Statistical traffic classification by Boosting Support Vector Machines

Gabriel Gómez Sena
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
ggomez@fing.edu.uy

Pablo Belzarena
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
belza@fing.edu.uy

ABSTRACT

In recent years, traffic classification based on the statistical properties of flows has become an important topic. In this paper we statistically analyze the data length of the first few segments exchanged by a transport flow. This traffic classification method may be useful for early traffic identification in real time, since it takes into account only the beginning of the flow and therefore it can be used to trigger on-line actions. This work proposes the use of a supervised machine learning method for traffic identification based on Support Vector Machines (SVM). We compare the SVM classification accuracy with a more classical centroid based approach, obtaining good results. We also propose an improvement of the classification accuracy preformed by one single SVM model, introducing a weighted voting scheme of the verdicts of a sequence of SVM models. This sequence is generated by means of the boosting technique and the proposed method improves the classification accuracy of poorly classified classes without noticeable detriment of the other traffic classes. This work analyzes the behavior of both TCP and UDP transport protocols.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management, Network monitoring*

Keywords

Traffic identification, Traffic classification, Support Vector Machines, Boosting

1. INTRODUCTION

Identification and classification of Internet traffic is essential to manage large networks. Common tasks such as network design, provisioning, optimization and quality of service management, require a good knowledge of the types of traffic traversing the network. Traffic identification and classification is also a relevant topic in the context of network security for intrusion and anomaly detection.

Traffic classification based on simple transport header information, like TCP or UDP port numbers, is a well known method but has become inappropriate nowadays. Many modern applications use dynamic port negotiation or try to hide themselves using well known ports to bypass firewalls and other network security devices [1, 2, 3].

Classification methods by payload inspection, based on searching for specific patterns within TCP flows, have been proven to be accurate, but have some disadvantages. First, searching for patterns within the payload may be a slow task. Second, they are sometimes questioned because they need access to private information. Third, those methods do not work for encrypted traffic [1, 4, 5].

In recent years, traffic classification based on the statistical analysis of flow properties has been addressed by many authors. Techniques based on the analysis of packet sizes, inter arrival time or other features have been proposed to classify internet traffic [6, 7, 8, 9, 1, 10, 11].

This work looks toward on-line traffic classification, focusing on simple and fast statistical analysis methods.

We analyze statistically the data length of the first segments exchanged by a transport flow, showing that it is a relevant feature. Our proposal includes the use of a supervised based machine learning method, specifically Support Vector Machines (SVM) [12]. We also improve the classification accuracy by a weighted voting scheme of the verdicts of a sequence of SVM models. This sequence is generated using the boosting technique [13].

We improve and extend the results of our previous work in this topic [14]. First of all, we propose an improvement of the traffic classification based on a single SVM model, introducing the boosting technique and a voting scheme of multiples SVM models. We also extend the analysis to UDP transport protocol, not analyzed previously. Furthermore, we have changed the flow representation and the way we use to construct the training sets for the machine learning methods, as explained later.

Three traffic datasets are considered, over which two different machine learning based methods are applied, and their accuracy compared. In the first approach, we use a classical centroid based classification to establish a comparison base and in the second one we use our proposed method based on SVM. Finally we propose the use of the boosting technique to generate a sequence of SVM models. The final traffic class assigned to each flow will result of a weighted voting scheme of the verdicts of the SVM models previously generated. This technique improves the classification accuracy of poorly classified classes without detriment of right classified traffic classes.

The main contributions of this work can be summarized as follows:

- Apply SVM to classify TCP and UDP traffic considering only the data length of the first few segments exchanged in a flow
- Show that SVM classification obtains better accuracy than classical centroid classification approach
- Show that a voting scheme of SVM models generated by the boosting technique, can improve the classification accuracy of poorly classified classes.

The remainder of this paper is structured as follows. In Section 2 we give the methodological overview, defining the flow representation and some features of the data traces analyzed. In Section 3 we describe the implemented centroid approach and its results. In Section 4 we detail our proposed implementation based on SVM and its results. We compare and analyze both implemented methods in Section 5, and in Section 6 we propose a classification accuracy improvement by applying the boosting technique. Section 7 addresses some aspects of the influence of *unknown* traffic. Finally, in Section 8 we present the most relevant conclusions of this work and some trends for future work.

2. METHODOLOGY OVERVIEW

This work proposes a statistical analysis of the data length of the first N segments exchanged by a transport flow using a supervised machine learning approach. We apply the SVM technique which is known to be a method with high discriminative power.

As shown in [14, 6], the data length of the first segments in a flow, is a relevant feature for traffic classification. Other properties like inter-arrival time, jitter, and variance of packet size, provide less discrimination power.

