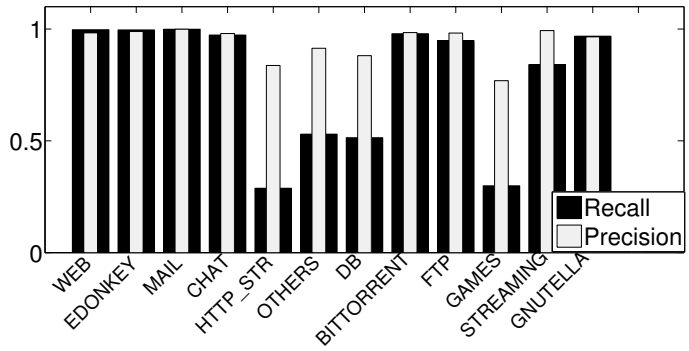
Figure 5.2: Recall and precision using packet sizes (set A) for static case.

To keep the figure clear we indicate only the misclassification higher or equal to 2%. We found that for the case of HTTP-STREAMING, almost all misclassified flows fall into the WEB class, which is understandable as it might be difficult to discriminate between a http streaming and a Web browsing transfer. In contrast, Webmail and HTTP-file transfers, are correctly classified in the WEB and FTP class respectively. This outlines that the application semantics is more important than the lower level protocols in those cases. This is especially important for the case of HTTP as it becomes a bearer for more and more diverse applications.

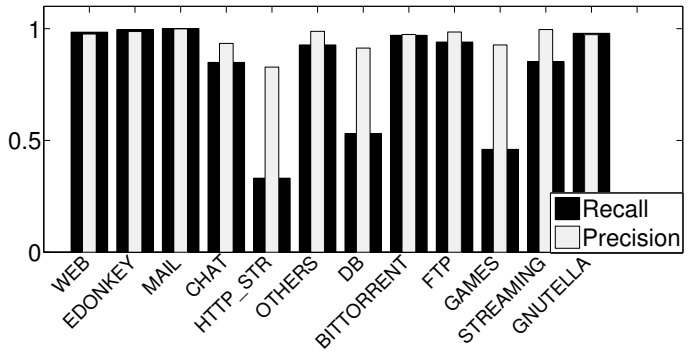
For the case of GAMES and STREAMING, misclassified flows are scattered mostly in the WEB and EDONKEY classes. For the case of GAMES, we note that this class aggregates applications with widely different behaviors. This heterogeneity might explain the difficulties faced by the statistical classifier. This observation is further backed by the fact that classification performance are poor for both features sets that we use – see Figures 5.2 and 5.3.



(a) MS-I



(b) R-II



(c) T-I

Figure 5.3: Recall and precision using set B for static case.

5.1.3 Static Results - Discussion

Results of statistical classification per site are in line with the current knowledge about the state of the art flow features classifiers. Using both set of features we obtained good results for most application classes. However, we would like to assess feasibility of statistical classification as a stand-alone solution not accompanied by any DPI tool. In such a scenario static experiment as we have just presented is not sufficient. We need to verify if the model built over one site is representative enough to be applied on other sites (portability). We discuss this issue in the next section.

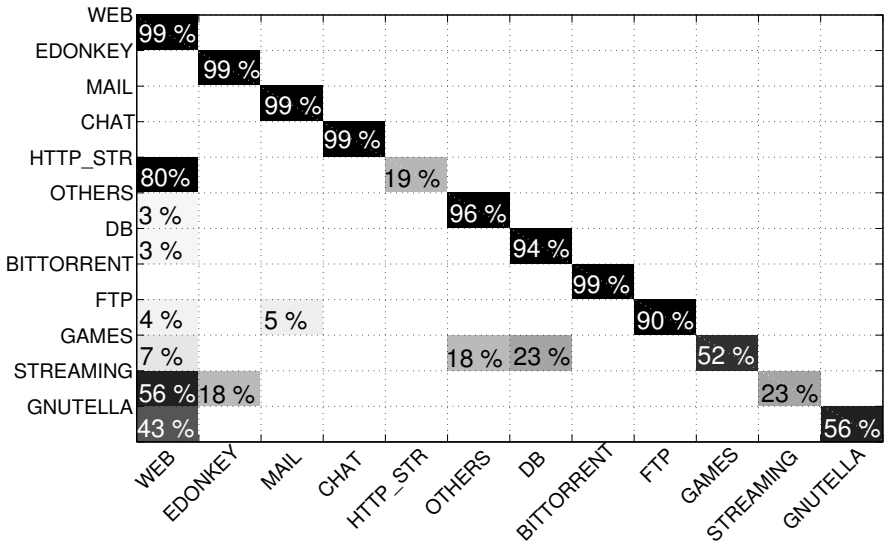


Figure 5.4: Confusion matrix for MSI trace, features set A (class considered on Y axis is classified as classes on X axis).

5.2 Classification - Cross Site

In this section, we address the problem of training a classifier on one site and then applying it to an other. Such a technique could be useful for an ISP that would deploy some deep packet inspection tool on one of its major PoPs, train a statistical classifier there and then apply it to its other PoPs. As in the static case, we will first look at the overall performance of the classifier, which means that we focus on the dominant classes. In a second stage, we will detail results per application to illustrate the main outcome of this section, namely the overfitting problem faced by statistical classifiers in cross-site studies.

5.2.1 Overall Results

In Figure 5.6, we present the overall recall obtained using one trace as a training set (on the y axis) and the others as test sets (on the x-axis). The left matrix corresponds to the use of set A (packet sizes) while the right matrix correspond to set B (flow features). Results are qualitatively similar: the overall recall is in general high for the two feature sets, though not as large as in the static case - see Figure 5.2. The more pronounced degradation is when the T-I trace is considered (as a training or test trace). This might be due to the fact that this trace is older (Dec. 2006) than the other ones. Let us now dig into the details of each class for each different feature sets.

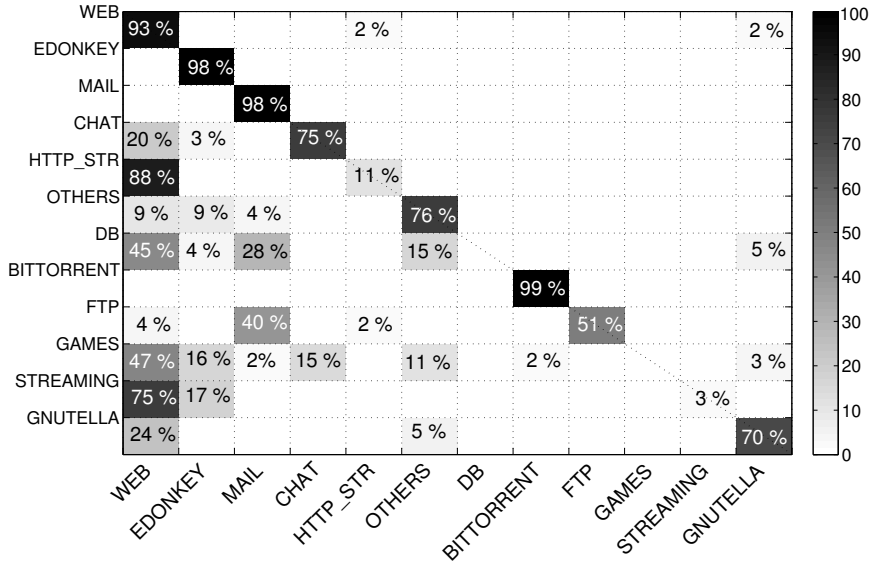


Figure 5.5: Confusion matrix for TI (training) on MSI (testing) (class considered on Y axis is classified as classes on X axis).

5.2.2 Detailed Results for Set A (packet sizes)

Let us now dig into the details of each class. We focus in this section on the case where the first feature set (set A) is used. Figure 5.7 depicts the per class recall[1] in the cross-site process. Note that we provide results only for the classes that performed well (high recall and precision – See Figures 5.2 and 5.3) in the static case: WEB, BITTORRENT, CHAT, FTP, MAIL, EDONKEY, GNUTELLA and OTHERS.

A first striking result is that EDONKEY appears immune to performance degradation in a cross-site context[2]. This it not the case for the other classes, even if most of the problems seem to stem from the T-I trace (older trace). This is however not the only explanation behind the observed degradations as there are also problems with BITTORRENT, GNUTELLA, FTP and OTHER classes for the three traces captured in 2008 (See Table 4.1).

As indicated in Section 4.1.1, we have two interesting pairs of traces in our dataset. R-II and R-III have been captured on the same site while MS-I and R-III were captured simultaneously. We do observe from Figure 5.7 that spatial similarity seems more important than temporal similarity. Indeed, for R-II and R-III results are consistently good: over 95% for all classes except OTHERS, which is at 85%. However, the latter class is a potpourri class and we are not certain of having an homogeneous set of applications for this class in the two traces. The picture is different when we focus on MS-I and R-III, as here results can degrade significantly. For FTP, recall falls to 52% when MS-I is used as a

[1]Please note that Figures 5.6 and 5.7 use different color scales.

[2]Note that the 99% recall in cross-site case comes from the fact that size of some packets for each eMule transfer is deterministic.

training trace and R-III as a test trace (and 69% for the other way around). This is in clear contrast with the static case where the recall was above 90% for the two traces.

We further investigated the case of FTP that seems extremely surprising. We picked on purpose one of the worse performing cases (T-I against MS-I) in order to highlight the problem. While the T-I trace is older, our focus is on FTP and there is no reason to believe that its fundamental characteristics have changed between the end of 2006 and the beginning of 2008. The confusion matrix is a useful tool to pinpoint problems. Figure 5.5 presents the confusion matrix for the case of training over T-I trace and testing over MS-I. We observe that a significant fraction of FTP is categorized as MAIL. It turns out that the root of this problem is that the distribution of packet sizes on different sites for FTP and MAIL classes sometimes overlap. We made similar observations for other cases where a significant degradation was observed from static to cross-site case. We will study this issues in details in chapter 7.

Confusion matrix (Figure 5.5) shows that misclassification take place for almost all traffic classes. In most cases we observe significant bias toward most popular classes, namely EDONKEY and WEB. Some applications are also confused with MAIL (like the FTP case discussed above) and OTHERS.

We face here the data set over fitting problem, when **the classifier learns trace specific characteristics that not necessarily generalize well.** One might argue that the over fitting problem we have highlighted is directly related to the feature set we use. This is however not the case as we will exemplify in the next section with our second set of features.

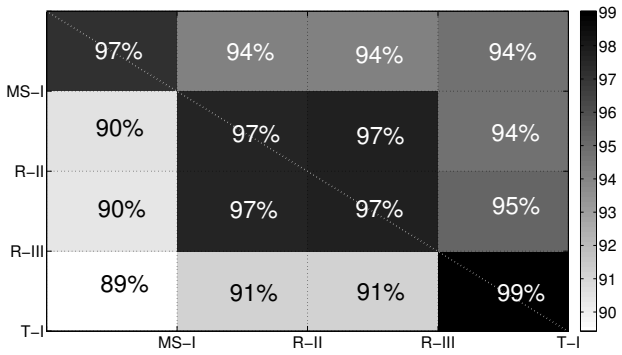
5.2.3 Detailed Results for Set B (advanced statistics)

Similarly to the case of set A, we observed significant degradations during our cross-site study with set B. For instance, the CHAT or BITTORRENT classes perform well in the static case but significantly degrade in cross-site studies. Set B consists of several features, each of them being a potential source of data over fitting. We will analyze precisely feature stability in chapter 7. Here we will present the problem focusing on one feature, namely port number, for which data over fitting is easy to explain.

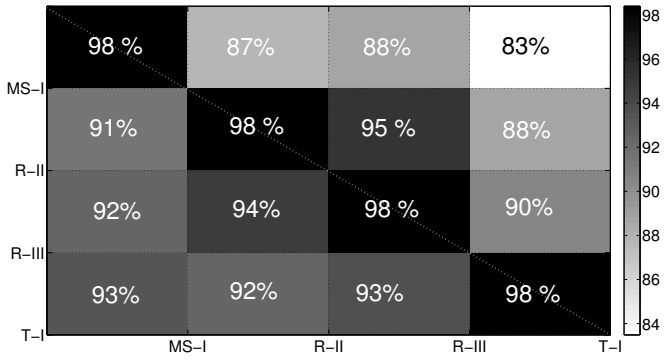
5.3 Impact of the Port Numbers

It has been claimed in a number of studies [58, 62] that ports have high predictive power and thus should increase classification recall. The use of port number is however puzzling as it is often treated as a quantitative (numeric) and not qualitative (category) value. Indeed, most classification algorithms make use of similarity metrics (distances) among the features of the different samples, and from this perspective, port 80 is closer to port 25 than to port 443 or 8080.

To gain a better understanding of the impact of the port number, we applied our second set of features *with* and *without* the port number on the static and



(a) Set A - Sizes of packets



(b) Set B - flow features

Figure 5.6: Cross site overall recall (training trace on Y axis, test trace on X axis).

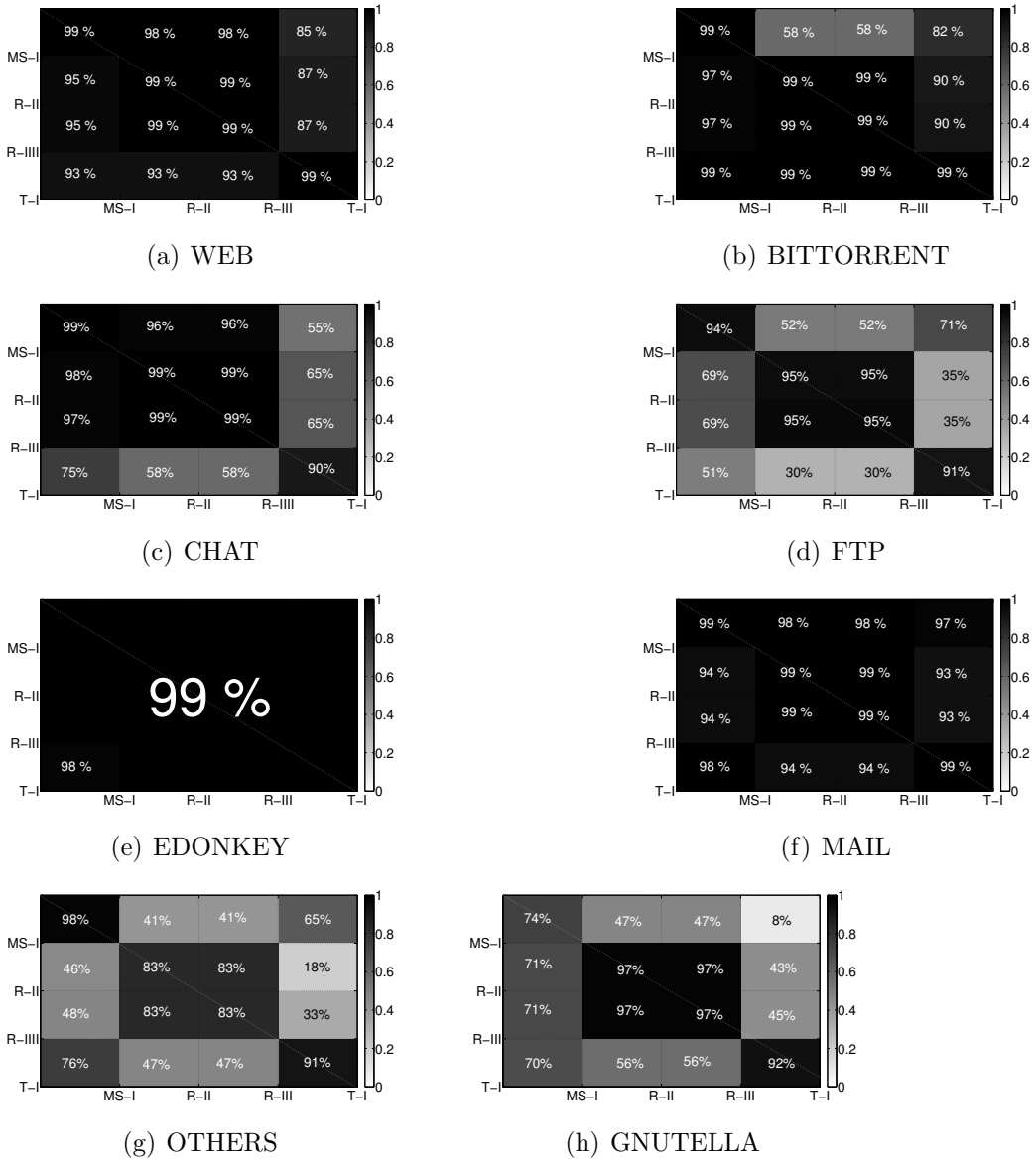
cross-site cases. We detail these two cases below.

5.3.1 Impact of the Port Numbers - Static Case

In all static cases, including port numbers increases both recall and precision, typically by a few percent in the case of p2p applications to as much as 38% in the case of FTP class. Let us detail the results for WEB and p2p classes:

- The WEB class is almost unaffected, i.e., ports have minor impact on this class. This is good news given that Web use widely different ports, esp. 80, 443, and 8080.

- Recall and precision of p2p classes, especially the EDONKEY class, are significantly increased when using the port number, even though we observed that the legacy ports of those applications are rarely used: 18 to 40% of the flows for EDONKEY and at most 16% for BITTORRENT.



(a) WEB (b) BITTORRENT

(c) CHAT (d) FTP

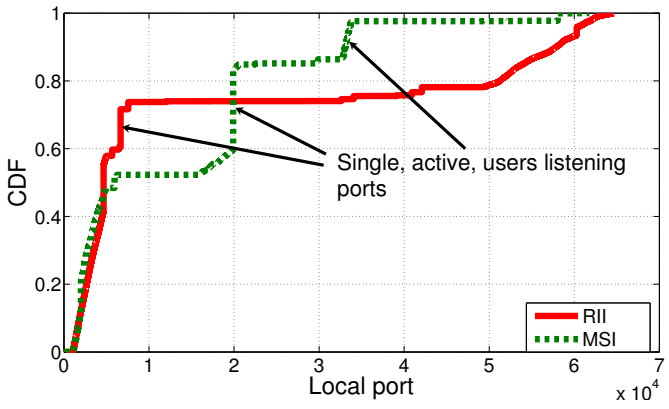
(e) EDONKEY (f) MAIL

(g) OTHERS (h) GNUTELLA

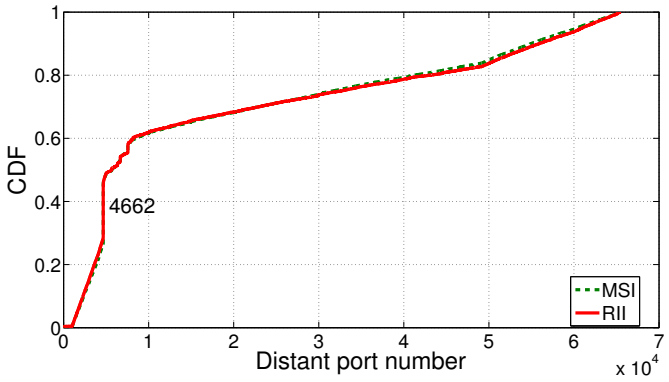
Figure 5.7: Cross site recall per application using packet sizes (training trace on Y axis, test trace on X axis).

5.3.2 Impact of the Port Numbers - Cross Site Case

In a cross-site study, using the port number is detrimental, especially for p2p traffic. In fact, in the static case, when the port number is used, the classifier learns particular non legacy port numbers of users. They are predictive in the static case, but misleading in the cross-site case because the non legacy port numbers are not the same between two sites. This is illustrated by Figure 5.8 for the MS-I and R-II traces (that were captured two weeks apart). We observe that the distribution of remote port numbers is very similar for both traces (Figure 5.8(b)) while the distribution of local ones clearly differ (Figure 5.8(a)). The former was to be expected due to the way p2p networks work. As for the latter, it is partly due to some heavy-hitters, i.e. local clients that generate a lot of transfers using e-Donkey. The presence of heavy-hitter being a known and global phenomenon, we can expect to observe a similar phenomenon irrespectively of



(a) Local port



(b) Distant port

Figure 5.8: Ports for EDONKEY MSI and RII.

the actual size of a PoP. To sum up, the port number, although it has a strong predictive power, must be *used with caution*, as we might run into the problem of over fitting the data. This issue is clearly related to the current usage of p2p applications.

5.4 Impact of the Algorithm

So far, we have considered a single machine learning algorithm, namely C4.5 and different features sets. In this section, we address the other dimension of the problem, namely the impact of the classification algorithm. We consider two alternatives to C4.5: Naive Bayes with kernel estimation and Bayesian Network. As we will see shortly, the issues described in the previous sections persist and can be even more pronounced with these algorithms.

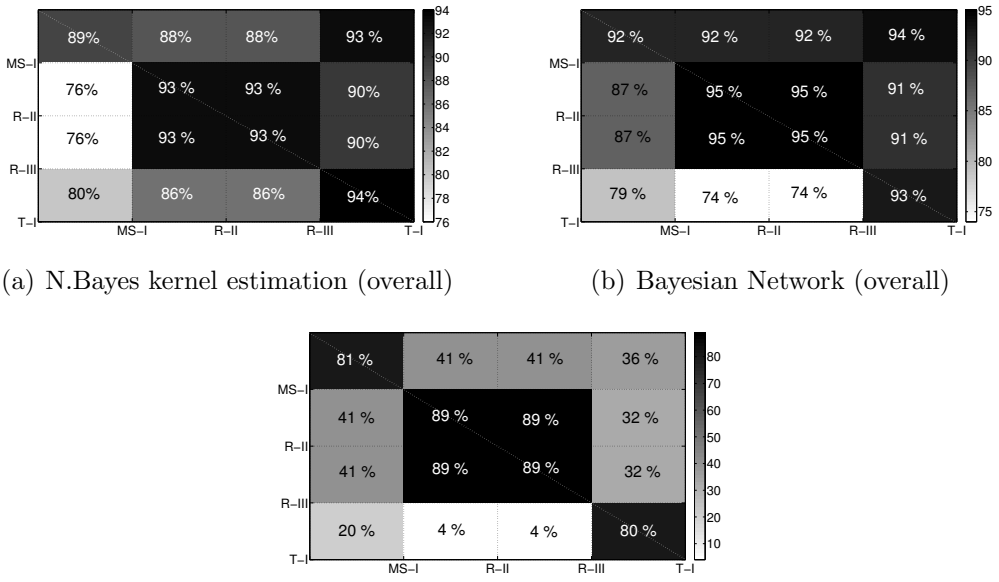
In Figures 5.9(a) and 5.9(b) we depict the overall recall for both algorithms considered using set A. While using C4.5 for the cross-site studies, we observed that the FTP case turned out to be a complex one. In Figure 5.9(c), we present recall for FTP using Bayesian Network. Detailed, per application, results are omitted for the sake of clarity. From those figures we conclude that:

- In almost all cases C4.5 performs the best in terms of overall recall in

both static (diagonal elements) and cross-site experiments (non diagonal elements).

- Degradation of overall recall for Naive Bayes with kernel density estimation and Bayesian Network in cross-site cases is similar or higher (17 % in the worse case) than with C4.5.
- Per application recall degradation, can be even more pronounced for Naive Bayes with kernel density estimation and Bayesian Network than with C4.5. We also observed issues with the same classes of applications (e.g., FTP) that caused problems for the decision tree.

Those results confirm our previous findings. The data overfitting issue turns out to be a complex problem that apparently persists when one varies the features set or the machine learning algorithm.



(a) N.Bayes kernel estimation (overall)

(b) Bayesian Network (overall)

(c) Bayesian Network (FTP)

Figure 5.9: Cross site recall for other algorithms (features A, training trace on Y axis, test trace on X axis).

5.5 Discussion

The main lesson from this cross site study is that although the degradation in terms of overall recall is often acceptable, some classes, that work correctly in the static case, might suddenly degrade.

We have demonstrated that data over fitting is at the root of the problem. To the best of our knowledge, such a phenomenon was never pointed out before. From this point on, the conclusion is twofold. On one hand, it shows that training a classifier on one site before running on other can lead to unpredictable results. On the other hand, it shows that cross site studies allow us to pinpoint problems that can not be observed otherwise.

A last conclusion suggested by our results is that once a classifier has been trained on a site, it can be used for a significant period of time on this site (as the results between traces RII/RIII are consistently good). However, more work needs to be done to validate this observation that we made for two traces collected 2 weeks away on the same PoP[3].

We will analyze the root cause of the cross site issue pointed out in this chapter in the chapter 7.

[3]We do not address this issue in this thesis

Chapter 6

Methods evaluation - Principal Components

In previous chapter we observed a degradation of performance of statistical classifier based on two state of the art feature sets (set A and set B) for cross site scenarios.

This chapter evaluates an orthogonal approach. Starting from a large set of flow features, we first apply a dimensionality reduction method (PCA - Principal Component Analysis), and use principal components as flow features. The data sets in use and the rest of the procedure remains unchanged.

This strategy aims at capturing the discriminative information from a large set of flow descriptors without increasing computational complexity due to a large set of features used. We test a superset of features proposed in previous chapters, and provide both single site and cross site results.

6.1 PCA

Principal component analysis [50] seeks to maximize the variance of uncorrelated linear combinations of original variables, called principal components. If a small number of principal components explains a large proportion of the total variance of the p original variables, then PCA can successfully be used as a dimension reduction technique. Moreover, the analysts are also interested in the interpretation of the principal components, i.e., the meaning of the new directions where the data is projected.

Given a set of n observations (flows) on the random variables $X_1, X_2, \ldots, X_p$, (p being number of features in use), the k-th principal component (PC k) is defined as the linear combination,

$$Z_k = \alpha_{k,1}X_1 + \alpha_{k,2}X_2 + \ldots + \alpha_{k,p}X_p, \tag{6.1}$$

such that the loadings of Z_k, $\boldsymbol{\alpha}_k = (\alpha_{k,1}, \alpha_{k,2}, \ldots, \alpha_{k,p})^t$, have unitary Euclidean norm ($\boldsymbol{\alpha}_k^t\boldsymbol{\alpha}_k = 1$). Z_k has maximum variance and PC k, $k \geq 2$, is uncorrelated with the previous PCs, which in fact means that $\boldsymbol{\alpha}_i^t\boldsymbol{\alpha}_k = 0$, $i = 1, \ldots, k-1$. Thus, the first principal component is the linear combination of the observed

variables with maximum variance. The second principal component verifies a similar optimal criterion and is uncorrelated with PC 1, and so on. As a result, the principal components are indexed by decreasing variance, i.e., $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_s$, where λ_i denotes the variance of PC i and $s = \min(n, p)$ is the maximum number of PCs that can be extracted (in the present chapter $n > p$).

It can be proved [50] that the vector of loadings of the k-th principal component, $\boldsymbol{\alpha}_k$, is the eigenvector associated with the k-th highest eigenvalue, λ_k, of the covariance matrix of $X_1, X_2, \ldots, X_p$. Therefore, the k-th highest eigenvalue of the sample covariance matrix is the variance of PC k, i.e. $\lambda_k = \mathrm{Var}(Z_k)$.

The proportion of the total variance explained by the first r principal components is

$$\frac{\lambda_1 + \lambda_2 + \ldots + \lambda_r}{\lambda_1 + \lambda_2 + \ldots + \lambda_p}. \tag{6.2}$$

If this proportion is close to one, then there is almost as much information in the first r principal components as in the original p variables. In practice, the number r of considered principal components should be chosen as small as possible, taking into account that the proportion of the explained variance, (6.2), should be large enough.

Once the loadings of the principal components are obtained, the score of flow i on PC k is given by

$$z_{ik} = \alpha_{k1} x_{i1} + \alpha_{k2} x_{i2} + \ldots + \alpha_{kp} x_{ip}, \tag{6.3}$$

where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^t$ is the i-th (multivariate) observation, i.e., a collection of measurements on p different variables made on flow i.

PCA being scale dependent, we need a pre-treatment of the variables. In some cases the variability of the data can be dominated by the variable with largest scale, which will make interpretation of the results harder. We thus apply logarithmization of variables to avoid this issue.

6.2 Description of the Flow Features

In this section we present a set of over eighty features that are used to create the Principal Components. The main types of the features are provided in the Table 6.1. Features names are prefixed with DIP which is a name of the internal tool used to extract the statistics. A brief description of the main types of features grouped in two subsets is provided below.

We use two kinds of metrics: metrics in the first group characterize the whole connection while metrics in the second group are measured for each packet and then statistics are computed over the connection. Almost all these metrics are defined both upstream and downstream for each flow, except for the duration which is considered equal in both directions.

DIP-Duration-Sum-MilliSeconds-Effective. It is the duration between the first and the last packets of the connection, whatever their direction is, removing all the silences between any two packets of the flow when they are larger

Num	Abbreviation	Description
1	DIP-FlowPortSource	TCP local port
2	DIP-FlowPortDest	TCP distant port
3	DIP-Duration-Sum-MilliSeconds-Effective	Effective flow duration (silence longer 5s removed)
4	DIP-Volume-Number-Packets-Down	Number of packets down
5	DIP-Volume-Number-Packets-Up	Number of packets up
6	DIP-Volume-Sum-Bytes-Down	Sum of bytes up
7	DIP-Volume-Sum-Bytes-Up	Sum of bytes down
8	DIP-Thp-Number-Kbps-Down	Volume down/Flow duration
9	DIP-Thp-Number-Kbps-Up	Volume up/Flow duration
10	DIP-Thp-Number-Kbps-Down-Effective	Volume down/Flow duration effective
11	DIP-Thp-Number-Kbps-Up-Effective	Volume up/Flow duration effective
12	DIP-PushCounter-Up	Number of packets with push flags up
13	DIP- PushCounter-Down	Number of packets push flag down
14	DIP-Min-Segment-Size-Bytes-Up	Smallest TCP segment up
15	DIP-Min-Segment-Size-Bytes-Down	Smallest TCP segment down
16	DIP-IPT-Mean-ms-Up	Inter arrival time between packets up [ms]
17	DIP-IPT-Mean-ms-Down	Inter arrival time between packets down [ms]
18	DIP-PS-Mean-o-Up	Packet size mean up (whole flow)
19	DIP-PS-Mean-o-Down	Packet size mean down (whole flow)
20	DIP-TPM-Mean-Mbps-TCP-Up	Mean throughput over RTT up
21	DIP-TPM-Mean-Mbps-TCP-Down	Mean throughput over RTT up
22	DIP-TPI-Mean-Mbps-TCP-Up	Packet size divided by IPT up
23	DIP-TPI-Mean-Mbps-TCP-Down	Packet size divided by IPT down
24	DIP-DSQ-NbMes-sec-TCP-Up	IPT between out of sequence packets up
25	DIP-DSQ-NbMes-sec-TCP-Down	IPT between out of sequence packets down
26	DIP-RTM-NbMes-sec-TCP-Up	IPT between two retransmitted packets up
27	DIP-RTM-NbMes-sec-TCP-Down	IPT between two retransmitted packets down
28	DIP-Volume-Number-Segments-Up	Number of data segments up
29	DIP-Volume-Number-Segments-Down	Number of data segments down
30	DIP-Payload-Number-Bytes-Up	Total number of payload bytes up
31	DIP-Payload-Number-Bytes-Down	Total number of payload bytes down
32	DIP-data-pkt1-size	Size and direction of first payload packet
33	DIP-data-pkt2-size	Size and direction of second payload packet
34	DIP-data-pkt3-size	Size and direction of third payload packet
35	DIP-data-pkt4-size	Size and direction of fourth payload packet

Table 6.1: Per flow features (main types).

than five seconds. It is especially important to remove these silences for short transfers as it has been observed in [39] that in this case the TCP tear-down delay represents most of the duration of the connections.

DIP-Volume-Number-Packets-* and DIP-Volume-Sum-Bytes-* give respectively the size in packets and the volume in bytes of the connection. The metrics are observed by the probe and retransmitted packets are then counted twice if they have been lost after the probe, i.e., between the probe and the distant host for upward transfer and between the probe and the local host for downstream transfer. The volume in bytes takes also into account the TCP/IP headers so the size of acknowledgments is not null. Headers of lower layers (Ethernet on our traffic traces) are not taken into account. It should be noted that most of the connections use TCP options and the size of the TCP/IP header is then usually larger than 40 bytes.

DIP-Thp-Number-Kbps-* Is the volume of the connection in kilobits (DIP-Volume-Sum-Bytes-*) divided by the total duration of the connection, without removing the silences, while **DIP-Thp-Number-Kbps-*-Effective** is computed with DIP-Duration-Sum-MilliSeconds-Effective.

DIP-Push-counter-* Is the number of packets with TCP Push flag set.

Min-Segment-Size-* - The smallest TCP segment size in bytes.

Payload-Number-Bytes-* - Sum of all the bytes of the payload (headers excluded).

The second group of metrics are measured each time a packet is detected by the probe. Many statistics are then updated based on the history over of the packets of each connection. For many types of metrics we consider a number of derived statistics, the minimum and the maximum values (Min and Max), the mean and the median (Mean and Dec5), the standard deviation and the coefficient of variation (StdDev and CoefVar), and the number of observations (NbMes). Thus each of the metrics 16-23 presented in Table 6.1 has a number of variants (min/max/stddev/coefvar/mean/dec5/NbMes) which we omitted in the table for clarity.

DIP-PS-* Measures in each direction the size of each packet, again taking into account the TCP/IP headers but not the lower layers.

DIP-IPT-* Measures in each direction the inter-arrival time between two consecutive packets.

DIP-DSQ-* Measures in each direction the inter-arrival time between two consecutive out-of-sequence packets. We consider that a packet is out-of-sequence when its TCP sequence number is lower than the TCP sequence number of the previous packet of the same connection in the same direction. The first interval is measured from the first packet of the connection. It should be noted that if many packets are lost in a given round-trip time, only one out-of-sequence packet will be detected. For this performance criterion we only consider in this work the number of observations.

DIP-RTM-* Measures in each direction the inter-arrival time between two retransmitted packets. We consider that a packet is retransmitted when its TCP sequence number is seen twice by the probe. It must be noted that RTM takes only into account the packets lost between the probe and the destination. Furthermore, if many packets are lost (after the probe) in a given round-trip time and are retransmitted they will be all counted as retransmitted.

DIP-TPI-* Measures in each direction the instantaneous throughput: the size of the current packet (PS) divided by the inter-arrival time with the previous one (IPT). TPI is limited by the link with the lowest capacity between the source and the probe, and of course by the application on top of TCP.

DPI-TPM-* Is a measure of the mean throughput over a round-trip time; It is the ratio of the number of outstanding bytes divided by the round trip time. The round trip time is measured as the delay between the detection of a packet by a probe and the return of its acknowledgments. As the probe is on the access router the round trip time estimation measures actually the local delay for downstream traffic and the distant delay for the upstream traffic. For each acknowledgment, the number of outstanding bytes is the difference between the highest sequence number and the currently acknowledged sequence number. One of the objectives of TCP is to assess the available bandwidth on the path through this TPM. Nowadays, for most of the flows, the limitations are not inside the network, but at the application level on the server side [87]. Many methods have been proposed to compute much more precise approximations, but we have to analyze here a large number of flows and we are not interested by the

Original				Logarithm			
RII		RIII		RII		RIII	
Nr PC	% Var.	Nr PC	% Var.	Nr PC	% Var.	Nr PC	% Var.
1	(13.4%)	1	(14.3 %)	1	(23.3%)	1	(24.6 %)
2	(22.2%)	2	(23.8 %)	2	(41.9%)	2	(43.5 %)
3	(30.1%)	3	(31.7 %)	3	(52.6%)	3	(53.8 %)
4	(35.0%)	4	(36.9 %)	4	(59.4%)	4	(60.5 %)
5	(39.2%)	5	(41.0 %)	5	(65.0%)	5	(65.8 %)
6	(43.2%)	6	(44.5 %)	6	(68.8%)	6	(69.2 %)
17	(71.3%)	16	(70.2 %)	7	(71.7%)	7	(71.9 %)
20	(76.2%)	19	(75.0 %)	9	(76.0%)	9	(76.2 %)
23	(80.7%)	23	(80.7 %)	12	(81.2%)	12	(81.3 %)
27	(85.6%)	27	(85.5 %)	15	(85.0%)	15	(85.2 %)
33	(90.7%)	33	(90.8 %)	20	(90.2%)	20	(90.4 %)
41	(95.0%)	41	(95.3 %)	28	(95.0%)	28	(95.2 %)
57	(99.0%)	56	(99.1 %)	45	(99.1%)	45	(99.1 %)
71	(99.9%)	70	(99.9 %)	65	(99.9%)	65	(99.9 %)
85	(100%)	85	(100 %)	85	(100%)	85	(100.0 %)

Table 6.2: Explained percentage of the total variability (% Var.) by each number of principal componentes (Nr PC), for the two traces.

exact throughputs but only by a rough value as a parameter which may help to distinguish the applications.

