EURECOM
Department of Networking and Security
2229, route des Crètes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-10-238

# Hybrid Traffic Identification

Marcin Pietrzyk,Taoufik En-Najjary, Guillaume Urvoy-Keller and
Jean-Laurent Costeux

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {ennajjar,urvoy}@eurecom.fr,
{marcin.pietrzyk,jeanlaurent.costeux}@orange-ftgroup.com

[1]EURECOM's research is partially supported by its industrial members: BMW Group Research & Technology - BMW Group Company, Bouygues Téécom, Cisco Systems, France Télécom, Hitachi, SFR, Sharp, ST Microelectronics, Swisscom, Thales.

## Abstract

Traffic classification is a key function for ISPs and companies in general. 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. However, there is no one-fit-all method, i.e., no method that offers the best performance for all applications.

In this paper, we propose a framework, called Hybrid Traffic Identification (HTI) that enables to take advantage of the merits of different approaches. Any source of information (flow statistics, signatures, etc) is encoded as a feature; the actual classification is made by a machine learning algorithm. We demonstrated that 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.

Using multiple traces from a large ISP, we demonstrate that HTI outperforms state-of-the-art methods as it can select the best sources of information for each application to maximize its ability to detect it. We further report on an ongoing live experiment with our HTI instance in production network of the large ISP, which already represents several weeks of continual traffic classification.

## 1 Introduction

Recent work on Internet traffic classification has yielded a host of proposals. Each method has its strengths and weaknesses. Transport layer ports fail to classify large fractions of p2p traffic. Still, recent work shows that ports constitute a valuable source of information if used with caution [1]. Deep packet inspection techniques, that rely on signatures are accurate, but fail in the case of encrypted or obfuscated protocols. Alternatively, a broad family of statistical methods relying on flow features, i.e., that do not need to inspect the packet payload have been proposed. However, for many application classes they have questionable robustness, especially when the classifier is used on a site different from the one on which it was trained [2].

Our starting point in this work 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 fail at zooming inside HTTP traffic, e.g., to isolate HTTP streaming or Webmail, while deep packet inspection can mine the HTTP header to detect the application over HTTP.

In this work, we propose a framework – called Hybrid Traffic Identification (HTI) – that 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, 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 statistical machine learning algorithm based on this enriched set of features. Furthermore, HTI is not bound to any specific machine learning algorithm. We evaluate its performance with three different algorithms, namely C4.5, SVM and logistic regression.

The main contributions of the paper are as follows:

- We propose a novel, generic classification framework (HTI), enabling the synergy between diverse classification methods in a single modular tool.
- We provide a full evaluation of the HTI framework based on ADSL traces. We demonstrate that HTI outperforms classical classification methods, by offering high performance both in terms of bytes and flows for each application of interest. We further highlight the resilience of HTI to the data over-fitting problem of supervised machine learning approaches identified in [2].
- 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 weeks 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.
- We take advantage of a specific feature of logistic regression (used in our HTI instance), namely the rating it assigns to each input feature, to discuss which method, among the ones we consider, are relevant for the classification of each application.

The remainder of this paper is organized as follows. After reviewing related work in Section 2, we present the HTI framework in Section 3. Sections 4 and 5 describe the evaluation of the HTI instance using passive captures. We discuss HTI operational deployment in Section 6 and summarizes the major findings in Section 7.

## 2 Related work

Many different methods have been introduced to classify traffic. Traditional approaches relied on port numbers. However, early works [3, 4] quantitatively demonstrated the decrease of accuracy 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 [5, 6]. The increasing use of encryption and obfuscation of packet content, the need of constant updates of application signatures and governments regulations, might however undermine the ability to inspect packets content.

To address these problems, recent studies have relied on statistical classification techniques to probabilistically map flows to applications [1, 7, 8, 9, 10, 11, 12, 13]. Hereafter, we cite a representative sample of traffic classification research. For a much more complete survey, see the work by Nguyen et al. [14].

Moore et al. in [10] presented a statistical approach to classify the traffic into different types of services based on a combination of flow features. A naive Bayes classifier combined with kernel estimation and a correlation-based filtering algorithm was used to classify TCP traffic. They used 10 flow features and obtained accuracy between 93% and 96%, however, their data set contains mainly Web traffic.

Bernaille et al. presented in [8] 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 Hidden Markov model. These models featured accuracy over 98%. The authors further extended their work to the identification of encrypted traffic in [9].

Karagiannis et al. [15] studied traffic behavior 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% accuracy. In their recent work [16], they applied those techniques to profile the users' activity, and to analyze the dynamics of host behaviours.

There exists also a lot of works focusing on specific applications. For example Bonfiglio et al. [13] 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 [3, 17, 18].

There exists a number of works that approached the problem by trying to combine existing methods in several ways. Authors of [19] 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 [20]. Work conceptually closest to our efforts is [21]. The authors design an hybrid classifier for peer-to-peer traffic only, combining several methods used as a input for a Flexible Neural Tree model. The classifier is then implemented on a network processor and evaluated in a small and isolated test network.

Several papers [1, 2] highlighted the main problem that concerns purely statistical methods, namely the portability issues for many applications when the model is applied on different site than it was trained on. One of the aims of our work is to provide a remedy to this problems and build a robust and portable classifier.

## 3 HTI - Hybrid Traffic Identification

In this section we present the design requirements we impose to our traffic classification method, followed by a formal presentation of the HTI framework, the machine learning algorithms we consider and a typology of flow features. This chapter aims at presenting HTI in a generic way. We exemplify and evaluate HTI instances on off-line traces in Section 5 and through live experiments in a production network in Section 6.