We discuss two supervised machine learning methods for clustering. These methods need a training phase and a testing phase. In the training phase, part of the pre-classified traffic is used to train the machine, defining the traffic classes. In the testing phase, we apply the machine learning algorithm to classify each flow into the learned traffic classes. Comparing the pre-classified traffic class with the traffic class provided by the testing phase, we can calculate the accuracy of the classification method. The accuracy is defined as the the fraction of well classified traffic (true positive plus true negative) over the total tested traffic.

In both machine learning methods, we have constructed the training set using the same percentage of the total flows available for each class. For example, the training set is made with 30% of available *smtp* flows, plus 30% of the available *http* flows, and so on. This approach leads to greater accuracy results than the reported in our previous work [14] where the training set was constructed with the same amount of flows of each class.

The first implemented classification method uses a centroid approach, applying the euclidean distance to assign flows to defined classes [6]. The second –and proposed– method, is based on the use of SVM. Afterwards the classification accuracy of both methods is compared. Moreover, and in order to improve the accuracy of poorly classified traffic types, we propose a boosting scheme to generate a sequence of SVM models and a voting procedure to obtain the final verdict.

We test our approach using three different traffic datasets as detailed later.

Flow representation

We represent a transport flow by a vector of data lengths and a *label* identifying the traffic class. We consider only those segments whose application data length is greater than zero. This means that segments like TCP connection establishment and acknowledgements are not taken into account. Since these segments are generated by TCP, and not the application, it is natural not to consider them. We take into account the sequence of segments exchanged by the flow by adding a minus sign to the data length of the segments in the reverse path of the flow, reflecting the application behavior.

This flow representation is a major change from our previous work [14] where we consider the data length of N segments in each direction of the flow (N forward and N backwards). This new approach reflects not only the size of the exchanged segments of a flow but also the sequence in which they are exchanged. Also it gives better results.

Datasets and ground truth

Good initial traffic classification labels, called "ground truth" are essential for a proper evaluation of traffic classification methods [15]. Some efforts have been made in recent years in order to develop frameworks that can be used to compare traffic classification methods, like TIE [16] or NeTraMark [17]. Projects like WEKA [18], focusing on machine learning algorithms comparison and evaluation, can also be useful. However, the main challenge for a proper evaluation and comparison of techniques is the lack of public availability of pre-classified datasets. The low development activity of these cited projects seems to confirm this assertion. When we started working on this issue [14], those tools had not been reported yet, so we started developing our own tools. In this work we used and extended our own tools.

In this work we consider three different traffic datasets.

1. The Telecommunication Networks group @ UniBS has made available a tcpdump dataset with associated ground truth information (*unibs* dataset) [19]
2. In [20], the authors provide a tcpdump dataset with associated ground truth information (*measurement* dataset)
3. Residential traffic in tcpdump format provided by a national ISP's

The first two datasets included associated ground truth information. For these cases we extract the data length of the segments exchanged by each flow and correlate them with ground truth information. The third dataset contains bidirectional flows, limited in capture to 200 bytes per packet. This dataset has multiple protocol hierarchy: VLAN, Ethernet, PPPoE, PPP, IP, TCP/UDP and Application. Therefore, after stripping packet headers, we are left with only 130 bytes of layer 7 payload for TCP traffic. This fact may reduce the accuracy of the classification method used to associate ground truth information, based on payload inspection.

For classifying the *isp* dataset we implemented a classifying architecture based on Linux L7-Filter [21]. We re-send the captured data to a Linux machine with L7-Filter, obtaining by deep packet inspection, the ground truth classification for the flows. After that, by means of a *perl* [22] script we extract the data length segment sizes exchanged by each flow and then correlate them with the ground truth labels assigned in the previous phase. For this correlation, the flows are identified by the classical 5-tuple: source and destination IP address, source and destination port and transport protocol (TCP or UDP).

3. CENTROID BASED CLASSIFICATION

Just to make this paper self-contained, we will give a brief description of the naive centroid classification used to establish a comparison base.

As mentioned in Section 2, a flow j of traffic class m can be described as a vector V_j^m of segments data length where positive values represents the forward way of the TCP connection and negative values, the backward way: $V_j^m = (v_{j1}^m, v_{j2}^m, .., v_{jN}^m)$ where:

- m is the ground truth associated traffic class,
- N is the number of segments considered to characterize the flow (the first N exchanged),
- v_i^m is the data length of the i^{th} segment exchanged by the flow

In the training phase, we consider t^m vectors V_j^m to define the centroid vector for traffic class m. As stated earlier, to train each class we use the same percentage of the total flows available for this class, so t^m is the number of training flows of class m.