6.3 Variable Logging

In Table 6.2 we present the percentage of the total variability (% Var.) explained by each number of principal componentes (Nr PC), for the two traces. We show two options: (i) with the original variables and (ii) with the log of the variables.

Apart from smoothing the discriminators, Table 6.2 demonstrates that with the log of the variables we need a much lower number of PC to explain the same amount of variability in the data. For instance to explain 80% of the variability in the data we need 12 and 23 components for the cases of loged and raw variables respectively. This is a desired feature, as we want to use as low number of PC's as possible.

6.4 Interpretation of the Principal Components

In this section we give an interpretation of the first principal components. Note that each principal component is a linear combination of the features. Thus, for each principal component, we identify the set of features that contribute most to the component. There are two criteria: choose the variables with the highest loadings or the features with the highest correlations (with the principal component). While there is some controversy whether to use one criterion or the other, in our case both approaches lead to the same set of features. Some of the results, helpful for the interpretation, are presented in Table 6.3.

PC1 of R-II - The most important types of features are Volume-Sum-Bytes, Payload-Number-Bytes, PS, Thp, Thp-Effective, TPI, and IPT, all in the downstream direction. These features are only of 3 types: size, rate and duration. Recall that we are working with the logarithms of the features and the logarithm

Id	Feature (X_i)	$\widehat{Cor}(PC_1, X_i)$	$\hat{\alpha}_{i1}$	$\widehat{Cor}(PC_5, X_i)$	$\hat{\alpha}_{i5}$
4	Volume-Sum-Bytes-Down	-0.911	-0.205		
6	Thp-Number-Kbps-Down	-0.873	-0.196		
8	Thp-Number-Kbps-Down-Effective	-0.854	-0.192		
15	IPT-Mean-ms-Down	0.721	0.162		
25	IPT-Dec5-ms-Down	0.741	0.167		
27	PS-Mean-o-Down	-0.836	-0.188		
29	PS-StdDev-o-Down	-0.789	-0.177		
35	PS-Max-o-Down	-0.781	-0.176		
39	TPM-Mean-Mbps-TCP-Down	-0.779	-0.175	-0.216	-0.099
45	TPM-Min-Mbps-TCP-Down			-0.288	-0.131
47	TPM-Max-Mbps-TCP-Down	-0.767	-0.172		
49	TPM-Dec5-Mbps-TCP-Down	-0.762	-0.171	-0.220	-0.100
50	TPI-Mean-Mbps-TCP-Up			0.306	0.139
51	TPI-Mean-Mbps-TCP-Down	-0.760	-0.171		
56	TPI-Min-Mbps-TCP-Up			0.340	0.155
57	TPI-Min-Mbps-TCP-Down			-0.246	-0.112
58	TPI-Max-Mbps-TCP-Up			0.303	0.138
59	TPI-Max-Mbps-TCP-Down	-0.747	-0.168		
60	TPI-Dec5-Mbps-TCP-Up			0.294	0.134
61	TPI-Dec5-Mbps-TCP-Down	-0.743	-0.167		
62	DSQ-NbMes-sec-TCP-Up			-0.451	-0.206
63	DSQ-NbMes-sec-TCP-Down			0.535	0.244
64	DSQ-Mean-sec-TCP-Up			-0.510	-0.232
65	DSQ-Mean-sec-TCP-Down			0.497	0.226
66	DSQ-StdDev-sec-TCP-Up			-0.490	-0.223
67	DSQ-StdDev-sec-TCP-Down			0.550	0.250
68	DSQ-CofVar-sec-TCP-Up			-0.474	-0.216
69	DSQ-CofVar-sec-TCP-Down			0.444	0.202
70	DSQ-Min-sec-TCP-Up			-0.356	-0.162
71	DSQ-Min-sec-TCP-Down			0.383	0.175
72	DSQ-Max-sec-TCP-Up			-0.520	-0.237
73	DSQ-Max-sec-TCP-Down			0.488	0.222
74	DSQ-Dec5-sec-TCP-Up			-0.490	-0.223
75	DSQ-Dec5-sec-TCP-Down			0.459	0.209
76	RTM-NbMes-sec-TCP-Up			-0.376	-0.171
81	Payload-Number-Bytes-Down	-0.730	-0.164		

Table 6.3: Highest values of the loadings and correlations associated with the first and fifth principal component of RII.

of a rate is the difference between the logarithm of size and the logarithm of duration. Thus, given that all features related with size and throughput have positive loadings and all features related with durations have negative loadings, we may interpret this PC as a measure of the global download throughput.

PC2 of R-II - The second principal component characterizes short and rather symmetric flows with small up and down volumes, small variations of up and down instantaneous throughput, but with a large upstream throughput (and also downstream to a lesser extent).

PC3 of R-II - The third principal component groups the flows with high upstream instantaneous throughput and median packet size, and small downstream (instantaneous and over a RTT) throughputs, and a few grouped upstream losses.

PC4 of R-II - The forth principal component classifies the flows with a large instantaneous upstream throughput, and a high downstream loss rate. This correlation is rather surprising. It is maybe due to saturations of the upstream buffer and loss of the acknowledgments.

PC5 of R-II - This component can be interpreted as a contrast between throughput in both directions. The most important types of features are TPM-Down, TPI-Down, DSQ-Up, with negative loadings, and TPI-Up and DSQ-Down with negative loadings. Thus high throughput values in one direction (and high losses) are accompanied by low throughput values (and low losses) in the other direction. This is a strong characteristic of asymmetric flows, which are present in many applications. This example shows clearly that principal components are able to describe a difference between two features.

PC6 of R-II - The sixth principal component concerns the flows with a small and slightly varying instantaneous and total upstream throughput, with varying upstream packet sizes and with sparse packets in both directions.

PC7 of R-II - The seventh principal component groups the flows with variable downstream packet sizes, a large instantaneous downstream throughput and small upstream throughput.

6.5 Classification Results

In this section we present the results of the classification using PC's as features. As a learning algorithm we use C4.5 introduced in Section 4.2.1. Once more we demonstrate the results in two scenarios, static case (part of the same trace used for training and testing) and the cross site case.

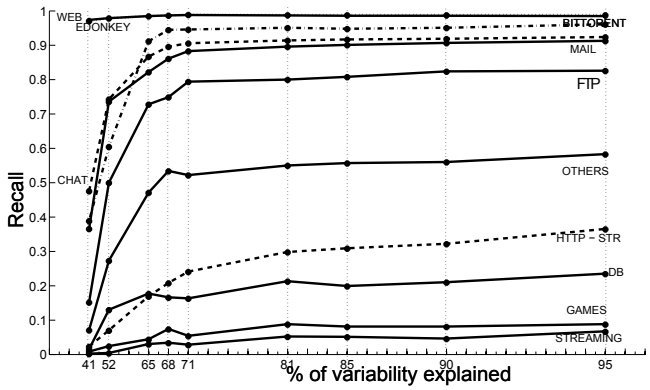
Class	Recall	Precision
WEB	94%	88%
EDONKEY	96%	96%
MAIL	85%	92%
CHAT	86%	62%
HTTP-STR	7%	4%
OTHERS	9%	22%
DB	4%	8%
BITTORRENT	39%	71%
FTP	82%	83%
GAMES	1%	4%
STREAMING	1%	9%
GNUTELLA	62%	59%

Table 6.4: Cross trace results, training using RII test using RIII.

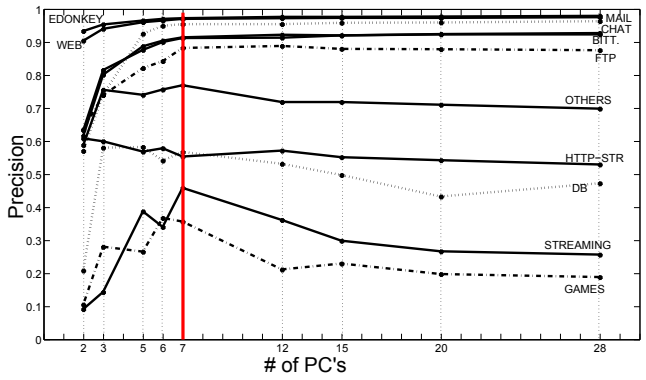
6.5.1 Static Case

Figure 6.1 presents the recall and precision of the classifier depending on the number of PC used. The results were obtained with R-II using ten fold cross validation technique. We obtained qualitatively similar results to the classifier using subset of features - Set A (see Section 5.1). Most popular applications (WEB, EDNOKEY) are well classified using small number of principal components. Also the set of misclassified applications is similar HTTP_STREAMING, DB,GAMES, STREAMING. In general the results for most applications are slightly lower than the ones obtained with raw features. We also observe that using seven Principal Components we obtain the best trade-off between the performance and the number of the flow features which should be kept low. In fact after that number we do not observe large changes in recall/precision.

In conclusion, it turns out that our technique of using large number of features reduced to principal components offers similar results to strategy of using small subset of raw features (in single site). Such result is disappointing and we need to seek for alternative approach to obtain satisfactory results.



(a) Recall versus principal components R-II



(b) Precision versus principal components R-II

Figure 6.1: Recall and precision versus principal components R-II.

6.5.2 Cross Site Case

We also evaluated the cross site results using all the combination of traces, similarly to the experiment done in Section 5.2. Examples of the scores obtained are presented in Table 6.4. In all the cases the results are much worse than the ones obtained with the raw features.

6.6 Conclusions

In this chapter we evaluated traffic classification strategy that uses principal components derived from large set of discriminators as a features. Our a priori assumption was that by combining a large set of features we might improve the discriminative properties of the classifier. The experiments that we carried out clearly demonstrated that it is not the case. It turns out that the strategy adopted introduced noise rather than a better discriminative power, and lead to degradation of the recall and precision as compared to the usage of the row features.

Although, in static case the technique provides results similar (slightly worse) than the usage of the raw features, we increase the complexity by a large factor (for each flow we need to compute over 80 features). In conclusion, the evaluated strategy did not bring the expected gains and we will seek for other techniques in the following chapters.

Chapter 7

The Root Cause of the Cross-site Issue

In chapter 5 we uncovered the of portability of statistical classifier when using State-of-the-Art algorithms and flow features. We further uncovered the role of some features (e.g. port numbers) in this issue. In Chapter 6, we revisited the problem of choosing the flow features using principal component analysis.

This chapter goes one step further in understanding the root cause of the problem, with a detailed study of the features stability between sites. To illustrate the issues we once more use the sizes of packets (set A) introduced in Chapter 4. As we will see shortly, using statistical tests, the stability of the features is highly dependent on the application, and helps explaining the reasons of the phenomena discovered in previous chapters.

7.1 Portability Problem

When addressing the stability of features across datasets, most works apply a specific feature selection method to the mix of flows from multiple data sets and choose the most common features as the stable ones. However, even if one set of features can lead to good classification results across several data sets when the classifier is trained and tested on the same dataset, it can happen that a classifier trained with one dataset obtains poor classification results when tested on another dataset. The example given in appendix A clearly illustrates this fact.

7.2 Statistical Test and Effect Size

The lesson learned from the previous section is that to ensure portability the features considered should be similar in distributions, we formalize this property below.

Given k datasets, $S_1, \ldots, S_k$, characterized by the same set of p features, we want to determine the features that have the same distribution on the k sets, i.e., we want to determine all X_i such that $X_i|S_1 \stackrel{d}{=} \ldots \stackrel{d}{=} X_i|S_k$, $i = 1, ..., p$, where

equality is in the distributional sense[1]. In order to test the equality of the flow features distributions, we adopt a procedure with three steps:

1. Test equality of means, i.e., $H_{0a} : E(X_i|S_1) = \ldots = E(X_i|S_k)$ versus $H_{1a} : \exists j \neq l : E(X_i|S_j) \neq E(X_i|S_l)$. A feature not rejecting this hypothesis is said *mean*-stable.

2. If H_{0a} is not rejected, test equality of variances, i.e. $H_{0b} : Var(X_i|S_1) = \ldots = Var(X_i|S_k)$ versus $H_{1b} : \exists j \neq l : Var(X_i|S_j) \neq Var(X_i|S_l)$. A feature not rejecting this hypotheses is said *variance*-stable.

3. If both H_{0a} and H_{0b} are not rejected, do graphical comparisons based on Q-Q plots, boxplots, and density kernel estimation to obtain evidence that the distributions are equal.

Step 1 can be accomplished using analysis of variance (ANOVA) with one factor and k levels [69] and step 2 through the Levene test [10]. In the special case of $k = 2$ the statistical tests have a simpler form: step 1 is a t-test and step 2 an F-test. Step 3 prevents the case of different distributions with the same expected value and variance being considered stable.

7.2.1 Effect Size

The first two steps of the method described above involve performing statistical tests. However, it is known that the use of statistical tests must be exercised with care[2]. In particular, Pires and Branco [7] have shown that the null hypothesis will always be rejected for large enough sample sizes, even if there is strong evidence from the data that it should not be rejected. This is an important and common problem that is not very often discussed in the literature, and can seriously affect Internet traffic studies.

One possible way to overcome this problem is to rely on the so-called *effect size*. The effect size is a family of indices that measures the strength of the result established by the null hypothesis, associated with a certain statistical test. These indices are enjoying an increasing acceptance in various fields, such as medicine, behavioral sciences and psychology.

Effect size is being used essentially for two purposes: as a complement to hypotheses testing and to determine appropriate sample sizes for these tests. For example, Nix and Barnette [72] argue that when comparing two means, reporting p-values only indicates that a difference should not be attributed to chance, while effect size assess a quantitative measure of the difference, in a scale tied to the real world, and that these measures lead to a more complete understanding and interpretation of the results, assessing the practical significance (meaningfulness) of findings.

[1] Strictly speaking the stability problem is a joint distributional one, as statistical classifiers operate jointly on a set of features. However, finding a statistical test for joint distributions is too complex.

[2] *Some hesitation about the unthinking use of significance tests is a sign of statistical maturity*, David Moore, Statistics: Concepts and Controversies, 4th edition, 1997, Freeman.

Taking into account the findings of [7], we propose the following methodology: first perform the statistical test and then, if the null hypotheses is rejected, evaluate the effect size, to assess if the rejection was due to meaningful differences between the features or the impact of the sample size. The null hypothesis will only be definitely rejected if the effect size estimate is not considered small.

The effect size for comparing two means is defined as $d = (\mu_1 - \mu_2)/\sigma$, where μ_i is the expected value of the feature in i-th population and σ^2 is the variance, considered equal in the two populations, a usual assumption of the t-test. Several indices for estimating this effect size have been proposed in the literature [18, 43]. The major difference among them is in the way σ^2 is estimated. We use the Hedges's effect size,

$$g = \frac{\bar{y}_1 - \bar{y}_2}{s_p} \tag{7.1}$$

where $\bar{y}_i$ is the sample mean of the feature in S_i, $i = 1, 2$ and s_p^2 is the sample pooled variance, i.e.

$$s_p^2 = \frac{s_1^2\,(n_1 - 1) + s_2^2\,(n_2 - 1)}{n_1 + n_2 - 2}, \tag{7.2}$$

s_i^2, and n_i are the sample variances and the samples sizes of the datasets S_i, $i = 1, 2$. For example, $g = 0.25$ indicates that the difference between the means is one-fourth of the pooled standard deviation. For the comparison between two variances, we consider as effect size

$$\Delta_\sigma = \frac{s_1^2}{s_2^2}. \tag{7.3}$$

Different areas of application and authors suggest different thresholds for what is meant by small, medium and large effects. For example, Cohen [18] proposed the value 0.2, 0.5 and 0.8, for Cohen's d: $d = (\bar{y}_1 - \bar{y}_2)/s$. Cohen did not explicitly define how s^2 was calculated, but is commonly accepted to use the maximum likehood estimator for σ^2, considering that the two population have the same variance [43], thus $g = \sqrt{(n_1 + n_2 - 2)/(n_1 + n_2)}\,d$. The thresholds suggested by Cohen were based on extensive work on areas like psychology and behavioral sciences. Even though, authors like Nix and Barnette [72] argue that "*he pretty much arbitrarily chose three values that had been used extensively as standards for effect sizes*". Nevertheless, Cohen [18] also recommended precautions in the use of this rule of thumbs blindly in other areas of application. In our work, and given the lack of history about the use of these indexes in Telecommunications, we consider the threshold suggested by Cohen, i.e. when comparing two means, an effect size below 0.20 is considered small. Note that, for large sample sizes the correction of Hedge does not differ much from Cohen's d, so we use the value 0.20, without any correction.

When comparing k means, several measures of effect size have also been proposed. The most common ones are: η^2, partial η^2, and ω^2. These measures have a different interpretation. Given a certain variable measured in different conditions (from different traces) these indices provide an indication of the proportion of

variance that can be attributed to the trace (treatment). In the case of ANOVA, with one factor, η^2 and partial η^2 are equal. The η^2 is known to be biased thus ω^2 was proposed to reduce this bias [42]. It is known that the bias decreases with the sample size, so in our cases we consider both:

$$\eta^2 = \frac{SS_{Treatment}}{SS_{Error}},$$

where $SS_{Treatment} = \sum_i^k n_i(\bar{y}_{i\bullet} - \bar{y}_{\bullet\bullet})^2$, $SS_{Error} = \sum_i^k \sum_j^{n_i}(y_{ij} - \bar{y}_{i\bullet})^2$, $\bar{y}_{i\bullet} = \sum_j^{n_i} y_{ij}/n_i$, $\bar{y}_{\bullet\bullet} = \sum_{i=1}^k n_i \bar{y}_{i\bullet}/N$, n_i is the number of observation in set S_i, $N = \sum_{i=1}^k n_i$ is the total number of observations, and y_{ij} corresponds to the value on the feature on j-th flow from set S_i, $j = 1, \ldots, n_i$ and $i = 1, \ldots, k$. The ω^2 effect size is defined as:

$$\omega^2 = \frac{SS_{Treatment} - (k-1)MS_{Error}}{SS_{Total} + MS_{Error}},$$

where $SS_{Total} = SS_{Treatment} + SS_{Error}$ and $MS_{Error} = SS_{Error}/(N-k)$. Both measures lead to similar results, and because of that only results referring to ω^2 are reported. What is meant by small-medium-large effect, in this context, changes as well. For η^2 and ω^2 the thresholds usually recommended for small, medium and large effect size are 0.010, 0.059 and 0.138, respectively [59].

The definition of effect size when comparing k variances is not so straight forward. But we can take advantage of the fact that Levene's test is based on the ANOVA statistic applied to absolute deviations of observations from the corresponding group mean [10]. Another version of this test can be obtained considering ANOVA applied to the absolute deviations of observations from the group trimmed mean instead of the group means. The estimates of the effect sizes as well as the thresholds proposed previously, can then be used.

In this chapter, we will study the stability of features, per application, among three traces. After assessing mean-stability (per application) among the three traces, we make two by two comparisons, once again on a per application basis. This allows us to discuss which are the stable features for each application when comparing two traces, and will give explanations for the performance of the C4.5 classifier, when trained with one trace and used to classify flows from another trace.

7.3 Stability Analysis

In this section we analyze the impact of the stability of the flow features (the sizes of the first 4 packets of a TCP connection, Set A) in the cross performance results presented in previous chapters. We then try to explain the stability of these features by the protocol analysis of the most frequent behaviors of the considered applications. The analysis in this section is based on a log-transformation of the data, which has the merit of smoothing extreme values, down weighting the impact of possible outliers. More precisely, if y is an observed packet size then $x = sgn(y)ln(|y| + 1)$ is the value considered, where $sgn(y) = 1$ if $y > 0$ and

Application	pkt1	pkt2	pk3	pk4
FTP	0.051	0.025	0.037	**0.008**
BITTORENT	0.101	0.076	0.022	0.013
CHAT	0.019	0.038	0.041	0.023
WEB	**0.003**	**0.000**	0.013	**0.001**
EDONKEY	**0.003**	**0.005**	**0.002**	**0.000**
MAIL	0.036	0.030	**0.006**	0.012
OTHERS	0.026	0.104	0.030	0.151

Table 7.1: ANOVA effect size, ω^2 (Mean-stable features in bold.)

Application	MS-I/R-II				MS-I/R-III				R-II/R-III			
	pkt1	pkt2	pk3	pk4	pkt1	pkt2	pk3	pk4	pkt1	pkt2	pk3	pk4
FTP	-0.53	-0.33	-0.4	**-0.19**	**-0.19**	-0.29	**0.06**	**-0.18**	0.36	**0.07**	0.44	**0.03**
BITTORENT	0.87	-0.8	0.57	0.34	0.49	-0.26	0.4	**0.10**	-0.52	0.48	**0.14**	**-0.20**
CHAT	-0.25	0.3	0.29	0.23	**0.04**	**-0.16**	**-0.18**	**-0.12**	0.29	-0.42	-0.46	-0.34
WEB	**-0.11**	**0.03**	0.24	**0.07**	**-0.11**	**-0.01**	**0.20**	**0.06**	**0.00**	**-0.04**	**-0.04**	**-0.01**
EDONKEY	**-0.09**	**0.20**	**0.18**	**-0.07**	**-0.19**	0.29	**0.16**	**-0.07**	**-0.08**	**0.06**	**-0.01**	**0.00**
MAIL	-0.51	0.5	-0.23	0.29	-0.37	0.28	**-0.09**	**0.20**	**0.13**	**-0.17**	**0.12**	**-0.09**
OTHERS	-0.44	0.98	0.6	1.2	-0.4	0.74	**0.15**	0.98	**0.16**	-0.38	-0.41	-0.28

Table 7.2: Effect size from the comparisons of means, Hedge's g.

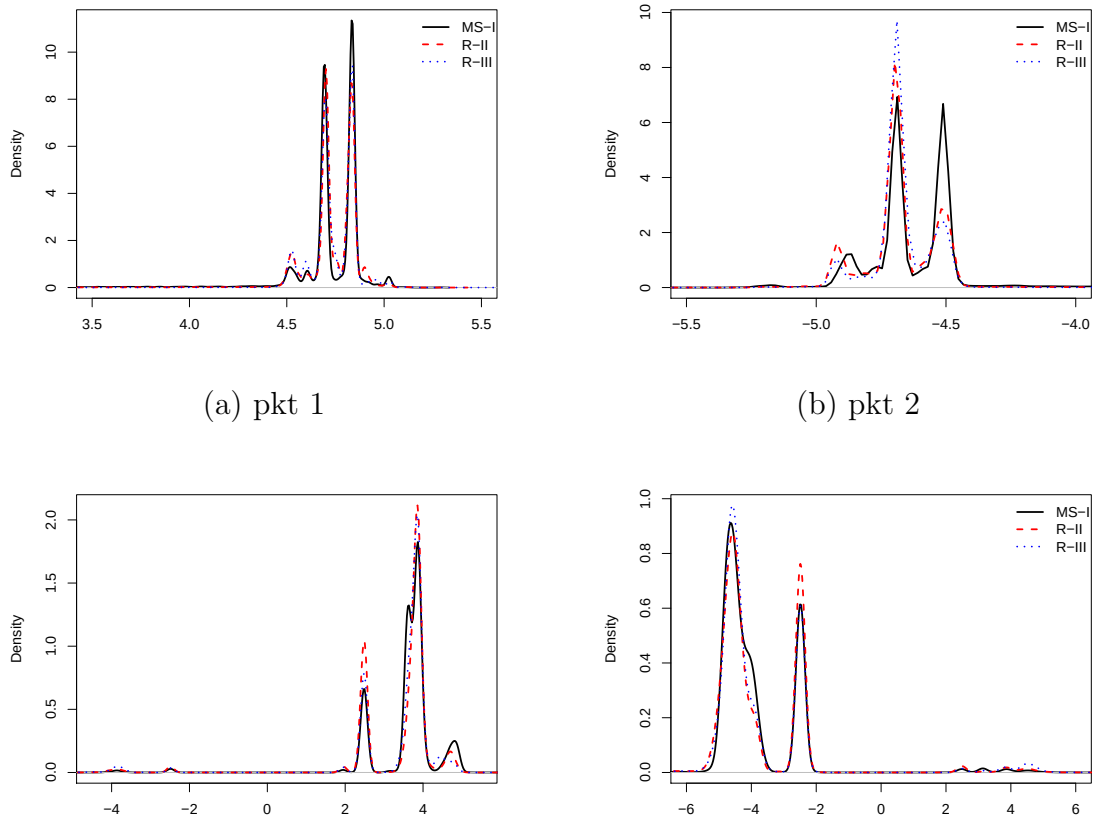
$sgn(y) = -1$ otherwise. Table 7.1 shows the effect sizes obtained with ANOVA (comparison of 3 means). The features considered mean-stable are shown in boldface. The threshold is 0.010.

7.3.1 EDONKEY and WEB

Results show that for EDONKEY and WEB all features are mean-stable, except pkt3 for WEB. The analysis of the densities of the 4 packet sizes for EDONKEY (Figures 7.1) also reveals an excellent agreement. These results confirm the high recall and precision results of the cross performance studies (chapter 5): the performance is always 99%.

Since EDONKEY and WEB are the most numerous applications, the excellent cross performance results explain why the overall performance is high. For example, in the case of R-II, EDONKEY is 40.7% of the traffic and WEB is 44.9%. However, as pointed out before and is clear both from the cross performance and stability results, this may hide lack of performance on some applications.

As in the case of the ANOVA tests, the results of EDONKEY and WEB match the cross performance ones. The cross performance is very good, higher or equal to 99% of recall for EDONKEY and 95% of recall for WEB. The features are mean-stable in almost all cases, except pkt2 in MS-I/RIII, for EDONKEY, and pkt3 for MS-I/R-II, for WEB. In these two cases the Hedges g coefficient is close to the threshold value (0.292 for EDONKEY and 0.245 for WEB), so we are not far from stability. However, a detailed analysis of the densities in these two cases shows that, while the features are considered unstable in a distributional sense, they keep their discriminating properties. We will discuss this issue in more detail when referring to BITTORRENT.



(a) pkt 1

(b) pkt 2

(c) pkt 3

(d) pkt 4

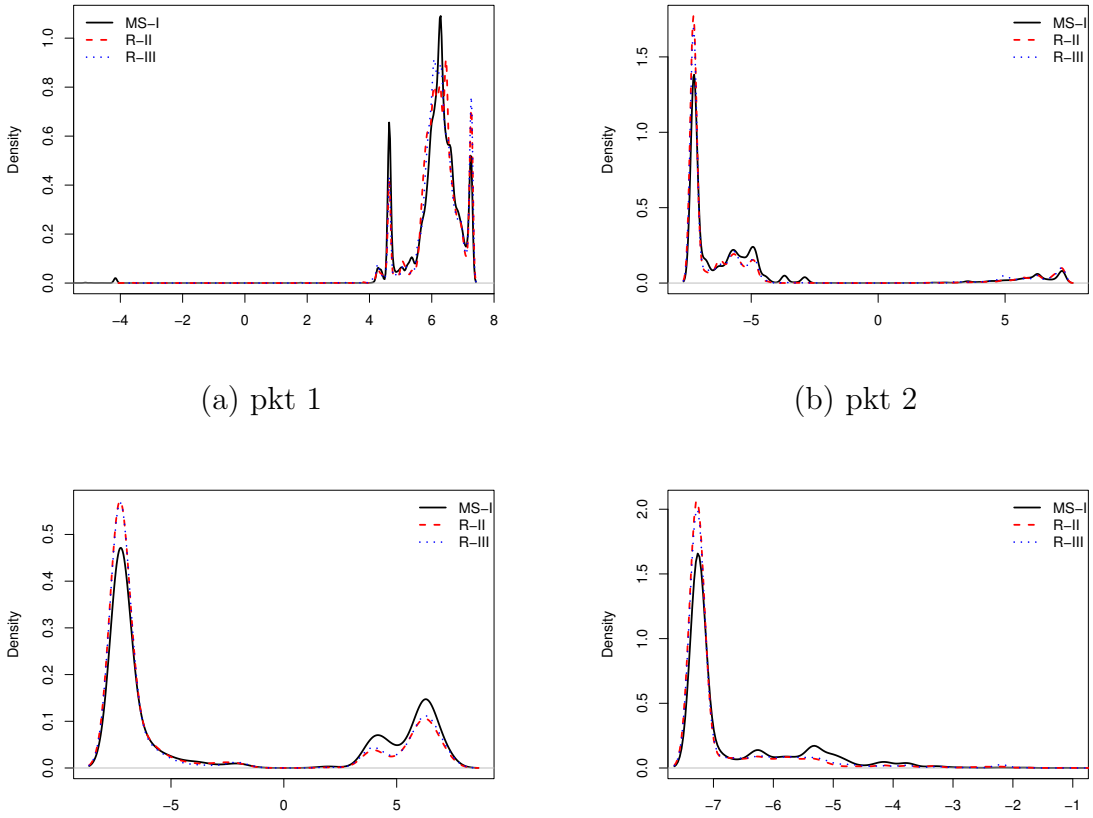
Figure 7.1: EDONKEY packet sizes distributions (log).

From the analysis of Figure 7.1 it is clear that the first 4 packets of EDONKEY are mostly signaling: there is an alternate exchange of small packets of approximately 100 bytes for pkt1 (from the client) and pkt2 (from the server) and then 11 bytes for pkt3 (from the client) and pkt4 (from the server). Moreover, the dispersion around the main peaks is very low. This behavior is consistent across all datasets which explains the good stability properties of these features.

We show the packet size densities associated with WEB in Figure 7.2. However, the stability of these features has a clear explanation based on the operation of the HTTP protocol. The first packet, issued by the client, is an HTTP request, usually a GET, which contains a list of standard fields whose size cannot vary a lot. We have observed a tight peak at 100 bytes. We also observed that the following three packets are mostly sent by the server and have full size (1460 bytes). This may be explained by that fact that after the HTTP request the server usually sends back an HTML page larger than 5 kilobytes, filling at least the 3 next packets.

7.3.2 BITTORRENT

BITTORRENT has a good cross performance only between R-II and R-III (captured on the same site 2 weeks apart). The recall is low when training with MS-I



(a) pkt 1

(b) pkt 2

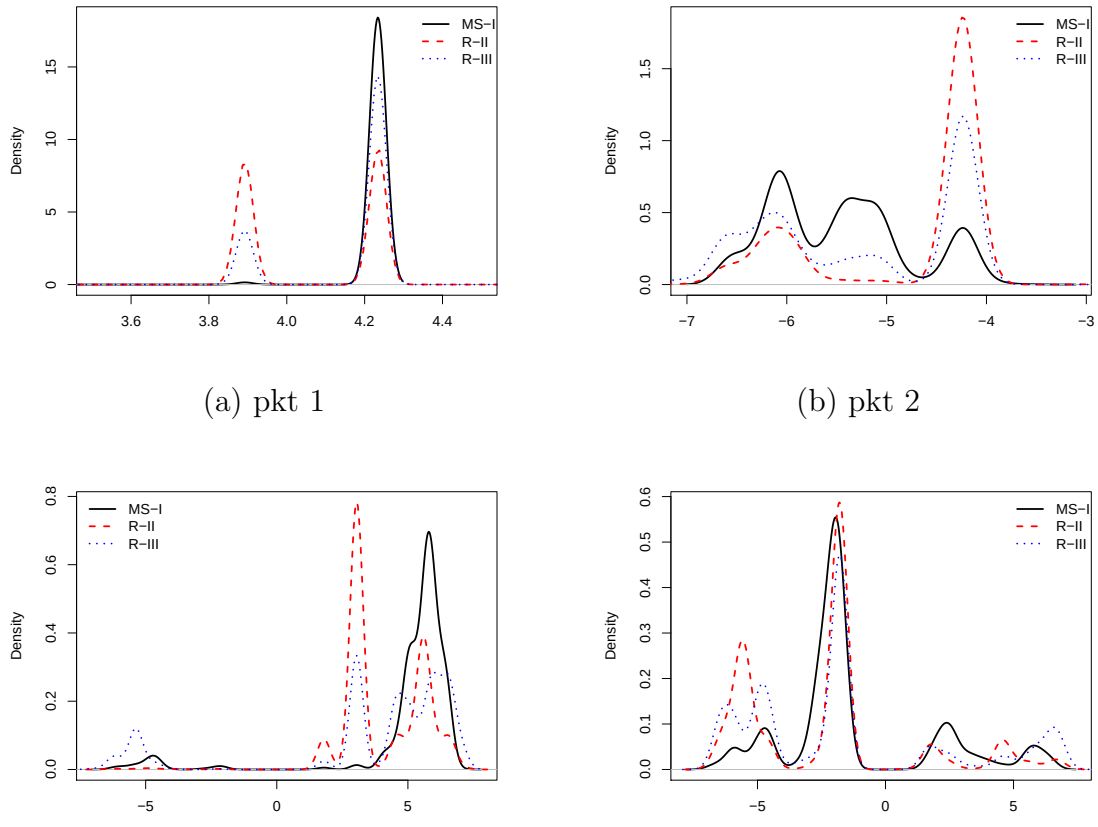
(c) pkt 3

(d) pkt 4

Figure 7.2: WEB, packet sizes distributions (log).

and testing with R-II or R-III and the precision is low in the opposite case. In these cases, except for pkt4 in MS-I/R-III, the features are not mean-stable (Table 7.2). Thus the stability results explain and confirm that the packet sizes and directions are not good discriminators for BITTORRENT in situations requiring portability. For BITTORRENT we observe significant differences in the distributions of packets between MS-I and R-II or R-III, especially on pkt2 and pkt3. We attribute these differences to the protocol behavior which can be dependent on the client version and the user configuration. Indeed, users may choose among different protocol obfuscation methods offered by the client. Detailed reverse engineering of the protocol is out of the scope of this work, however BITTORRENT is known for its efforts to evade detection applying, among others techniques, random padding on the initial packets that may alter its distributions.

The case of R-II and R-III datasets deserves further attention. In fact, the cross performance is very high (99% in all cases) but pkt1 and pkt2 features are not stable. It can be seen in Figure 7.3 (a), that, for pkt1, the densities of R-II and R-III are both bimodal with the mixture components having similar locations and dispersions; they differ in the weights of the components, which explain why the feature is not stable. However, because the mixture components have similar location and dispersion, the classifier will learn the same range of packet sizes whether trained on one dataset or the other. The weights of the



(a) pkt 1

(b) pkt 2

(c) pkt 3

(d) pkt 4

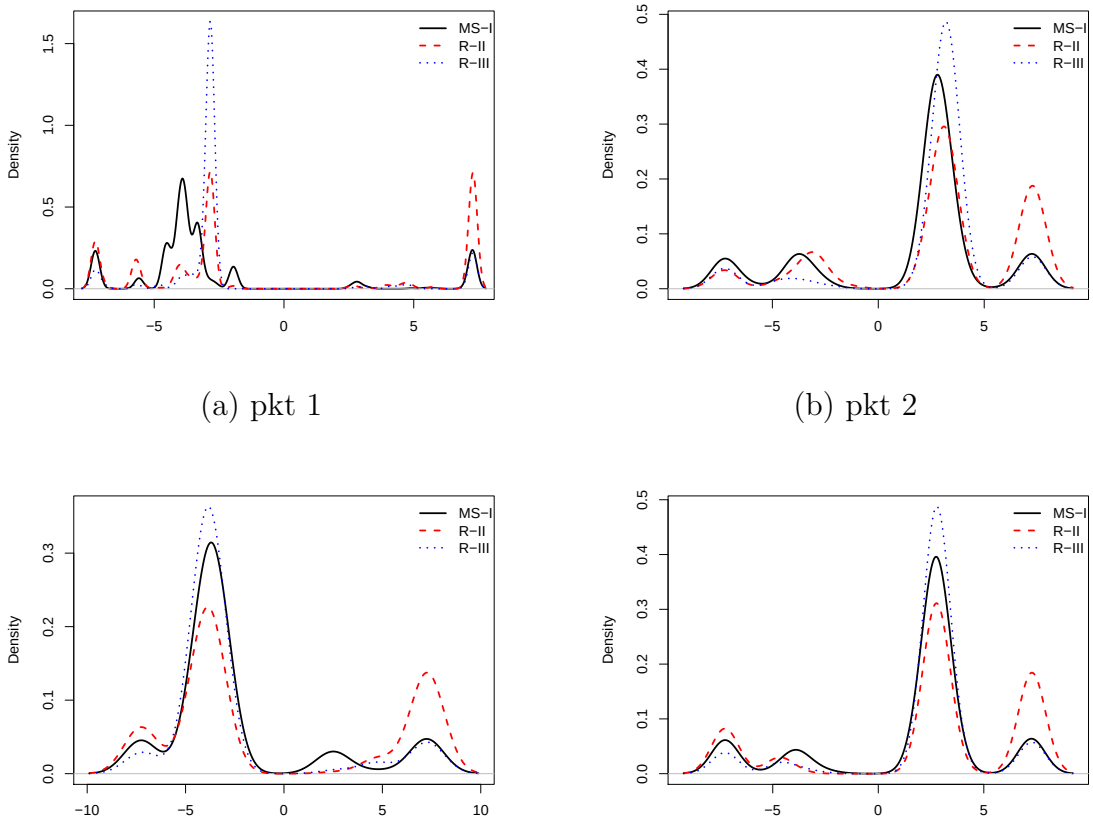
Figure 7.3: BITTORRENT, packet sizes distributions (log).

mixture components will not affect the performance of the classifier provided one is not too low, so that a minimum amount of packet size values is made available to train the classifier. We have observed this phenomenon in several other cases. This in fact means that distributional stability is, in some cases, a too strict requirement for classifier portability.