### 3.1 Design requirements

The design requirements we impose to a traffic classification tool stems from the position we take in this paper, namely an ISP managing residential customers. Those requirements are:

- **Accuracy**: the classification tool should feature high performance, measured in accuracy and precision (definition in Section 4.4) in terms of both flows and bytes, for all applications of interest.
- **Automatic, per-class, features selection**: the tool should be able to pick automatically the features (thus methods) that best fits to a given traffic class during its training phase.
- **Efficiency:** the classification tool should work in real-time on high speed link of at least 1Gbit/s.
- **Extensibility**: it should be easy to add the support of a new application class (and detection method) to an already deployed tool instance, without disturbing other classes.
- **Flexibility**: the tool should accept as input several sources of information. For instance, payload signatures, statistical flow features, port numbers, and others.
- **Portability**: the classifier should be able to be used on sites (PoPs) different from the one it was originally trained. We believe it is a fundamental property for ISPs that might not have the ability to train a classifier on each and every PoP over which the tool should be used. Instead will train the classifier on a large PoP before deploying it elsewhere. We demonstrated in [2] that classifiers that rely purely on statistical features can fail at fulfilling the portability constraint.

### 3.2 High level description

HTI is designed to enable the synergy, between diverse classification methods. It associates on a per application basis, a weight to each input method. HTI is a

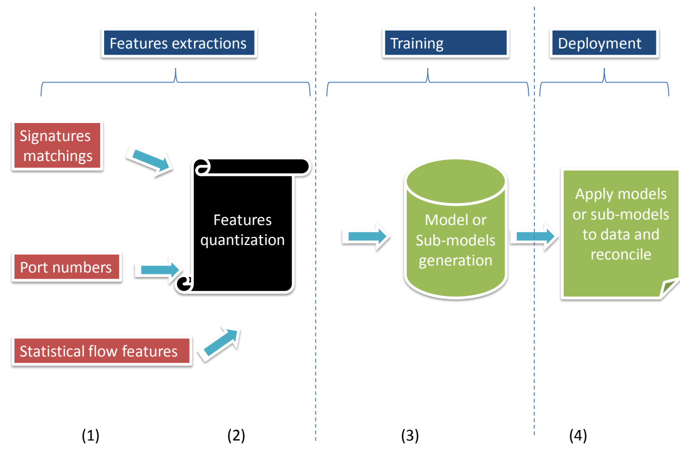

Features extractions
Training
Deployment
Signatures matchings
Port numbers
Statistical flow features
Features quantization
Model or Sub-models generation
Apply models or sub-models to data and reconcile
(1)
(2)
(3)
(4)

Figure 1: HTI framework.

supervised machine learning technique, thus its life cycle consists of three major phases (Figure 1 provides a high level description of the HTI framework):

- **Features Extraction**: (1) raw characteristics are collected for each flow. (2) feature quantization is applied if necessary (e.g. DPI signature encoded as a binary value).
- **Training**: (3) the parameters of the machine learning algorithm are set during the training phase, leading to a so-called model. For instance, if the algorithm is a decision tree, the model consists of the tree along with the conditions used at each branch. An oracle is necessary at this stage to know which flow was generated by which application in the training set. This oracle is in general a deep packet inspection tool, though alternative approaches are possible, see Section 4.1.1.
- **Deployment**: (4) the model is applied to stored or live data, depending on the operational needs and the complexity of extracting the flow features.

The above steps are common to all supervised machine learning approaches. What distinguishes HTI from previous supervised machine learning techniques used in the traffic classification domain are:

- With HTI, one model(sub-model) is generated per application, in contrast to previous techniques. This, preserves design modularity and allows to have specific detection method for each application.
- Features quantization. Any source of information can be used as an input to HTI, and be encoded with one or more features.

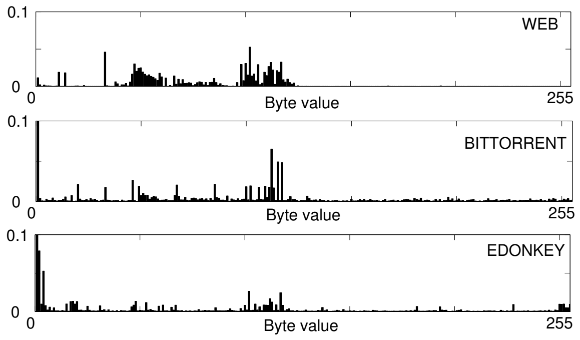


Figure 2: Histogram of the first data packet payload.

We detail the above key characteristics of HTI in the next two sections. In addition, as stated earlier, HTI is a framework rather than a specific machine learning approach with a specific set of input features and a specific machine learning algorithm. We present at the end of this chapter three machine learning algorithms that we have used in our evaluation of HTI on stored traces in Section 5.

## 3.3 Features

HTI uses input features that are used by classification techniques that differ in nature from the ones used in other works [14]. To do so, we need to adapt the features used by these other techniques to fit in the machine learning framework. We detail below the set of features we use in the present work. Note however that the strategy we use can be easily extended to incorporate as features other techniques, e.g., discriminating information obtained with heuristics [15], end points profiling [22], flows statistics [8, 4] or payload pattern matching information [23, 5].

- **Data packets:** Information about first four data packets of each flow. As noticed in [8], size and direction of few first data packets carry information about the application. It appears to hold for some key applications, for example eDonkey, in our traces [2]. 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 [1] include port numbers in the features set treating

[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 1: Payload signatures used. (Case sensitive)

them as quantitative features which is one of the reasons of the data overfitting issues described in [2].

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. Note 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 use, 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 1. Presence of a signature is denoted as a 1, its absence as a 0.

- **Payload histograms:** Some of the applications carry valuable information in the packet payload, but this information can not be easily transformed into a classical signature. To address this issue, we propose an additional feature, which is inspired by the work in [13] and [24]. During the model building phase we create a model histogram of bytes distribution for a fraction of the payload of all the flows of a given application. During the classification phase, the histogram of a flow under test is compared to the reference histogram using the Kullback-Leibler distance as suggested in [25]. We encode the result into a binary value that takes value 1 if the histogram of flow under test is close enough to the model signature and 0 otherwise. Example of model histograms are presented on Figure 2.

A key advantage of HTI is that all the information collected during the feature extraction phase is cast into a new statistical decision problem. This means for instance that the presence of a signature is not treated as a deterministic indication leading to a unique decision. Instead, HTI will assess the relevance of this information and assign a weight to this feature. This relieves the practitioner from the burden of formulating heuristic rules when various sources of information lead to contradictory results, e.g, the deep packet inspection phase states that the flow is a Web flow while statistical packet inspection states that it is a p2p flow.