The centroid is a vector $C^m(t^m)$ that can be defined as:

$$C^m(t^m) = \left(c_1^m(t^m), c_2^m(t^m), .., c_N^m(t^m)\right)$$

where:

$$c_i^m(t^m) = \frac{\sum_{j=1}^{t^m} v_{ji}^m}{t^m}, \forall i = 1 \dots N$$

We also define a standard deviation vector:

$$S^m(t^m) = \left(s_1^m(t^m), s_2^m(t^m), .., s_N^m(t^m)\right)$$

where:

$$s_i^m(t^m) = \sqrt{\frac{1}{t^m - 1}\sum_{j=1}^{t^m}\left(v_{ji}^m - c_i^m(t^m)\right)^2}\ ,\ \forall i = 1 \dots N$$

Therefore, each traffic class m is represented by the couple of vectors $C^m(t^m)$ and $S^m(t^m)$.

In the testing phase we calculate the distance of each test flow to all traffic classes and we assign it to the class with the minimum distance.

To improve the classification accuracy for this method we have included the standard deviation into the distance definition. This distance definition is useful to weight down the influence of high variable coordinates [14]. The distance of a test flow $W^l = (w_1^l, w_2^l, .., w_N^l)$ to traffic class m characterized by $C^m(t^m)$ and $S^m(t^m)$, is defined as:

$$d^m(N, t^m) = \sqrt{\sum_{i=1}^{N}\left(\frac{w_i^l - c_i^m(t^m)}{s_i^m(t^m)}\right)^2}$$

In all three datasets we could observe that considering more than 3 or 4 segments of the flow does not improve accuracy. This is consistent with the results stated in [14, 6].

It can also be verified that 30% of flows of each class are enough to characterize traffic classes, as shown in figures 1, and 2. Moreover, it can be seen that different traffic classes exhibit different accuracies. The graphical results per traffic type are shown in Section 5 where we compare the accuracy of the implemented methods.

Due to space limitations and to improve the readability, we have chosen some representative graphs to illustrate the obtained results.

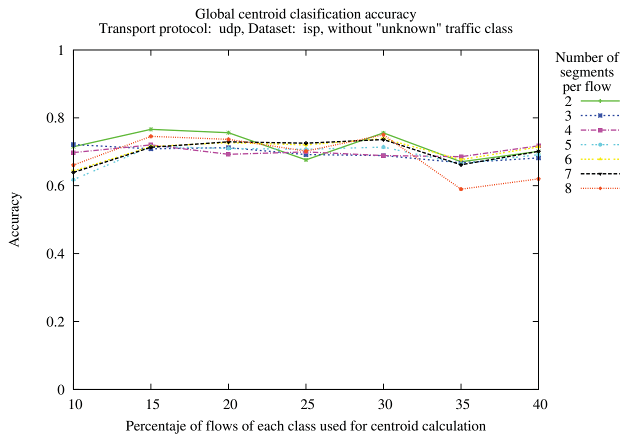

Figure 1: Centroid accuracy (UDP, *ISP*)

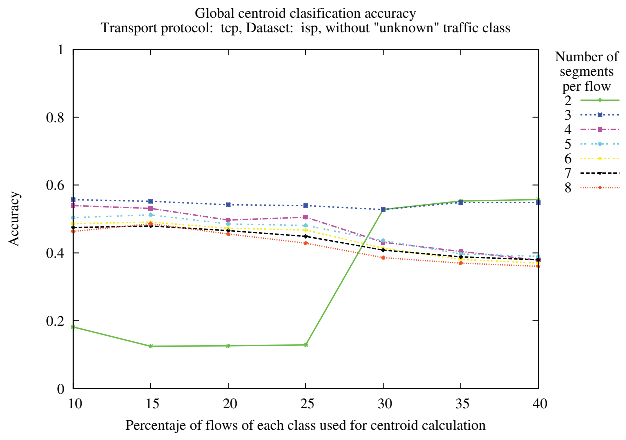

Figure 2: Centroid accuracy (TCP, *ISP*)

4. SVM CLASSIFICATION

This Section describes our proposal for traffic clustering based on SVM.

This technique has been proposed for traffic classification [23, 3, 24, 25], with other goals or using different traffic features. As far as we know, traffic classification using the segment sizes and the SVM technique has been proposed in our previous work [14] and also in [26] at almost the same time.

Brief introduction to SVM

We now present a brief introduction to SVM. This is a supervised learning tool introduced by Vladimir Vapnik in 1995 [27] and well known for its high discriminative power. The basic SVM scheme classifies data by separating it with hyperplanes, a procedure which is extended to non-linear decision boundaries using the kernel trick described below. A complete description of the method can be found in [28].

Given a set of empirical data:

$$(x_1, y_1) \dots (x_m, y_m) \in \mathbb{X} \times \{\pm 1\}$$

where $\mathbb{X}$ is a set of training samples or inputs x_i with associated labels or outputs y_i, which will be used to train a model. In the first approach we consider only 2 possible outputs thus labeled as $+1$ or -1. This case corresponds to the binary classification problem and we will review the multiclass classification problem later.