The case of BITTORRENT exposes another interesting property. In some cases, the cross performance is asymmetric: training with MS-I and testing with R-II or R-III yields poor recall (58%), but the opposite is not true; training with RII or R-III and testing with MS-I yields good recall (97%). As can be seen in Figure 7.3 (c) for the case MS-I/R-III, the density of R-III is multimodal but the one of MS-I has a single mode. Thus a classifier trained with MS-I will not learn packet sizes similar to the first peak of R-III leading to a poor recall. Otherwise, training with R-III prompts the classifier to learn the packet sizes that are most frequent in MS-I, resulting in a much higher recall. Moreover, in this case, since the classifier learns two ranges of sizes, other applications from MS-I with values of pkt3 close to the first peak of R-III can be wrongly classified as BITTORRENT. This is the reason why the precision is so low.

7.3.3 FTP

The cross performance of FTP is poor in many cases, in particular those involving crossing MS-I with R-II and R-III. Features are mean-stable in some cases, but this is merely a consequence of the fact that 1st order stability does not necessarily implies distributional stability. Indeed, pkt2 (RII/R-III), pkt3 (R-II/R-III) and pkt4 (MS-I/R-III) turn out to be variance-unstable. In fact, in these 3 cases the p-values associated with the F-test are approximately 0 and the effect sizes are 1.653, 2.210 and 1.474, respectively. An in-depth analysis of the Q-Q plots shows that pkt3 (MS-I/R-III) is also not stable. Thus only pkt4 (MS-I/R-II) and pkt1 (MS-I/RIII) are left as stable features, which explains the poor cross performance results.



(a) pkt 1

(b) pkt 2

(c) pkt 3

(d) pkt 4

Figure 7.4: FTP, packet sizes distributions (log).

Figure 7.4 shows that the main mode of pkt1 is negative, which means that FTP transfers are mostly initiated by the server. This seems surprising at first sight but has a clear explanation. FTP includes both signaling and data connections, on ports 21 (ftp) and 20 (ftp-data), respectively. Signaling connections are started by the client but it is the server that sends the first FTP packet, a welcome message that depends on the server implementation or on text inserted by the network manager. The data connections are initiated by the server in response to requests issued by the client in signaling ones. So, again in this case, the first FTP packet is sent by the server, and its size depends on the clients

request that may range from the download of a file, producing full size packets, to simple commands like cd or ls, producing quite small packets. The fact that FTP is composed of two dissimilar types of connections, signaling and data, and that the tasks performed over the data connection can produce (initial) packets of very different sizes, and indeed that some of this behavior can be configuration specific, clearly supports the lack of stability of these features. To confirm this we looked at the FTP servers used in each dataset. R-II and R-III share 3 among 5 of the most used servers, which explain why the packet size distributions are so close. MS-I, because it was captured on a different site, share none.

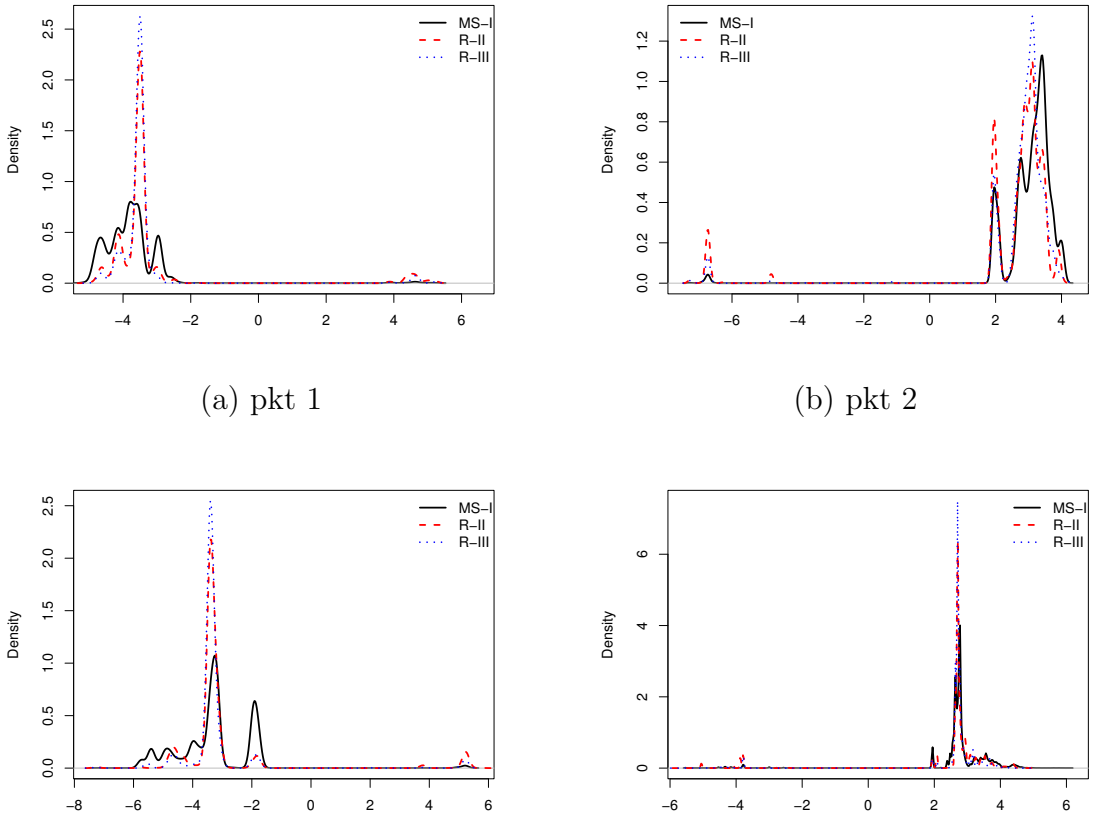
7.3.4 CHAT and MAIL

MAIL and CHAT have both a very good cross-site performance: for CHAT the minimum recall-precision is 96%-92%; for MAIL it is 94%-90%. However, the features are not stable in some cases. As in the case of BITTORRENT, this is only due to dissimilar weights of the mixture components, which does not compromise portability. Again, this behavior can be clearly explained on the basis of the protocols operation. For example, in CHAT, irrespective of the specific protocol, clients initiate a transfer with a request included in a small packet, and the servers answer sending a bunch of data that fills at least the three following packets. The packet sizes distributions are shown in Figures 7.6 and 7.5.

7.4 Discussion

We performed an in depth analysis of the root cause for the poor cross performance of classifiers, as reported in previous chapters, focusing on features Set A. Using a variant of statistical tests capable of dealing with large sample sizes, we demonstrated that in case of many applications, the degradation is explained by a lack of stability of the feature between sites. This suggests that purely statistical methods are often not sufficient to fulfill the classifier portability constraints. More precisely, statistical methods perform well only in case of some applications. Per application study of the feature stability should be a mandatory component for building stable classifiers.

Although the analysis presented here sheds some light on the problem, it seems that the criterion of equality of distributions is too strict, as for instance CHAT/MAIL classifier works well, despite the differences in distributions. In order to solve this issue, we would need a more appropriate metric of the features stability. Our analysis suggests that the the crucial issue is the feature distribution components locations and dispersions, whereas the weights of the mixture components should not affect the performance of the classifier.



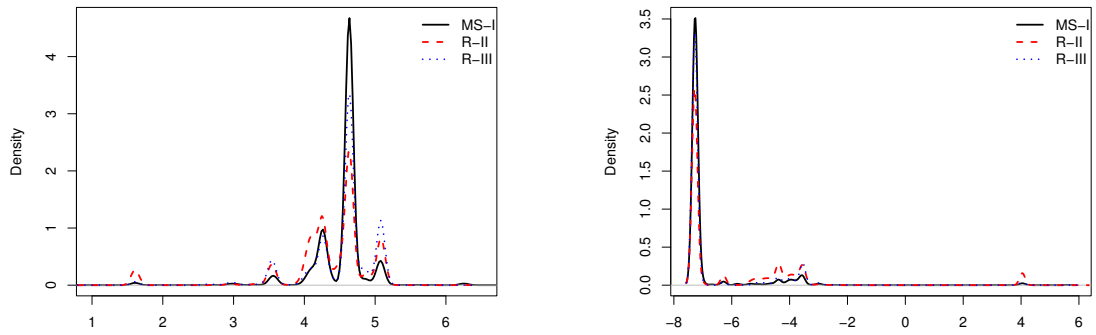
(a) pkt 1

(b) pkt 2

(c) pkt 3

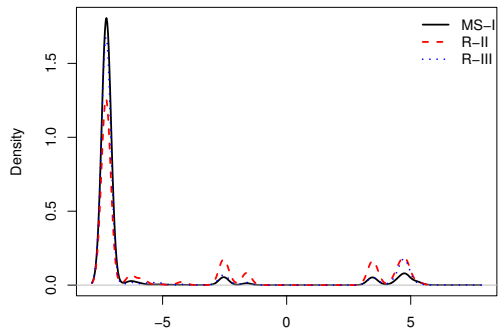
(d) pkt 4

Figure 7.5: MAIL, packet sizes distributions (log).

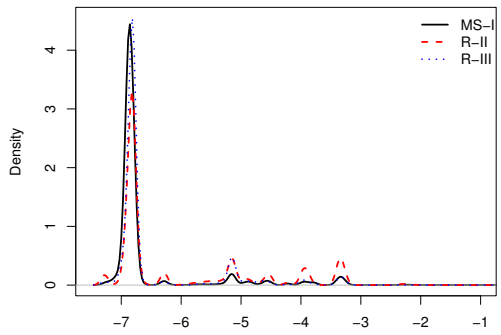


(a) pkt 1

(b) pkt 2



(c) pkt 3



(d) pkt 4

Figure 7.6: CHAT, packet sizes distributions (log).

Chapter 8

Mining the Unknown Class

Previous chapters addressed the difficulties in building a portable (resilient to cross site issues) statistical classifier. In this chapter we revisit the static case, focusing on the traffic that the ground truth tool was not able to recognize.

Indeed, in most studies where supervised machine learning algorithms are used, results from the statistical classifier are benchmarked against the known traffic, i.e., the traffic identified by the ground truth tool that is used. The rest of the traffic, that we term unknown traffic, is excluded from further analysis. This was the case in previous chapters where we evaluated statistical classification as an *alternative to DPI*.

In this chapter, we present a different approach and investigate results obtained when the statistical classifier is used over the UNKNOWN class. We propose solution, where statistical methods complement DPI classifier. Such a classifier could be included as a module of tools like ODT and used as source of information or help in the process of the tool development, in case an increase of unknown traffic is noted. To the best of our knowledge this is the first study that tackles this problem using supervised methods.

8.1 Unknown Mining Method

Study of the filtering scenarios (see Table 4.5) revealed that the UNKNOWN class consists of a large fraction of connections (61% to 84% depending on the trace) for which the beginning is missing. Those truncated connections carry the majority of bytes in this class, from 79% to 86%. To maximize the number of bytes for which a prediction could be made, we adopted the following strategy:

1. We used the second set of features (Set B, see table 4.4). The first one (packet sizes) would have de facto reduced the number of flows and bytes for which a prediction could be made (see Table 4.5).

2. We trained the classifier on all known traffic for which a three-way handshake was observed (S/S).

3. We apply the classifier on all flows of the UNKNOWN class, without any a priori filtering.

4. Our classifier outputs for each flow a class prediction associated with a confidence level.

5. We make use of the confidence level returned by the C4.5 algorithm to select the flows for which we consider the prediction as plausible.

The high level procedure is presented in Figure 8.1. In the latter step of the methodology described above, we used a threshold of the 95% confidence level.

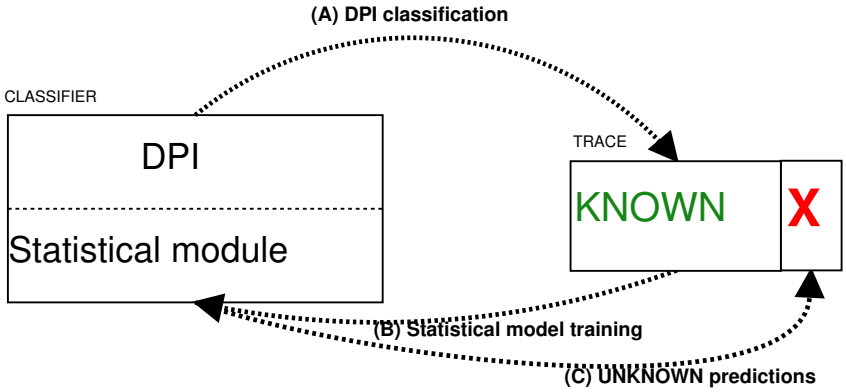


Figure 8.1: Mining the unknown - schema.

8.2 Predictions

Figure 8.2 depicts the cumulative distribution function of per flow confidence levels for the flows in the UNKNOWN class. With a threshold of 95%, we observe that, depending on the trace, a fraction between 40% to 70% of the flows are kept for further analysis.

Predictions (classifications) are reported in Table 8.1. We present only results for classes that performed well in the static case and carry at least 1% of bytes for at least one of the traces. Those results are in line with the ones obtained for the known traffic as we observe a majority of Web, e-Donkey and BitTorrent traffic.

Class	MSI	RII	RIII	TI
EDO.	18%/32%	17%/46%	26%/42%	28%/71%
BT.	1%/15%	5%/14%	8%/12%	2%/9%
GNU.	1%/3%	1%/10%	2%/3%	3%/≤1%
WEB	8%/≤1%	5%/≤1%	9%/≤1%	3%/≤1%
$\sum$	28%/50%	28%/71%	37%/58%	34%/81%

Table 8.1: Unknown class predictions, [flows%/bytes%].

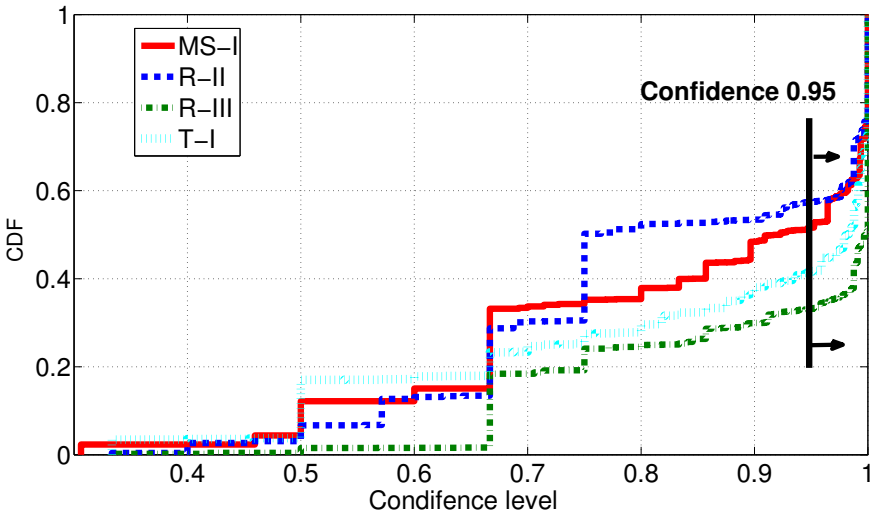


Figure 8.2: Confidence level vs. fraction of flows.

8.3 Validation

As in this section we operate on unknown traffic, ODT (our DPI tool) does not provide us any reference point. We need to validate the predictions of the statistical classifier using some other methods. In this section, we perform several side tests to challenge the predictions we obtained for the unknown traffic. We will mainly use the knowledge about the {*IP, port*} pairs of the endpoints of the flows.

8.3.1 Peer-to-peer Predictions

For the case of peer-to-peer predictions we use the following additional sources of information per flow:

- **Port numbers.** Even for p2p applications, there is still a fraction of users that use legacy ports [58]. A list of legacy ports for popular p2p applications is given in Table 8.2. If ever such a port is observed for a flow for which the classifier outputs "P2P class", we consider that this information backs the result of the classifier.

- **Endpoint information:**

  – We search for connections to the same remote endpoint, e.g., the same {IP,port} pair, in the known set. This method was inspired by the work in [55].

  – We perform **reverse dns lookups** for each remote IP searching for ADSL machines. Most of the providers use simple syntax consisting of IP address and some keywords to identify the hosts of their users. The list of keywords we used is provided in Table 8.3. It is inspired by

[91][1], and based on the hypothesis that communication between two ADSL hosts is likely to be due to a p2p application.

The above procedure is formalized in Algorithm 1. Results for the p2p predictions are presented in Figure 8.3. Overall, we obtained that at least half of the bytes and flows classified with high confidence are further reinforced by the results of Algorithm 1. The reason why a fraction of p2p flows were not classified by ODT lies in the method used to detect these applications. In most cases, DPI tools need to monitor the beginning of the flows.

Algorithm 1: Endpoints profiling.

```
foreach f=flow in P2P do
    if f.prediction.confidence ≥ 0.95 then
        if f.remote.endpoint in known set then
            Known.insert(f)
        else
            if f.local.port==legacy OR f.remote.port==legacy then
                Port.insert(f)
            else
                if f.remote.endpoint in adsl set then
                    ADSL.insert(f)
                else
                    Reject.insert(f)
                end
            end
        end
    else
        Reject.insert(f)
    end
end
```

Class	Port
WEB	80, 8080, 443
P2P-EDONKEY	4662, 4672
P2P-BITTORRENT	6881-6889
P2P-GNUTELLA	6346

Table 8.2: Legacy ports used.

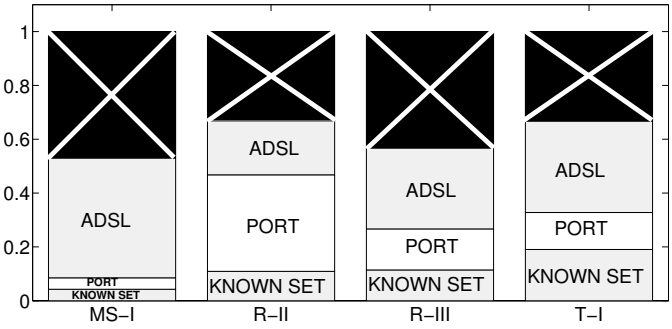
8.3.2 Web Predictions

For the flows classified as Web, we perform connections attempts to each endpoint, using *wget*, searching for active Web servers. The hit ratios was very low, below

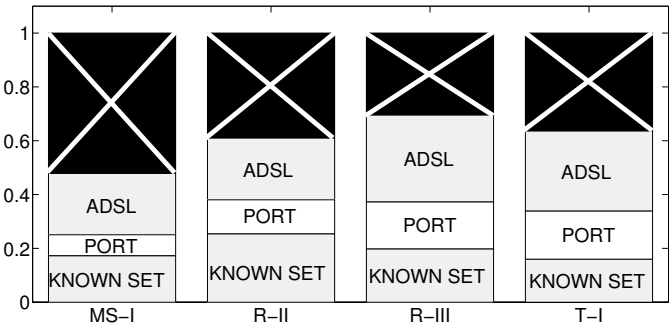
[1]We also implemented a simple google querying tool proposed in [91]. This method relies on parsing the google answers for the {*IP, port*} pairs of the flows seeking for application indication. However the number of hits obtained was too low.

Keyword	Provider
wanadoo	Orange
proxad	Free
dsl/DSL/ADSL	Other providers

Table 8.3: Keywords used to detect DSL hosts.



(a) Flows



(b) Bytes

Figure 8.3: Results of the validation algorithm 1 for the P2P applications. Fractions corresponding to each validation method on Y axis.

3%. However traces are more than one year old, so we can not verify how many servers were really active during the time of the capture.

Using reverse dns queries, we verified that most of the endpoints involved in the flows predicted as WEB flows were residential hosts. In such a case, the existence of transient Web servers can be due to malicious activities like Fast Flux networks [29], which are botnets where compromised machines are used as proxies to hide a Web server. There is also an increasing trend of using HTTP protocol to control bots which is supposed to make the detection more difficult [57]. Such behavior could explain results of our classifier and the fact that the flows were unknown to ODT. We leave for future work an in-depth study of this hypothesis.

8.3.3 Throughput Distribution Comparison

A last technique we used to challenge the predictions made by the statistical classifier is to plot distributions of throughput for flows in a given class in the known and unknown sets. We present the resulting cdfs in Figure 8.4. We observe from this figure that EDONKEY and BITTORRENT predictions seem reasonable as the throughputs for both sets are similar. In addition, those throughputs are clearly smaller than the throughputs of the flows in the known WEB class, which is in line with the fact that residential end hosts are less provisioned than Web servers in general. On the contrary, the unknown WEB class significantly differs from the known one, which is in line with the observation made on the previous section that the remote server was a residential host, and gives further weight to the hypothesis that malicious activities are possibly at play.

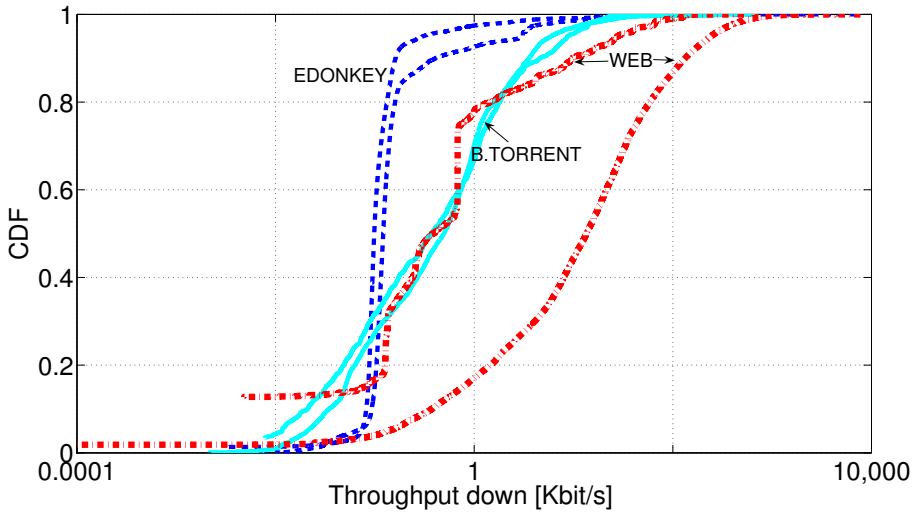


Figure 8.4: Throughput distributions for known and predicted sets. Trace MSI.

8.4 The Unknown Class - Discussion

We have shown that a supervised model of traffic classification can be useful to mine the unknown traffic. High confidence predictions were further validated by a number of heuristics based on a variety of endpoint informations and port numbers. We presented the usage of statistical classifier as a complementary method for tools like ODT. A prediction module, based on the statistical classifier, can be included in a tool like ODT and used as a additional source of information in the labor intensive process of updating signatures for new versions of emerging applications.

Chapter 9

Conclusions of Part I

In this part, we adopted the perspective of an ADSL provider, and critically evaluated the potential benefits coming from usage of state of the art statistical methods for application identification.

Our conclusions are manifold. On the positive side, statistical classification offers high performance when applied on the same site where they were trained. It also turns out to be useful to mine the traffic left unidentified by DPI tools.

On the negative side, we have demonstrated that statistical classification tools might suffer from data over fitting, which prevents a simple strategy such as: train on the largest PoP (where ground truth is available) and deploy on all other sites. To the best of our knowledge, this has never been observed before. This problem is complex as it persisted over the whole range of features sets and machine learning algorithms we considered. We also investigated the reason for the poor portability and we showed that in many cases it can be explained by the variability of features between the datasets.

An important by product of this study is to highlight the need to test new classifiers not simply on traces collected on a given site, but also on traces collected at different sites. The latter needs to be done on "homogeneous" traces in terms of type of traffic and capture time. Indeed, previous attempts to address the cross-site issue, namely [58] and [62], either considered overly heterogeneous traces [58] or traces collected in academic environments [62] and with long periods of time (one year) between subsequent traces.

One last important take away message from this part of the thesis is the observation that the complexity of detecting an application varies greatly from one application to another. Despite the cross site problem, some applications (e.g. EDONKEY) are very well classified using statistical features. As a consequence, it is hard to classify all applications using a single method, thus the detection technique should be tailored to the each class of traffic. We build on this observation in Part II of the thesis which describes a hybrid classification method that enables synergy between diverse sources of information.

Part II

Hybrid Classifier

Chapter 10

Introduction

Our starting point in Part II is the observation that the complexity of detecting an application varies greatly from one application to another. As a consequence, it is hard to classify all applications using a single method. For instance, deep packet inspection techniques are blind when traffic is encrypted unlike statistical approaches. Conversely, statistical approaches might be less accurate than DPI at zooming inside HTTP traffic, e.g., to isolate HTTP streaming or Webmail. What is more, many applications can not be reliably classified using statistical methods as it was demonstrated in Part I of the thesis.

To address the above issue, we propose a technique that we call Hybrid Traffic Identification (HTI), which lets us benefit from the synergy between various classification approaches, e.g., deep packet inspection techniques and statistical classification methods relying on flow features.

We treat each source of information as a feature, e.g., the presence of a signature in the payload of a packet becomes a feature of the corresponding flow along with other discriminators like the size and direction of packets. Virtually any classification method can be incorporated as its output can be encoded as a feature. The classification decision in HTI (where a flow is attributed to a specific application) is made by a machine learning algorithm based on this enriched set of features. We provide a full evaluation of HTI based on the ADSL traces used in Part I. We demonstrate that HTI offers high performance both in terms of bytes and flows for key application of interest. We further highlight the enhanced resilience of HTI to the data over-fitting problem of supervised machine learning approaches identified in Part I, where we showed that the statistical traffic classification methods tend to learn site specific characteristics of traffic, which deteriorates their performance when applied at other sites.

Furthermore, we report on the *production deployment* of an HTI instance in the network of a large ISP, which connects several thousands of customers to the Internet. Results span several months of continuous 24/7 classification. To the best of our knowledge, this is the first time that the supervised machine learning traffic classifier leaves the lab to be deployed in an operational network.

10.1 Contributions - Part II

Based on the observations from Part I, we propose a novel traffic identification schema that enables to take advantage of the merits of different approaches. Any source of information (flow features, DPI decision, etc.) is encoded as a binary feature. Furthermore, the quantization of features offers the additional advantage that it allows proper treatment of qualitative indicators (like port number).

Apart from the hybrid nature of our classifier, we propose the usage of logistic regression, a simple yet powerful learning model. This is a classical machine learning algorithm that so far hasn't been used in the context of traffic classification. Among other features, it offers modular and easy to interpret models that allow judging the discriminative role of different methods in use.

We first heavily test HTI using traces described in Part I of the thesis. We demonstrate that not only HTI elegantly integrates the classical classification schemes, but, in addition, has higher resilience in a cross-site classification scenario.

In the last chapter of Part II we report on the implementation and operational deployment of an HTI instance that works live in 24/7 manner on an ADSL platform. We also report on the selection of the statistics collected by HTI during the course of the measurement.

10.2 Relevant Publications for Part II

[77] M. Pietrzyk; T. En-Najjary, G. Urvoy-Keller, J-L. Costeux, Hybrid traffic identification, Eurecom, Report RR-10-238

[25] T. En-Najjary, G. Urvoy-Keller, M. Pietrzyk, Application-based feature selection for internet traffic classification ITC 2010, 22nd International Teletraffic Congress, September 7-9, 2010, Amsterdam, The Netherlands

Chapter 11

Design and Evaluation

In this chapter we present our hybrid classifier and its evaluation in off-line mode. The next chapter reports on its deployment in operational network.

11.1 HTI - Hybrid Traffic Identification

In this section we introduce Hybrid Traffic Identification (HTI). We exemplify and evaluate HTI on off-line traces in section 11.5 and through live experiments in a production network in chapter 12. Hybrid Traffic Identification aims, as its names suggest, at combining several existing traffic classification techniques into a single framework, enabling the synergy between them. HTI features the three following key characteristics:

Everything is a feature: We encode diverse sources of information as feature. For instance, the presence of a signature in the payload of a packet becomes a feature of the corresponding flow along with other discriminators like the size and direction of packets. Virtually any classification method can be incorporated as its output can be encoded as a feature. We present the set of features we use in this work in Section 11.1.1. Note that this is a fundamentally different strategy than the one typically followed by traffic classification studies [71]. In most cases authors restrain themselves to single class of features.

Self learning: HTI relies on supervised machine learning. During its learning phase, based on traffic sample HTI rates, i.e. assigns a weight, to the pieces of information (encoded as features) originating from different classification techniques. This relieves the practitioner from the burden of formulating heuristic rules when various sources of information lead to contradictory results. As a learning algorithm we use logistic regression (detailed in Section 11.2).

Per-application sub-models: In its learning phase, HTI creates a dedicated classifier for each application. We term them sub-models. This allows for flexibility as different methods can be used for each application. In addition, inspecting the sub-model enables to understand the usefulness of each input fea-

ture. We detail this issues in Section 11.3.

Let us now detail the features used in HTI along with details on the logistic regression algorithm and specific issues related to the use of sub-models.

11.1.1 Features

We describe here the features used by HTI, along with the quantization/encoding strategies that enable a statistical machine learning algorithm to process them.

- **Data packets:** Information about first four data packets of each flow. As noticed in [5], size and direction of few first data packets sometimes carry enough information to detect the flow behind an application. It appears to hold for some key applications, for example eDonkey, in our traces (See Part I). We use three binary features for each of the first k packets of a transfer: direction (up/down), size (small/big)[1] and the presence of Push flag.

- **Port numbers:** Port numbers carry valuable information, even though they can not be used blindly as there is no way to enforce that a specific application uses a specific port number. In addition, port numbers are qualitative rather than quantitative values. Indeed, comparing port number values is meaningless, as for instance port 80 is closer to port 25 than 8080, while 80 and 8080 are often associated to HTTP and 25 is in general associated to SMTP. Many studies [58] include port numbers in the features set treating them as quantitative features which is one of the reasons of the data over-fitting issues described in Part I.

  In this work, we use a pre-processing phase where port numbers are quantized. The quantization technique used depends on the application of interest. For applications using the HTTP protocol, we assign the port variable to 1 if the source or destination port number belongs to the set $80, 8080, 443$ and 0 otherwise. For p2p applications, we assign the port variable to 1 if both the source and destination ports are above 1024 but not equal to 8080. This procedure is used in the training and testing phases. In the training phase, we know the ground truth and so we set the ports according to the procedure. In the testing phase, we assign the port number value used as input for the classification depending on the test we want to perform. When we want to test if this is an application on top of HTTP, we check if it flows on ports $80, 8080, 443$ or not to the set the corresponding feature and when we test for a p2p application, we check if ports are dynamic (above 1024) or not (again to set the corresponding feature).

  Note also that other quantization strategies are possible. For instance, for p2p applications, one could have used legacy port numbers of considered p2p applications. It turned out however that the quantization technique we

[1] We use a fixed threshold, derived from empirical distributions of packet sizes, of 200 bytes for all applications and all traces

Class	Signature
WEB	(not!)(Content-Type: application/x-shockwave‖Content-Type: video‖ ‖GNUTELLA‖X-Gnutella‖Content-Type: audio)
HTTP-STR	Content-Type: application/x-shockwave‖Content-Type: video
EDONKEY	– none –
BITTORRENT	0x13BitTorrent protocol
MAIL	– none –

Table 11.1: Payload signatures used (case sensitive).

propose, which makes no use of such a priori information, offers satisfactory results.

- **Payload signatures:** For many application classes, valuable information is carried in the packet payload. We thus enrich our set of features with payload signatures. A list of signatures used in this work is presented in Table 11.1. The presence of a signature is denoted as a 1, its absence as a 0.

Note that the strategies used above to introduce new types of information within the supervised statistical learning framework can be easily extended to incorporate as features informations used by other techniques, e.g., discriminating information obtained with heuristics [55], end points profiling [91], flows statistics [66] or payload pattern matching information [85, 92].

11.2 Machine Learning Algorithm

HTI relies on supervised machine learning techniques. We use logistic regression in the remaining of the Part II (unless stated otherwise), as it offers simple and easy to interpret models[2]. We demonstrate in Section 11.6.3 that other algorithms (Support Vector Machine, Decision Tree C4.5) can offer similar accuracy. It confirms that the strength of our approach lies in its hybrid nature rather than any specific characteristics of logistic regression.

11.2.1 Logistic Regression

The use of logistic regression has proliferated during the past decade. From its original use in epidemiological research, the method is now commonly used in many fields including biomedical research [95], business and finance [93], criminology [97], health policy [95] and linguistics [74]. Logistic regression is designed to model the relation between a binary variable (true vs. false) and a set of covariates (features in our case).

[2]A model is the output of the training phase. We use interchangeably the terms model and classifier.

When applied to traffic classification, logistic regression will generate c sub-models, one per application. This is because it tests one binary variable – the flow has been generated by a certain application or not – against a set of covariates – the flow features. Thus, Y takes values in the set $\{0, 1\}$, where $Y = 1$ if the flow was generated by a certain application and 0 if it was generated by one of the other $c - 1$ applications, or an unknown application.

Each sub-model is fully defined (during the training phase) by its feature vector $x = (x_1, x_2, \cdots, x_n)$ parameterized by the weights vector $\beta = (\beta_0, \beta_1, \cdots, \beta_n)$. The β vector is easy to interpret when one wants to understand which features have been selected during the training phase – see Section 11.5.3.

Consider a flow with the following feature vector $x = (x_1, x_2, \cdots, x_n)$. We wish to have the probability of whether this flow is generated by application A or not. Formally, we can state this as[3]

$$p(Y = 1|X = x) = P(x, \beta), \tag{11.1}$$

where $p(Y = 1|X = x)$ is the conditional probability that the flow with features $x = (x_1, x_2, \cdots, x_n)$ is generated by application A and P is a function of x parametrized by the weights vector $\beta = (\beta_0, \beta_1, \cdots, \beta_n)$. Since the function P represents a probability, it must take values between 0 and 1. Within the Logistic regression framework, one assumes a specific function P:

$$P(x, \beta) = \frac{e^{\beta_0 + \sum_{j=1}^{n} \beta_j x_j}}{1 + e^{\beta_0 + \sum_{j=1}^{n} \beta_j x_j}}, \tag{11.2}$$

From the above equation, we can derive a linear function between the odds of having application A and the feature vector x, called the logit model:

$$\ln\left(\frac{P(x, \beta)}{1 - P(x, \beta)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n, \tag{11.3}$$

Unlike the usual linear regression model, there is no random disturbance term in the equation for the logit model. This does not mean that the model is deterministic because there is still room for randomness in the probabilistic relationship between $P(x, \beta)$ and the application A.

A logistic regression model is fully defined by its vector $\beta = (\beta_0, \beta_1, \ldots, \beta_n)$. Those values are estimated during the training phase, which is usually done using maximal likelihood estimation, numerically computed with the Newton-Raphson algorithm [41]. Please note that for each application we have a distinct β vector.

We next turn our attention to the issue of reconciling multiple sub-models in the case of logistic regression. Each flow to be classified is evaluated against all the sub-models. Each answer being probabilistic, we consider only sub-classifiers results above a certain threshold. In this work we assume a threshold of $P(x, \beta) \geq$

[3]Please note that, for the sake of clarity, we avoided indexing each variable with application A. However, we would like to point out that the following procedure is done for each application of interest. In particular, it leads to β vectors that are application dependent in both length and value.

0.5 which is equivalent to:

$$\beta_0 + \sum_{j=1}^{n} \beta_j x_j > 0 \tag{11.4}$$

This can potentially lead to multiple contradictory classification decisions. If for a given flow, we have several sub-model decisions above the threshold, the one with the highest probability score is picked:

$$\arg \max_{k=1,\ldots,c} \left\{ P(x, \beta^{(k)}) | P(x, \beta^{(k)}) \geq 0.5 \right\} \tag{11.5}$$

where $\beta^{(k)}$ is the beta vector for application number k. If for a given flow, no sub-model decision is above the threshold, the flow is declared as unknown.

11.2.2 Training Phase

We now describe the *model building phase* (β estimation) for the logistic regression. We exemplify the process for the case of a single application denoted as A. The same procedure as below needs to be repeated for each application of interest. Consider a training data set of N flows characterized by the feature vectors $X = (X_1, X_2, \cdots, X_N)$, where $X_i = (x_1^i, x_2^i, \cdots, x_n^i)$ is the feature vector of flow i, and let the vector $Y = (y_1, y_2, \cdots, y_N)$ be such that $y_i = 1$ if flow i is generated by the application A and $y_i = 0$ otherwise. The likelihood function is given by a standard formula [41]:

$$\begin{aligned} P(X, \beta) &= \prod_{j=1}^{N} p(Y = y_j | X_j) \\ &= \prod_{j=1}^{N} (p(Y = 1 | X_j)^{y_j} (1 - p(Y = 1 | X_j))^{1-y_j} \end{aligned} \tag{11.6}$$

As the values of p are small, it is common to maximize the log-likelihood $L(X, \beta) = \log P(X, \beta)$ instead, to avoid rounding errors [41].