### 3.4 Sub-models

To the best of our knowledge, all 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 [14]. For instance, if the machine learning algorithm is a decision tree, a global decision is constructed during the training phase that is to be operated as follows during the deployment phase:

- One first extracts for the newly arriving flow the set of input features used in the decision tree;
- Starting from the root of the tree, one follows the decision tree, evaluating the conditions associated to each branching point.
- One eventually ends up in a leaf of the tree which specifies the most likely application behind the flow.

We adopt a different strategy in this paper where we generate one model per application during the training phase. This slightly modifies the training.

Let us take a concrete example to illustrate this modified procedure. Assume we have only three possible applications, Web, eDonkey and Mail. One first generates the model for Web traffic by merging eDonkey and Mail traffic together into a data set labeled as “non-Web”. The training phase will lead to a model whose output is a binary decision: the tested flow is a Web flow or not. We proceed similarly for the two other classes. At the end of the training phase, we have one model per application. We call these models sub-models to emphasize the fact that they take a decision for one out of the many applications one seeks to detect. Sub-models present the clear advantage in an operational setting that they can be updated independently from each other, e.g., to account for a new release of an application. What is more, sub-models allow for using different detection methods for each class. It is still possible to generate one global model for all applications of interest. We discuss the relative merits of each approach in Section 5.4.

If sub-models are used, a reconciling phase is needed in case a flow matches many sub-models. We rely on the statistical confidence indices provided by the machine learning algorithm to do so, i.e., we pick among the matching sub-models, the one that offers the highest probability score level. We exemplify this reconciling procedure for the case of the logistic regression algorithm that we present in the next section.

## 3.5 Machine Learning Algorithms

We consider three supervised machine learning algorithms (Logistic Regression, Support Vector Machine (SVM) and C4.5) to instantiate the HTI framework. Before detailing those algorithms, we formally define the problem of traffic classification.

Traffic classification consists in associating each flow to an application, based on the features that have been extracted for this flow. A flow is defined as a sequence of packets with the same source IP address, destination IP address, source port, destination port and protocol number. Let $X$ be the n-dimensional random variable corresponding to the flow features. A vector $x$ 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 $y$.

### 3.5.1 Algorithm I - Decision tree C4.5

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. The version we use [26] incorporates a number of improvements such as pruning that aims at reducing data over-fitting. More details about the algorithm can be found in [27]. C4.5 has been widely used for traffic classification, [7, 1, 28].

### 3.5.2 Algorithm II - Support Vector Machine (SVM)

Support Vector Machine (SVM) is a technique for solving classification problem with high dimensional features space. The principle of SVM is to find a separating hyperplane, which maximizes the distance between the closest sample data points in the (reduced) convex hulls for each class, in a multi dimensional features space [29, 30]. Originally the algorithm was designed for binary classification, but multi class extensions exists as well. SVM has been already applied to the traffic classification problem in several works e.g. [1, 30]. We use the implementation in [26].

### 3.5.3 Algorithm III - Logistic regression model

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 [31], business and finance [32], criminology [33], health policy [31] and linguistics [34]. Logistic regression is designed for dichotomous variables. To model the relation between a binary variable (true vs. false) and a set of covariates. To the best of our knowledge the method has never been used in the domain of traffic classification, we thus describe the model in more details than C4.5 and SVM.

When applied to traffic classification, logistic regression will generate $c$ sub-models, one per application. This is because it can test 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 value 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. Note that while logistic regression leads to $c$ sub-models, we have greater flexibility with C4.5 and SVM where one can generate one global model or $c$ sub-models. We discuss this issue in Section 5.4.

Consider a flow with the following features 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[2]

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

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 $x$ parameterized by the weights vector $\beta = (\beta_0, \beta_1, \cdots, \beta_n)$. Since the function $P$ represents a probability, it must take value between 0 and 1. Within the Logistic regression framework, one assumes a specific function P:

$$P(x, \beta) = \frac{e^{\beta_0 + \sum_{j=1}^{n} \beta_j x_j}}{1 + e^{\beta_0 + \sum_{j=1}^{n} \beta_j x_j}}, \tag{2}$$

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

$$\log \left( \frac{P(x, \beta)}{1 - P(x, \beta)} \right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n, \tag{3}$$

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 [35]. Further details on the logistic regression algorithm are provided in Appendix A. Please note that for each application we have a distinct $\beta$ 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

[2]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 $\beta$ vectors that are application dependent in both length and value.

| **Set** | **Date** | **Start** | **Dur** | **Size [GB]** | **Flows [M]** | **TCP [%]** | **TCP Bytes [%]** | **Local users** | **Distant IPs** |
|---|---|---|---|---|---|---|---|---|---|
| **M-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 |

Table 2: Traces summary.

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 $P(x, \beta) \geq 0.5$ which is equivalent to:

$$\beta_0 + \sum_{j=1}^{n} \beta_j x_j > 0 \tag{4}$$

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:

$$\arg \max_{k=1,\ldots,c} \left\{ P(x, \beta^{(k)}) | P(x, \beta^{(k)}) \geq 0.5 \right\} \tag{5}$$

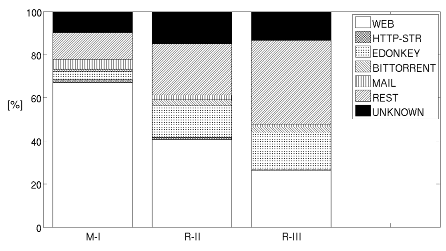
where $\beta^{(k)}$ is the beta vector for application number $k$. If for a given flow, no sub-model decision is above the threshold, the flow is declared as unknown.