The classification problem must predict the associated label $y \in \{\pm 1\}$ to any new sample $x \in \mathbb{X}$, so that the (x, y) pair be similar to the experimental samples used to train the model. Thus, in the binary classification, we need to evaluate the similitude between two samples x and x' searching for a function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$, which assigns a real number to the similitude of two samples x and x'.

A simple function that can be used is the canonical internal product defined as:

$$(\mathbf{x} \cdot \mathbf{x'}) := \sum_{i=1}^{N} x_i x'_i$$

where x_i is the i^{th} coordinate of vector $\mathbf{x}$. In order to use the internal product as as similitude measurement the samples must be represented as vectors in a $\mathbb{H}$ space (known as the "feature space"), by mean of a transformation $\Phi : \mathbb{X} \to \mathbb{H}$. This transformation, known as *kernel*, will associate the vector $\mathbf{x} := \Phi(x)$, in $\mathbb{H}$ space, to each sample x in $\mathbb{X}$ space.

With this representation, the similitude between samples x and x' can be defined as:

$$k(x, x') := (\mathbf{x} \cdot \mathbf{x'}) = \big(\Phi(x) \cdot \Phi(x')\big) \qquad (1)$$

and we can work with the samples as vectors, using lineal algebra and analytic geometry. Moreover, the transformation Φ can be chosen according to the problem.

In the $\mathbb{H}$ space we can use the hyperplane classification technique considering the hyperplanes like $(\mathbf{w} \cdot \mathbf{x}) + b = 0$ where $\mathbf{w} \in \mathbb{H}, b \in \mathbb{R}$, corresponding to decision functions like:

$$f(\mathbf{x}) = sgn\big((\mathbf{w} \cdot \mathbf{x}) + b\big) \qquad (2)$$

If the samples are lineally separables by an hyperplane, we can chose the maximum margin hyperplane, given by:

$$\max_{\mathbf{w},b} \ \min \Big\{ \parallel \mathbf{x} - \mathbf{x}_i \parallel, \mathbf{x} \in \mathbb{H}, (\mathbf{w} \cdot \mathbf{x}) + b = 0, i = 1, \dots, m \Big\}$$

To find this maximum margin hyperplane we have to solve:

$$\min_{\mathbf{w}\in\mathbb{H}, b\in\mathbb{R}} \ \tau(\mathbf{w}) = \frac{1}{2} \parallel \mathbf{w} \parallel^2$$
$$subject\ to: \ y_i\big((\mathbf{w} \cdot \mathbf{x}_i) + b\big) \geq 1, \ i = 1, \dots, m \qquad (3)$$

This is a optimization problem with inequality constraints, solvable with the introduction of Lagrange multipliers and the Lagrangian:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \parallel \mathbf{w} \parallel^2 - \sum_{i=1}^{m} \alpha_i \Big(y_i\big((\mathbf{w} \cdot \mathbf{x}_i) + b\big) - 1 \Big) \qquad (4)$$
$$with\ \boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)\ and\ \alpha_i \geq 0$$

The Lagrangian $\mathcal{L}$ has to be minimized respect to the primary variables $\mathbf{w}$ and b, and maximized respect to the dual variables α_i. By the Karush-Kuhn-Tucker (KKT) [29] optimization conditions, the solution should satisfy:

$$\frac{\partial}{\partial b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \ \frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0 \qquad (5)$$
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \qquad (6)$$

The solution vector $\mathbf{w}$ can be expressed in terms of the training samples, particulary those with non zero α_i, called Support Vectors (SV).

By the KKT conditions,

$$\alpha_i \big[y_i\big((\mathbf{w} \cdot \mathbf{x}_i) + b\big) - 1 \big] = 0, \ i = 1, \dots, m \qquad (7)$$

the SVs lie on the margin and the hyperplane is determined solely by the training samples closest to it, as shown in figure 3.

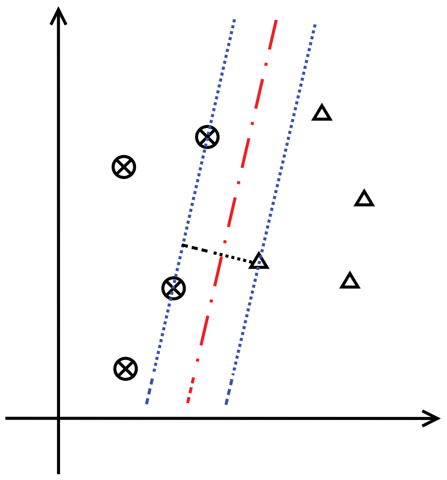

Figure 3: Separables elements. In red (dot-dash) the maximum margin hyperplane. In blue (dots) the margins determined by the SVs

By substituting the equations 5 and 6 into equation 4 one eliminate the primary variables $\mathbf{w}$ and b, arriving to the so called dual optimization problem which is the one usually we solve in practice:

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \qquad (8)$$
$$subject\ to: \begin{cases} \alpha_i \geq 0, \quad i = 1, \dots, m \\ \displaystyle\sum_{i=1}^{m} \alpha_i y_i = 0 \end{cases}$$