By substituting the value of $p(Y = 1|X_j)$ by its value defined in Equation (11.2) we get the log-likelihood for the logistic regression:

$$L(X, \beta) = \sum_{i=1}^{N} \left[y_i \beta^T X_i - log(1 + e^{\beta^T X_i}) \right] \tag{11.7}$$

In the logistic regression model, we wish to find β that maximizes Equation (11.7). Unfortunately, this can not be achieved analytically. In this work, we compute it numerically using the Newton-raphson algorithm [41]. This algorithm requires two main components: the first derivative of the loglikelihood and the Hessien matrix, i.e., the second derivative matrix with respect to β.

From Equation (11.7) we can derive the first derivative

$$\frac{\partial L(X,\beta)}{\partial \beta} = \sum_{i=1}^{N} X_i(y_i - p(x_i,\beta)) \tag{11.8}$$

We now derive the Hessien matrix

$$\frac{\partial^2 L(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} X_i X_i^T p(x_i,\beta)(1 - p(x_i,\beta)) \tag{11.9}$$

The pseudo code of Newton-Raphson algorithm is depicted in Algorithm 2. We start with a first guess of β, then we use the first derivative and the Hessien matrix to update β. Using the new β we compute the new loglikelihood. This is repeated until there is no further change of β. The Newton-Raphson algorithm has been shown to converge remarkably quickly [44]. In this work, it takes less than one second to output an estimate of β.

Algorithm 2: Newton-Raphson algorithm

1: initialize β
2: **while** $\|\beta_{new} - \beta_{old}\| > thr1$ and abs$(L_{new} - L_{old}) > thr2$) **do**
3: Calculate $g = \partial L/\partial \beta$
4: Calculate $H = \partial^2 L/\partial \beta^2$
5: Set $\beta_{old} = \beta_{new}$
6: Calculate $\beta_{new} = \beta_{old} - H^{-1}g$
7: Set $L_{old} = L_{new}$
8: Calculate L_{new}
9: **end while**
10: Calculate variance matrix $\hat{V}$

11.2.3 Selection of Relevant Features

As we estimate a new model for each application, the weight β_j given for each feature emphasis the importance of the corresponding feature to this application. Moreover, logistic regression provide a way to test the relevance of a given feature to the classification output. This can be done through the formulation and testing of a statistical hypothesis to determine whether the corresponding variables in the model are "significantly" related to the outcome variable Y. In other words, for each feature j, we test the hypothesis that the corresponding weight β_j is equal to zero. If we can't reject this hypothesis, this means that this parameter is not relevant to classify this application and, thus, can be removed from the model [44].

In this work, we use the Wald test [44] that tests, individually, for each β_j the null hypothesis that $\hat{\beta}_j = 0$. The Wald statistic $W(j)$ is obtained by comparing the maximum likelihood estimate of each parameter $\hat{\beta}_j$ to an estimate of its standard deviation $\hat{V}(\hat{\beta}_j)$.

$$W(j) = \frac{\hat{\beta}_j}{\hat{V}(\hat{\beta}_j)} \tag{11.10}$$

The standard deviation $\hat{V}(\hat{\beta}_j)$ of β_j is given by the j^{th} diagonal element of the variance matrix given by Equation (11.11) [41], that is computed as the last iteration of the Newton-Raphson algorithm (Alg. 2).

$$\hat{V} = \left\{ -\frac{\partial^2 L(\beta)}{\partial\beta\partial\beta^T} \right\}^{-1} \tag{11.11}$$

Under the *null hypothesis* that $\beta_j = 0$, $W(j)$ follows a standard student *t-distribution* with $n-1$ degree of freedom t_{n-1}.

For a given significance level α, for each β_j we compute the p-value $pv_j = p(t_{n-1} > W(j))$, and we reject the hypothesis of $\beta_j = 0$ if $\alpha > pv_j$. Otherwise, if we fail to reject the hypothesis of $\beta_j = 0$, we exclude the corresponding feature from our model. By doing so, we can keep a minimum number of features relevant to the application under study.

A crucial aspect of using logistic regression is the choice of an α level to judge the importance of features. Bendel et al [3] have shown that the choice of α smaller than 0.01 is too stringent, often excluding important variables from the model. In this work, we use $\alpha = 0.01$.

It was demonstrated in [25] that presented strategy does not decrease the accuracy of the method while allowing to reduce the number of features for each application. In our case the set of features is relatively small and in many cases common for all applications (they anyway need to be computed). We thus do not perform this step in the following sections and rely on β coefficients to assess importance of the features.

11.3 Per Application Sub-models

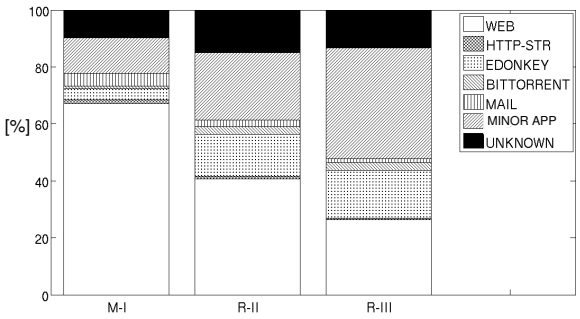
Many approaches relying on machine learning techniques in the domain of traffic classification, lead to the generation of one model for all the applications of interest [71]. For instance, if the machine learning algorithm is a decision tree, a global decision is constructed during the training phase, using the same shared set of features for all applications.

We adopt a different strategy that leads to one model per application during the training phase. This slightly modifies the training as each application has now to be tested against all the others, leading to as many models as applications. We call these models sub-models to emphasize the fact that they take a decision for a specific application only. If sub-models are used, a reconciling phase is needed in case a flow matches many sub-models. We rely on the statistical confidence value provided by the machine learning algorithm to do so. This means that we pick among the matching sub-models, the one that offers the highest probability score level (equation 11.5). An advantage of sub-models is that they enable to understand, for each application one wants to detect, which features have

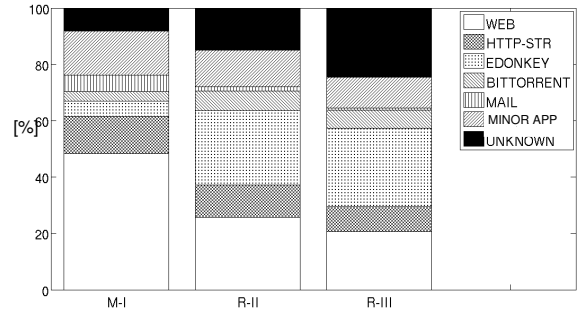
been declared as important by the machine learning algorithm during its training phase. We illustrate this point in Section 11.5.3. Furthermore, such a modularity allows for greater flexibility as detection method (features set) can be distinct and tailored to each application.

11.4 Off-line Evaluation: warming-up

For the evaluation we use the same datasets, reference point, performance metrics (recall/precision) and flow definition as in Part I, which are described in chapter 4. The main difference is that we now aggregate less popular applications into a single class we term MINOR_APP. Figure 11.1 shows classification results obtained by ODT, in flows and bytes, for our three traces.



(a) Breakdown in flows



(b) Breakdown in bytes

Figure 11.1: Application breakdown in the data sets.

In the remaining of this section, we present the calibration results of our (logistic regression based) instance of HTI, for the following subset of applications: BITTORENT, EDONKEY, HTTP-STREAMING, WEB and MAIL. Please note that when building the sub-models for these classes, we keep the flows in the UNKNOWN and MINOR-APP classes (their removal could lead to too optimistic results).

Recall [flows% | bytes%]

WEB

↓Training	MS-I	R-II	R-III
MS-I	99% | **96%**	98% | **92%**	98% | **92%**
R-II	95% | **93%**	99% | **95%**	99% | **95%**
R-III	95% | **93%**	99% | **95%**	99% | **95%**

HTTP-STR

↓Training	MS-I	R-II	R-III
MS-I	98% | **99%**	96% | **99%**	98% | **99%**
R-II	98% | **99%**	96% | **99%**	98% | **99%**
R-III	98% | **99%**	96% | **99%**	98% | **98%**

EDONKEY

↓Training	MS-I	R-II	R-III
MS-I	99% | **99%**	98% | **98%**	98% | **98%**
R-II	97% | **98%**	96% | **97%**	97% | **97%**
R-III	97% | **99%**	98% | **98%**	97% | **98%**

BITTORRENT

↓Training	MS-I	R-II	R-III
MS-I	100% | **100%**	99% | **99%**	97% | **98%**
R-II	100% | **100%**	99% | **100%**	99% | **99%**
R-III	100% | **100%**	99% | **100%**	99% | **99%**

MAIL

↓Training	MS-I	R-II	R-III
MS-I	94% | **97%**	99% | **100%**	100% | **99%**
R-II	90% | **95%**	99% | **100%**	99% | **100%**
R-III	90% | **95%**	99% | **100%**	99% | **99%**

Precision [flows% | bytes%]

WEB

↓Training	MS-I	R-II	R-III
MS-I	99% | **97%**	99% | **95%**	99% | **95%**
R-II	99% | **97%**	99% | **94%**	99% | **92%**
R-III	99% | **97%**	99% | **95%**	99% | **95%**

HTTP-STR

↓Training	MS-I	R-II	R-III
MS-I	93% | **96%**	96% | **98%**	95% | **99%**
R-II	93% | **96%**	96% | **98%**	95% | **99%**
R-III	91% | **96%**	95% | **98%**	94% | **99%**

EDONKEY

↓Training	MS-I	R-II	R-III
MS-I	91% | **95%**	95% | **94%**	98% | **98%**
R-II	92% | **95%**	97% | **95%**	98% | **98%**
R-III	92% | **96%**	95% | **94%**	98% | **98%**

BITTORRENT

↓Training	MS-I	R-II	R-III
MS-I	96% | **98%**	98% | **99%**	98% | **99%**
R-II	99% | **98%**	99% | **100%**	99% | **100%**
R-III	99% | **98%**	99% | **100%**	99% | **100%**

MAIL

↓Training	MS-I	R-II	R-III
MS-I	94% | **99%**	99% | **100%**	99% | **100%**
R-II	99% | **99%**	99% | **100%**	100% | **100%**
R-III	99% | **100%**	99% | **100%**	99% | **100%**

Table 11.2: Off-line classification results [flows%/bytes%].

The reason why we focus on this specific set of applications is to be put in perspective with our research goals. First, we demonstrated in Part I that legacy statistical techniques failed at separating HTTP streaming from the rest of HTTP traffic. However, detecting HTTP streaming is arguably an important objective for an ISP servicing residential customers due to the key importance of this traffic for its clients and the load that this type of traffic imposes on an ISP infrastructure. Second, while eDonkey is the dominating p2p application in the network we monitor in terms of bytes and flows and turns out to be easy to detect, we observed that BitTorrent challenges the robustness of purely statistical classifiers [76]. Last but not least, it appears that monitoring this set of 5 applications is enough to classify on average 84% of bytes[4] on a daily basis for a continuous observation of an ADSL PoP over several months – see chapter 12.

The training set for each application class contains 10K flows: 2K flows from the considered class and 8K flows from the rest of the traffic. These values offered a good trade-off between the resulting classifier's performance and the training time. The training and testing sets never overlap. The procedure is the same in single-site and in cross-site experiments. For each case we repeat the experiment with randomly chosen training sets five times and present averaged results.

[4]Considering only the flows with more at least 4 data packets.

Class	Recall [flows% | bytes%]	Precision [flows% | bytes%]
WEB	$\leq 1\%$ | $\leq$ **1%**	$\leq 1\%$ | **1%**
HTTP-STR	$\leq 1\%$ | $\leq$ **1%**	$\leq 1\%$ | $\leq$ **1%**
EDONKEY	$\leq 1\%$ | $\leq$ **1%**	$\leq 1\%$ | $\leq$ **1%**
BITTORRENT	$\leq 1\%$ | $\leq$ **1%**	$\leq 1\%$ | **3.7%**
MAIL	$\leq 1\%$ | $\leq$ **1%**	1.7% | **1%**

Table 11.3: Maximum deviations of off-line classification results, for random training sets [flows%/bytes%].

11.5 Off-line Evaluation: Performance Results

In this section, we report on the performance of HTI in an off-line scenario using ADSL traces. We report on its robustness for a full cross-site scenario. We have highlighted in Part I the importance of performing cross site studies to detect data over fitting issues. We further contrast the results of HTI with several state of the art methods. We also discuss the impact of the machine learning algorithm used in HTI.

11.5.1 Overall Results

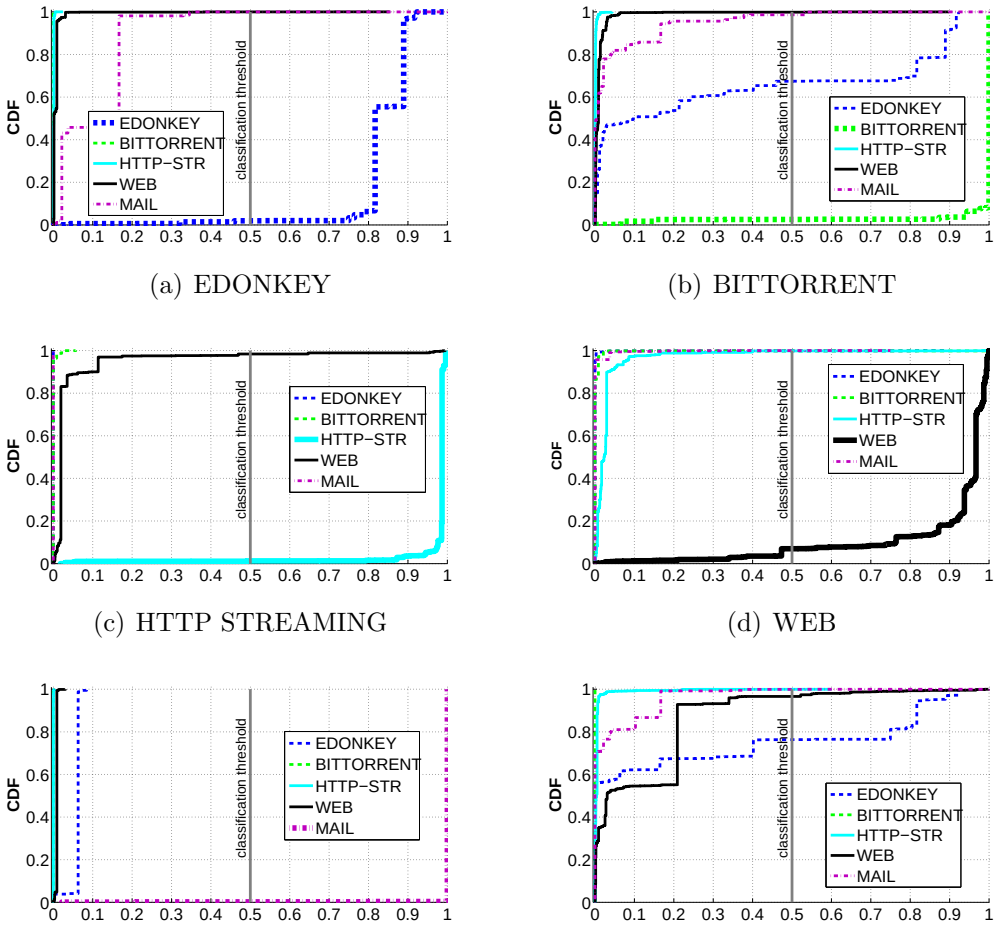
Table 11.2 presents classification results for each application class in terms of recall and precision in both flows and bytes in a full cross-site scenario. HTI achieves its accuracy objective as no class obtains recall and precision scores below 90%, both in bytes or flows. Please note that all the traffic (including MINOR_APP and UNKNOWN) is kept in this step for testing the performance. We repeat each experiment five times using randomly selected training sets to confirm that results are stable. Table 11.3 presents the maximum deviation from the results for each class and metric. Clearly, the choice of the training set has almost no impact on the results.

11.5.2 Multiple Classifications

In the classification phase each flow is tested against all the possible sub-model, and so, multiple matches might occur. As explained in Section 11.3, for each flow, we pick the class corresponding to the highest probability score. However, it is interesting to note that multiple classifications are rare events affecting at most a few percents of the bytes. Figure 11.2 depicts the cumulative distribution functions of each sub-model, along with the classification threshold. For example Figure 11.2c shows scores obtained by the HTTP-STR sub-model for all flows in the trace. Each curve on the plot represents distribution of scores for a given class (obtained using ground truth ODT).

Figure 11.2b shows that edonkey is the only application that may suffer from double classifications as almost 35% also match the BitTorrent sub-model. However, when picking the class with highest probability, all those flows are correctly classified as eDonkey.

In conclusion Figure 11.2 shows that the HTI sub-models offer very good separation between the application we consider and the rest of the traffic.



(a) EDONKEY (b) BITTORRENT

(c) HTTP STREAMING (d) WEB

(e) MAIL (f) MINOR_APP

Figure 11.2: Probability scores for each class sub-model. Training on MS-I test on RIII.

		1st packet			2nd packet			3rd packet			4th packet			port	signature
	β_0	dir.	push	size	dir.	push	size	dir.	push	size	dir.	push	size	X	X
WEB	-22.95	1.47	1.88	7.14	0.46	0.81	1.82	-0.24	1.90	-1.67	0.95	-1.87	-0.24	6.49	11.87
HSTR.	-96.75	88.90	-1.32	5.65	-2.15	-0.19	-3.85	3.34	-6.78	-3.86	1.42	0.37	-0.66	6.90	16.96
EDON.	-36.52	-0.45	-2.49	-33.46	-0.03	-2.81	-55.05	2.68	0.30	-64.26	2.28	34.30	-2.23	7.05	-
BITT.	-102.57	-0.00	-0.00	0.00	-0.00	-0.00	-0.00	0.00	-0.00	-0.00	0.00	0.00	-0.00	-0.00	205.13
MAIL	-41.03	-0.54	0.69	-89.78	-1.40	1.13	-2.58	-1.00	0.86	5.12	-0.33	31.64	-4.27	16.26	-

Table 11.4: Example of beta coefficients.

11.5.3 Which Method for Which Application?

The use of logistic regression has an additional advantage here in that it assigns the weights (β vector) to each input feature and a value close to zero reveals that the feature is useless to classify the corresponding class. Other machine learning algorithms provide interpretable models as well. For instance, by inspecting the height in the tree at which a feature is used as a discriminator provides insight about its importance in methods based on decision trees like C4.5. Note however that if trees are large, an accurate evaluation of the merit of each feature might be difficult. For instance using non quantized features in Part I typical decision tree we obtained had O(1000) nodes, which made interpretation practically impossible.

Table 11.4 presents the β values for each application of interest. We observed that some classes, like BITTORENT, could be classified with signatures only. In this case HTI narrows down to a standard deterministic DPI. For some other classes, like HTTP-STR, mixing signatures and legacy flow features is the key to obtain high recall and precision. HTI thus offers a powerful method to combine many sources of a priori heterogeneous information without requiring the practitioner to choose among contradictory results that could be obtained, e.g., by using in parallel a DPI and a (classical) statistical approach.

We report below on the key insights obtained with the study of the β values in Table 11.4:

- HTTP-STR and WEB: The use of purely statistical features allows to discriminate between both classes taken together and the rest of the traffic, but are not sufficient to differentiate between WEB and HTTP-STR. To prevent misclassification between those classes, we introduced payload signature (see Table 11.1). Doing so results in consistently high performance for both classes. The payload signature used for the WEB class is the negation of a similar regular expression used for the HTTP-STR class. The payload feature is thus set to 1 if the signature is *not* matched, and to 0 otherwise. This kind of signature would be useless in standard DPI, but here plays an important role for differentiating the HTTP-STR class from the WEB class.

- BITTORRENT: While Edonkey and BitTorrent are two p2p applications, different methods need to be used to detect them. Detecting BitTorrent using statistical features can lead to poor results in some cross-site experiments (see Part I). Adding payload signature leads to a perfect recall and precision[5].

- EDONKEY: We use only statistical features for eDonkey, which turns out to be sufficient (no additional payload signature). It is important to note that the statistical features of eMule are not altered by the obfuscation of protocol. If we used a classical payload signature instead, we would miss all the obfuscated eMule flows.

[5]Please note that our ODT version did not detect encrypted Bittorrent, thus we might find some of this traffic in the unknown class.

- MAIL: This is an a priori easy to detect class. Our HTI instance relies on port numbers and statistical features to take its decision.

Class	Feature		
	Statistical	**Port**	**Payload Signature**
WEB	V	V	V
HTTP-STR	V	V	V
EDONKEY	V	V	–
BITTORRENT	–	–	V
MAIL	V	V	–

Table 11.5: Methods playing important role in the model for each class.

11.6 HTI vs. State of the Art Methods

To further highlight the benefits of HTI, we contrast its performance with the one obtained with other state of the art methods on our traces (recap of Part I results). We also report on cross-site experiments reported in other studies (obviously on different traces).

11.6.1 Legacy Ports Method

Class	Recall [f.% | b.%]	Precision [f.% | b.%]	Port
WEB and HTTP-STR	99% | 93%	95 | 95%	80,443,8080,8081
EDONKEY	1% | 0.7%	99% | 99%	4662
BITTORRENT	6% | 3%	87% | 82%	6881-6889
MAIL	88% | 94%	99% | 100%	25,110

Table 11.6: Recall and precision of the legacy ports method (Trace R-III).

The classical method of relying on transport layer port numbers might still be effective for some applications. We present the results obtained with the port number method in Table 11.6. As HTTP-STREAMING and WEB use the same ports, we merged here both classes together. Almost all p2p traffic is exchanged on non legacy ports, leading to poor recall scores for the port number methods. However, if a flow uses a legacy port of EDONKEY, it is almost surely generated by this application, resulting in a high precision score for the port number method. These results are in line with the ones reported in a recent study by H. Kim et al. on other traces [58].

For the case of HTTP and MAIL, we observe from Table 11.6 that port numbers have a strong predictive power. However, note that they offer suboptimal performance as compared to HTI, which combines port number with other sources of information. In addition, the port number methods prevents from drilling down into the HTTP traffic to identify the application on top of HTTP.

11.6.2 Statistical Methods

In Part I, we evaluated several machine learning algorithms with purely statistical features (used in [5] and [62]) on *the same traces* as in the current study. We provide here a brief summary of our main findings.

We observed performance degradation for some applications in cross-sites experiments, e.g., recall of BitTorrent falls to 58% when trained on MS-I and tested on RII and RIII. Moreover, the classification of HTTP streaming with those methods lead to recall scores below 20% and precision scores below 60%.

Other studies tackled the problem of cross-site classification with statistical features. In [58] using heterogeneous traces from different networks, the authors present an SVM-based classifier trained on trace KAIST-I that achieved an overall recall of 49.8% when applied to trace PAIX-II. Similarly, when trained on trace KEIO-I and used on trace KAIST-I, the overall recall was only 63.6%. The authors in [62] reported a recall score of 58% in a cross-site experiment for p2p applications.

In conclusion, similarly to the port number method, purely statistical features can have a high predictive power, but not for all applications we consider. This is yet another confirmation that combining traffic classification methods as enabled by HTI is a promising avenue to devise robust traffic classifiers.

11.6.3 Impact of the Classification Algorithm

In this section, we discuss the impact of the exact machine learning algorithms used to instantiate HTI, We present only a summary of the experiments, indicating the most significant results. We considered two alternative algorithms to logistic regression, namely C4.5 and SVM (details on these algorithms can be found in Section 4.2.1).

Note that (i) the input features are the ones described in Section 11.4 and (ii), we re-generated all static cases and cross-site results.

Comparing these scenarios with the results obtained with logistic regression in Table 11.2 we observed that: There is a perfect match between all 3 HTI instances, with a largest deviation of 3% in scores (recall/precision for bytes or flows) over all single-site and cross-site cases.

We conclude that the results obtained in Table 11.2 are not specific to the logistic regression algorithm but can also be obtained with other machine learning algorithms. This further highlights that the strength of our approach lies in its hybrid nature (by combining heterogeneous sources of information) and proper feature quantization technique, rather than any specific characteristic of the logistic regression algorithm.

Chapter 12

HTI in the Wild - Production Deployment

Results of the chapter 11 demonstrated that our logistic regression instance of HTI is accurate and robust when applied on passively captured traces from different PoPs. We now report on its deployment in a production environment consisting of a large ADSL platform.

We first detail some implementation and measurements platform issues, followed by the validation of the classifier accuracy in live experiment. Finally we present the traffic measurements obtained with the classifier during six months of its run, and discuss efficiency issues of the HTI.

12.1 Implementation Details

We developed HTI on-line classifier in C using pcap library [63]. The simplified procedure is presented in Algorithm 3. We treat all incoming traffic packet by packet, and store flows (identified by four tuples) in a hash table. A new flow entry is created each time we observe the first SYN packet of a flow. UDP and ICMP packets are stored in a separate hash table for accounting purposes. In the current version of HTI we classify only TCP traffic.

Packets belonging to already classified flows are used only to compute statistics (number of bytes, flows, packets etc.). The ones belonging to the flows for which we missed the beginning or unidirectional flows are ignored for the classification and just counted for keeping track of statistics.

For each flow the most important decisions are made in the beginning of its existence. After the three way handshake, for each packet containing payload the function updating features vectors is called (it computes features for each application of interest). After the PKT_LIMIT (4 in our case) for packets with payload is reached the classifier function is called.

The simplified classifier procedure is presented in Algorithm 4. It takes as input flow (f) with the already computed per application features table, and takes advantage of the per application β vectors output during the training phase. The classification boils down to computing the probabilities that a flow belongs to

each application. The maximum probability is picked out of the results higher than a threshold TH (here we assume TH=0.5). In case none of the sub-models offers probability score higher than TH, the flow will remain unknown.

After running the classification procedure, flow label will remain unchanged. Further flow packets are used only for updating the flows statistics which are periodically dumped to a database. Both statistics dumping and garbage collection are triggered periodically every TIMER=20 seconds. Time is computed based on the packets timestamps.

Due to the architecture of our network, the MAC address of the switch is used to decide the direction of the packets. The direction (upstream/downstream) is important due to the nature of the features we are using (see Section 11.1.1 for details).

Algorithm 3: Traffic processing

```
Data: packet: IP packet
Data: f: struct {pkts, label, features[][]}
/* Structure store all flow information, here simplified for clarity */
procedure process_packet(packet)
if packet.SYN AND !packet.ACK then
    f ⟵ new_flow() /* Create new flow and add table entry */
end
if packet ∈ f then
    update_stats(f, packet) /* update flow statistics (bytes, packets) */
    if size(packet.payload) > 0 then
        if f.pkts < PKT_LIMIT then
            update_features(f, packet) /* Compute features for each appli. */
            f.pkts ⟵ f.pkts + 1
            if f.pkts == PKT_LIMIT then
                classify(f) /* Run classifier once we have enough packets */
            end
        end
    end
end
if packet.ts − last_ts < TIMER then
    last_ts = packet.ts
    output_stats() /* Periodically dump traffic statistics to RRD */
    remove_inactive_flows() /* Clean hash table */
end
```

12.2 Platform Details

We deployed our HTI classifier at an aggregating links of an ADSL platform servicing around 16,000 customers. It is a different PoP from the ones used in

Algorithm 4: Classification

Data: $\underline{f}$: struct {pkts, label, features[][] }
/* Structure store all flow information, here simplified for clarity */
Data: $\underline{Ai}$: struct {betas[],label}
/* Structure storing application model (beta vector) and label */
procedure classify(f)
foreach $A_i \in Applications$ **do**
  $beta[] \longleftarrow A_i.betas[]$ /* Model from the training phase */
  $X_i \longleftarrow f.features[A_i]$ /* Features for each aplli. computed from the flow */
  $P_i(A_i|X_i) = \beta_0 + \sum_{j=1}^{n} \beta_i x_i$ /*We compute probability for each appli.*/
end
if $\exists\ A_i such that P_i(Ai|Xi) > TH$ **then**
  $class \longleftarrow Ai.label$ that fulfills: $\arg\max_{k=1,\ldots,c}(P_i(Ai|Xi))$ and $P_i(Ai|Xi) > TH$
  $f.label \longleftarrow class$ /* Label flow with highest probability class */
end
else
  $f.label \longleftarrow unknown$ /* No match, we label flow as unknown */
end

Section 11.5 that were smaller in size. The traffic enters and leaves the ADSL platform through 4 high-end switches with two load balancers working at the IP level before and after the switches. As a consequence of this topology, the traffic of a user, which is mapped at a specific time to a certain IP address (the ISP allocates addresses dynamically), will go always through the same switch on its outbound and inbound path. However, the inbound and outbound switches need not to be the same as the two load balancers are working independently. Hence, there is a probability of 1/16 that the two directions of the traffic to and from a local IP address transits through the same switch. HTI is currently deployed on the machine connected to one of those switches and we need to observe the two directions of traffic to take a classification decision. As a consequence, we observe at a given point of time 1/16-th of the platform traffic. However, as IP addresses are reallocated to customers, the set of users we observe varies over time and on a long period of time, we can assume to see most of the users connected to the platform.

HTI runs on a core-duo Intel(R) Xeon(TM) CPU 2.80GHz machine with 2 GB of RAM equipped with a standard Ethernet NIC. The operating system used is Mandriva Linux.

12.3 Live Experiment Results

We have HTI running on our platform without interruption for more than 8 months (still ongoing). Starting in the beginning of 2010, every five minutes fresh statistics are shipped to a Round Robin Database [84]. RRD is specially is designed for the continuous monitoring of time series. Below we present selection

of some results obtained during this measurement.

Figure 12.1: One week of layer four breakdown [%].

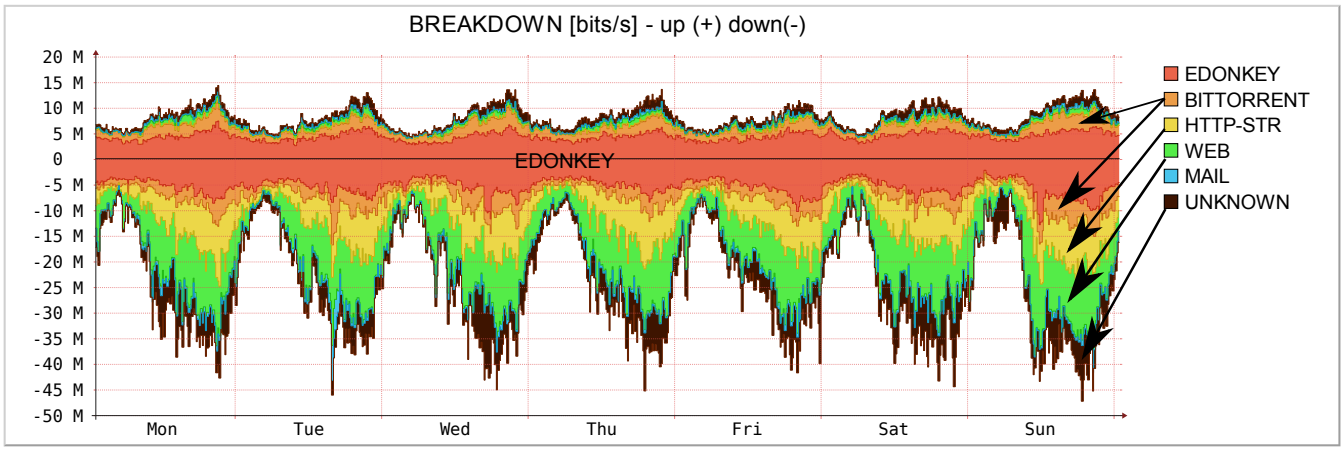


Figure 12.2: One week of application rates (only TCP, flows with 4+ data packets).

Figure 12.3: 180 Days of traffic classification (only TCP, flows with 4+ data packets).

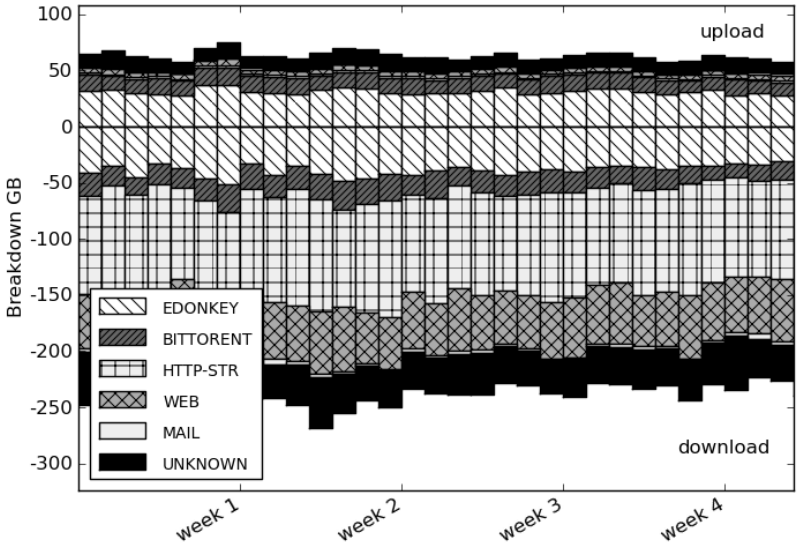


Figure 12.4: Daily traffic breakdowns for one month classified by HTI. In total 9.7 TB of the data (only TCP, flows with 4+ data packets).

12.3.1 One Week of Traffic

We present in Figure 12.2 the rates breakdown in upstream (+) and downstream (-) directions for the traffic of the platform. Some statistics about the traffic are presented in Table 12.1. Figure 12.1 shows transport protocol breakdown for the same week. We report key findings:

- As show in Figure 12.1, TCP clearly dominates as a transport layer in terms of bytes. Fraction of UDP based traffic is slightly more significant in the up direction. Other types of traffic including ICMP are negligible in terms of bytes, but still present - in order of 30 Kb/s.
- Using just a small number of classes we are able to classify 82 % of the bytes (of the long flows) during one week.
- Peer-to-peer traffic exhibits small daily variations in contrast to HTTP-Streaming and WEB, which are clearly dependent on users interaction (see Figure 12.2).
- HTTP based traffic clearly dominates in terms of volume (over half of the bytes) followed by EDONKEY and BITTORENT.
- HTTP_STREAMING achieve highest maximal rates followed by WEB and EDONKEY. The two HTTP based classes are also burstier in terms of the maximal daily amplitudes.
- UNKNOWN traffic although kept low in terms of aggregate bytes fractions (e.g. per day), tend to exhibit high short term peeks.

Class	Breakdown		Rate per class [Mb\s]			Daily amplitude [Mb\s]		
	[GB]	[%]	min	mean	max	min	mean	max
EDONKEY	527.40	25.44	3.37	7.14	16.77	5.60	8.43	12.50
BITTORENT	206.24	9.95	0.46	2.79	7.45	4.36	5.61	6.98
HTTP_STR	587.07	28.32	0.12	7.95	26.05	16.98	19.93	24.81
WEB	374.84	18.08	0.20	5.07	17.87	10.10	14.50	17.44
MAIL	41.05	1.98	0.00	0.56	4.37	2.44	3.05	4.36
UNKNOWN	336.43	16.23	0.44	4.55	15.11	9.61	12.11	14.44

Table 12.1: HTI statistics, from 5 July 2010 to 12 July 2010, total classified data: 1707.83GB.

The week that we present is qualitatively representative to the sets obtained with other time periods.

12.3.2 Six Months of Traffic

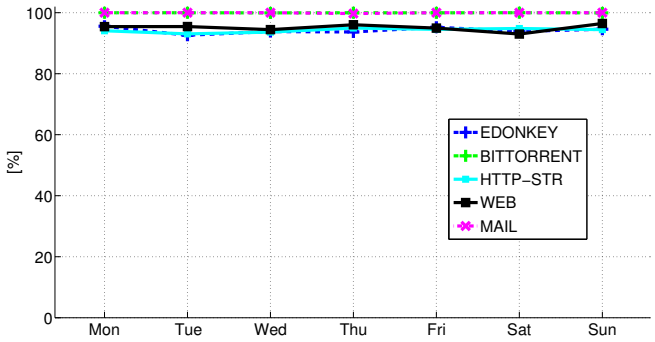
In Figure 12.3 we present the relative fraction of the traffic per day for almost half a year (180 consecutive days). Every bar plot represents a single day. Figure 12.4 shows one month of traffic represented with absolute values. We report key findings:

- Uplink traffic is still dominated by peer-to-peer applications: eDonkey followed by Bittorrent. In contrast, HTTP-STR and WEB traffic dominates the downlink direction. Increasing importance of the HTTP driven applications as compared to peer-to-peer is in line with the findings in [64, 28] where the traffic from large populations of ADSL users (30K and 100K respectively) was analyzed in 2009.
- The accumulated size of classified data was more than 9.7 TB during a month and represent more than 75 millions flows. While thanks to RRD properties the size of the database was in order of MBytes.
- The volumes for each type of traffic is fairly stable over the course of the six months period, despite the fact that the set of users we observe varies over time due to our measurement set-up. Although, we observe periodically peeks of some types of traffic, looking at the aggregates (e.g. Figure 12.3) we observe that traffic fractions remain constant over time.