## 4 Off-line evaluation: warming-up

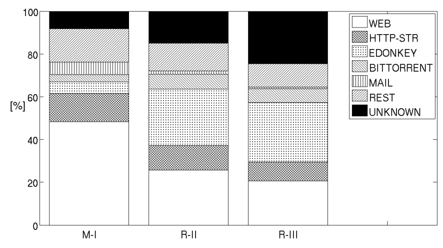
In this section, we report on the performance of an HTI instance, based on logistic regression algorithm, in an off-line scenario using ADSL traces. We first present the data set, reference point establishment method and the flow features considered. We next report on the portability results obtained for all possible cross-site scenarios. By cross-site, we refer to the case where, for traces collected on different sites, we train the classifier on the trace collected on one site and apply the resulting model on a different site. We call static case, the case where training is performed on the same site on which the classifier is used (training and testing sets are chosen so as to not intersect in this case). If we have $n$ sites, each with one trace, we thus have $n$ static results and $n(n-1)$ cross site results. We have highlighted in [2] the importance of performing cross site studies to detect data over fitting issues. A typical example is the direct use of port number that leads the classifier to learn the ports used by the p2p clients of local customers (managed by the ISP), which improves the results on static cases, but is detrimental on cross-site cases.

### 4.1 Traffic Data

Let us present our data set, how we establish the reference point (ground truth) that is used as benchmark for our HTI classifier, the definition of our traffic classes



(a) Breakdown in flows



(b) Breakdown in bytes

Figure 3: Application breakdown in the data sets.

and the traffic breakdown.

Our data set consists of three recent packet traces collected at two 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 contains over four million TCP flows. Each trace contains at least one hour of full bidirectional traffic, with similar number of active local users varying between 1380 and 2100. For details, see Table 2.

### 4.1.1 Reference point

In order to benchmark the performance of any classification method, a training set, that we term our reference point (a.k.a ground truth) is needed. Signatures commonly used in recent works [1, 12] provide deceptive results with our traces, as more than 55 % of the flows are classified as unknown. In this work, 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 have compared ODT to Tstat [5], whose latest version features DPI functions, in [36]. Specifically, we have shown that ODT and Tstat v2 offer similar performance and outperform signature based tools used in the literature [1, 12]. As ODT embeds a larger set of signatures than Tstat v2, we rely on the former to establish the ground truth in our study.

| Class | Application/protocol |
|---|---|
| WEB | HTTP and HTTPs browsing |
| HTTP-STR | HTTP Streaming |
| EDONKEY | eDonkey, eMule obfuscated |
| BITTORRENT | Bittorrent |
| MAIL | SMTP, POP3, IMAP, IMAPs<br>POP3s, HTTP Mail |
| REST | MSN, FTP, NBS, NEWS, GAMES, STREAMING<br>P2P other, Attacks |
| UNKNOWN | Other not recognized by ODT |

Table 3: Application classes.

We are aware that the wording ground truth remains tricky as even DPI tools might fail. We face here the same issue as former studies in the domain. However, there barely exists any alternative to DPIs. Some approaches have been recently proposed to obtain high quality reference data sets. In [37], the authors propose a network driver installed on end hosts. This middleware flags flows according to the application generating traffic. However, this solution is not applicable to the case of large ADSL traces. A similar approach was recently proposed in [38].

### 4.1.2 Traffic breakdown

Classes used in this work are summarized in Table 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 two classes depending on the application implemented on top: HTTP streaming and WEB, i.e., all other HTTP based applications. Second, the most popular p2p applications among the customers of the ISP[3] have their own class. Less popular p2p applications are endemic and merged into the REST class. More generally, the REST class aggregates less popular classes of traffic, including unidirectional flows (See Table 3).

Figure 3 shows classification results obtained by ODT, in flows and bytes, for our three 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 p2p applications, most bytes and flows are due to eDonkey (more precisely eMule client [41]) followed by Bittorrent. Concerning eDonkey, we observed that obfuscated traffic accounts typically for half of the bytes in the EDONKEY class. Less popular applications (REST class) generated a small amount of bytes (8-15%). We exclude them from our subsequent analysis, as we exemplify HTI framework using most popular application classes.

[3]Studies of ADSL networks in Germany and the USA suggest that eDonkey and BitTorrent are also the two major p2p applications in those countries [39, 40].

The vast majority of traffic in the HTTP Streaming class is due to Dailymotion [42] and Youtube [43], which account for 80% of the bytes. P2P streaming applications, that fall into the REST 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 [24]. A large fraction of flows in the REST class are unidirectional flows to ports 135, 445 and 139. Those Windows services are targeted by a large family of self-propagating malware (see for instance [44]).

Overall, ODT provides fractions of UNKNOWN bytes that range between 8% and 24% depending on the trace.

In the remaining of this section, we present the calibration results of our (logistic regression based) instance of HTI, restricting to the following subset of applications: BITTORENT, EDONKEY, HTTP-STREAMING, WEB and MAIL. These classes were chosen for two reasons. First, they explain the vast majority of the bytes on the ADSL platform we consider (see Figure 3). Second, they encompass a diversity of protocols that enable us to challenge our HTI instance.

## 4.2 Flow definition

We seek to classify bidirectional TCP flow. We use the definition of a flow based on its 5-tuple {*source IP address, destination IP address, protocol, source port, destination port*}. 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.

Since we use information related to the first few packets of each TCP transfer, we restrict our attention to TCP flows for which a three-way handshake is observed and at least four data packets are exchanged. This can lead to a significant reduction of the number of flows we analyze but has little impact on the number of bytes due to the heavy-tailed nature of the Internet traffic. For instance, for the case of trace M-I, we end up analyzing 34% of the flows and 69% of the bytes. Note that if we had restricted ourselves to TCP flows for which a three-way handshake is observed, we would have respectively analyzed 77% of flows and 75% of bytes. We refer the reader to our previous work [2] where we discuss in details different flow definitions for the same traces as the ones used in the present work.

## 4.3 Training procedure

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. 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.

### 4.4 Performance metrics

We use accuracy and precision to assess the quality of our statistical classifier. 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 fraction of WEB flows that are labeled (resp. not labeled) as WEB by the statistical classifier. FPs (resp. TNs) are the fraction of flows not labeled as WEB by ODT that are labeled (resp. not labeled) as WEB by the statistical classifier.