Using equation 6, the decision function shown in equation 2, can be written as:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} y_i\alpha_i(\mathbf{x}\cdot\mathbf{x}_i) + b\right) \tag{9}$$

and b can be obtained exploiting the equation 7:

$$b = y_j - \sum_{i=1}^{m} y_i\alpha_i k(w_j, x_i) \tag{10}$$

To express this formulas in terms of the input samples, we need to get back to equation 1, which express the dot product in terms of the kernel k applied to the input samples. This substitution in equation 9, called the kernel trick, leads to:

$$\begin{aligned} f(\mathbf{x}) =& sgn\left(\sum_{i=1}^{m} y_i\alpha_i\big(\Phi(x)\cdot\Phi(x_i)\big) + b\right) = \\ & sgn\left(\sum_{i=1}^{m} y_i\alpha_i\big(k(x,x_i)\big) + b\right)\end{aligned}$$

and the dual problem to solve (equation 8) is now:

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i,x_j) \tag{11}$$

$$subject\ to: \begin{cases} \alpha_i \geq 0, \quad i = 1,\dots,m \\ \sum_{i=1}^{m}\alpha_i y_i = 0 \end{cases}$$

If we need a more general surface to separate samples, we can use non linear kernel transformation so that in the feature space $\mathbb{H}$ the samples can be separated by byperplanes. If we increase the dimensions of the feature space, it is easy to imagine that we will increase the possibility of a hyperplane separation (see figure 4). The idea is to properly choose the kernel k.

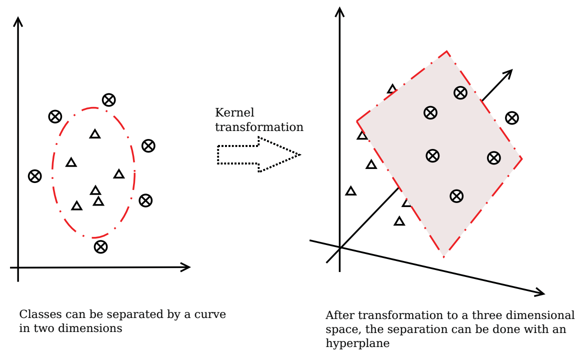
Kernel transformation
Classes can be separated by a curve in two dimensions
After transformation to a three dimensional space, the separation can be done with an hyperplane

Figure 4: Classes can be more easily separated in $\mathbb{H}$ space

In many cases, classes can not be separated by an hyperplane so we have to relax the constraints introducing the adjusting variables:

$$\xi_i \geq 0, \quad i = 1,\dots,m \tag{12}$$

transforming the equation 3 into:

$$y_i\big((\mathbf{w}\cdot\mathbf{x}_i) + b\big) \geq 1 - \xi_i, \quad i = 1,\dots,m \tag{13}$$

Thus, we can obtain one posible solution for this classifier by minimizing the function:

$$\tau(\mathbf{w},\boldsymbol{\xi}) = \frac{1}{2}\|\ \mathbf{w}\ \|^2 + C\sum_{i=1}^{m}\xi_i \tag{14}$$

subject to the constraints stated by equations 12 and 13. The constant $C > 0$ represents the tradeoff between maximizing the separation margin and minimizing the training error. Again, by means of the Lagrange multipliers we arrive to the following optimization problem:

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i,x_j)$$

$$subject\ to: \begin{cases} 0 \leq \alpha_i \leq C, \quad i = 1,\dots,m \\ \sum_{i=1}^{m}\alpha_i y_i = 0 \end{cases}$$

The difference with the linear separable case is that we only need to care about the samples with $\alpha_i \leq C$, thus reducing the impact of outliers samples. The solution is the same as in the linear separable classes and b can be computed in the same way because for the SVs with $\alpha_i < C$, the adjusting variable ξ_i is zero.

The most used transformation kernels are of the form:

$$\begin{aligned} &\text{(i) polinomial: } k(x,x_i) = (x\cdot x_i)^d \\ &\text{(ii) gaussian: } k(x,x_i) = e^{-\gamma\|x-x_i\|^2} \\ &\text{(iii) sigmoid: } k(x,x_i) = \tanh(\kappa(x\cdot x_i) + \theta \end{aligned} \tag{15}$$

In this work we use the gaussian kernel (equation 15) which is the most used in practice.

So far we have discussed the binary separation problem, but for traffic classification we need a multiclass classification framework. To extend the exposed technique to the multiclass problem there are two common approaches. The first one is the classification referred as "*one-versus-all*", which proposes the construction of a M-class classifier by means of M binary classifiers trained to separe one class from the rest of the samples. After classifying a particular sample with the M binary classifiers, we choose the class for which the decision function of the classifier is maximum. The second approach is the classification called "*one-versus-one*", which need to construct binary classifiers trained with pairs of classes. In this case we need $M(M-1)/2$ classifiers, more than in the previous case, but each one require less samples to be trained. The final class for a particular sample is obtained by a voting process.