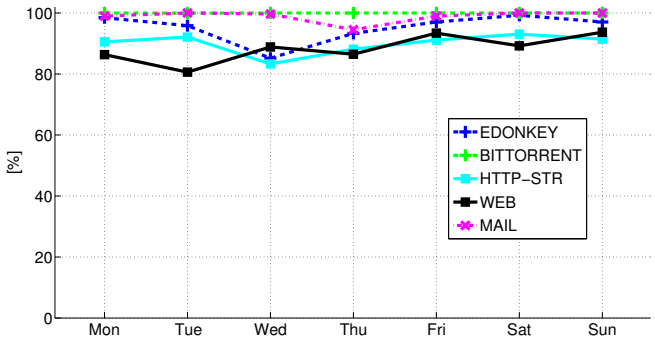
12.4 Model Validity

We used a model (consisting of several sub-models) generated out of the MS-I trace (see Section 11.5). Given that the MS-I trace was captured more than one year ago, we need to assess the recall and precision of the model.

To do so, we performed multiple captures in parallel to the constantly running HTI instance and cross-verified the results with our ground truth tool, ODT. Each



(a) flows



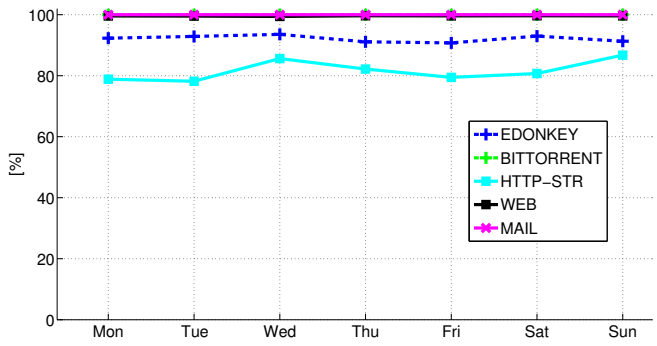
(b) bytes

Figure 12.5: Recall of HTI in live experiment.

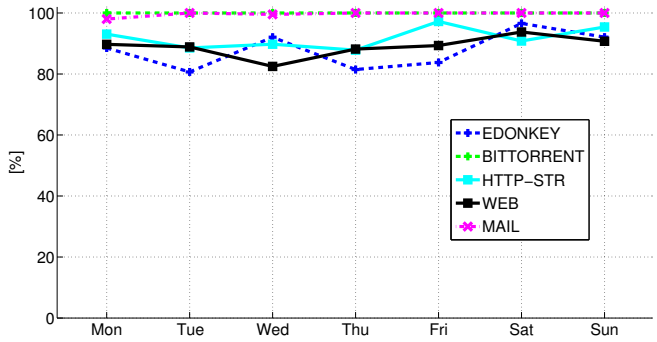
day (during a week) four captures (including payload) of one hour are performed, one in the morning, one in the afternoon and two in the evening, which aim at taking into account the diurnal traffic variations. Each capture contains 17-23 GB of data. We test our classifier for each of the traces (against ODT) and compute the average performance results for each day.

We present in Figures 12.5 and 12.6 the recall and precision in flows and bytes respectively of the model trained over the MS-I trace against the ground truth as obtained by ODT. We observe a very good agreement between the model and the ground truth. The discrepancy observed between recall in bytes and flows stems from a handful of large HTTP-STREAMING flows not correctly flagged by our HTI instance. More specifically as HTTP streaming flows are fairly small in numbers (as compared to HTTP flows in general) but large in size, mis-classifying them by labeling them as WEB and not HTTP-streaming affects our results both in bytes or flows.

As HTI relies partly (but not only, see Table 11.5) on the signatures in Table 11.1 to separate HTTP streaming flows from WEB flows, one can expect to improve results by extending our set of signatures.



(a) flows



(b) bytes

Figure 12.6: Precision of HTI in live experiment.

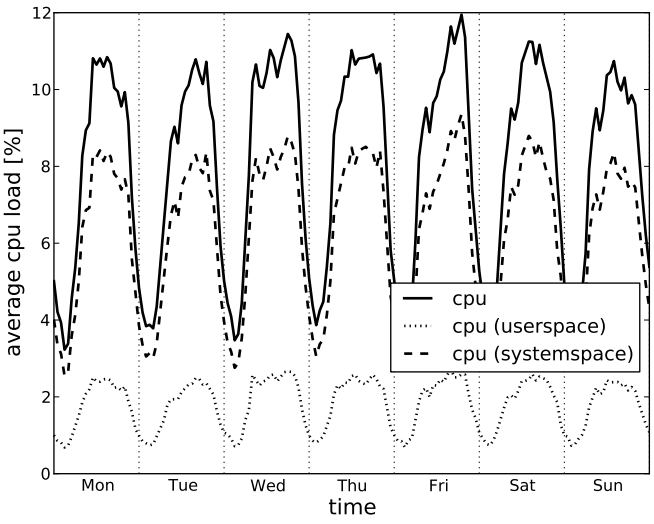


Figure 12.7: HTI CPU consumption evolution - one week (one hour aggregation).

12.5 Efficiency Issues

We report here on the efficiency of HTI in terms of CPU and memory consumption and also the time it takes to reach a classification decision.

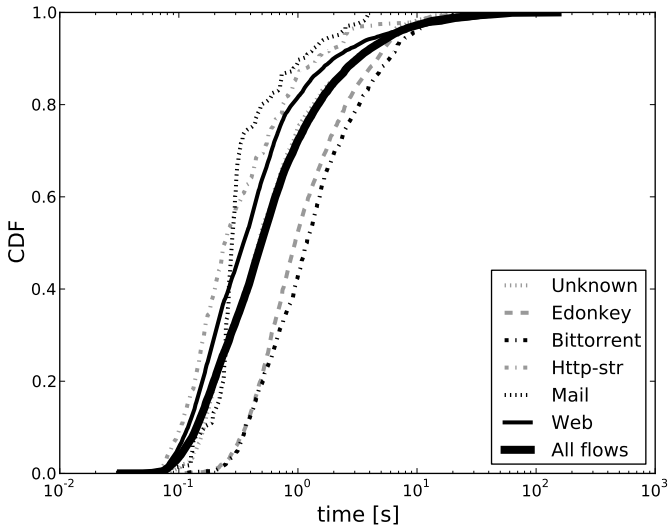


Figure 12.8: CDF of per flow time to take the classification decision.

12.5.1 CPU

We recorded the CPU footprint of HTI during its whole runtime. The average CPU consumption ranged between 3.7-12% and followed almost linearly the daily traffic rates evolutions. We present the load evolution in Figure 12.7. We observe that 80% of the CPU time is spent in the system (kernel) space (packets handling). We further analyzed which parts of the code are consuming most of the CPU using Valgrind profiler [94]. The profiler confirmed that the CPU is spending about 80% of its time handling the packets (pcap related functions). This is in line with our expectations as we use Ethernet cards and pcap library [63], which are not optimized for large scale captures (each packet causes several CPU interruptions [32]). The picture would be different if we had dedicated traffic capturing devices [26]. The good news is that the runtime of HTI related to the classification function (testing sub-models) is in the order of a few percents. The remaining time relates to flow management, garbage collection, etc.

We also increased the number of sub models (classes) to test in the live experiment. These additional models were faked ones obtained by picking random values of the weights associated to the features. The set of features used was the same as the other models. We added 5 models each 24 hours to reach a total of 50 sub-models. The total duration of this experiment was 10 days. We observed that increasing the number of sub-models to test for each flow had very little impact on the CPU utilization, less than 0.05% per sub-model added. Obviously if submodels would use more computationally intensive features, we would observe higher load on the CPU.

In conclusion, for our current deployment, the number of classes is not the bottleneck of our system and does not require to consider performance optimizations techniques like multi-threading.

12.5.2 Memory

In order to keep the memory usage low, a garbage collection mechanism periodically dumps non active or completed flows on the disk and removes them from the data structure stored in main memory. This mechanism permits to keep the memory consumption low, in the order of 20MB (1%). Only the state of the long flows needs to be stored for long periods of time. Note that while the classification process extracts all input features (port numbers, flow features and presence of signatures) from the first 4 data packets of a flow, we continue tracking flows longer than 4 data packets as we need to know their size in bytes and packets along with their ending time for accounting purpose.

12.5.3 Time to Classify

It is important to know how fast our system reaches a classification decision for each flow. The distribution of decision time for each application is depicted in Figure 12.8. The decision for 90% of the flows is made in less than three seconds, with a median value of 480 ms. Discrepancies among the per-class distributions primarily stem from the way those applications work, as it influences the time it takes to collect 4 data packets. It turns out that it is longer in case of peer-to-peer applications (median around 1s) than in case of mail or web based applications. We alsoe observe few flows with a very high values (more than 100s), although we did not analyse in details the reasons for this anomalies.

12.6 Discussion and Limitations

Although we believe that HTI, by allowing the combination of several classification methods, is an important step forward, it does not solve all classification issues. Some of the still existing limitations are:

- Currently, benchmarking our method we are in fact chasing ODT. We might face the situation in which our results appear good only because we use similar method (feature). However, for the moment there barely exist alternative. The building phase of the machine learning approaches (like HTI) requires some sort of ground truth. The ground truth problem might be solved by the recent efforts to generate reference sets, described in [35, 38].

- Results presented are sufficient for the passive monitoring of the network. However, the error rate for many applications is too high to risk QoS/Shaping actions.

- Statistical features can handle well some of the obfuscated protocols (e.g. EDONKEY); however handling VPN tunneling or other encrypted applications is still an open issue. For the moment we do not observe significant fractions of the unknown traffic, which might change with open source VPN solutions [73] gaining on popularity.

- We lack an oracle to trigger a re-training of the sub-models upon the detection of abnormal variations of the traffic in the corresponding class. It is easy to imagine that a new release of a protocol can alter the features we rely on. However, as the deployment of this version might be progressive in the user community, this would result in a simultaneous decrease in the rate of traffic of this class and an increase of the unknown traffic. One might consider integrating an anomaly detection method to detect those cases [14].

- Last but not least, most of the features/methods we rely on are very easy to imitate for a malicious application. Hence, the need to continue designing new robust classification techniques. HTI can help integrating such new techniques with previously proposed methods, due to its ability to test the discriminative power of different features.

12.7 Conclusion

Traffic classification is a key function in the management process of corporate networks and also for ISPs. Several different classes of methods have been proposed, especially deep packet inspection (DPI) and machine learning based approaches. Each approach is in general efficient for some classes of applications. To put it differently, the difficulty of detecting an application varies from one application to the other. It is a function of the peculiarities of the application or the result of the will of the designer to evade detection techniques. Based on these observations, we propose a technique, called Hybrid Traffic Identification (HTI) that enables to take advantage of the merits of different approaches. Any source of information (flow features, DPI decision, etc.) is encoded as a binary feature; the actual classification is made by a machine learning algorithm. We demonstrated that the good performance of HTI is not-dependent on a specific machine learning algorithm, and that any classification method can be incorporated to HTI as its decision could be encoded as a new feature.

We heavily tested HTI using different ADSL traces and 3 different machine learning methods. We demonstrated that not only HTI elegantly integrates the classical classification schemes, but, in addition, it is suitable for cross-site classification.

We further reported on the operational deployment of an HTI instance that takes its decision on the fly for an ADSL platform. We presented half a year continuous traffic measurement study, and reported on the key findings.

Overall, the HTI constitutes a significant step forward in the issue of traffic classification as it is flexible enough to enable the integration of new approaches that could be proposed in the future and thus enables an easy comparison of the added value they bring as opposed to legacy methods.

Part III

User Profiling

Chapter 13

User Profiles

13.1 Introduction

In Part II of the thesis we demonstrated how application classification can be used for the long term monitoring of residential platforms. We observed the evolutions of traffic profiles over several months. Our analysis, among other conclusions, confirmed some of the findings of other recent studies, namely the come back of HTTP driven traffic, which tends to cannibalize the peer-to-peer applications.

The statistics collected by the hybrid classifier can be used for monitoring and dimensioning purposes. In this chapter we address a different but related issue, we aim at understanding the traffic profile by analyzing the relative contributions of the users to the traffic mix.

User profiling is an important yet not enough explored area of traffic analysis. The current chapter aims at filling the gap between the low-level (network) level performance study and high level (application) study by profiling ADSL users. We use several techniques including hierarchical clustering to aggregate users' profiles according to their application mix. We pay a particular attention to the users that generated large fractions of traffic, which are commonly called "heavy hitters".

13.2 Relevant Publications for Part III

[79] M. Pietrzyk, L. Plissonneau,G. Urvoy-Keller,T. En-Najjary On profiling residential customers, To appear in third Workshop on Traffic Monitoring and Analysis TMA'11, April 2011, Vienna, Austria.

13.3 Related Work

A large scale study of Japanese residential traffic [17, 16], where almost 40% of the Internet traffic of the island is continuously observed, has revealed specific characteristics of the Japanese traffic: a heavy use of dynamic ports, which suggests a heavy use of P2P applications and a trend of users switching from ADSL

to FTTH technology to run P2P along with gaming applications. A recent study in the US [28], where the traffic of 100K DSL users has been profiled with a Deep Packet Inspection tool, has revealed that HTTP traffic is now the dominant protocol at the expense of P2P for the considered ISP, and probably for the US in general. This significant change in the traffic breakdown is not due to a decrease of P2P traffic intensity but a surge of HTTP traffic driven by HTTP streaming services like YouTube and Dailymotion. Similar results have been obtained in European countries. In Germany, a recent study [64] analyzed about 30K ADSL users and also observed that HTTP was again dominant at the expense of P2P traffic, for the same reason as in the US: a surge of video content distribution over HTTP. Early studies in France [86] for an ADSL platform of about 4000 users highlighted the dominance of P2P traffic in 2007 but a subsequent studies on the same PoP [82] or other PoPs under the control of the same ISP revealed similar traffic trend of HTTP traffic increasing at the expense of P2P both for ADSL [76] and FTTH access technology [96].

In the above studies, the application profiling of residential traffic was used to inform network level performance aspects, e.g., cachability [28] of content or location in the protocol stack of the bottleneck of transfers performed on ADSL networks [86], [64]. The study in [64] further reports on usage of the ADSL lines with a study of the duration of Radius sessions.

In the analysis carried in [56], the authors take a *graphlet* approach to profile end-host systems based on their transport-layer behavior, seeking users clusters and "significant" nodes. Authors in [31], take advantage of another clustering technique (namely Kohonen Self-Organizing Maps) to infer customers application profiles and correlate them with other variables (*e.g.* geographical location, customer age).

13.4 Data Set

To perform the user analysis we captured two dedicated data sets. Note that we could not take advantage of the data sets collected in Part I nor Part II as the user tracking information was not available in those cases. Using IP address to identify users is not an option for the reasons that we present in section 13.5.

The raw data for our study consists of two packet level traces collected on an ADSL platform of a major ISP in France (Table 13.1). Each trace lasts one hour and aggregates all the traffic flowing in and out of the platform. Both traces are fully indexed in the sense that both IP to user and connection to applications mapping are available. We use this data to discuss different options to profile the users.

Table 13.1: Trace summary.

Label	Start time	Duration	Bytes	Flows	TCP Bytes	TCP Flows	Local Users	Local IPs	Distant IPs
Set A	2009-03-24 10:53 (CET)	1h	31.7G	501K	97.2 %	30.7 %	1819	2223	342K
Set B	2009-09-09 18:20 (CET)	1h	41 G	796K	93.2 %	18.3 %	1820	2098	488K

13.5 Reliable Users Tracking

In this platform, ATM is used and each user is mapped to a unique pair of Virtual Path, Virtual Channel, identifiers. As the packet level trace incorporates layer 2 information, we can identify users thanks to this ATM layer information. This approach allows for reliable users tracking. Indeed, 18% of the users change their IP address at least once, with a peak at 9 for one specific user (during one hour). One could expect that the only source of error made when considering the IP address is that the session of the user is split onto several IP level sessions. However, we also noticed in our traces that a given IP could be reassigned to different users during the periods of observation. Specifically, 3% of the IPs were assigned to more than one user, with a peak of 18 re-assignments for one specific IP. Those results are in line with the ones obtained in [64] for a German residential operator.

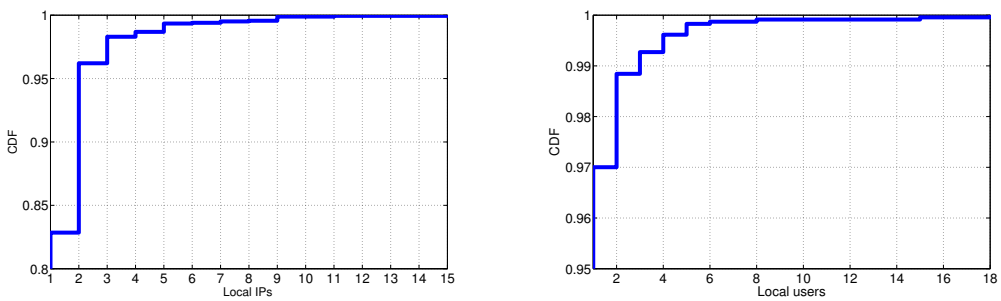
We present the details of the IP/users issues in Figure 13.1. The results clearly demonstrate that using IP for users identification would introduce a significant bias even in short time scales.

We developed an ad-hoc C++ trace parser that relies on libpcap to extract the per user statistics from the raw traces. Users' data was anonymized prior to analysis.

13.6 Applications Classes and Traffic Breakdown

Both traces are indexed thanks to ODT. This is the same tool that we have used in Part I and II for the benchmarking of the statistical classifiers (for the details see section 3.3).

The classes of traffic we use are reported in Table 13.2, along with the corresponding applications. The classes are very similar to the ones used in Part I. The main difference is that we now aggregate all the peer-to-peer applications into a single class (P2P). Moreover, the DOWNLOAD class is now separated from the former WEB class. This is due to the emergence of large file transfers from one-click hosting services [2], which are growing competitors of P2P file sharing services. As previously, the flows not classified by ODT (e.g. some encrypted



(a) IPs per user (b) Users per IP

Figure 13.1: IP aliasing illustration.

applications) are aggregated in the UNKNOWN class.

Table 13.2: Application classes.

Class	**Application/protocol**
WEB	HTTP and HTTPs browsing
UNKNOWN	–
P2P	eDonkey, eMule obfuscated, Bittorrent Gnutella, Ares, Others
MAIL	SMTP, POP3, IMAP, IMAPs POP3s, HTTP Mail
CHAT	MSN, IRC, Jabber Yahoo Msn, HTTP Chat
STREAMING	HTTP Streaming, Ms. Media Server, iTunes, Quick Time
OTHERS	NBS, Ms-ds, Epmap, Attacks
DB	LDAP, Microsoft SQL, Oracle SQL, mySQL
DOWNLOADS	HTTP file transfer, Ftp-data, Ftp control
GAMES	NFS3, Blizzard Battlenet, Quake II/III Counter Strike, HTTP Games
VOIP	Skype
NEWS	Nntp

13.6.1 Traffic Breakdown

We report in Table 13.3 the bytes breakdown views of the two traces, where the DB, CONTROL, NEWS, CHAT and GAMES classes have been omitted as they do not represent more than 1% of bytes and flows in any of the traces.

The traffic breakdown of our platform is similar to the ones from Part II of the thesis. Indeed, when summing all HTTP-based traffic in sets A or B, namely Web, HTTP Streaming and HTTP Download, more than 50% of the bytes in the down direction is carried over HTTP. Clearly, HTTP driven traffic dominates at the expense of background traffic that is due to P2P applications.

Table 13.3: Traffic Breakdown (classes with more than 1% of bytes only).

	Set A	**Set B**
Class	Bytes	Bytes
WEB	22.68 %	20.67 %
P2P	37.84 %	28.69 %
STREAMING	25.9 %	24.91 %
DOWNLOAD	4.31 %	6.47 %
MAIL	1.45 %	0.54 %
OTHERS	1.04 %	0.44 %
VOIP	0.36 %	1.67 %
UNKNOWN	5.26 %	15.79 %

13.7 Distribution of Volumes Per User

Understanding the relative contribution of each user to the total amount of bytes generated by dominating applications is important. Indeed these results, even if not surprising, justify the approach of focusing on heavy hitters in the last section of the chapter.

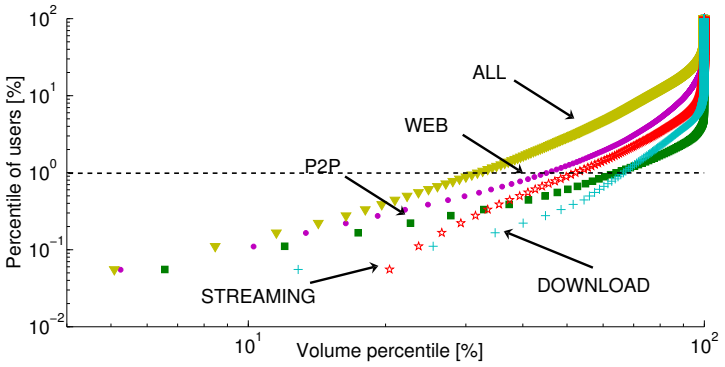


Figure 13.2: Contribution of users to traffic aggregate (global and per application, Set A).

In Figure 13.2, we present the contribution of users to the total traffic aggregate per application, with users sorted by decreasing volumes for the considered application (sets A and B being similar we focus on set A here). Note that we sum up, for a user, her bytes in both directions. We also include in the graph the overall contribution by user without distinguishing per application.

The fraction of users contributing to the majority of bytes in each application and even overall is fairly small. When looking at the global volumes generated, 90% of the bytes are generated by about 18% of users. For the same volume quantile, the fraction of users involved is even smaller when focusing on the applications generating most of the bytes (those represented in the graph). For the case of P2P traffic for instance, only 0.3% of the users contribute to 90% of the bytes uploaded or downloaded. We confirm here the well known phenomenon explored for instance in [30, 8]. This also holds for the STREAMING and WEB classes, which are two key classes in the dimensioning process of links of ISPs (for example bulk of Streaming users is active in the evenings).

A consequence of these highly skewed distributions is that the arrival or departure of some customers on the platform can potentially have an important impact on the traffic shape. For instance, the first four heavy users of STREAMING are responsible for about 30% of all streaming traffic.

The above observations further motivates our approach in the next section which is on profiling customers (and especially heavy hitters) from their application usage perspective.

13.8 Users Profiling

In this section, we address the issue of building an application level profile of customers that would characterize their network usage. The problem is challenging as it can be addressed from many different viewpoints. Here are some questions that one might want to answer: *Which amount of bytes or alternatively which number of flows should be observed to declare that a user is actually using a specific application? Can we characterize users thanks to the dominant application they use? What is the typical application profile of a heavy hitter? What is the typical application mix of the users?*

We address the above questions in the next paragraphs. We discuss several options to map applications to users. Our first approach focuses on the dominating applications for each user, we further discuss the precise profile of the top ten heavy hitters in both traces. Last paragraph presents typical users application mixture using a clustering technique.

13.8.1 Users Dominating Application

We present here a simple approach that provides an intuitive high level overview of the users activity: we label each user with her dominating application, the application that generated the largest fraction of bytes. Such an approach is justified by the fact that for both of our data sets, the dominating application explains a significant fraction of the bytes of the user. Indeed, for over 75% of the users, it explains more than half of the bytes. This phenomenon is even more pronounced when considering heavy users. Figure 13.3 presents the distribution of the fraction of the bytes explained depending on which application dominates users activity.

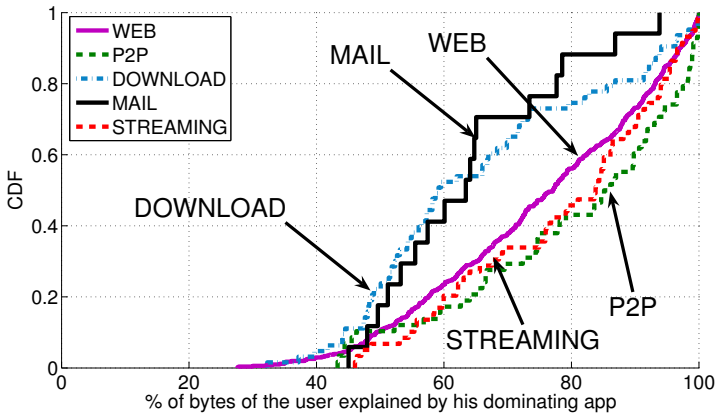


Figure 13.3: CDF of the fraction of bytes explained by the dominant application of each user (Set B).

The distribution of users per application with such an approach (dominant application) is reported in Table 13.4. As expected, the dominating class is WEB. We have more STREAMING than P2P dominated users. This complies with the intuition that every user, even if not experienced, can watch a YouTube

Table 13.4: Users dominating applications breakdown. Each user is labeled with his dominant application in terms of bytes (only users that transfered at least 100B: 1755 users, Set B).

Class	Fraction of Users	Fraction of Bytes explained
UNKNOWN	21%	12%
WEB	35%	19%
P2P	4%	35%
DOWN	5%	$\leq 1\%$
MAIL	1%	$\leq 1\%$
DB	9%	$\leq 1\%$
OTHERS	8%	$\leq 1\%$
CONTROL	7%	$\leq 1\%$
GAMES	$\leq 1\%$	$\leq 1\%$
STREAMING	7%	25%
CHAT	1%	$\leq 1\%$
VOIP	1%	2%

video, whereas using a P2P application requires installing a specific software (P2P client). The remaining dominant applications correspond to clients that generate a small amount of bytes most of the time. For instance, users that have DB, Others, Control or Games as dominating application generate an overall number of bytes that is extremely low.

We present in Figure 13.4 the users to application mapping for set B using the above dominant application approach. We adopt a representation in which

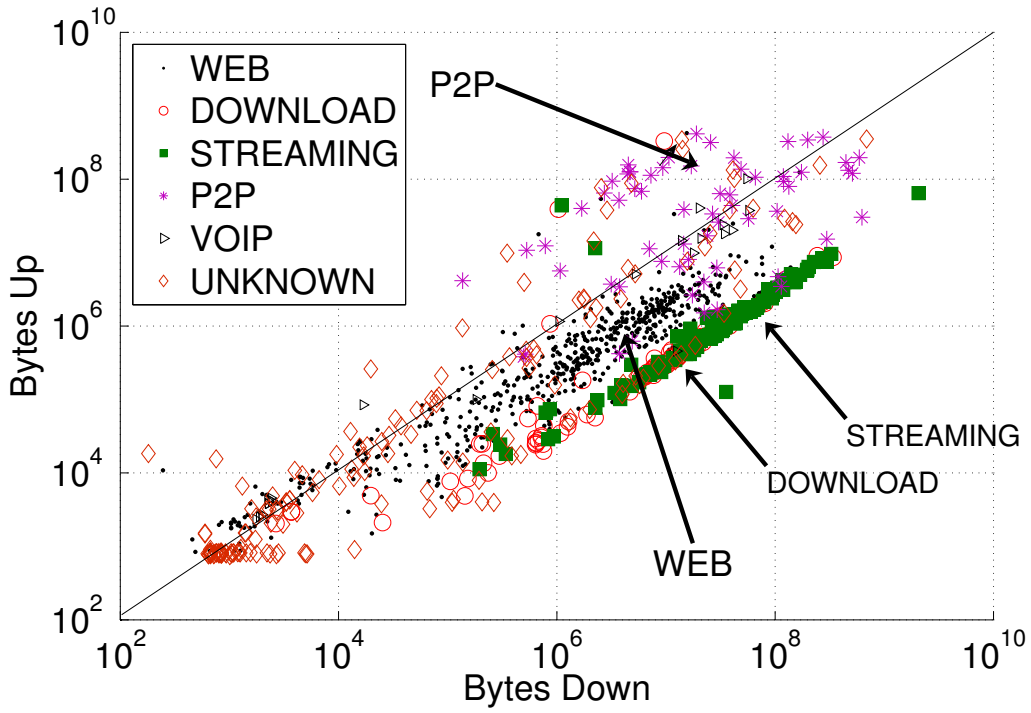


Figure 13.4: Users bytes Up/Down. Dominating application marked (Set B).

each user is characterized by the **total number of bytes** she generates in the up and down direction and label the corresponding point in a two dimensional space with the dominant application of the user in terms of bytes. We restricted the figure to a list of 6 important applications: Web, Streaming, VOIP, Download and P2P. We further added the users having majority of bytes in the Unknown class to assess their behavior.

Most important lesson of Figure 13.3 is that labeling a client with her dominating application is meaningful. Indeed, the dominating application in terms of bytes usually generates the vast majority of users' total volume. Customers with the same dominating applications are clustered together, and exhibit behavior typical for this application, which we detail below.

We observe from Figure 13.4 that:

- P2P heavy hitters tend to generate more symmetric traffic than Download and Streaming heavy hitters, which are far below the bisector.
- Web users fall mostly in between the bisector and the heavy hitters from the Download and Streaming classes. This is also in accordance with intuition as Web browsing often requires data exchange from clients to servers, e.g., when using Web search engines. This is in contrast to Streaming or Download where data flow mainly from servers to clients.
- Concerning Unknown users, we observe first that a significant fraction of them generated almost no traffic as they lay in the bottom-left corner of the plot. As for Unknown heavy hitters, we observe that they are closer on the figure to P2P heavy users than to client-server heavy users. This might indicate that there exist some P2P applications that fly below the radar of our DPI tool (encrypted Bittorrent would be a candidate). We further investigate this issue in the next section.

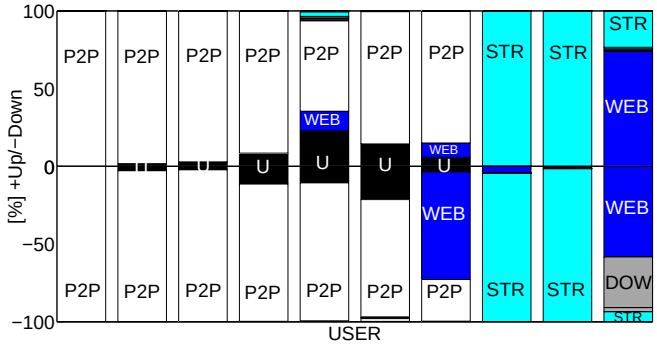
A last key remark is that the equivalent of Figure 13.4 for set A is qualitatively very similar, emphasizing the overall similarity of users activity in the two data sets (even if several months apart and at a different time of day).

The above analysis has again underlined the crucial role of (per application) heavy hitters. In the next section, we will focus on the top 10 heavy hitters in each trace. Each of them generated at least 0.6 GB of data and up to 2.1 GB and, overall, they are responsible for at least 1/4th of the bytes in each trace. We profile these users by accounting simultaneously for all the applications they use.

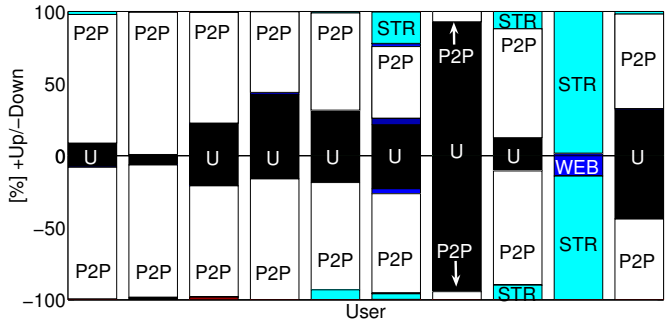
13.8.2 Top Ten Heavy Hitters

In this section, we focus on the top 10 heavy hitters for sets A and B. Note that these are distinct sets of users. It is a small, but very important group of customers from the ISP perspective, and better understanding of this group (aggregating 1/4 of total volume) might have significant impact on network provisioning and dimensioning. Figure 13.5(a) and 13.5(b) show the fraction of bytes they have generated in the up (positive values) and down direction (negative values) for each application. For sake of clarity, we put in the figure only the labels

of the significant applications for each user. We do observe from Figure 13.5(a) and 13.5(b) that heavy hitters, for the most part, use P2P applications. Streaming and (at least for one user) download activities seem also to give birth to some heavy hitters.



(a) Set A (Users generating 31% of the bytes in the trace)



(b) Set B (Users generating 24% of the bytes in the trace)

Figure 13.5: Top 10 heavy hitter users. Application usage profiles expressed in bytes fractions (U stands for UNKNOWN).

We also observe that unknown traffic seems to be associated mostly with P2P users (which is in line with Figure 13.4). This is an important finding from the perspective of the traffic classification. This user level information could be used as a feature in the hybrid classifier. It is also in line with the findings of chapter 8 where it is shown that a significant fraction of bytes in the unknown category is generated by P2P applications. In the present case, 67 % and 95 % of unknown bytes are generated by the users having in parallel a peer-to-peer activity for set A and B respectively. The reason why some of the P2P traffic might be missed by our DPI tool is out of the scope of this chapter. We note that there are at least two possible explanations: either we missed in our trace the beginning of a long P2P transfer and the DPI tool might not have enough information[1] to take a decision, or these users run currently unknown P2P applications in parallel.

[1]Application level information are often at the onset of transfers [5].

13.9 Users Application Mix

In the previous sections, we analyzed our users profile taking only bytes into account. This approach is informative and makes sense from a dimensioning viewpoint. However as the per applications volumes are very different – e.g., P2P applications tend to generate much more bytes than Web browsing – we miss some usage information with this purely byte-based approach. In this section, we explore a different perspective. We associate to each user a binary vector, which indicates her usage of each application. We take advantage of clustering techniques to present typical application mixtures.

13.9.1 "Real" *vs.* "Fake" Usage

We represent each customer with a binary vector: $A = [appli_1 \cdots appli_n]$ where n is the number of applications we consider. Each $appli_i \in \{0, 1\}$ is indication if the customer used application i or not.

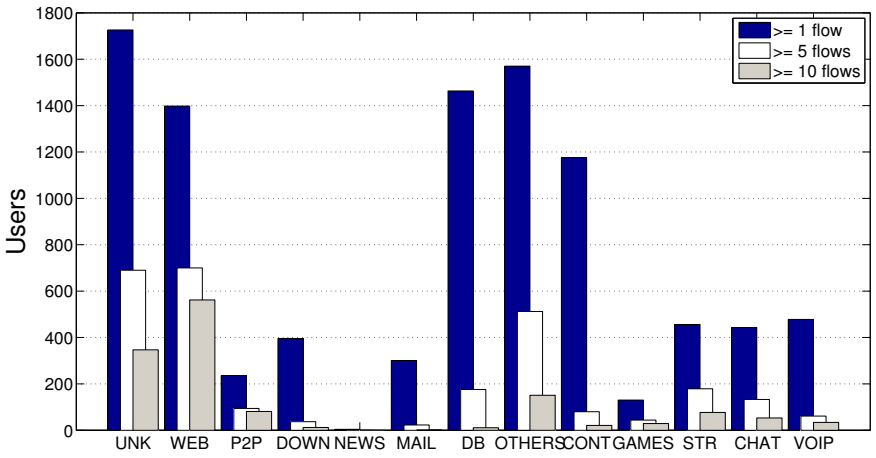
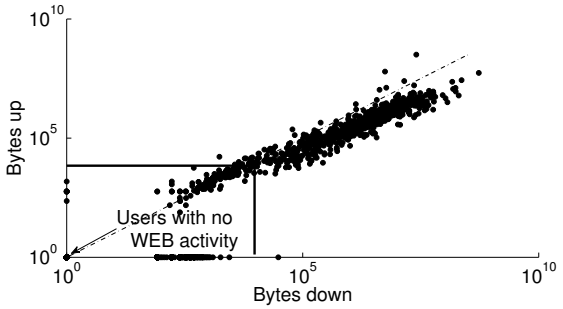
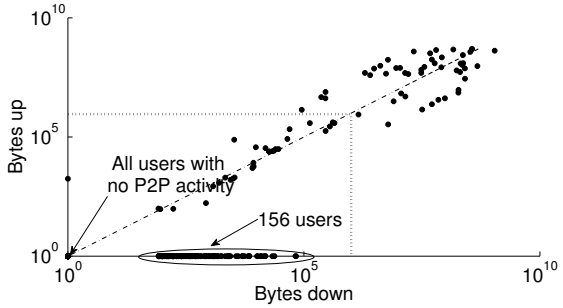


Figure 13.6: Application rank according to flows only.