Accuracy and precision are defined as follows:

- **Accuracy**, **a.k.a. Recall**: Accuracy corresponds to the 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, an accuracy of 50% for the WEB class means that only half of the WEB flows/bytes are labeled correctly by the statistical classifier.

- **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 accuracy.

A classifier works well if it offers, not only high overall accuracy, but also a high precision for all classes. Depending on the precise requirements, flows or bytes figures might be considered more important.

## 5 Off-line evaluation: performance results

### 5.1 Overall results

We evaluate our classifier in both single site and cross site scenarios. Table 4 presents classification results for each application class in terms of accuracy and precision in both flows and bytes. Overall results are clearly impressive as no class goes below 90% of accuracy and precision in both bytes and flows.

We repeat each experiment five times using randomly selected training sets to confirm that results are stable. Table 5 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.

To further highlight the benefits of the HTI approach, it is important to contrast the above results with the ones obtained in previous studies:

- In [2], we evaluated several machine learning algorithms with purely statistical features (used in [8] and [7]) on the same traces. We observed performance degradation for some applications in cross-sites experiments, e.g.,

| | | Accuracy [flows% \| bytes%] | | | Precision [flows% \| bytes%] | | |
|---|---|---|---|---|---|---|---|
| | ↓Training | **M-I** | **R-II** | **R-III** | **M-I** | **R-II** | **R-III** |
| WEB | **M-I** | 95% \| **92%** | 98% \| **93%** | 97% \| **96%** | 99% \| **97%** | 99% \| **95%** | 99% \| **95%** |
| | **R-II** | 94% \| **92%** | 97% \| **92%** | 97% \| **96%** | 99% \| **97%** | 99% \| **94%** | 99% \| **92%** |
| | **R-III** | 92% \| **92%** | 96% \| **92%** | 96% \| **96%** | 99% \| **97%** | 99% \| **95%** | 99% \| **95%** |
| HTTP-STR | **M-I** | 98% \| **99%** | 96% \| **99%** | 98% \| **99%** | 93% \| **96%** | 96% \| **98%** | 95% \| **99%** |
| | **R-II** | 98% \| **99%** | 96% \| **99%** | 98% \| **99%** | 93% \| **96%** | 96% \| **98%** | 95% \| **99%** |
| | **R-III** | 98% \| **99%** | 96% \| **99%** | 98% \| **98%** | 91% \| **96%** | 95% \| **98%** | 94% \| **99%** |
| EDONKEY | **M-I** | 99% \| **99%** | 98% \| **98%** | 98% \| **98%** | 91% \| **95%** | 95% \| **94%** | 98% \| **98%** |
| | **R-II** | 97% \| **98%** | 96% \| **97%** | 97% \| **97%** | 92% \| **95%** | 97% \| **95%** | 98% \| **98%** |
| | **R-III** | 97% \| **99%** | 98% \| **98%** | 97% \| **98%** | 92% \| **96%** | 95% \| **94%** | 98% \| **98%** |
| BITTORRENT | **M-I** | 100% \| **100%** | 99% \| **99%** | 97% \| **98%** | 96% \| **98%** | 98% \| **99%** | 98% \| **99%** |
| | **R-II** | 100% \| **100%** | 99% \| **100%** | 99% \| **99%** | 99% \| **98%** | 99% \| **100%** | 99% \| **100%** |
| | **R-III** | 100% \| **100%** | 99% \| **100%** | 99% \| **99%** | 99% \| **98%** | 99% \| **100%** | 99% \| **100%** |
| MAIL | **M-I** | 94% \| **97%** | 99% \| **100%** | 100% \| **99%** | 94% \| **99%** | 99% \| **100%** | 99% \| **100%** |
| | **R-II** | 90% \| **95%** | 99% \| **100%** | 99% \| **100%** | 99% \| **99%** | 99% \| **100%** | 100% \| **100%** |
| | **R-III** | 90% \| **95%** | 99% \| **100%** | 99% \| **99%** | 99% \| **100%** | 99% \| **100%** | 99% \| **100%** |

Table 4: Off-line classification results. [flows%/bytes%]

| Class | Accuracy [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 5: Maximum deviations of off-line classification results, for random training sets. [flows%/bytes%]

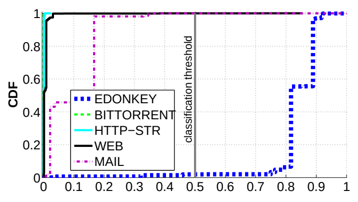
accuracy of BitTorrent falls to 58% in some cases. Moreover, the classification of HTTP streaming with those methods lead to accuracy scores below 20% and precision scores below 60%.

- Other studies tackled the problem of cross-site classification. In [1], a SVM-based classifier trained on trace KAIST-I achieved an overall accuracy of 49.8% when applied to trace PAIX-II. Similarly, when trained on trace KEIO-I and used on trace KAIST-I, the overall accuracy was only 63.6%. The authors in [7] reported an accuracy score of 58% in a cross site experiment for p2p applications.

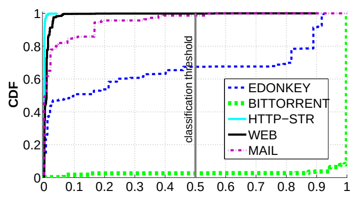
## 5.2 Multiple classifications

In the classification phase each flow is tested against all the possible classifiers, and so, multiple classifications might occur, for instance, a flow might be associated to more than one class. As explained in section 3.4, for each flow, we pick the class corresponding to the highest probability score. However, even if we keep all the classification results, multiple classifications are rare events affecting at most a few percents of the bytes. Figure 4 depicts the probability distributions for each sub-classifier with classification threshold. For example Figure 4c 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).

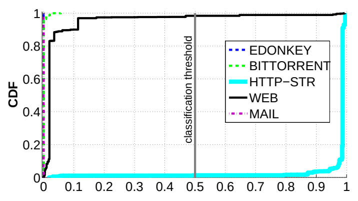
Figure 4b shows that eDonkey is the only application that may suffer from double classifications as almost 35% also match our BitTorrent sub-model. However, when picking the class with highest probability, all those flows are correctly classified as eDonkey.