Particulary for the multiclass classification problem, the SVM implementation used in this work, *LIBSVM* [30], performs the "*one-versus-one*" method.

SVM classification process

In this work we use the flow representation stated in Section 2 and the following process recommended by the LibSVM authors [31]:

- Convert and scale flow data into LibSVM vector format. The scaling procedure is recommended by the LibSVM authors to avoid the prevalence of higher coordinate values over the smallest ones
- Construct train and test sets vectors

- Train and test adjusting parameters for better accuracy

The SVM classification process depends on parameters C, which represents the tradeoff between maximizing the separation margin and minimizing the training error (equation 14) and γ, included in the exponential of the gaussian kernel (equation 15). LibSVM authors recommend a grid-search varying C and γ as the best method to find the pair values that maximize classification accuracy.

As done in centroid classification, we analyze the influence of the number of segments considered per flow and the percentage of flows used for training. As can be seen in figures 5 and 6, 40% flows are enough for training. Moreover, considering only the data length of the first 3 or 4 segments of a flow gives the best accuracy when using SVM classification. Note that for better appreciation, the y-axis range has been changed for the following graphs.

This result is consistent with centroid based classification and previously cited publications. The accuracy is variable with the traffic class and those graphical results are given in Section 5.

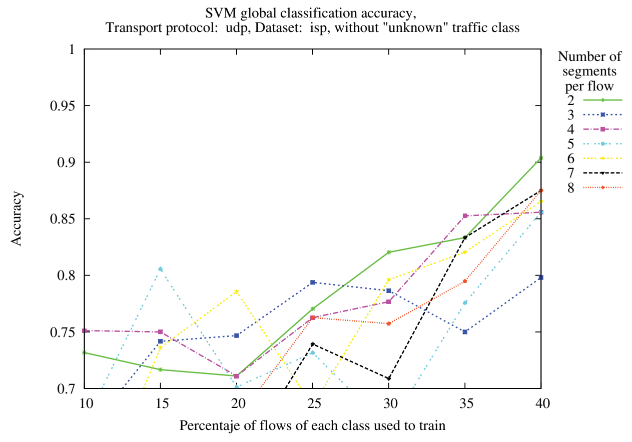

Figure 5: SVM accuracy (UDP, *ISP*)

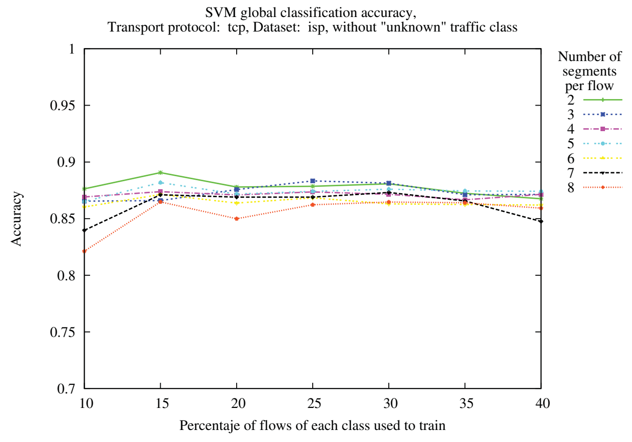

Figure 6: SVM accuracy (TCP, *ISP*)

5. CENTROID AND SVM CLASSIFICATION ACCURACY COMPARED

In previous sections we detailed the methodology and showed some results of both implemented methods for traffic classification.

Comparing the exposed methods accuracy, we can note that the global traffic classification accuracy using SVM is around 85% while the global traffic classification accuracy by centroid distances is around 70% for UDP traffic and 60% for TCP traffic. This result shows the potential of using SVM for traffic classification.

Analyzing in detail the classification accuracy per traffic type, the proposed classification approach based on SVM performs better for almost all traffic classes as shown in figures 7 and 8. The worst results of SVM classification, occur for poorly represented traffic classes, like *tsp* in UDP or *whois* in TCP. Other poor results are surely by cause of errors in ground truth assignment, like *ntp* in the TCP data, because this protocol is known to runs over UDP.