The major concern in this part is how to declare that a user is actually using a certain application and we are not facing measurement (e.g. DPI) artifacts. Our first attempt was to use the number of flows to map users and applications. Figure 13.6 presents the distributions of the number of users per application (application popularity rank) for three different threshold values: 1 flow, 5 flows and 10 flows. We observe a significant drop of the number of users, when we consider 5 flows instead of one. Depending on the application, between 50% and 90% of users generated less than 5 flows. For instance, most of the users on our platform generated one web transfer, however, less than half of them generated more than 5 flows. This is against intuition that browsing activity should generate a single transfer due to the interconnection between sites, e.g., advertisements delivered by a third party or a content distribution network [48]. Current web sites tend to generate multiple connections for a single site (single search without browsing on google.com shows up to 7 connections). Similar problems might occur with other applications, for instance peer-to-peer user that closed his application, might still



(a) WEB



(b) P2P

Figure 13.7: Example of how the threshold is selected (Set A).

Table 13.5: Ad-hoc, per application and user minimum thresholds to declare application usage.

Class	**Volume**		**Number**	**Policy**
	Down	**Up**	**of Flows**	
WEB	300kB	500kB	20	All
P2P	1 MB	1 MB	10	Any
STREAMING	1 MB	1 MB	–	Any
DOWNLOAD	2 kB	1 kB	–	Any
MAIL	30kB	3 kB	–	All
GAMES	5 kB	5 kB	–	Any
VOIP	200kB	200kB	–	All
CHAT	10kB	10kB	–	Any

receive file requests for some time due to the way some P2P overlays work.

A conclusion we draw from this initial study is that accounting for flows only in the profiling process is not sufficient and might provide misleading results. We thus propose a different strategy.

We define per application heuristics to declare that a customer actually uses a class of application. To do that, we define minimal thresholds for three metrics: bytes up, bytes down and number of flows. Depending on the application any or all of the three thresholds need to be matched. We summarize the heuristics in Table 13.5. The values were derived from the data as it is exemplified in Figure 13.7 for P2P and WEB traffic.

13.9.2 Choice of Clustering

We have considered several popular clustering techniques to be able to understand the application mix of each user; see [40] for a complete reference on main clustering techniques. As explained in the previous paragraph, we have discretized the user's characteristics according to some heuristic threshold in order to keep only "real" application usage.

We have first tried the popular k-means clustering algorithm, and observed that the resulting clusters are difficult to match to applications. Moreover the choice of the number of clusters can dramatically change this representation.

Hierarchical clustering offers an easily interpretable technique for grouping similar users. The approach is to take all the users as tree leaves, and group leaves according to their application usage (binary values). We choose an agglomerative (or down-up) method:

1. The two closest nodes[2] in the tree are grouped together;
2. They are replaced by a new node by a process called linkage;
3. The new set of nodes is aggregated until there is only a single root for the tree.

With this clustering algorithm, the choices of metric and linkage have to be customized for our purpose.

We want to create clusters of users that are relatively close considering the applications mix they use. Among comprehensive metrics for clustering categorical attributes the Tanimoto distance [90] achieves these requirements. It is defined as follows:

$$d(x, y) = 1 - \frac{x^t \cdot y}{x^t \cdot x + y^t \cdot y - x^t \cdot y} \tag{13.1}$$

This means that users having higher number of common applications will be close to each other. For example, consider 3 users having the following mix of applications[3]:

User	Web	Streaming	Down	P2P
A	1	1	0	0
B	1	1	1	0
C	1	1	0	1

With Tanimoto distance, users B and C will be closer to each other because they have same total number of applications even if all 3 users share same common applications.

We use a complete linkage clustering, where the distance between nodes (consisting of one or several leaves) is the maximum distance among every pair of leaves of these nodes. It is also called farthest neighbor linkage.

[2] at first occurrence, nodes are leaves

[3] 1 means application usage and 0 means no application usage.

Due to the chosen metric, and as we chose not to prune the resulting tree, the hierarchical clustering leads to as many clusters as there are applications combinations: $\sum_{i=1}^{n} \binom{n}{i}$. In our case, we restrict the set of applications only to Web, Streaming, P2P and Download.

13.9.3 Application Mix

We present in Figure 13.8 and 13.9 the clustering results for the top 50 and second 50 most active users respectively. In total, the first one hundred users of the platform are responsible for 75% of the volume. We first consider only the classes generating most of the traffic, as described by Table 13.3 namely: Web, P2P, Streaming, and Download.

Each bar plot represents a single user and expresses his total volume share. Bar plots (thus users) are grouped into the sorted clusters. Each cluster, indicated by a different color groups the users that had the same applications. Thus close clusters in the graph are similar with respect to their application mix.

Considering only four applications, we have 15 possible combinations. What we observe is that some combinations are clearly more popular than others, while a few of them never occurred in our data. We present below a more precise analysis that reveals some insights about the typical users profiles.

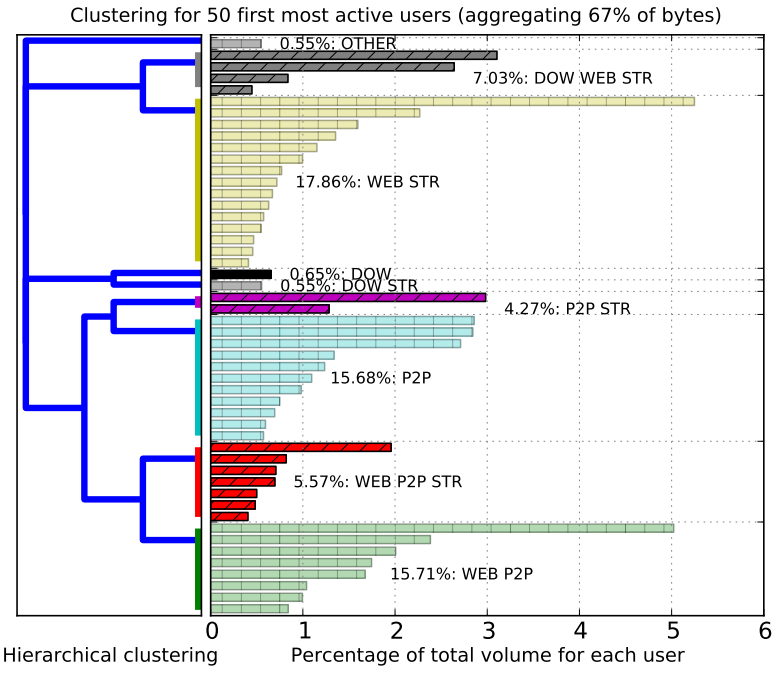


Figure 13.8: Application clustering for top 50 most active users (Set A).

Looking at the top 50 active users, we see that the P2P related clusters (P2P only, P2P + Web, P2P + Web + Streaming) dominates the top heavy hitters with 28 users. These P2P related clusters aggregate 36% of the total volume of the trace. Pure Web + Streaming profiles are the largest cluster in volume (18% of total), and have the biggest heavy hitter of the trace (over 5% of the whole traffic).

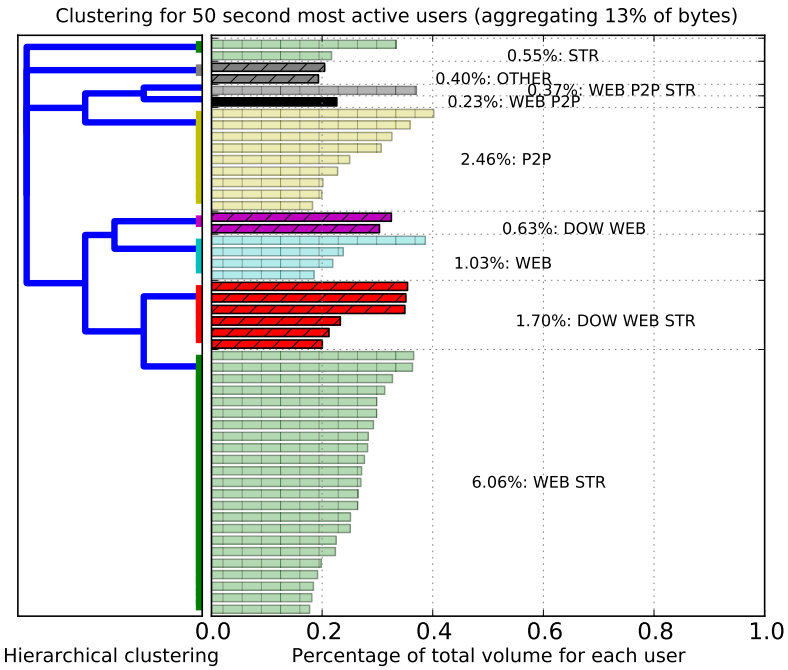


Figure 13.9: Application clustering for top 51-100 most active users (Set A).

The second set of 50 most active clients reveals a different picture. Here, over 23 clients use only Web and Streaming, while the group of P2P users is much smaller. In these users, the *usual browsing* activity is much present with Web related clusters regrouping 2/3 of users.

It is interesting to see that P2P and Streaming users form very distinct groups as only 10 of 100 most active users mix these 2 applications. This is also the case with Download whose profile never overlaps P2P. This shows that there is a set of clients that prefer classical P2P and another set of clients that use one click hosting to download contents.

The clustering process indeed partitions the first 50 heavy hitters according to P2P first, whereas for the second 50 heavy hitters the first partition occurs on Web.

13.9.4 Profile Agreement

Our data sets were captured several months apart. Two short snapshots are not enough to draw strong conclusions about the profile stability or evolution. We still took advantage of the ability to track users to make some initial comparisons.

First of all we observe almost perfect overlap of the users active between both sets. Deeper look at both sets revealed some interesting properties. Both traces have qualitatively similar traffic profiles, in terms of traffic breakdowns and heavy hitters profiles. But, the sets of users that are "responsible" for the traffic generated is different. For instance there is no single common user in the set of twenty most active users in both sets. What is more, looking at the fraction of common users in both sets that were "dominated" by certain application the overlaps are very low.

13.9.5 Application Mix - Discussion

Focusing on the first heavy hitters we observe that this family of users is dominated by P2P heavy-hitters. Even if streaming activity can also lead a user to become a heavy user, the main part of the volume generated by this class comes from a majority of medium users.

We conjecture that this situation will persist as the popularity of streaming continues to increase. Indeed, this increase of popularity is likely to translate into more users streaming more videos rather than a few users streaming a lot. If the main content providers switch to High Definition video encoding (which has bit-rates up to 4 times larger than standard definition), this could have a dramatic impact for ISPs.

13.10 Conclusions

In this part, we have proposed and illustrated several simple techniques to profile residential customers, with respect to their application level characteristics.

We have first presented an approach where the focus is on the dominant application of a user, which is justified by the fact that the dominant application explains a large majority of bytes for most users (in our data sets at least). This approach enables us to observe overall trends among moderately heavy and heavy users in a platform. We have next focused more deeply on the heavy hitters. Those heavy hitters are mostly P2P users, even though the global trend of traffic shows that Web and Streaming classes dominate. It is however understandable as P2P applications naturally tend to generate a few heavy hitters, while Web and Streaming tend to increase the volume of traffic of the average user.

We also devised an approach that seeks for common application mixes among the most active users of the platform. To this aim, we defined per application thresholds to differentiate real usage of an application from measurement artifacts. We use hierarchical clustering, that groups customers into a limited number of usage profiles. By focusing on the 100 most active users, divided in two equal sets, we demonstrated that:

- P2P users (pure P2P or mixed with other applications) are dominant in number and volume among the first 50 most active users;
- whereas in the second set of 50 most active users, the killer application is the combination of Web and Streaming.

Moreover while almost all P2P bytes are generated by the first 50 most active users, the Web + Streaming class is used by many users, and generates a fraction of bytes comparable (or higher) to P2P.

Our study sheds light on the traffic profile of the most active users in a residential platform, which has many implications for ISPs. However, we have only scratched the surface of the problem. Application at a larger scale of similar techniques, e.g., on much longer traces, would bring more insights than the snapshots

we analyzed. As part of our future work, we plan to further extend the analysis, by tracking the evolution of users profiles on the long term.

We strongly believe that hierarchical clustering on discretized attributes is a good approach because it greatly eases interpretation of the resulting clusters. Still, we plan to extend the discretization process from binary to (at least) ternary variables to take into account low/medium usage of an application *vs.* high usage.

Chapter 14

Thesis Conclusions

Traffic analysis and classification are relatively young, but already quite well explored domains. Building on the results of other teams, we tried to enrich the domain with our own contributions. Taking advantage of the access to quality trace data, we were able to complete some goals which would otherwise have been difficult to do for academic researchers. We now revisit the thesis claims and discuss the thesis work in general, highlighting the main contributions. Finally, we give our opinion on how this research could be extended in the future.

Portability is an issue. Part I of the thesis revisited the performance of flow feature methods using residential platform traces. We have demonstrated that statistical classification tools might suffer from data over-fitting, which prevents a simple strategy such as: train on the largest PoP (where the ground truth is available) and deploy on all other sites. Validation techniques, such as n-fold cross validation, might not be enough to draw conclusions about classifier portability and might lead to over-optimistic results. Portability issues are present for several sets of features and learning algorithms, as well as for their combinations, based on advanced techniques, such as principal components analysis.

Hybrid strategy pays off. To provide a remedy for some of the known issues of traffic classification, we propose a novel hybrid approach. By quantizing different sources of information into features, we enable synergy between diverse methods. We further illustrated how our method could be used in practice with a subset of important applications. An additional feature of our method is its transparency: the algorithm we use allows for an easy interpretation of the models, thus judging the discriminative role of features on an application basis.

Hybrid tool deployment. In Part II we reported on production deployment of our hybrid tool in the network of a large ISP which connects several thousands of customers to the Internet. We reported the results of several months of continuous 24/7 classification. To the best of our knowledge, this is the first time that the supervised machine learning traffic classifier has left the lab to be deployed in an operational network.

Logistic regression and feature quantization. We proposed to use a classical

machine learning algorithm along with a uniform features quantization technique. This technique offers high classification accuracy along with an easy to interpret model, as compared to the state-of-the art methods used in the community.

Case study of user profiling. The last part of the thesis shows one of the possible applications of traffic classifiers beyond the simple monitoring of the traffic aggregates. We propose several methods of building the application level profiles of the residential customers. Using clustering and data mining techniques, we group clients into a limited set of profiles that shed more light on the way the users benefit from their Internet access.

Our work tackled some aspects of traffic classification. Despite extensive efforts in the community, we believe there are still many issues to be addressed. Let us present a few examples of interesting research avenues that can be taken:

Formalizing the domain. As was mentioned earlier, a host of methods and algorithms have been proposed to solve the issue of traffic classification in recent years. However, comparing results between different works is very hard due to the lack of structure and a common understanding of notions in the domain, and the lack of quality shared data sets. Even such a priori obvious notions as, for example, the definition of application classes, have never been agreed upon. This leads to the situation in which class A in two works might contain significantly different types of traffic. We took a first attempt at addressing these ambiguities and proposed the first systematic traffic class definitions by taking advantage of ontology paradigms in [78][1]. In order to enable collaborative efforts regarding the problem, more standardization is required.

Measuring feature stability. As was demonstrated in Part I, it is easy to incorporate data set specific features into a classifier. The feature selection methods typically tend to optimize the classifier performance for a given data set. We proposed a method to test the stability of distribution, that could help with the issue uncovered in Chapter 7, but we think that further research is necessary to solve the problem and to differentiate between discriminative features and the site specific ones. One possibility would be to test the accuracy of the model involving the application itself.

Scalability. Although the common belief in the community is that DPI scales poorly to high bandwidth rates, a recent study has undermined this view [12]. It seems that the scalability of the classification method is highly dependent on the precise setup and traffic profiles. We believe that a detailed investigation of the scalability of the methods could be a very interesting and useful research subject.

Hide and seek. So far, with some exceptions, e.g. Skype, there is no extensive effort from most application developers to evade the classification. However, recently, in response to some governmental regulations, commercial anonymity

[1]The results of this work are not included in this thesis

solutions have been proposed. For a flat monthly fee, the user gains access to a bittorent proxy which offers enhanced anonymity [11]. We have not yet observed the popularity of such services, however, this might change with stricter regulations causing classification to be more challenging than ever before.

End user collaborative classification. Involving the end user in the process might be a possible solution. For such a system to be successful, it would first of all need to preserve users privacy, while at the same time enhancing the quality of service he/she receives. In such a scenario, instead of playing the hide and seek game, the ISP would offer added value services for the end users cooperation in the classification of his/her flows (for instance, by installing a software agent).

References

[1] Allot. http://www.allot.com/.

[2] D. Antoniades, E. P. Markatos, and C. Dovrolis. One-click hosting services: a file-sharing hideout. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet Measurements*, pages 223–234. ACM, 2009.

[3] R. Bendel and A. Afifi. Comparison of stopping rules in forward regression. *Journal of the American Statistical Association*, pages 46–53, 1972.

[4] L. Bernaille and R. Teixeira. Early recognition of encrypted applications. In S. Uhlig, K. Papagiannaki, and O. Bonaventure, editors, *Passive and Active Network Measurement*, volume 4427 of *Lecture Notes in Computer Science*, pages 165–175. Springer Berlin / Heidelberg.

[5] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12. ACM, 2006.

[6] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing skype traffic: when randomness plays with you. *SIGCOMM Comput. Commun. Rev.*, 37(4):37–48, 2007.

[7] J. A. Branco and A. P. Pires. A small data set may not be a drawback and a large data set may not be an advantage - in portuguese. Actas do XIV Congresso Anual da Sociedade Portuguesa de Estatistica, 2009.

[8] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *INFOCOM*, 1999.

[9] Bro-IDS. http://bro-ids.org.

[10] M. B. Brown and A. B. Forsythe. Robust tests for equality of variances. *Journal of the American Statistical Association Volume 69, Issue 10*, 1974.

[11] Btguard. http://btguard.com/.

[12] N. Cascarano, L. Ciminiera, and F. Risso. Improving cost and accuracy of dpi traffic classifiers. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 641–646, New York, NY, USA, 2010. ACM.

[13] N. Cascarano, A. Este, F. Gringoli, F. Risso, and L. Salgarelli. An experimental evaluation of the computational cost of a dpi traffic classifier. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 30 2009.

[14] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:15:1–15:58, July 2009.

[15] Z. Chen, B. Yang, Y. Chen, A. Abraham, C. Grosan, and L. Peng. Online hybrid traffic classifier for peer-to-peer systems based on network processors. *Appl. Soft Comput.*, 9(2):685–694, 2009.

[16] K. Cho. Broadband Traffic Report. *Internet Infrastructure Review*, 4:18–23, August 2009.

[17] K. Cho, K. Fukuda, H. Esaki, and A. Kato. The impact and implications of the growth in residential user-to-user traffic. In *SIGCOMM*, 2006.

[18] J. Cohen. *Statistical power analysis for the behavior sciences (2nd ed.)*. Laurence Erlbaum Associates, new Jersey, 1988.

[19] F. Constantinou and P. Mavrommatis. Identifying known and unknown peer-to-peer traffic. In *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on*, pages 93–102, July 2006.

[20] Dailymotion. http://dailymotion.com/.

[21] A. Dainotti, W. Donato, and A. Pescapé. Tie: A community-oriented traffic classification platform. In *Proceedings of the First International Workshop on Traffic Monitoring and Analysis*, TMA '09, pages 64–74, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] M. Dusi, F. Gringoli, and L. Salgarelli. Quantifying the accuracy of the ground truth associated with internet traffic traces. *Comput. Netw.*, 55:1158–1167, April 2011.

[23] eMule. http://www.emule-project.net/.

[24] T. En-Najjary and G. Urvoy-Keller. A first look at traffic classification in enterprise networks. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, pages 764–768, New York, NY, USA, 2010. ACM.

[25] T. En-Najjary, G. Urvoy-Keller, M. Pietrzyk, and J.-L. Costeux. Application-based feature selection for internet traffic classification. In *Teletraffic Congress (ITC), 2010 22nd International*, pages 1 –8, 2010.

[26] Endace. http://www.endace.com.

[27] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, New York, NY, USA, 2006. ACM.

[28] J. Erman, A. Gerber, M. T. Hajiaghayi, D. Pei, and O. Spatscheck. Network-aware forward caching. In *WWW*, 2009.

[29] FastFlux. http://www.darkreading.com/security/perimeter/showarticle.jhtml?articleid=208804630.

[30] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: methodology and experience. *IEEE/ACM Trans. Netw.*, 9(3):265–280, 2001.

[31] F. Fessant, V. Lemaire, and F. Clarot. Combining Several SOM Approaches in Data Mining: Application to ADSL Customer Behaviours Analysis. In *Data Analysis, Machine Learning and Applications*. 2008.

[32] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, and D. Rossi. Live traffic monitoring with tstat: Capabilities and experiences. In *WWIC*, pages 290–301, 2010.

[33] A. Finamore, M. Mellia, M. Meo, and D. Rossi. Kiss: Stochastic packet inspection classifier for UDP traffic. *Networking, IEEE/ACM Transactions on*, 18(5):1505 –1515, 2010.

[34] A. Finamore, M. Meo, D. Rossi, and S. Valenti. Kiss to abacus: A comparison of p2p-tv traffic classifiers. In F. Ricciato, M. Mellia, and E. Biersack, editors, *Traffic Monitoring and Analysis*, volume 6003 of *Lecture Notes in Computer Science*, pages 115–126. Springer Berlin Heidelberg, 2010.

[35] S. M. G. Szabo, D. Orincsay and I. Szabo. On the validation of traffic classification algorithms. In *Passive and Active Measurement conference (PAM 08)*, 2008.

[36] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos. Link homophily in the application layer and its usage in traffic classification. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –5, 2010.

[37] O. Grémillet, P. Gonçalves, P. V.-B. Primet, and A. Dupas. Traffic classification techniques supporting semantic networks. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, pages 463–467, New York, NY, USA, 2010. ACM.

[38] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and k. c. claffy. Gt: picking up the truth from the ground for internet traffic. *SIGCOMM Comput. Commun. Rev.*, 39(5):12–18, 2009.

[39] A. Hafsaoui, D. Collange, and G. Urvoy-Keller. Revisiting the performance of short TCP transfers. In *Networking 2009, 8th IFIP International Conference on Networking, May 11-15, 2009, Aachen, Germany / Also published in "Lecture Notes in Computer Science", Vol 5550/2009, ISBN: 978-3-642-01398-0*, 05 2009.

[40] J. Han. *Data Mining: Concepts and Techniques (Second Edition)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[41] J. W. Hardin and J. W. Hilbe. *Generalized Linear Models and Extensions, 2nd Edition*. StataCorp LP, 2007.

[42] W. L. Hayes. *Statistics for Psychologists*. Holt, Rinehart & Winston, New York, 1963.

[43] L. V. Hedges and I. Olkin. *Statistical methods for meta-analysis*. New York: Academic Press, 1985.

[44] D. W. Hosmer and S. Leshow. *Applied Logistic Regression*. New York ; Chichester ; Brisbane : J. Wiley and Sons, cop., 2001.

[45] Iana. http://www.iana.org/assignments/port-numbers.

[46] M. Iliofotou, B. Gallagher, T. Eliassi-Rad, G. Xie, and M. Faloutsos. Profiling-by-association: A resilient traffic profiling solution for the internet backbone. In *ACM CoNEXT*, Philadelphia, CA, USA, December 2010.

[47] M. Jaber and C. Barakat. Enhancing application identification by means of sequential testing. In *Networking*, pages 287–300, 2009.

[48] Y. Jin, E. Sharafuddin, and Z.-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 49–60, 2009.

[49] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 5^{th} edition, 2002.

[50] I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[51] J.R.Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

[52] Y. L. Jun Li, Shunyi Zhang and J. Yan. Hybrid internet traffic classification technique. *Journal of Electronics*, 26(1):685–694, 2009.

[53] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is p2p dying or just hiding? [p2p traffic measurement]. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, pages 1532–1538 Vol.3, Nov.-3 Dec. 2004.

[54] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy. Transport layer identification of p2p traffic. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134, New York, NY, USA, 2004. ACM.

[55] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, 35(4), 2005.

[56] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos. Profiling the end host. In *Passive and Active Measurement conference (PAM 07)*, April 2007.

[57] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.

[58] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *CONEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12, New York, NY, USA, 2008. ACM.

[59] J. Kittler, W. Menard, and K. Phillips. Weight concerns in individuals with body dysmorphic disorder. *Eating Behaviors*, 8:115–120, 2007.

[60] L7filters. http://l7-filter.sourceforge.net/.

[61] Leurrecom. http://www.leurrecom.org.

[62] W. Li, M. Canini, A. W. Moore, and R. Bolla. Efficient application identification and the temporal and spatial stability of classification schema. *Computer Networks*, 53(6):790 – 809, 2009.

[63] Libpcap. http://sourceforge.net/projects/libpcap/.

[64] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 90–102, New York, NY, USA, 2009. ACM.

[65] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41 –48, Aug. 1999.

[66] A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In *Passive and Active Measurement conference (PAM 05)*, pages 41–54, 2005.

[67] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005.

[68] D. Nechay, Y. Pointurier, and M. Coates. Controlling false alarm/discovery rates in online internet traffic flow classification. In *INFOCOM 2009, IEEE*, pages 684 –692, 2009.

[69] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied Linear Statistical Models*. Irwin, Chicago, 5^{th} edition, 2004.

[70] T. Nguyen and G. Armitage. Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks. In *in Proceedings of the IEEE 31st Conference on Local Computer Networks*, pages 369–376, 2006.

[71] T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys and Tutorials, IEEE*, 10(4):56–76, 2008.

[72] T. Nix and J. Barnette. The data analysis dilemma: ban or abandon. a review of null hypothesis significance testing. *Research in the Schools*, 5(2):3–14, 1998.

[73] OpenVPN. http://openvpn.net/.

[74] J. C. Paolillo. *Variable Rule Analysis: Using Logistic Regression in Linguistic Models of Variation*. Chicago University Press, 2002.

[75] B.-C. Park, Y. Won, M.-S. Kim, and J. Hong. Towards automated application signature generation for traffic identification. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 160 –167, 2008.

[76] M. Pietrzyk, J.-L. Costeux, T. En-Najjary, and G. Urvoy-Keller. Challenging statistical classification for operational usage : the adsl case. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 2009.

[77] M. Pietrzyk, T. En-Najjary, G. Urvoy-Keller, and J.-L. Costeux. Hybrid traffic identification. Technical Report EURECOM+3075, Institut Eurecom, France, 04 2010.

[78] M. Pietrzyk, L. Janowski, and G. Urvoy-Keller. Toward systematic methods comparison in traffic classification. In *Proceedings of the 2nd International Workshop on TRaffic Analysis and Classification TRAC 2011 (to appear)*, TRAC '11, pages xx–xx, 2011.

[79] M. Pietrzyk, L. Plissonneau, G. Urvoy-Keller, and T. En-Najjary. On profiling residential customers. In *Proceedings of the Third International Workshop on Traffic Monitoring and Analysis (to appear)*, TMA '11, pages 1–14, Berlin, Heidelberg, 2011. Springer-Verlag.

[80] M. Pietrzyk, G. Urvoy-Keller, and J.-L. Costeux. Digging into kad users' shared folders. *in poster of SIGCOMM Comput. Commun. Rev.*, 38, August 2008.

[81] M. Pietrzyk, G. Urvoy-Keller, and J.-L. Costeux. Revealing the unknown adsl traffic using statistical methods. In *COST 2009 : Springer : Lecture Notes in Computer Science, Vol 5537, 2009.*, May 2009.

[82] L. Plissonneau, T. En-Najjary, and G. Urvoy-Keller. Revisiting web traffic from a DSL provider perspective : the case of YouTube. In *ITC Specialist Seminar on Network Usage and Traffic*, 2008.

[83] Qosmos. http://www.qosmos.com/.

[84] RRDtool. http://oss.oetiker.ch/rrdtool/.

[85] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 512–521, New York, NY, USA, 2004. ACM.

[86] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. W. Biersack. Performance Limitations of ADSL Users: A Case Study. In *PAM*, 2007.

[87] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, and D. Collange. A root cause analysis toolkit for TCP. *Computer Networks. Volume 52, Issue 9, 26 June 2008*, 2008.

[88] Snort. http://www.snort.org/.

[89] 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, June 2007.

[90] T. Tanimoto. An elementary mathematical theory of classification and prediction. In *IBM Program IBCLF*, 1959.

[91] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci. Unconstrained endpoint profiling (googling the internet). *SIGCOMM Comput. Commun. Rev.*, 38(4), 2008.

[92] Tstat. http://tstat.tlc.polito.it/.

[93] J. Tucker and D. J. Tucker. Neural Networks Versus Logistic Regression In Financial Modelling: A Methodological Comparison. In *in Proceedings of the 1996 World First Online Workshop on Soft Computing (WSC1)*, 1996.

[94] Valgrind. http://valgrind.org/.

[95] E. Vittinghoff, D. V. Glidden, and S. C. Shiboski. *Regression methods in biostatistics : linear, logistic, survival, and repeated measures models.* Springer New York, 2005.

[96] G. Vu-Brugier. Analysis of the impact of early fiber access deployment on residential Internet traffic. In *ITC 21*, 2009.

[97] J. T. Walker and S. Maddan. *Statistics in criminology and criminal justice, Third Edition.* Sudbury, Mass., USA, Jones and Bartlett Publishers, 2009.

[98] WEKA. http://www.cs.waikato.ac.nz/ml/weka/.

[99] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, 2006.

[100] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4p: provider portal for applications. *SIGCOMM Comput. Commun. Rev.*, 38(4), 2008.

[101] Youtube. http://youtube.com/.

Appendix A

Portability Problem - Formally

Consider a classifier based on Linear Discriminant Analysis [65], that classifies the flows of two datasets S_1 and S_2, belonging to two applications A_1 and A_2, based on a set of p flow features, $\mathbf{X} = (X_1, \ldots, X_p)^t$. Assume that in each dataset and each application the features have a multivariate normal distribution, with the same covariance matrix ($\boldsymbol{\Sigma}$) and different mean values $\boldsymbol{\mu}_{ij}$, $i = 1, 2$, and $j = 1, 2$, i.e. $\mathbf{X}|(A_i, S_j) \sim \mathcal{N}_p(\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}_{i1} = \boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_{i2} = \boldsymbol{\mu}_i + k\mathbf{1}$, $k \in \mathbb{R}^+$, $\mathbf{1}$ is a vector of ones, and $i = 1, 2$. Thus, for each application the means in the two datasets differ by k. The classification rule that minimizes the total probability of misclassification (TPM), for S_j, $j = 1, 2$ is given by [49]:

$$\begin{cases} \text{Classify } \boldsymbol{x}_0 \text{ in A}_1 \; if \; \boldsymbol{\alpha}^t \boldsymbol{x}_0 \geq m_j + \ln \frac{p_2}{p_1}; \\ \text{Classify } \boldsymbol{x}_0 \text{ in A}_2, \text{ otherwise,} \end{cases}$$

where $\boldsymbol{\alpha} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{1j} - \boldsymbol{\mu}_{2j})$, $m_1 = \frac{1}{2}\boldsymbol{\alpha}^t(\boldsymbol{\mu}_{1j} + \boldsymbol{\mu}_{2j})$, $m_2 = m_1 + k\boldsymbol{\alpha}^t\mathbf{1}$, and p_i is the a priori probability of A_i, considered the same for both traces (note that $p_1 + p_2 = 1$). In this case,

$$TPM = p_1\Phi\left(\frac{\ln\left(\frac{p_2}{p_1}\right) - \frac{\Delta^2}{2}}{\Delta}\right) + p_2\Phi\left(-\frac{\ln\left(\frac{p_2}{p_1}\right) + \frac{\Delta^2}{2+}}{\Delta}\right),$$

where Δ is the Mahalanobis distance between the two mean vector $\boldsymbol{\mu}_{1j}$ and $\boldsymbol{\mu}_{2j}$, which does not depend on j, i.e. $\Delta^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$. Note that in this example, the TPM is the same for both traces.

If we now use the optimal classification rule for S_1 to classify observations from S_2, it can be shown that the associated total probability of misclassification, TPM_{12}, is

$$\begin{aligned} TPM_{12} &= p_1\Phi\left(\frac{\ln\frac{p_2}{p_1} - \Delta^2/2 - k\boldsymbol{\alpha}^t\mathbf{1}}{\Delta}\right) \\ &+ p_2\Phi\left(-\frac{\ln\frac{p_2}{p_1} + \Delta^2/2 - k\boldsymbol{\alpha}^t\mathbf{1}}{\Delta}\right). \end{aligned}$$

Let us admit $\boldsymbol{\alpha}^t\mathbf{1} \neq 0$, if $k \to +\infty$ then $\text{TPM}_{12} \to p_1 I_{]-\infty,0[}(\boldsymbol{\alpha}^t\mathbf{1}) + p_2 I_{]0,+\infty[}(\boldsymbol{\alpha}^t\mathbf{1})$,

where $I_A(x)$ is 1 when $x \in A$ and 0 otherwise. In fact, this means that for k sufficiently large, the optimal classifier for S_1 assigns all S_2 flows in A_2 with high probability, thus the flows generated from application A_1 in trace S_2 are wrongly classified and the error rate is p_1. Figure A.1 illustrates this problem. Let us consider two different traces, S_1 and S_2, each generated by two applications. Each flows is characterized by two features such that $\boldsymbol{\mu}_1 = (-1, -1)^t$, $\boldsymbol{\mu}_2 = (1, -1)^t$, and $k = 10$. The lines in figure A.1 represent the optimal classification boundaries, considering each dataset separately. Clearly, using the optimal decision rule from S_1 to classify flows from S_2 results in all flows assigned to application A_2, and because of that TPM_{12} is equal to the probability of a flow being generated by application A_1, i.e. p_1.

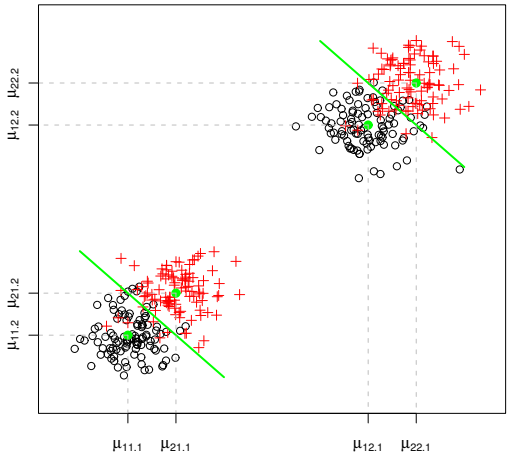


Figure A.1: Two traces where ∘ (+) represents a flow generated from application A_1 (A_2). The lines are the optimal boundaries, that minimizes TPM.

Appendix B

Synthèse en Français

B.1 Introduction

L'Internet a connu une croissance continue à un rythme élevé ces deux dernières décénnies. Nous sommes passés dans cette période de quelques applications client-serveur à une foule d'applications distribuées utilisées par des millions d'utilisateurs, presque 2 milliards en juin 2010. En principe, l'Internet et son modèle de transport TCP/IP est conçu dans une optique "best-effort", c'est-à-dire sans garantie de délivrance. Conçu il y a plus de 40 ans, ce modèle, qui permet toujours à l'Internet de fonctionner de manière satisfaisante, ne permet toutefois pas de répondre à certains besoins cruciaux de gestion du réseau.

Le besoin spécifique que nous traitons dans cette thèse est la classification des flux, c'est-à-dire la capacité d'établir une correspondance entre un flux IP et une application. Cette tâche est complexe car la pile TCP/IP n'offre pas de moyen fiable pour obtenir cette correspondance (car une application peut assigner elle-même le port TCP/UDP qu'elle utilise).

Les fournisseurs de service et plus généralement les administrateurs réseaux expriment un besoin clair d'avoir cette fonctionnalité afin de pouvoir : (i) surveiller le réseau et les tendances (les nouvelles applications à la mode), (ii) appliquer des politiques de service en fonction de l'application ou de la classe d'application, (iii) connaître les applications pour mieux profiler les utilisateurs, par exemple pour des études commerciales.

La tâche est rendue complexe par l'émergence continue de nouvelles applications. De plus, certaines applications, comme le pair-à-pair, utilisent diverses méthodes de camouflage (numéro de port non conventionnel voire chiffrement) de leur trafic pour ne pas être détectées.

Dans cette thèse, au travers de nombreuses études appliquées sur des traces réelles, nous avons mis à jour un certain nombre de problèmes qui avaient été négligés jusqu'alors, tel le problème de portabilité. Nos contributions consistent d'une part en une analyse précise des causes de ce problème, et d'autre part en la mise au point de techniques hybrides qui soient portables.

La majeure partie de la thèse traite de différents aspects de la classification de trafic et de ces applications. Ce travail comporte trois parties qui compren-

nent des chapitres courts et compacts. La première partie présente des résultats d'une évaluation systématique des techniques de classification statistiques existantes sur des traces ADSL, ce qui n'avait jamais pu être fait à une telle échelle. Dans la seconde partie, nous introduisons et évaluons notre méthode de classification hybride. Dans une dernière partie, nous présentons une application de la classification au problème de profiling des utilisateurs résidentiels.