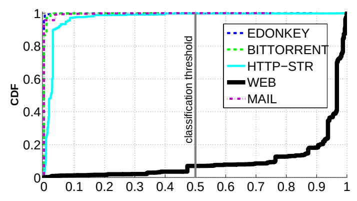
(a) EDONKEY



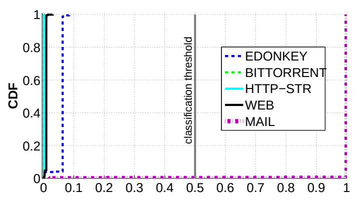
(b) BITTORRENT



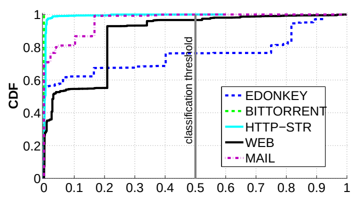
(c) HTTP STREAMING



(d) WEB



(e) MAIL



(f) REST

Figure 4: Probability scores for each class model. Training on MI test on RIII.

## 5.3 Which method for which application?

The use of logistic regression has an additional advantage here in that it assigns weights ($\beta$ vector) to each input feature and a too low value reveals that the feature is useless to classify the corresponding class. Table 7 (see Appendix B) presents the $\beta$ 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 accuracy and precision. HTI thus offers a powerful framework to combine many sources of a priori heterogeneous information without requiring the practitioner to choose among contradictory results that could be obtained by using in parallel a DPI and a (classical) statistical approach.

We report below on the key insights obtained with the study of the $\beta$ values in Table 6:

- 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 signatures (see Table 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 only leads to poor results in some cross site experiments [2]. Adding payload signature leads to a perfect accuracy and precision[4].

- 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 classical payload signature instead, we would miss a large fraction of bytes carried by obfuscated eMule flows.

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

As for the bytes histogram feature, after multiple tests it was removed from the classifier. It only improved the results for BITTORRENT class in the static case but deteriorates them in the cross site cases.

For the set of application classes we considered, the set of features we used was sufficient to obtain good results. However, HTI is a generic framework that can be tuned and extended as needed.

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

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

Table 6: Methods playing important role in the model for each class

## 5.4 Impact of the classification algorithm and of the use of sub-models

In this section, we discuss the impact of the exact machine learning algorithms used to instantiate HTI, as well as the impact of using single classifier for all classes versus one sub classifier per class. Finally, we report on the results obtained with raw (non quantized) statistical features. For the sake of clarity, we present only a summary of the experiments, indicating most significant results. We considered two alternative algorithms to Logistic regression, namely C4.5 and SVM. However, the results obtained with C4.5 and SVM are highly similar, and we thus report on C4.5 only.

We evaluated the following scenarios:

1. Single classifier - Quantized features.
   In this case, we have a *single* decision tree for all classes. Features are all binary values. Each application specific feature (like payload signature) is encoded using one binary value, which increases the total number of features. We use exactly the same data sets and features as in the case of logistic regression.

2. One sub-classifier per class - Quantized features.
   A *separate* decision tree is trained for each class of traffic. Features are all binary values. We use exactly the same data sets and features as in the case of the logistic regression.

3. Single classifier - raw statistical features.
   In this case we build a single decision tree for all classes of interest. However as opposed to previous cases we use the raw packet sizes as well as the direction of the first data packets and the port numbers.

Comparing scenarios (1) and (2) with the results obtained with logistic regression in Table 4 we conclude that:

- We have almost a perfect match between all three scenarios, with a largest deviation of 3% over all single-site and cross-site cases.

- Using one classifier for all classes (1) or multiple sub-classifiers (2) led to very similar results in terms of accuracy an precision.

As for scenario (3) (raw statistical features) we observe issues described in details in [2], namely:

- Degradation of performance in cross site cases. BITTORRENT is one of the particularly concerned classes.
- Non acceptable performance with HTTP-STREAMING.

Scenario (3) confirms that relying purely on statistical flow features is not sufficient for some applications.

The main lesson learned from those experiments with different algorithms is that the results obtained in Table 4 are not specific to the logistic regression algorithm but can also be obtained with other machine learning algorithms.

We use the logistic regression as it preserves the modularity (flexibility) of design, leading to easily expendable sub-classifiers. Moreover, it builds simple and very easy to interpret models ($\beta$ values) that allow intuitive understanding of the impact of each feature (method) in the classification process.

## 6 HTI in the wild - production deployment

Results of the previous section demonstrated that our Logistic regression instance of HTI is accurate and portable when applied on passively captured traces. We now report on an HTI deployment in a production environment of a large ADSL platform.

### 6.1 Platform Details

We deployed our HTI classifier in one of the aggregating links at an ADSL platforms servicing around 16,000 customers. It is a different PoP from the ones used in Section 5 that were fairly 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 instance is currently deployed on one of those switches and we need to observe the two directions of traffic to take classification decisions. 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 a vast majority of users of the platform.

Classification was performed on a two core Intel(R) Xeon(TM) CPU 2.80GHz machine with 2 GB of RAM. Operating system used was Mandriva Linux. The classification tool was developed in C.

## 6.2 Implementation Issues

The major implementation challenge we faced is a correct memory management as we keep per flow records. 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 memory consumption low – between 64 and 150 MB – as only 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 signature) 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.

As for the CPU consumption, it is as well kept low (12%-20%) as feature extraction is cheap and the classification process boils down to run the $c$ sub-classifiers ($c$ equals to number of classes considered) on each flow of more than 4 data packets.

## 6.3 Model Validity

We used a model (consisting of several sub-models) generated out of the M-I trace (see Section 5). Given that the M-I trace was captured more than one year ago, we need to assess the accuracy 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 day four captures of one hour are performed, one in the morning, one in the afternoon and two in the evening. 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 6 the accuracy[5] in flows and bytes respectively of the model trained over the M-I trace against the ground truth as obtained by ODT. We observe a very good agreement between the model and ground truth. The discrepancy observed between accuracy 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 labelling them as WEB and not HTTP-streaming affects our results both in bytes or flows.