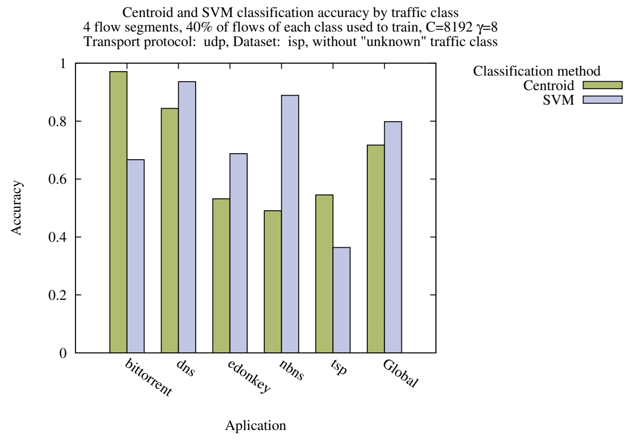

Figure 7: Centroid and SVM accuracy (UDP, *ISP*)

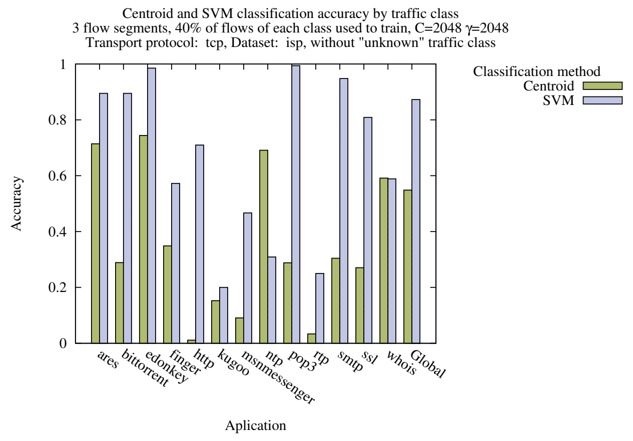

Figure 8: Centroid and SVM accuracy (TCP, *ISP*)

6. WEIGHTED VOTING OF SVM MODELS GENERATED BY BOOSTING

For some traffic classes we still obtain better classification accuracy using centroid classification, for example *bittorrent* in UDP traffic (figure 7). Thus, we propose the use of the *boosting* technique together with a weighted voting procedure to improve SVM classification accuracy for these traffic classes.

The *boosting* technique is based on the training algorithm *Hedge(β)* published in [13] and has been proposed to improve classification proceses in other fields [32, 33, 34]. In the networking area, particulary for intrusion detection, it has been also proposed [35].

Instead of taking random data to construct the training set in the learning phase, *boosting* proposes the generation of a sequence of SVM models trained with different datasets. The training dataset for each step of the sequence is generated by taking weighted data from the initial training set. Initially we take a random subset with uniform distribution from the initial training set and after testing the first model of the sequence, the weight of misclassified flows is increased. The training set for the next step model will be chosen based on these new weights. This procedure results in a biased training set on that "outliers" and therefore a SVM model that better fits to this samples. The AdaBoost algorithm [33] implements this approach and is described in Algorithm 1.

Algorithm 1 AdaBoost algorithm

```
Input: training set $T$ of $l$ labeled flows $x_i$
initialize weights $p_1(x_i) = 1/l$ (uniform distribution)
for $k = 1$ to $K$ do
    Construct $TR_k$ taking $m$ flows from $T$ based on $p_k(x_i)$
    Train SVM model $k$ using $TR_k$, varying $C$ and $\gamma$
    Construct $TE_k$ with $n$ flows from $T$ based on $p_k(x_i)$
    Test SVM model $k$ using $TE_k$, varying $C$ and $\gamma$
    Choose the $C$ and $\gamma$ which maximize the global accuracy
    Calculate error $e_k = \sum_{i=1}^{m} p_k(i) \mid i\ is\ misclassified$
    Calculate $\alpha_k = \frac{1}{2} \log \frac{e_k}{1 - e_k}$ (note: for $e_k < 0.5$, $\alpha_k < 0$)
    for $j = 1$ to $m$ do
        $p'_{k+1}(x_j) = p_k(x_j) \times \begin{cases} e^{\alpha_k} & if\ x_j\ is\ well\ classified \\ e^{-\alpha_k} & if\ x_j\ is\ misclassified \end{cases}$
    end for
    Normalize weights $p_{k+1}(x_j) = \frac{p'_{k+1}(x_j)}{N_k}$
    where $N_k$ is a normalization factor so that $\sum_{i=1}^{l} p_{k+1}(x_i) = 1$
end for
```

For this work, we generate arbitrarily 10 SVM models ($K = 10$) and as shown in figure 9 for TCP traffic, the accuracy per traffic class varies for each generated SVM model. The algorithm keeps the global (total) accuracy almost constant.

Our proposal is to perform a weighted voting procedure based on the verdicts of the sequence of SVM models generated by boosting, to assign the final classification class for a given flow. The proposed weighting factor is the precision of the verdict of each SVM model, that means the true positive percentage value.

Given a test flow, it must be tested against each of the SVM models generated by boosting. Each SVM model will assign a class to this flow. The verdict of each SVM model will have a precision that we can calculate in the testing phase of the model.

To clarify the method, we will illustrate the voting scheme with a real example. For TCP traffic taken from the *isp* dataset and considering 5 segments per flow and training with 15% of the flows for each class, we train 10 SVM models applying the described boosting algorithm. Each SVM model generated will get different precisions for each traffic class. We show in tables 2 and 3 the precision tables for two SVM models of the sequence generated for this particular case.