B.2 Évaluation des méthodes existantes

La technique la plus utilisée, voire la seule technique, dans les entreprises ou FAIs pour faire la classification de trafic est d'utiliser des méthodes DPI (Deep Packet Inspection), c'est-à-dire de chercher des signatures d'applications dans les données de l'utilisateur (au dessus de la couche transport TCP). Cette approche est féconde pour de nombreuses applications. Néanmoins, l'utilisation grandissante de méthodes d'obfuscation du trafic requiert l'utilisation de techniques alternatives de classification de trafic, qui viennent en remplacement ou en complément des techniques DPI.

Récemment, plusieurs solutions basées des méthodes d'apprentissage supervisées ont été proposées dans la littérature [62, 5, 4, 58, 67]. La majorité de ces techniques ont été testées sur des traces capturées dans des environnements académiques. Comparer les résultats est compliqué car chaque étude utilise des algorithmes différents et définit les classes de trafic de manière ad hoc.

Dans la première partie de cette thèse, nous avons évalué différents aspects des techniques de classification supervisées que nous nommons méthodes statistiques par la suite. Nous adoptons le point de vue d'un opérateur ADSL et évaluons la complémentarité entre méthodes statistiques et DPI.

Nous avons collecté plusieurs heures de traces (Tableaux B.2 et B.1) sur différents PoP (Point of Presence) d'un opérateur français. Nos données sont uniques car elles forment un ensemble homogène de données capturées dans un laps de temps court (début 2008) sur des PoPs sous le contrôle d'un même FAI. A l'aide de ces traces, nous avons posé les questions suivantes :

- Peut-on obtenir une méthode de classification statistique offrant de hautes performances, et ce pour un ensemble vaste d'applications?
- Est-ce que les méthodes statistiques peuvent aider à classer le trafic qui n'a pu être classé par les méthodes de DPI?
- Est-ce que les modèles statistiques obtenus sont représentatifs des applications et uniquement des applications, c'est-à-dire, est-ce que ces modèles ne capturent pas des spécificités des configurations des clients?
- Est-ce qu'une méthode statistique pourrait remplacer les outils DPI commerciaux?

Trace	Date	Début	Dur	Taille [GB]	Flots [M]	TCP [%]	Octets TCP [%]	Clients	IPs distantes
MS-I	2008-02-04	14:45	1h	26	0.99	63	90.0	1380	73.4 K
R-II	2008-01-17	17:05	1h 10m	55	1.8	53	90.0	1820	200 K
R-III	2008-02-04	14:45	1h	36	1.3	54	91.9	2100	295 K
T-I	2006-12-04	12:54	1h 48m	60	4.1	48	94.7	1450	561 K

Table B.1: Résumé des Traces.

[flows%/bytes%]				
Class	**MSI**	**RII**	**RIII**	**TI**
WEB	67/49	40/25	26/21	16/21
EDONKEY	4/6	15/27	16/28	33/40
MAIL	4/5	2/1	1/0.83	3/1
CHAT	1/0.51	2/1	0.79/0.25	0.42/1.44
HTTP-STR	1/12	0.83/13	0.61/9	0.27/3
OTHERS	8/2	15/0.16	33/0.36	18/1
DB	1/3	3/0.01	3/0.01	0.49/0.02
BITTORRENT	0.94/3	2/8	7/2	3/7
FTP	0.46/0.11	0.11/0.1	0.16/0.5	0.17/0.67
GAMES	0.08/6	0.12/0.41	0.11/0.09	0.27/0.4
STREAMING	0.05/0.054	0.13/1	0.13/1	0.12/1
GNUTELLA	0.09/1	1/4	0.76/3	1/2
UNKNOWN	9/7	14/15	13/24	21/18

Table B.2: Décomposition applicative des traces de trafic (en utilisant ODT)

B.2.1 Etudes croisées

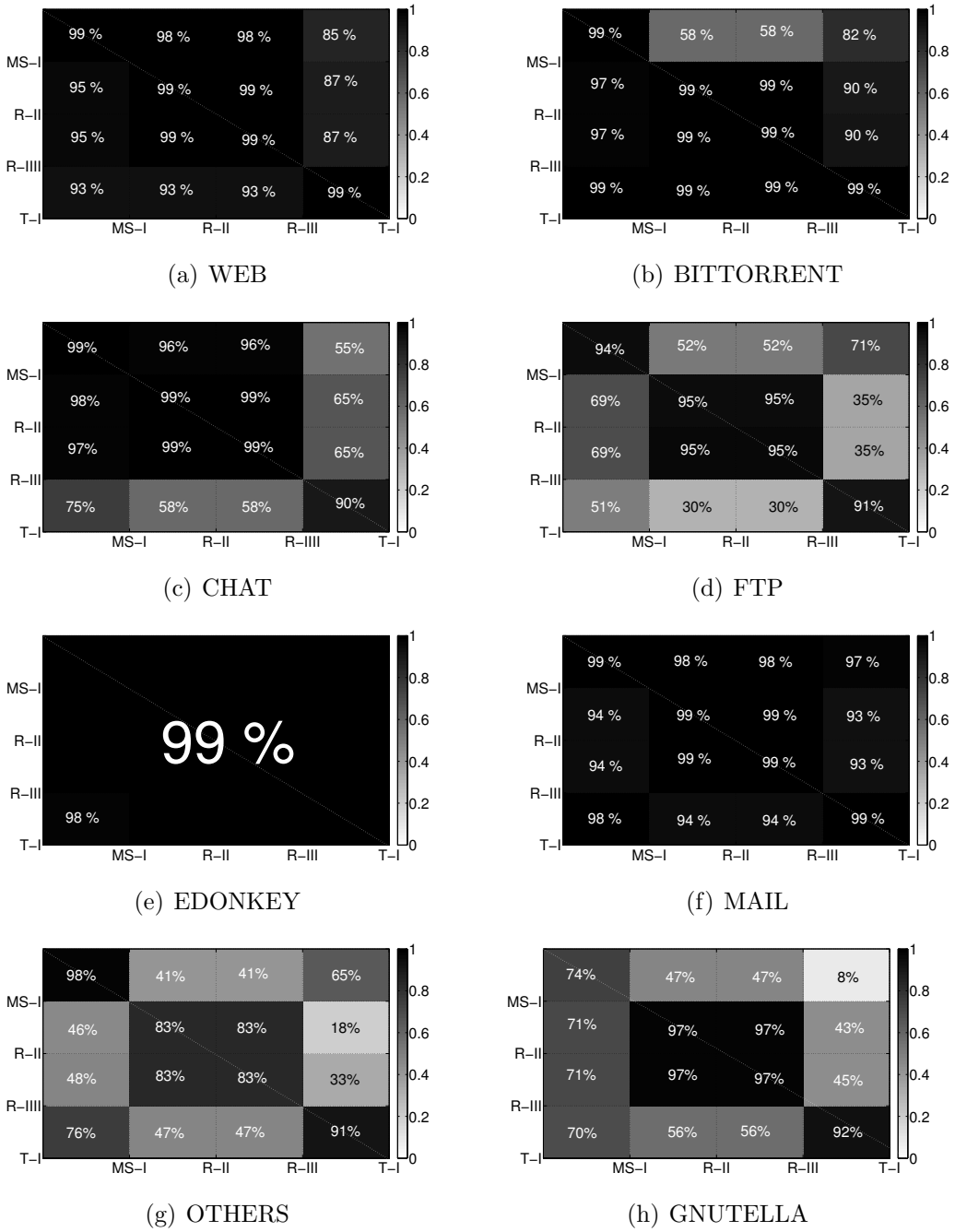
Dans la figure B.1, nous présentons, à titre d'exemple, les performances en terme de précision des algorithmes statistiques lorsque ceux-ci sont entraînés sur un PoP et testés sur un autre. Les caractéristiques utilisées sont la taille et la direction des quatre premiers paquets de données et l'algorithme d'apprentissage est C4.5.

La principale leçon que nous tirons des études croisées est que la dégradation des précisons reste acceptable sur certaines applications mais peut soudainement (suivant la trace d'apprentissage et la trace de test) chuter pour d'autres. Ce phénomène n'avait jamais été détecté auparavant. A partir de cette constatation, notre conclusion est double. D'une part, cela montre que calibrer sur un site puis tester sur un autre donne des résultats non prédictibles. D'autre part, cela montre que les études croisées sont un bon moyen de mettre des problèmes à jour.

Une dernière conclusion suggérée par ces résultats est qu'un modèle calibré sur un site peut sans doute être utilisé sur ce même site pendant une période de temps significative, comme le montrent les traces RII/RIII qui ont été obtenues sur un même site à plusieurs semaines d'écart et qui ne présentent pas de dégradation sensible des performances. Des travaux plus poussés seraient néanmoins nécessaires pour valider cette hypothèse. Nous n'avons pas creusé plus avant ce problème dans cette thèse.

B.2.2 Investigation des causes

Nous avons étudié de manière détaillée les causes des dégradations de performance observées dans les études croisées en nous attachant au problème de la stabilité, au sens statistique, des critères entre sites. Nous avons montré que certaines caractéristiques étaient stables pour certaines applications et non stables pour d'autres – voir par exemple les distributions présentées dans les figures B.2 et B.3.



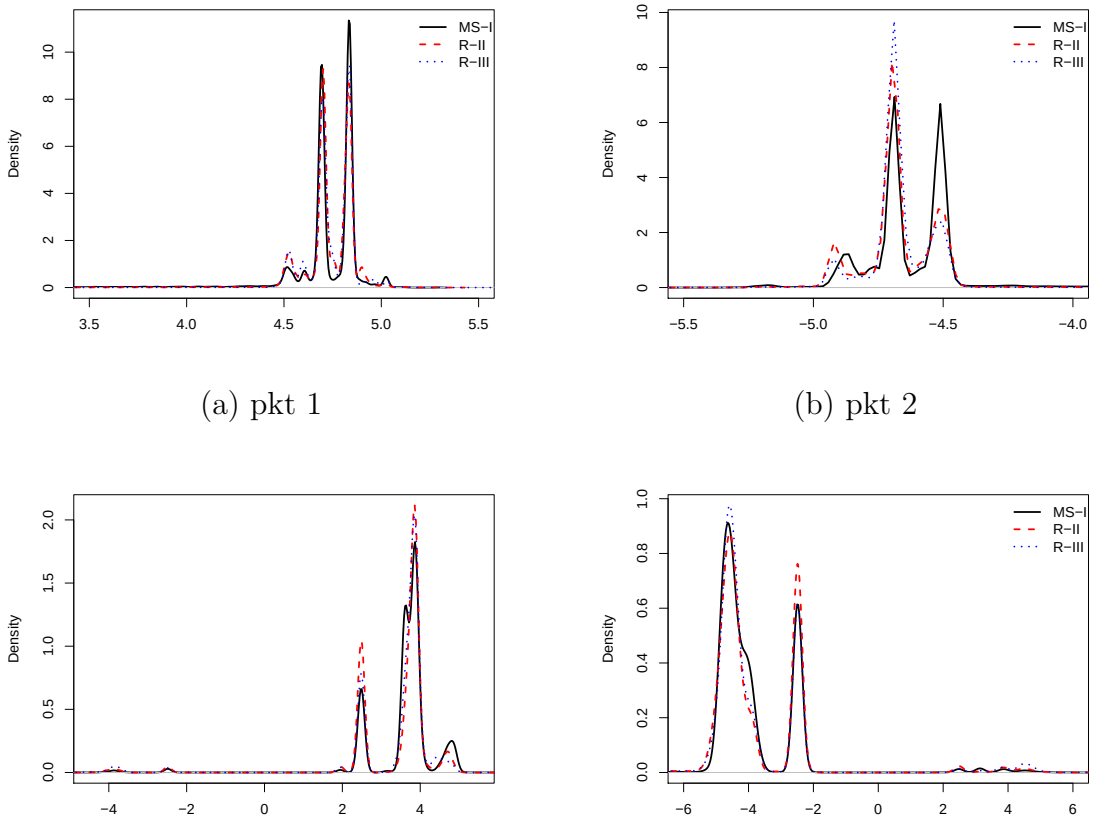
(a) WEB (b) BITTORRENT

(c) CHAT (d) FTP

(e) EDONKEY (f) MAIL

(g) OTHERS (h) GNUTELLA

Figure B.1: Précisions des études croisées utilisant comme critère la taille/direction des paquets (entraînement sur la trace en Y et test sur la trace en X).



(a) pkt 1

(b) pkt 2

(c) pkt 3

(d) pkt 4

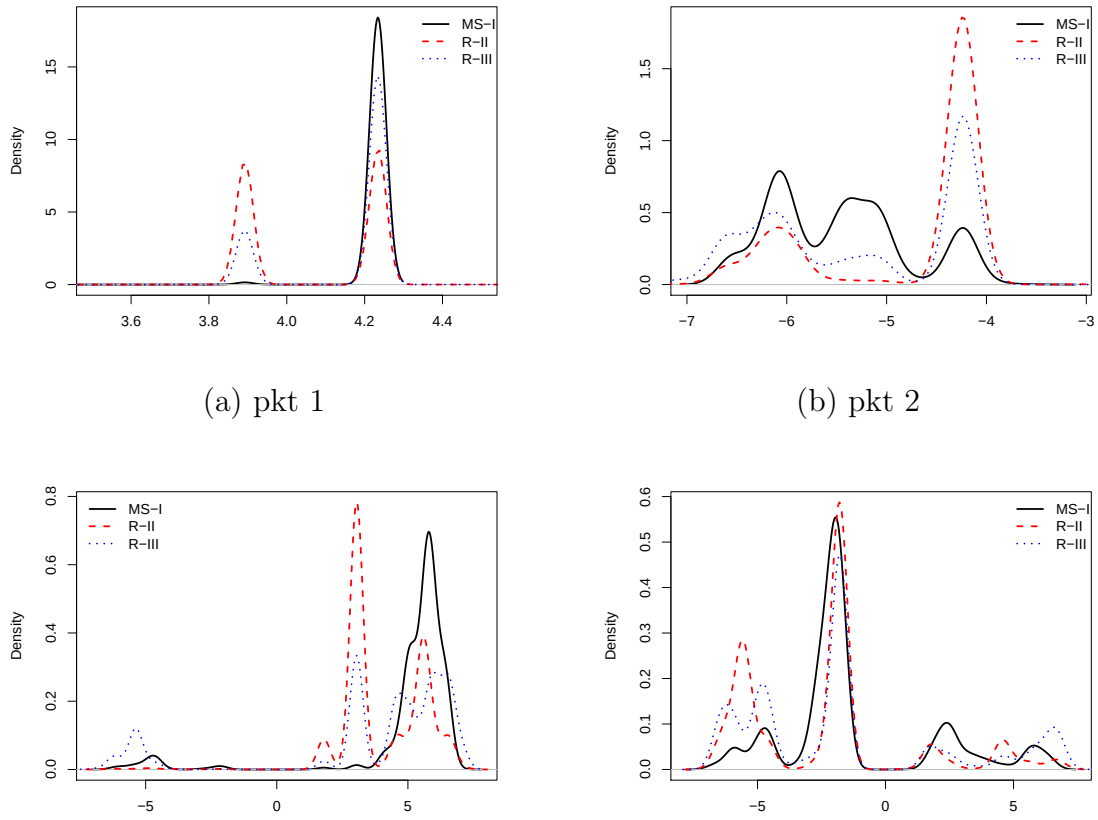
Figure B.2: Distributions de taille de paquets pour EDONKEY (log).

B.2.3 Évaluation des techniques statistiques - conclusions

Nos conclusions sont multiples. Point positif, les méthodes statistiques offrent de bonnes performances lorsque le calibrage et le test se font sur une même trace. Elles sont aussi utiles pour fouiller le trafic laissé non classé par les méthodes DPI.

Point négatif, nous avons montré que les méthodes statistiques pouvaient souffrir de problèmes de sur-apprentissage, ce qui limite grandement l'utilisation d'une technique simple pour un FAI : calibrage d'un modèle statistique sur un site (où un outil de DPI est disponible pour obtenir la réalité terrain) et déploiement sur un autre. Pour autant que nous sachions, ce phénomène n'avait jamais été clairement mis à jour auparavant. Ce problème apparaît comme complexe car il persiste pour tous les algorithmes et ensembles de caractéristiques utilisées dans nos études. Nous avons montré que l'origine principale de ce problème – que nous avons nommé problème de portabilité – était le manque de stabilité des caractéristiques entre sites.

Un dernier message important de cette partie de la thèse est que la complexité à détecter une application varie grandement d'une application à l'autre. Malgré ce problème de portabilité, plusieurs applications, par exemple EDONKEY, peuvent être efficacement classées avec une approche statistique. En conséquence,



(a) pkt 1

(b) pkt 2

(c) pkt 3

(d) pkt 4

Figure B.3: Distributions de taille de paquets pour BITTORRENT (log).

il apparaît qu'il sera difficile de classer toutes les applications avec une même méthode. Cette dernière devra être ajustée pour chaque application ou classe d'applications. Nous partons de cette constatation dans la partie II de cette thèse pour construire un classificateur hybride qui permet d'obtenir une synergie entre diverses sources d'informations.

B.3 Classification Hybride

Une des observations clefs de l'évaluation menée dans la partie I est que la complexité de détection d'une application varie grandement d'une application à l'autre. En conséquence, il est difficile de classer toutes les applications en utilisant une unique technique. Par exemple, les méthodes DPI sont aveugles si le trafic est chiffré. Réciproquement, les méthodes statistiques sont moins efficaces que les méthodes DPI pour zoomer dans le trafic HTTP afin d'isoler, par exemple, le streaming HTTP ou le Webmail. De plus, dans le cas d'études croisées, les méthodes statistiques ne sont pas fiables pour de nombreuses applications.

Pour répondre à ces problèmes, nous avons mis au point une technique dite Hybrid Traffic Identification (HTI - technique de classification hybride). Cette méthode nous permet d'obtenir une synergie entre différentes approches de clas-

sification, par exemple les techniques DPI et les méthodes statistiques basées sur les caractéristiques des flots.

Nous traitons chaque source d'information comme une caractéristique, par exemple la présence d'une signature dans les données applicatives devient une caractéristique du flot à côté d'autres caractéristiques comme la taille et la direction des paquets. Virtuellement, toute technique de classification peut être incorporée car son résultat peut être encodé comme une caractéristique. La décision de classification en HTI est faite à l'aide d'un algorithme d'apprentissage supervisé avec cet ensemble de caractéristiques enrichi. Nous avons mené une évaluation complète d'HTI sur les même traces que celles utilisées dans la partie I et montré que HTI est globalement immunisé contre les problèmes observés auparavant.

De plus, nous présentons les résultats obtenus par un déploiement d'HTI sur le PoP d'un ISP; PoP qui connecte plusieurs milliers d'utilisateurs à l'Internet. Les résultats couvrent plusieurs mois d'observation continue 24h/24 de la plateforme. A notre connaissance, c'est la première fois qu'une telle expérimentation est menée dans un réseau opérationnel.

B.4 HTI

HTI vise, comme son nom le suggère, à combiner plusieurs techniques existantes de classification de trafic afin d'obtenir une synergie entre elles. Ces caractéristiques principales sont :

Tout est une caractéristique : Nous encodons différentes sources d'information comme une caractéristique. Par exemple, la présence d'une signature dans les données utilisateur devient une caractéristique du flot correspondant au même titre que les autres caractéristiques mesurées. D'un point de vue pratique, toutes les sources d'information utilisées par les méthodes proposées dans la littérature peuvent être encodées ainsi [71].

Auto-apprentissage : HTI utilise une méthode supervisée. Durant la phase d'apprentissage, HTI assigne un poids à chaque caractéristique. Cela soulage l'opérateur qui n'a pas à choisir entre des résultats contradictoires de différents outils puisque HTI l'aura fait au préalable pour lui. L'algorithme d'apprentissage supervisé que nous utilisons est la régression logistique.

Sous-modèle par application: Dans sa phase d'apprentissage, et pour chaque application, un modèle est créé. Nous les appelons sous-modèles. Cette approche permet une grande flexibilité puisque des méthodes (caractéristiques) différentes peuvent être utilisées pour chaque application. De plus, l'inspection de chaque sous-modèle permet de comprendre l'utilité de chaque caractéristique.

B.4.1 HTI - évaluation hors ligne

Pour l'évaluation, nous utilisons 3 traces parmi celles utilisées dans la partie I, qui sont décrites dans la section B.2. La principale différence est que les classes les

Rappel [flows% | bytes%]

WEB

↓Training	MS-I	R-II	R-III
MS-I	99% | **96%**	98% | **92%**	98% | **92%**
R-II	95% | **93%**	99% | **95%**	99% | **95%**
R-III	95% | **93%**	99% | **95%**	99% | **95%**

HTTP-STR

↓Training	MS-I	R-II	R-III
MS-I	98% | **99%**	96% | **99%**	98% | **99%**
R-II	98% | **99%**	96% | **99%**	98% | **99%**
R-III	98% | **99%**	96% | **99%**	98% | **98%**

EDONKEY

↓Training	MS-I	R-II	R-III
MS-I	99% | **99%**	98% | **98%**	98% | **98%**
R-II	97% | **98%**	96% | **97%**	97% | **97%**
R-III	97% | **99%**	98% | **98%**	97% | **98%**

BITTORRENT

↓Training	MS-I	R-II	R-III
MS-I	100% | **100%**	99% | **99%**	97% | **98%**
R-II	100% | **100%**	99% | **100%**	99% | **99%**
R-III	100% | **100%**	99% | **100%**	99% | **99%**

MAIL

↓Training	MS-I	R-II	R-III
MS-I	94% | **97%**	99% | **100%**	100% | **99%**
R-II	90% | **95%**	99% | **100%**	99% | **100%**
R-III	90% | **95%**	99% | **100%**	99% | **99%**

Précision [flows% | bytes%]

WEB

↓Training	MS-I	R-II	R-III
MS-I	99% | **97%**	99% | **95%**	99% | **95%**
R-II	99% | **97%**	99% | **94%**	99% | **92%**
R-III	99% | **97%**	99% | **95%**	99% | **95%**

HTTP-STR

↓Training	MS-I	R-II	R-III
MS-I	93% | **96%**	96% | **98%**	95% | **99%**
R-II	93% | **96%**	96% | **98%**	95% | **99%**
R-III	91% | **96%**	95% | **98%**	94% | **99%**

EDONKEY

↓Training	MS-I	R-II	R-III
MS-I	91% | **95%**	95% | **94%**	98% | **98%**
R-II	92% | **95%**	97% | **95%**	98% | **98%**
R-III	92% | **96%**	95% | **94%**	98% | **98%**

BITTORRENT

↓Training	MS-I	R-II	R-III
MS-I	96% | **98%**	98% | **99%**	98% | **99%**
R-II	99% | **98%**	99% | **100%**	99% | **100%**
R-III	99% | **98%**	99% | **100%**	99% | **100%**

MAIL

↓Training	MS-I	R-II	R-III
MS-I	94% | **99%**	99% | **100%**	99% | **100%**
R-II	99% | **99%**	99% | **100%**	100% | **100%**
R-III	99% | **100%**	99% | **100%**	99% | **100%**

Table B.3: Classification hors ligne [flots%/octets%].

moins populaires sont ici agrégées en une seule que nous nommons MINOR_APP. La figure B.1 présente les résultats de classification obtenus par notre outil de DPI nommé ODT, en terme de flots et d'octets sur 3 traces.

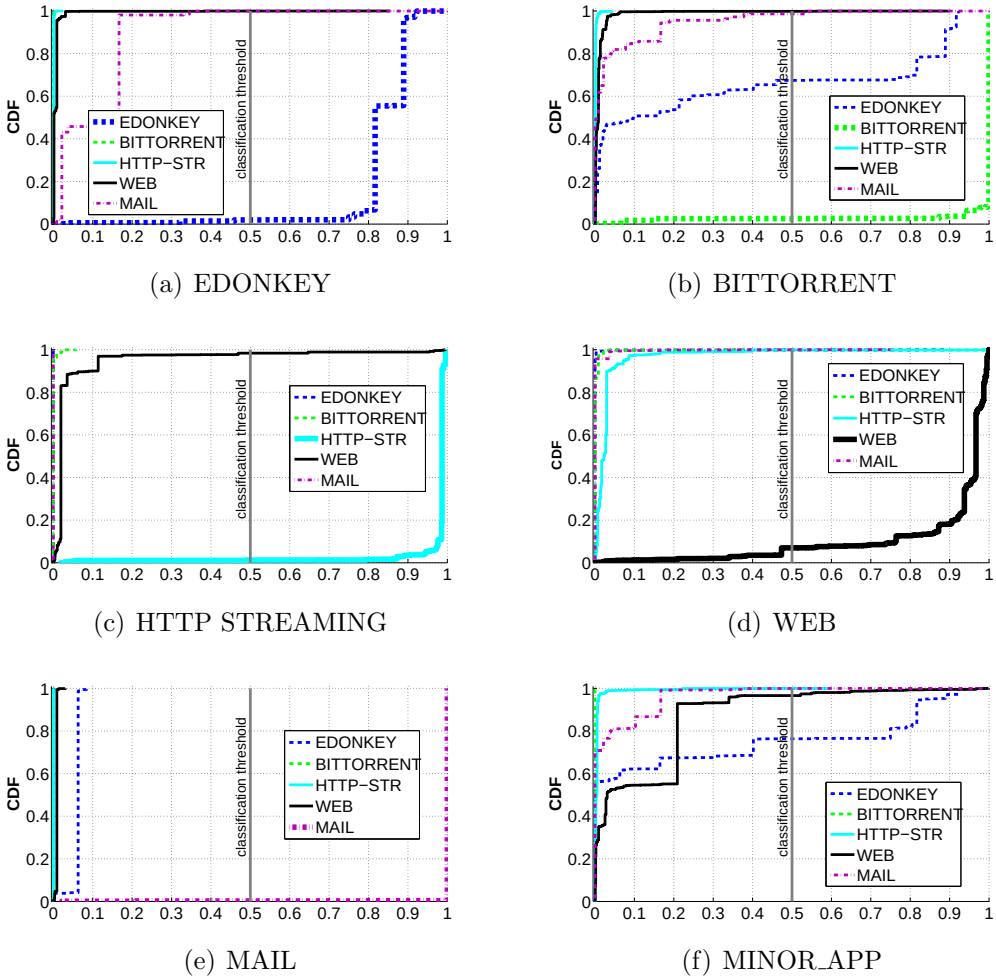
Le tableau B.3 présente les résultats de classification obtenus – précision et rappel – obtenus avec HTI, en terme de flots et d'octets pour le scénario croisé complet. Il faut noterd que tout le trafic, y compris, MINOR_APP et UNKNOWN, a été gardé durant la phase de test. Nous avons répété chaque expérience cinq fois pour confirmer la stabilité des résultats.

B.4.2 Classifications Multiples

Dans la phase de classification, chaque flot est testé contre tous les sous-modèles possibles et ainsi, des résultats positifs multiples sont possibles. Pour chaque flot, on choisit la classe qui a la plus haute probabilité. Il est important de noter que les classifications multiples sont rares et affectent au plus un faible pourcentage des octets.

La figure B.4 présente la fonction de répartition de chaque sous-modèle. Sur chaque graphe est indiqué le seuil de classification. Par exemple, la figure B.4c indique le score obtenu par tous les flots pour le sous-modèle HTTP-STR. Chaque courbe représente la distribution des scores d'une classe donnée (classe obtenue par l'outil de DPI ODT). La figure B.4b montre qu'edonkey est la seule application qui souffre de problèmes de double classifications puisque 35% des flots edonkey sont positifs au test BitTorrent. Néanmoins, quand on applique la règle de la probabilité la plus forte, tous ces flots sont correctement classés en edonkey.

En conclusion, la figure B.4 montre que les sous-modèles HTI offrent un bon pouvoir de séparation entre applications.



(a) EDONKEY (b) BITTORRENT

(c) HTTP STREAMING (d) WEB

(e) MAIL (f) MINOR_APP

Figure B.4: Scores de portabilité pour chaque sous-modèle. Entrainement sur MS-I et test sur R-III.

B.4.3 Quelle méthode pour quelle application?

L'utilisation de la régression logistique offre l'avantage que le résultat de la phase d'apprentissage (le vecteur β) permet de savoir quelle importance a été donnée à quelle caractéristique. Par exemple, une valeur proche de zéro signifie que cette caractéristique n'a pas été retenue dans la classification. D'autres algorithmes d'apprentissage supervisés offrent un modèle interprétable. Par exemple, les arbres de décision comme C4.5 qui permettent de savoir quelle importance a été donnée à telle caractéristique en fonction de la hauteur à laquelle il est utilisé comme facteur discriminant dans l'arbre. Néanmoins, l'arbre de décision est de taille variable et peut facilement atteindre O(1000) noeuds. Ainsi, son interprétation nous semble plus complexe que les poids assignés par la régression logistique.

Nous résumons ci-dessous les informations clefs obtenues à partir de l'étude des valeurs de β :

- HTTP-STR et WEB: L'utilisation de méthodes purement statistiques permet de séparer facilement ces deux classes du reste du trafic, mais ne permet pas de les séparer entre elles. Notre expérience nous a montré que seule l'utilisation de signatures permettait de séparer le HTTP-STR du reste du WEB (voir Tableau 11.1). La signature utilisée pour la classe WEB est en fait la négation de de la signature utilisée pour le HTTP-STR. La signature est transformée en caractéristique par l'utilisation d'un indicateur statistique qui vaut 1 si la signature est présente et 0 sinon.

- BITTORRENT: Alors qu'Edonkey et BitTorrent sont deux applications pair-à-pair, des méthodes différentes doivent être utilisées pour reconnaître ces deux applications. Détecter BitTorrent en utilisant les méthodes statistiques nous a mené systématiquement à de mauvais résultats. Ajouter une signature nous permet d'obtenir des précisions/rappels quasi-parfaits. Il faut noter que notre outil de DPI ne détecte, à l'heure actuelle, que du BitTorrent chiffré.

- EDONKEY: Les caractéristiques statistiques se révèlent suffisantes pour une bonne classification. Il est important de noter que le chiffrage dans eMule n'altère pas fondamentalement les tailles et sens des premiers paquets de données et explique donc que l'on puisse détecter ce trafic, bien que chiffré.

- MAIL: C'est a priori une classe facile à détecter. La phase d'apprentissage nous apprend que les caractéristiques statistiques et les numéros de port sont des caractéristiques importants pour l'algorithme de régression logistique.

Class	**Feature**		
	Statistical	**Port**	**Payload Signature**
WEB	V	V	V
HTTP-STR	V	V	V
EDONKEY	V	V	–
BITTORRENT	–	–	V
MAIL	V	V	–

Table B.4: Méthodes retenues pour les sous-modèles HTI.

B.4.4 HTI en environnement de production

Les résultats des sections précédentes ont démontré que notre instance d'HTI basée sur la régression logistique est précise et fiable. Elle est notamment robuste au problème de portabilité que rencontrent les méthodes de l'état de l'art. Nous rapportons ici les résultats d'un déploiement en environnement de production de ce logiciel pour un PoP ADSL de grande taille.

Nous détaillons tout d'abord l'implantation et discutons des problèmes de mesure rencontrés sur la plateforme avant de discuter de la validation des résultats live obtenus. Enfin, nous présentons des résultats pour 6 mois de surveillance du trafic de la plateforme.

Nous avons développé HTI en C avec la librairie pcap [63]. La procédure simplifiée est présentée dans l'algorithme 3. Nous traitons chaque paquet l'un après l'autre et stockons les flots (identifiés par les quintuplet classique) dans une table de hachage. Un nouveau flot est créé à chaque fois que nous observons un nouveau SYN. Les trafics ICMP et UDP sont stockés dans une table à part. Globalement, ces 2 types de trafic sont minoritaires et nous nous concentrons sur le trafic TCP.

Nous avons déployé HTI sur un lien d'agrégation d'une plate-forme ADSL qui sert environ 16000 clients. C'est une plate-forme différente de celle utilisée dans les section précédentes de la thèse qui étaient de taille plus modeste. Le trafic entre et sort de la plate-forme au travers de 4 commutateurs haut-débit qui effectuent une répartition de charge au niveau IP. Cette répartition de charge se fait de manière indépendante entre les flux entrants et sortants. Ainsi, sur un commutateur donné, seul un quart du trafic est bi-directionnel. Nous travaillons sur un seul des 4 commutateurs à l'heure actuelle. Il faut noter que bien que nous ne voyions pas la totalité du trafic à un instant donné, le fait que les adresses IP soient allouées dynamiquement aux clients au cours du temps et le fait que nous fassions nos mesures sur de grands intervalles de temps (plusieurs mois) nous garantit, avec une forte probabilité, que nous balayons tous les clients de la plateforme.

HTI tourne sur une machine équipée d'un processeur Intel(R) Xeon(TM) CPU 2.80GHz core-duo. La machine dispose de 2 GigaOctets de RAM et est équipée d'une carte Ethernet standard. Le système d'exploitation est Mandriva Linux.

HTI a tourné sur la plate-forme de façon ininterrompue pendant 8 mois. Depuis début 2010, HTI écrit, toutes les 5 min, ses résultats dans une base RRD [84]. RRD est conçu spécifiquement pour un suivi continu de séries temporelles.

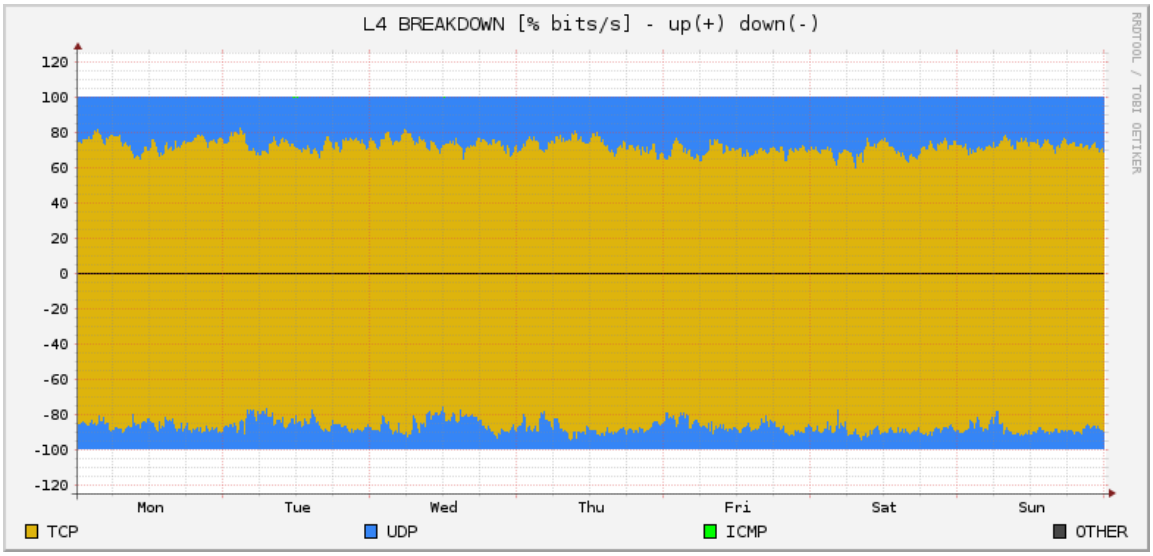


Figure B.5: Une semaine : décomposition au niveau transport [%].