[5]the precision figures are similar to the accuracy, we won't report on them due to space limitation

As HTI relies partly (but not only, see Table 6) on the signatures in Table 1 to separate HTTP streaming flows from WEB flows, one can expect to improve results by extending our set of signatures. Note that the signatures we used are the same as the ones used in [39].

## 6.4 Live Experiment Results

Results of the classification for one week are presented in Figures 5.
We note that:

- The accumulated size of classified data was more than 2 TB and represent more than 18 millions flows.
- Using just a small number of classes we are able to classify 84 % of the bytes.
- Uplink traffic is still dominated by peer-to-peer applications: Edonkey followed by Bittorrent. In contrast, the downlink carries additionally a significant amount of HTTP-STR and WEB traffic. Increasing importance of the HTTP driven applications as compared to peer-to-peer is in line with the findings in [39, 40] where the traffic from large populations of ADSL users (30K and 100K respectively) was analyzed in 2009.
- Peer-to-peer traffic exhibits small daily variations in contrast to HTTP-Streaming and WEB, which are clearly dependent on users interaction.
- The volumes for each type of traffic is fairly stable over the course of the week, despite the fact that the set of users we observe varies over time due to our measurement set-up. We intend to further analyze this phenomenon once several months of data will have been collected.

## 6.5 Live Deployment Benefits

The benefits that come from a live deployment of our logistic regression based HTI instance are:

- Constant monitoring of the network can be done with a low storage requirement. A large ADSL platform can generate more than 100 TB of data in just one month. Live classification lets us monitor this traffic continuously without storing any packet level traces. With this kind of design, we can monitor mid and long term trends in application usage.
- Our classification decision is based on the beginning of the flows (e.g., using the first four data packets of each transfer). This means that we can not only passively classify flows, but also take active decision about, e.g., QoS or even routing based on the traffic class.

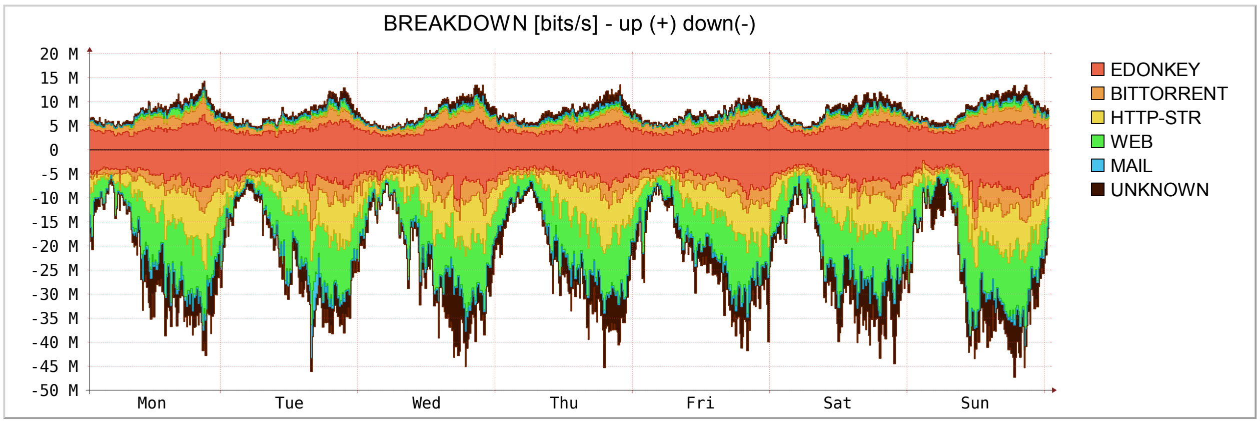


Figure 5: Traffic rate evolution - one week. Aggregation resolution of five minutes.

## 6.6 Discussion

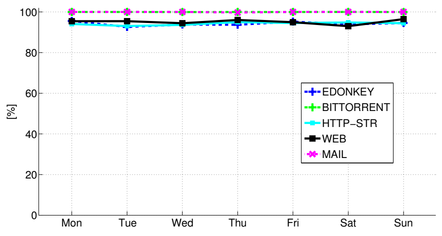
Although we believe that the HTI framework, by allowing the combination of several classification methods, is an important step forward, it does not solve all the issues. Some of the still existing limitations are:

- Building the classifier, method selection and calibration is still a labor intense iterative process requiring some sort of ground truth. The ground truth problem might be solved by the recent efforts to generate reference sets, described in [37, 38]. However, for the labor intensity of the process, there is no efficient alternative to the best of our knowledge.

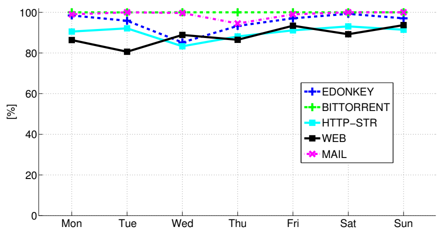
- We lack an oracle to trigger a re-training of the sub-models upon the detection of abnormal variations of the traffic in the corresponding class. It is easy to imagine that a new release of a protocol can alter the features we rely on. However, as the deployment of this version might be progressive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

- Statistical features can handle well some of the obfuscated protocols (e.g. EDONKEY), however handling VPN tunneling consisting of multiple applications is still an open issue.

- 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. The HTI framework can help integrating such new techniques with previously proposed method.



(a) flows



(b) bytes

Figure 6: Accuracy of HTI in Live experiment.

## 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 framework, 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 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 outperforms the classical classification schemes, but, in addition, it is suitable for cross-site classification. We further reported on the use of an HTI instance that takes its decision on the fly for all the traffic generated by the customers of an ADSL platform.

Overall, the HTI framework 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.

## A Logistic Regression - model building

We describe here the model building phase($\beta$ estimation) for the logistic regression. We exemplify the process for the case of a single application denoted as A. The same procedure as per 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 [35]:

$$\begin{aligned} P(X,\beta) &= \prod_{j=1}^{N} p(Y = y_j|X_j) \qquad (6) \\ &= \prod_{j=1}^{N} (p(Y = 1|X_j)^{y_j}(1 - p(Y = 1|X_j))^{1-y_j} \end{aligned}$$

As the values of $p$ are small, it is common to maximize the log-likelihood $L(X,\beta) = \log P(X,\beta)$ instead [35], to avoid rounding errors.

By substituting the value of $p(Y = 1|X_j)$ by its value defined in Equation (2) we get the log-likelihood for the logistic regression:

$$L(X,\beta) = \sum_{i=1}^{N} \left[ y_i\beta^T X_i - log(1 + e^{\beta^T X_i}) \right] \qquad (7)$$

In the logistic regression model, we wish to find $\beta$ that maximizes Equation (7). Unfortunately, this can not be achieved analytically. In this work, we compute it numerically using the Newton-raphson algorithm [35]. 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 $\beta$.

From Equation (7) we can derive the first derivative

$$\frac{\partial L(X,\beta)}{\partial \beta} = \sum_{i=1}^{N} X_i(y_i - p(x_i,\beta)) \qquad (8)$$

We now derive the Hessien matrix

$$\frac{\partial^2 L(\beta)}{\partial\beta\partial\beta^T} = -\sum_{i=1}^{N} X_i X_i^T p(x_i,\beta)(1 - p(x_i,\beta)) \qquad (9)$$

|  |  |  | 1st packet |  |  | 2nd packet |  |  | 3rd packet |  |  | 4th packet |  |  | port | signature |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | $\beta_0$ | 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 |
| HTTP-STR |  | -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 |
| EDONKEY |  | -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 | - |
| BITTORRENT |  | -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 7: Live experiment - beta coefficients

The pseudo code of Newton-Raphson algorithm is depicted in Algorithm 1. We start with a first guess of $\beta$, then we use the first derivative and the Hessien matrix to update $\beta$. Using the new $\beta$ we compute the new loglikelihood. This is repeated until there is no further change of $\beta$. The Newton-Raphson algorithm has been shown to converge remarkably quickly [45]. In this work, it takes less than one second to output an estimate of $\beta$.

**Algorithm 1** Newton-Raphson algorithm

```
1: initialize $\beta$
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 $\hat{V}$
```

## B Live experiment - beta coefficients

See Table 7.

## References

[1] 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*. ACM, 2008, pp. 1–12.

[2] M. Pietrzyk, J.-L. Costeux, G. Urvoy-Keller, and T. En-Najjary, "Challenging statistical classification for operational usage: the adsl case," in *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 122–135.

[3] 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*. ACM, 2004, pp. 121–134.

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

[5] Tstat, “http://tstat.tlc.polito.it/.”

[6] Bro, “http://bro-ids.org.”

[7] W. Li, M. Canini, A. W. Moore, and R. Bolla, “Efficient application identification and the temporal and spatial stability of classification schema,” *Computer Networks*, vol. 53, no. 6, pp. 790 – 809, 2009.

[8] L. Bernaille, R. Teixeira, and K. Salamatian, “Early application identification,” in *CoNEXT ’06: Proceedings of the 2006 ACM CoNEXT conference*. ACM, 2006, pp. 1–12.

[9] L. Bernaille, R. Teixeira, U. Pierre, and M. C. Lip-cnrs, “Early recognition of encrypted applications,” in *In Passive and Active Measurement conference (PAM 07)*, 2007.

[10] 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*, 2005, pp. 50–60.

[11] T. Nguyen and G. Armitage, “Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks,” Nov. 2006, pp. 369–376.

[12] 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*. ACM, 2006.

[13] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, “Revealing skype traffic: when randomness plays with you,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 37–48, 2007.

[14] T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *Communications Surveys and Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2008.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “Blinc: multilevel traffic classification in the dark,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, 2005.

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

[17] 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*, Nov.-3 Dec. 2004, pp. 1532–1538 Vol.3.

[18] 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*, July 2006, pp. 93–102.

[19] 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*, June 2007, pp. 1–8.

[20] Y. L. Jun Li, Shunyi Zhang and J. Yan, “Hybrid internet traffic classification technique,” *Journal of Electronics*, vol. 26, no. 1, pp. 685–694, 2009.

[21] 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.*, vol. 9, no. 2, pp. 685–694, 2009.

[22] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci, “Unconstrained endpoint profiling (googling the internet),” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 279–290, 2008.

[23] 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*. ACM, 2004, pp. 512–521.

[24] A. Finamore, M. Mellia, M. Meo, and D. Rossi, “Kiss: Stochastic packet inspection,” in *TMA ’09: Proceedings of the First International Workshop on Traffic Monitoring and Analysis*. Springer-Verlag, 2009, pp. 117–125.

[25] SPID, “http://spid.wiki.sourceforge.net.”

[26] WEKA, “http://www.cs.waikato.ac.nz/ml/weka/.”

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

[28] 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.*, vol. 36, no. 5, pp. 5–16, 2006.

[29] K. P. Bennett and C. Campbell, “Support vector machines: hype or hallelujah?” *SIGKDD Explor. Newsl.*, vol. 2, no. 2, pp. 1–13, 2000.

[30] A. Este, F. Gringoli, and L. Salgarelli, “Support vector machines for tcp traffic classification,” *Comput. Netw.*, vol. 53, no. 14, pp. 2476–2490, 2009.

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

[32] 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.

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

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

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

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

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

[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.*, vol. 39, no. 5, pp. 12–18, 2009.

[39] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, 2009, pp. 90–102.

[40] J. Erman, A. Gerber, M. T. Hajiaghayi, D. Pei, and O. Spatscheck, "Network-aware forward caching," in *WWW*, 2009, pp. 291–300.

[41] eMule, "http://www.emule-project.net/."

[42] Dailymotion, "http://dailymotion.com/."

[43] Youtube, "http://youtube.com."

[44] H. project., "http://www.leurrecom.org."

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