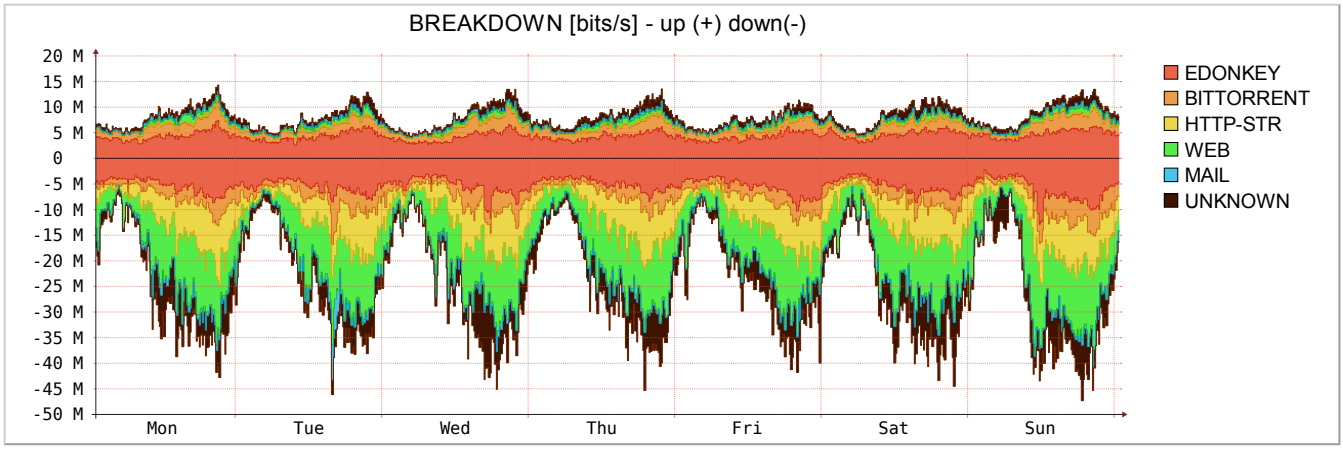


Figure B.6: Une semaine de trafic classé par HTI

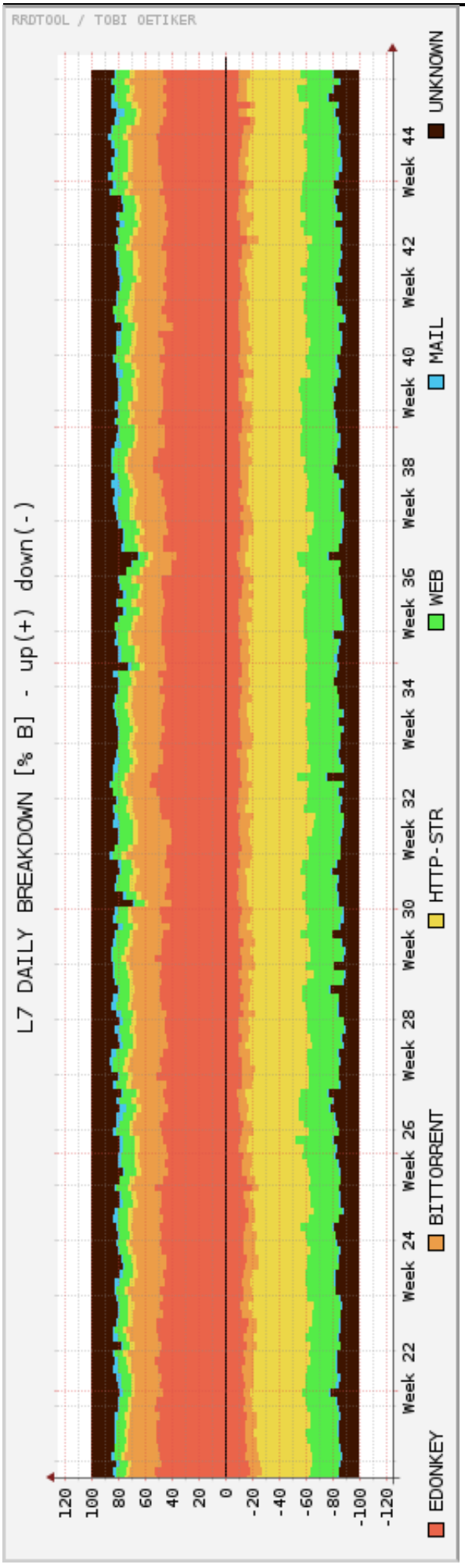


Figure B.7: 180 jours de trafic classé

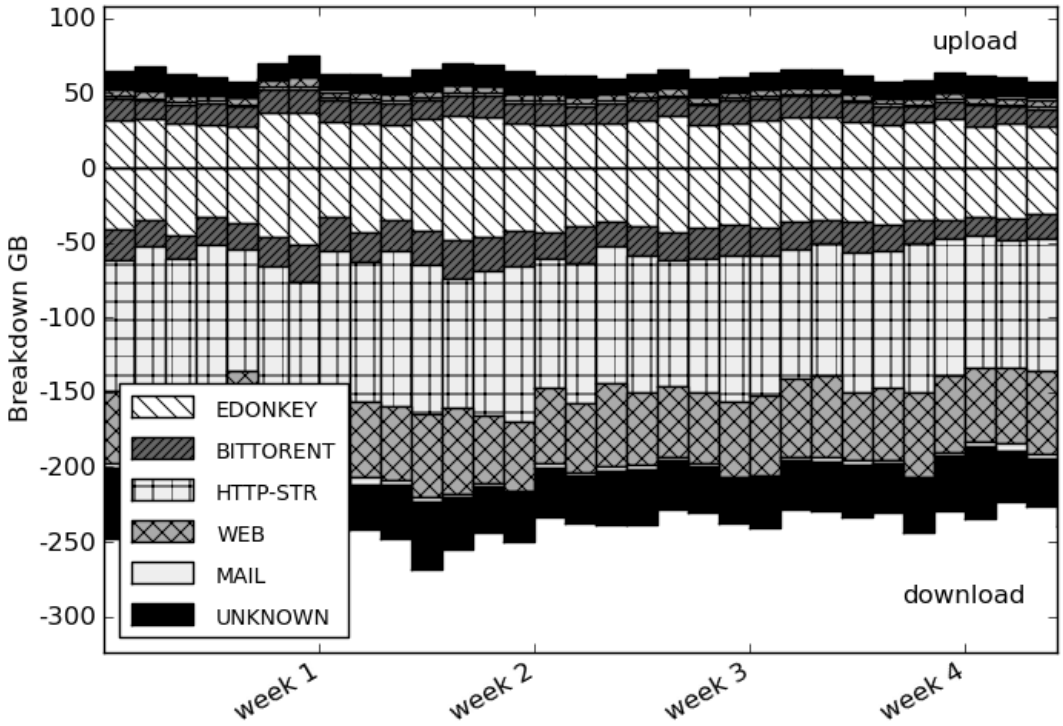


Figure B.8: Décomposition journalière de trafic pour un mois. Au total 9.7 Toctets de données (TCP seulement)

B.4.5 Une semaine de trafic

Nous présentons figure B.6 la décomposition en débit des trafics montant (+) et descendant (-) de la plate-forme. Des statistiques sur le trafic sont également présentées dans le Tableau B.5. La figure B.5 indique la décomposition en couches transport pour la même semaine. Les leçons clefs que l'on peut tirer de ces graphes sont :

- Comme le montre la Figure B.5, TCP domine clairement en terme de couche transport. La fration d'UDP est légèrement plus importante dans le sens montant. Les autres trafic, et notamment ICMP, sont négligeables en terme d'octets - de l'ordre de 30 Kb/s.

- En utilisant seulement un faible nombre de classes de trafic, 82% des octets sont reconnus durant cette semaine.

- Le trafic pair-à-pair montre de faibles variations d'un jour à l'autre, au contraire du trafic HTTP-STR qui est clairement dépendant des utilisateurs (voir la Figure B.6).

Class	Breakdown		Rate per class [Mb\s]			Daily amplitude [Mb\s]		
	[GB]	[%]	min	mean	max	min	mean	max
EDONKEY	527.40	25.44	3.37	7.14	16.77	5.60	8.43	12.50
BITTORENT	206.24	9.95	0.46	2.79	7.45	4.36	5.61	6.98
HTTP_STR	587.07	28.32	0.12	7.95	26.05	16.98	19.93	24.81
WEB	374.84	18.08	0.20	5.07	17.87	10.10	14.50	17.44
MAIL	41.05	1.98	0.00	0.56	4.37	2.44	3.05	4.36
UNKNOWN	336.43	16.23	0.44	4.55	15.11	9.61	12.11	14.44

Table B.5: Statistiques HTI, du 5 July 2010 au 12 July 2010, volume cumulé : 1707.83GB.

- Le trafic de type HTTP domine clairement le trafic en terme de volume (plus de la moitié des octets), suivi par EDONKEY et BITTORENT.
- HTTP_STREAMING domine le trafic WEB classique.
- Le trafic inconnu (UNKNOWN) bien que faible en moyenne, présente des variations à court-terme fortes parfois.

La semaine que nous avons présentée est représentative des nombreuses semaines d'observations que nous avons effectuées.

B.4.6 Six mois de trafic

En figure B.7, nous présentons la fraction relative des différentes classes, jour par jour, pour plus de 180 jours consécutifs. La Figure B.8 montre un mois de trafic en valeur absolue pour comparaison. Les leçons clefs à tirer de ces figures sont :

- Le trafic sur le lien montant reste dominé par le trafic pair-à-pair: eDonkey suivi de BitTorrent. Au contraire, HTTP-STR et le WEB dominent le lien descendant pour toute la période d'observation. Cette domination d'HTTP sur le trafic pair-à-pair corrobore les études similaires menées dans d'autres pays [64, 28] sur des populations d'utilisateurs ADSL de taille similaire et des périodes de temps récentes.
- Le volume cumulé de trafic observé dépasse 9.7 To et représente plus de 75 millions de flots. Notre base RRD a atteint de l'ordre de quelques Mo durant cette période.
- Les volumes de chaque type d'application sont stables durant cette période de 6 mois, malgré le fait que l'ensemble des utilisateurs varie du fait de notre technique de mesure.

B.5 Profiling des utilisateurs

Dans la partie II de cette thèse, nous avons montré comment la classification d'application pouvait permettre un suivi à long terme de l'activité d'une plate-

forme ADSL. Nos analyses ont confirmé les études récentes sur la part relative des différentes applications dans le trafic Internet.

Les statistiques collectées par notre classificateur hybride peuvent être utilisées pour le suivi mais également pour le dimensionnement. Dans cette partie, nous regardons un problème lié au précédent, à savoir comment établir le profil des utilisateurs au niveau applicatif.

Le profiling des utilisateurs est un sujet important qui a reçu peu d'attention. Cette partie vise à remplir le fossé entre les études de bas-niveau (niveau paquet) et les études de haut niveau (application) par l'ajout d'un niveau intermédiaire : le profiling des utilisateurs. Nous utilisons de multiples techniques, en particulier le clustering hiérarchique, pour grouper entre eux des utilisateurs ayant un profil similaire au niveau applicatif. Nous accordons aussi une attention particulière aux utilisateurs ayant généré beaucoup de trafic. Nous reprendrons pour ces derniers le terme anglais de "heavy-hitters".

Le problème que nous traitons implique plusieurs sous problèmes auxquels il faut répondre :

- Combien d'octets ou alternativement combien de flots doivent être observés pour déclarer qu'un utilisateur utilise une application donnée?
- Peut-on caractériser les utilisateurs par leur application dominante?
- Quel est le profil typique d'un heavy hitter?
- Quel est le mélange d'applications caractéristique d'un utilisateur?

Nous abordons ces questions dans les paragraphes suivants. Nous discutons plusieurs options pour établir une correspondance entre un utilisateur et les applications qu'il utilise.

B.5.1 Application dominante d'un utilisateur

Nous présentons ici une approche simple pour établir le profil d'un utilisateur : nous labelisons chaque utilisateur par l'application qui a généré le plus d'octets au total pour cet utilisateur. Cette approche est justifiée par le fait que l'application dominante représente en général la majorité des octets de l'utilisateur : chez 75% des utilisateurs, l'application dominante explique plus de la moitié des octets. Ce phénomène est encore plus prononcé chez les heavy hitters. La figure B.9 présente la distribution de la fraction d'octets expliquée par l'application dominante.

La distribution des utilisateurs par application avec une telle approche (application dominante) est reportée dans le tableau B.6. Comme anticipé, la classe dominante est le WEB. Nous avons plus de STREAMING que de P2P. Cela correspond à l'intuition que l'utilisateur lambda, même si il n'est pas chevronné, peut regarder des vidéos sur YouTube, alors que l'utilisation des applications P2P demande des connaissances informatiques plus avancées. Les cas d'autres applications dominantes (DB, Others, Control, Games) correspondent à de faibles volumes globaux et peuvent difficilement être considérés comme significatifs.

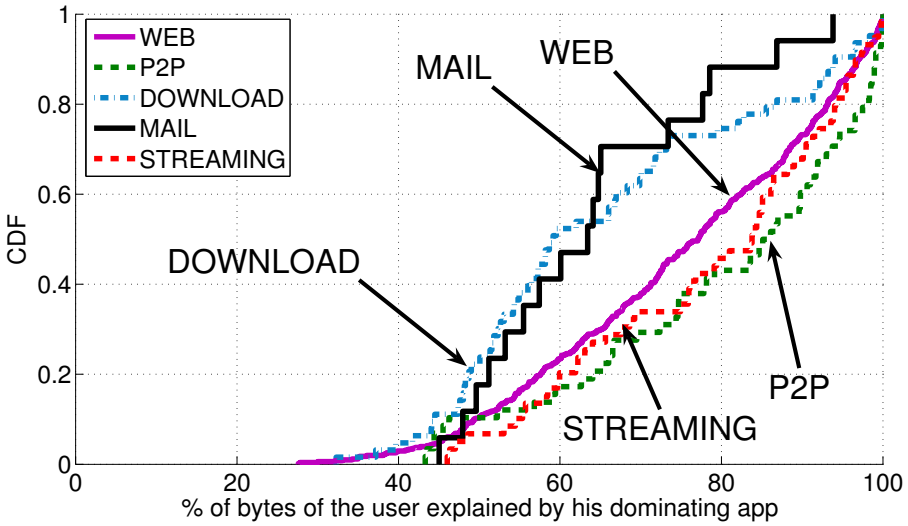


Figure B.9: Fonctin de répartition des octets dus à l'application dominante (Ensemble B).

Nous présentons en Figure B.10 le résultat de cette technique de labelisation pour les utilisateurs de l'ensemble B. Chaque utilisateur correspond à un point de la figure, qui a pour coordonnées son volume montant et son volume descendant total. Nous nous sommes restreints aux 6 applications dominantes : Web, Streaming, VOIP, Download and P2P. De plus, nous avons ajouté les utilisateurs dont la majorité des octets était dans la classe inconnue UNKNOWN.

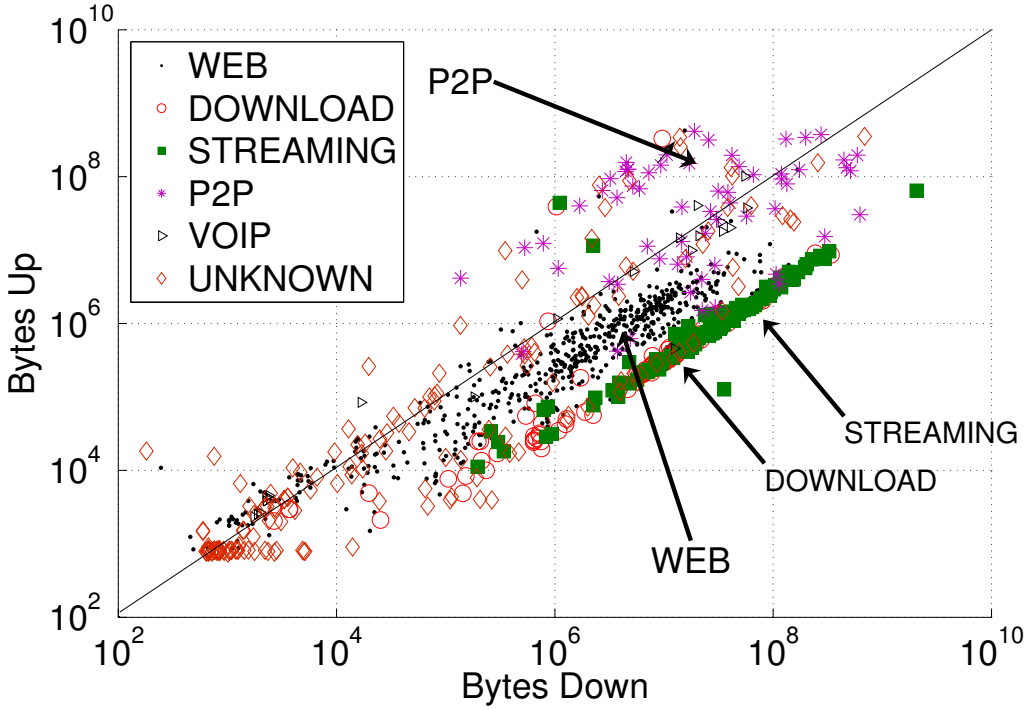


Figure B.10: Trafic utilisateur montant/descendant. Utilisateurs marqués par leur application dominante (Ensemble B).

Table B.6: Utilisateurs labelisés par leurs application dominante (pour ceux ayant émis plus de 100 octets : 1755 utilisateurs. Ensemble B).

Class	**Fraction of Users**	**Fraction of Bytes explained**
UNKNOWN	21%	12%
WEB	35%	19%
P2P	4%	35%
DOWN	5%	$\leq 1\%$
MAIL	1%	$\leq 1\%$
DB	9%	$\leq 1\%$
OTHERS	8%	$\leq 1\%$
CONTROL	7%	$\leq 1\%$
GAMES	$\leq 1\%$	$\leq 1\%$
STREAMING	7%	25%
CHAT	1%	$\leq 1\%$
VOIP	1%	2%

La leçon la plus importante de la Figure B.10 est que labeliser les utilisateurs par leur application dominante est porteur de sens. En effet, les utilisateurs avec la même application dominante se trouvent groupés ensemble sur la figure.

En particulier, nous observons que :

- Les heavy hitters P2P tendent à générer du trafic plus symétrique que les heavy hitters Download et Streaming qui sont sous la bissectrice.
- Les utilisateurs Web sont majoritairement sous la bissectrice et au dessus des heavy hitters Download et Streaming. Cela correspond bien à l'intuition que surfer sur le Web génère moins d'octets que le streaming ou le téléchargement en général. De plus ces échanges sont plus symétriques.
- Concernant les utilisateurs de type Unknown, nous observons qu'une grande partie d'entre eux génèrent très peu de trafic puisqu'ils sont dans le coin en bas à gauche de la figure. En ce qui concerne les heavy hitters de ce type, leur profil semble symétrique, ce qui suggère l'existence d'applications p2p que nous ne savons pas détecter (par exemple, BitTorrent chiffré).

L'analyse ci-dessus a également souligné le rôle clef des heavy hitters. Les 10 plus actifs ont généré 0.6 Go chacun. Globalement, ils sont responsables d'un quart des octets de la trace. Nous les avons profilé plus avant.

B.5.2 Top ten des heavy hitters

Dans cette section, nous regardons plus précisément les 10 heavy hitters de nos ensembles A et B. Bien que ces 2 ensembles correspondent à la même plateforme, ce sont des utilisateurs distincts. Les volumes qu'ils génèrent sont tels que le dimensionnement du réseau doit se faire à partir de leurs caractéristiques. En Figures B.11(a) et B.11(b), nous présentons les fractions d'octets par application

et par sens pour ces utilisateurs. Par soucis de clarté nous mettons sur la figure seulement les labels des applications visibles car ayant générées suffisamment d'octets. On observe que, pour la plupart, ces utilisateurs font du P2P ou du STREAMING.

Nous observons également que le trafic inconnu est associé aux utilisateurs P2P, ce qui va dans le sens de l'aggravation de nos soupçons concernant l'existence d'applications P2P que nous ne savons pas reconnaître avec notre outil de DPI. Dans le cas présent, 67 % et 95 % des octets de type inconnu sont générés par des utilisateurs faisant du P2P en parallèle, pour les ensembles A et B respectivement. Notons qu'une autre explication est possible : nos traces sont courtes (1h) et il est possible que nous ayons manqué le début de certaines connexion, alors que le début est souvent crucial pour la recherche de signatures, comme remarqué les outils DPI.

B.5.3 Profils utilisateurs

Dans les sections précédentes, nous avons analysé le profil des utilisateurs en se basant uniquement sur les volumes d'octets. Cette approche est intéressante, mais doit être complétée par une approche tenant compte du nombre de flots générés. En effet, si on tient juste compte du nombre d'octets, on introduit un biais envers les applications génératrices de gros transferts. Nous manquons ainsi certaines applications qui génèrent peu d'octets mais dont les performances restent cruciales pour l'utilisateur.

Dans cette section, nous explorons une perspective différente. Nous associons à chaque utilisateur un vecteur de variables binaires, chaque variable étant un indicateur qui indique si l'utilisateur utilise ou non une application. Nous appliquons ensuite un algorithme de clustering de type hiérarchique pour grouper entre eux les utilisateurs ayant un profil similaire.

Nous devons définir des seuils pour déterminer si un utilisateur utilise effectivement une application ou pas. Nous nous basons sur des heuristiques basées sur des seuils en octets (montants/descendants) et/ou nombre de flots. Ces seuils sont fonctions de l'application. Ces heuristiques sont rassemblées dans le Tableau B.7. Ces valeurs ont été obtenues à partir de valeurs observées sur du trafic réel, comme présenté par exemple en Figure B.13 pour le P2P et le WEB.

B.5.4 Mélanges applicatifs

Nous présentons figures B.14 et B.15 les résultats de clustering pour les 50 premiers et les 50 seconds utilisateurs les plus actifs. Au total, ces 100 premiers heavy-hitters sont responsables de 75% des octets. Nous considérons tout d'abord les classes de trafic ayant généré le plus de trafic, à savoir : Web, P2P, Streaming, and Download.

Chaque barre représente un seul utilisateur et exprime la fraction du volume total - sur la plateforme - dont il est responsable. Dans chaque cluster, les utilisateurs sont classés par volume. Chaque cluster représente un mélange applicatif différent. L'algorithme de clustering va placer l'un à côté de l'autre les clusters

Table B.7: Seuils d'usage

Class	Volume		Number of Flows	Policy
	Down	Up		
WEB	300kB	500kB	20	All
P2P	1 MB	1 MB	10	Any
STREAMING	1 MB	1 MB	–	Any
DOWNLOAD	2 kB	1 kB	–	Any
MAIL	30kB	3 kB	–	All
GAMES	5 kB	5 kB	–	Any
VOIP	200kB	200kB	–	All
CHAT	10kB	10kB	–	Any

similaires.

En considérant seulement 4 applications, on a 15 combinaisons possibles. Nous observons que certaines sont beaucoup plus populaires que d'autres et que certaines sont absentes.

En analysant le résultat pour les 50 premiers heavy hitters, on observe que le P2P est dominant puisqu'il apparaît pour 34 utilisateurs. Le cluster le plus populaire est du P2P pur, suivi des utilisateurs faisant du P2P et du Web.

Les clusters comportant du P2P aggrégent 40% du volume total de trafic. Le cluster Web + Streaming est minoritaire, bien qu'il contienne le plus gros client.

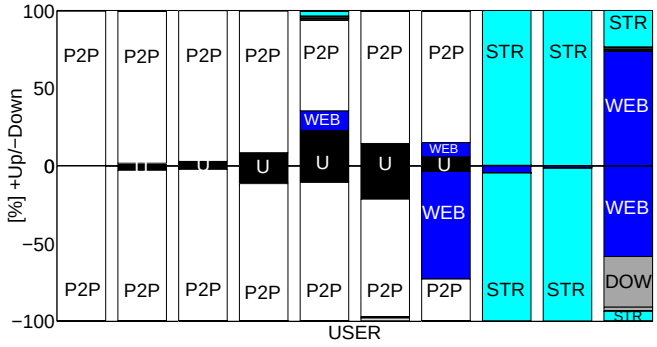
Le second ensemble de 50 heavy-hitters offre une image bien différente. Ici, plus de 30 utilisateurs ont un profil de type Web et streaming alors que les clusters avec du P2P sont minoritaires.

Il est intéressant de voir que les utilisateurs de type P2P et Streaming/Web forment des groupes distincts puisque seulement 10 des 100 premiers heavy-hitters contiennent ces 2 applications dans leur profil. C'est aussi le cas du DOWNLOAD qui n'est pas mélangé au P2P. Cela pourrait correspondre à des profils d'utilisateurs distincts ; mais cela devra être confirmé sur des traces plus longues.

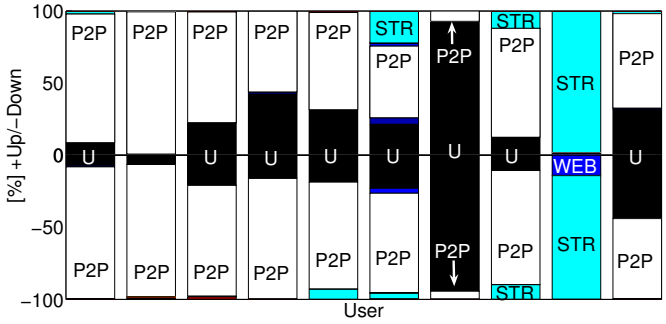
B.5.5 Mélange applicatif - Discussion

En ce concentrant sur les premiers heavy hitters, nous avons observé une famille d'utilisateurs dominée par le P2P. Même si le streaming apparaît dans ce premier ensemble, la majorité des octets générés par cette classe est due majoritairement à des utilisateurs "moyens", c'est-à-dire des heavy hitters de la seconde partie du tableau.

Nous conjecturons que cette situation va persister avec la monté en puissance continue du streaming. En effet, la croissance de popularité de cette classe va accroître la classe moyenne des heavy-hitters. De plus, si les distributeurs de contenu passent à de la vidéo haute définition, dont le débit est jusqu'à 4 fois plus important que le codage actuel, cela pourrait avoir un impact important pour le dimensionnement des liens des FAIs



(a) Ensemble A



(b) Ensemble B

Figure B.11: Top 10 des heavy hitter users. (U correspond à UNKNOWN).

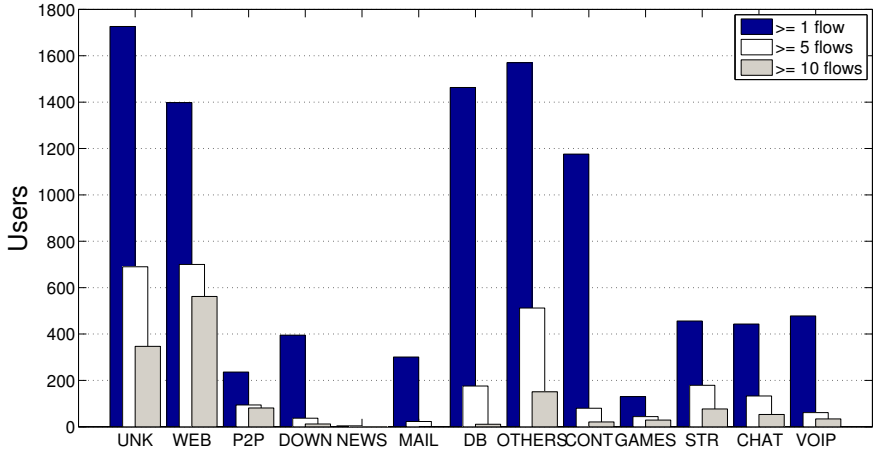
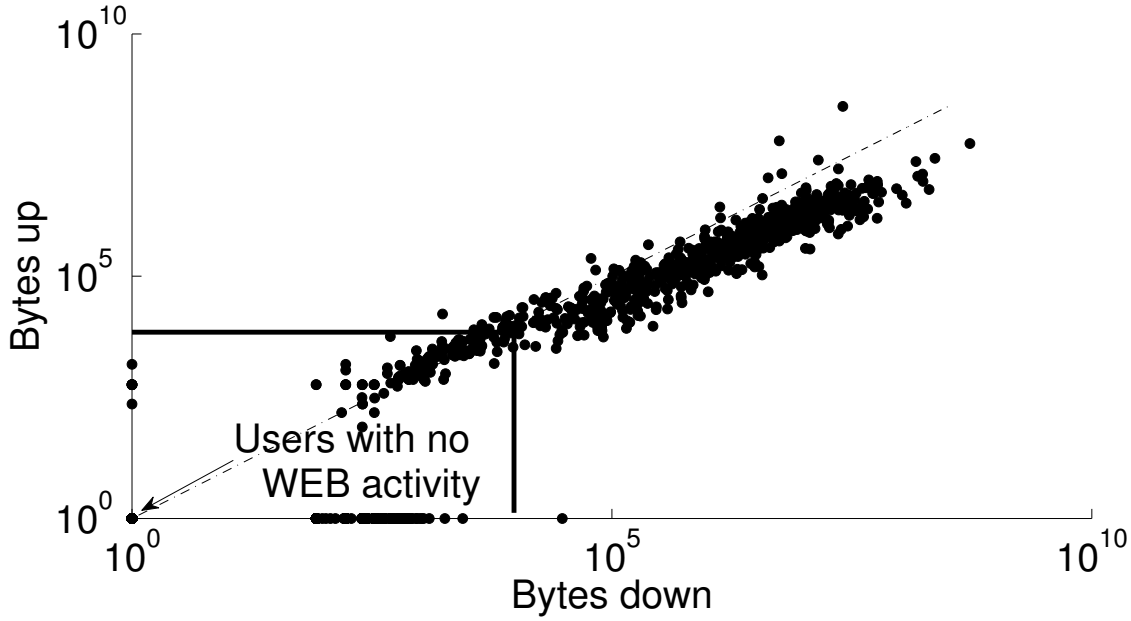
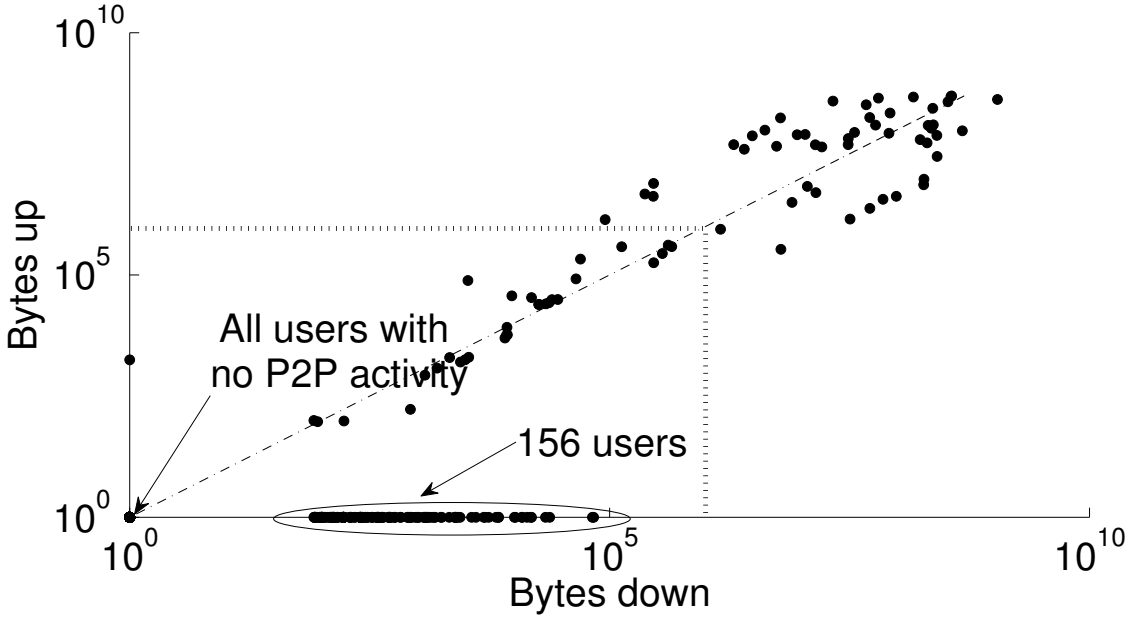


Figure B.12: Rang des applications en fonction des flots uniquement.



(a) WEB



(b) P2P

Figure B.13: Exemple montrant le choix des seuils.

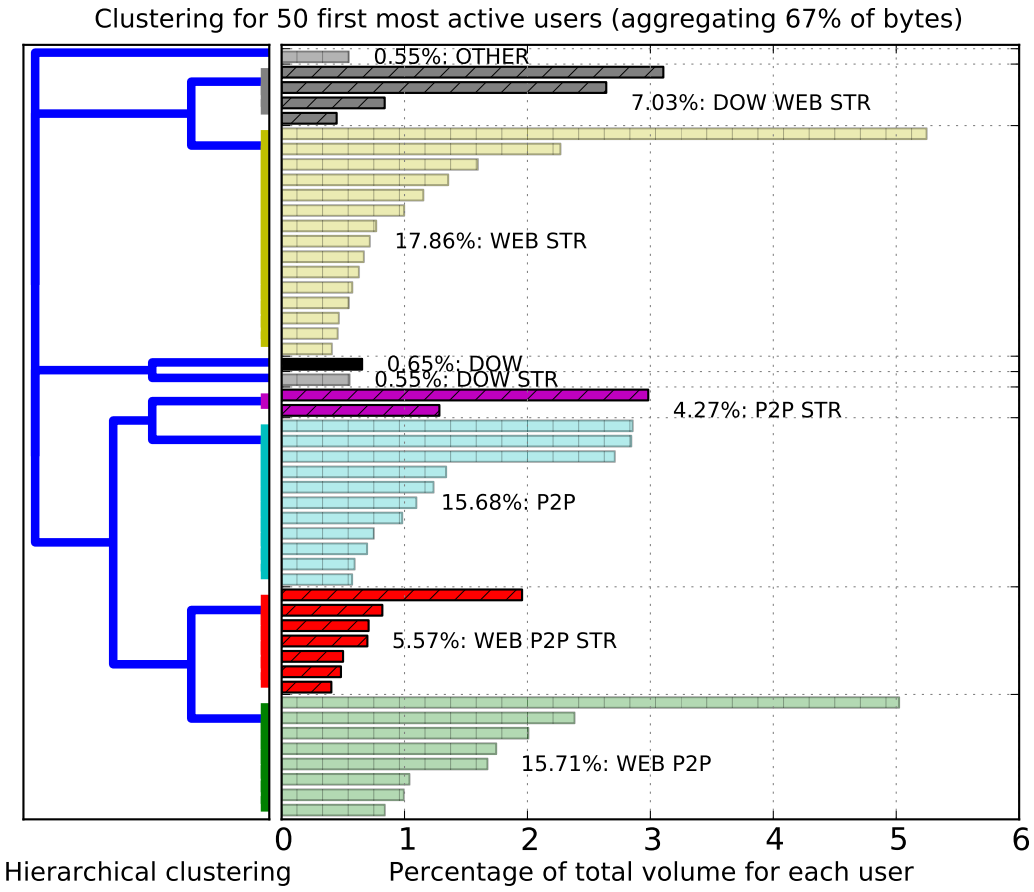


Figure B.14: Clustering applicatif des 50 premiers heavy hitters (ensemble A).

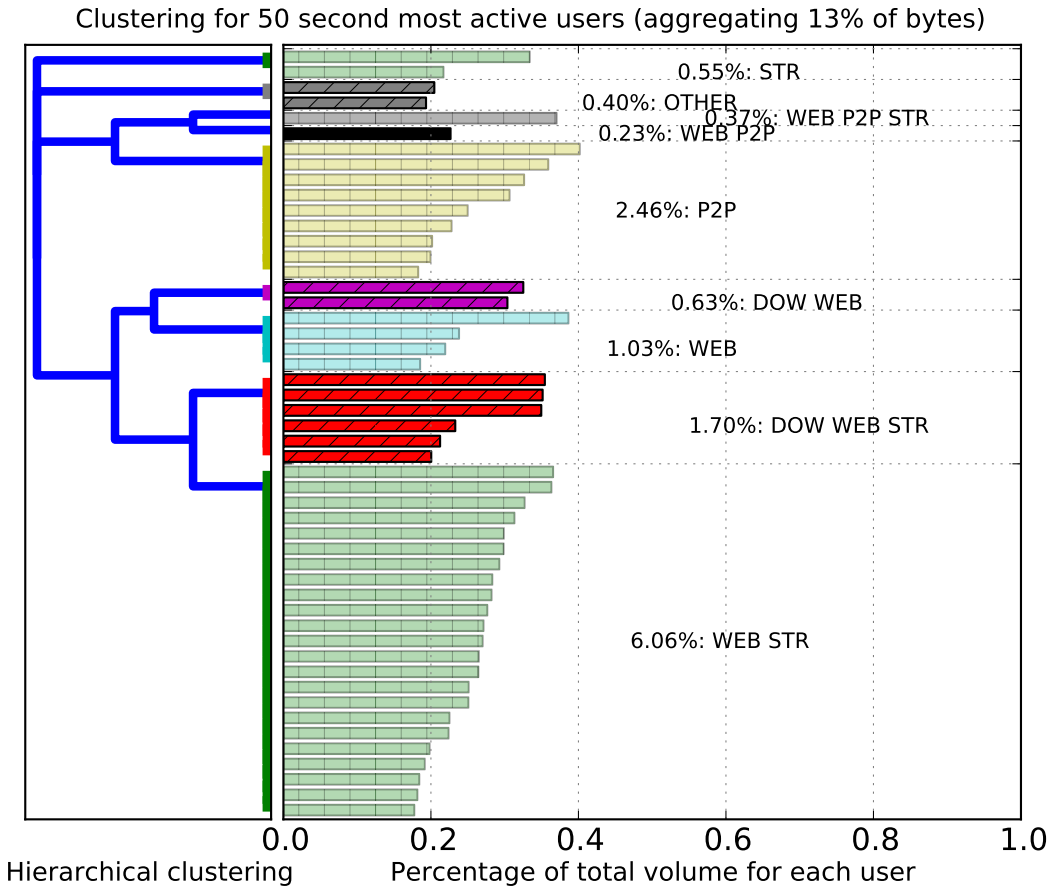


Figure B.15: Clustering applicatif des 50 seconds heavy hitters (ensemble A).