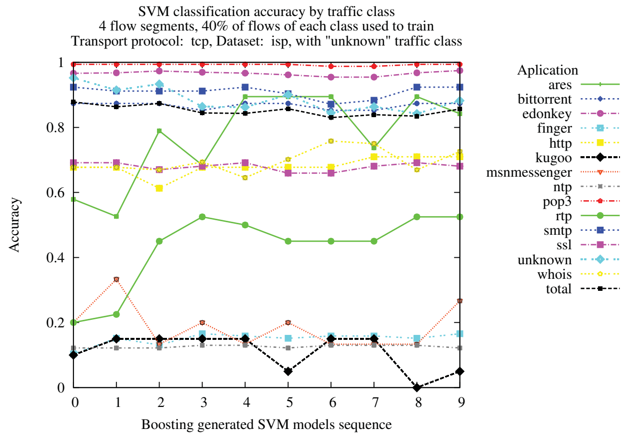

Figure 9: Accuracy with SVM boosting (ISP)

For a particular flow with **http** as base classification, each SVM model will give a verdict with an associated precision as shown in table 1. Thus, the final verdict will take into consideration that:

- the verdict of SVM models 0, 1, 2, 7, 8 and 9 is **http**, with a cumulative precision value of 592.467
- the verdict of SVM models 3 and 6 is **ssl**, with a cumulative precision value of 148.178
- the verdict of SVM models 4 and 5 is **edonkey**, with a cumulative precision value of 117.979

and finally, the proposed voting scheme will assign to this particular flow the **http** class, the one which obtains the higher cumulative precision.

	SVM model #									
	0	1	2	3	4	5	6	7	8	9
Verdict	http	http	http	ssl	edonkey	edonkey	ssl	http	http	http
Precision	99.051	97.836	97.562	82.949	59.280	58.699	65.229	98.369	99.968	99.681

Table 1: Verdicts and precisions for different SVM models for one particular *http* flow

Aplication	SVM model #0 verdict												
	whois	bittorrent	edonkey	finger	http	kugoo	ares	msnm...	ntp	pop3	rtp	smtp	ssl
whois	46.952	0.000	3.063	1.887	0.000	0.000	0.000	11.177	0.202	5.961	0.000	24.567	0.000
bittorrent	1.145	88.484	1.004	7.801	0.000	0.000	0.000	0.000	0.265	0.000	0.349	0.000	0.000
edonkey	0.128	0.188	65.054	0.734	0.033	0.669	0.000	0.000	0.393	0.000	0.026	0.000	0.059
finger	2.627	1.663	6.844	0.000	0.578	35.431	0.000	2.802	14.922	0.183	0.458	0.000	0.000
http	0.000	0.000	0.627	19.607	99.051	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
kugoo	1.374	3.043	5.780	34.685	0.000	25.932	0.000	2.929	21.589	0.000	1.675	0.000	0.000
ares	0.000	0.000	3.068	4.845	0.000	0.000	99.184	0.000	2.696	0.000	0.000	0.000	0.000
msnmessenger	43.743	0.000	1.278	6.056	0.000	0.000	0.000	81.627	1.685	0.000	0.000	0.000	0.000
ntp	0.440	1.218	5.398	11.106	0.338	31.828	0.816	0.000	55.909	0.000	1.073	0.000	2.130
pop3	0.000	0.000	0.000	0.168	0.000	0.000	0.000	0.000	0.000	90.674	0.000	1.282	0.000
rtp	0.000	3.803	5.780	8.215	0.000	4.322	0.000	1.465	1.270	0.000	77.042	0.000	6.652
smtp	3.591	0.000	0.076	0.289	0.000	0.000	0.000	0.000	0.000	3.182	0.000	73.907	0.000
ssl	0.000	1.600	2.027	4.608	0.000	1.818	0.000	0.000	1.069	0.000	19.378	0.244	91.159
Total	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Table 2: SVM model #0 precision for each traffic class. Dataset: *isp*, 5 segments per flow, trained with 15% flows of each class

As shown in figures 10 and 11, the decision taken by voting the judgement of the boosted SVM models increases the accuracy for almost all traffic types, compared to the original SVM classification. Also this proposed method outperforms

Aplicación	SVM model #3 verdict												
	whois	bittorrent	edonkey	finger	http	kugoo	ares	msnm...	ntp	pop3	rtp	smtp	ssl
whois	40.555	0.000	2.071	1.418	0.000	0.828	0.000	13.367	0.399	7.705	0.000	25.412	0.000
bittorrent	0.537	86.485	1.939	3.720	0.000	0.000	0.000	0.000	0.785	0.000	0.963	0.000	0.758
edonkey	0.000	0.154	62.994	0.525	0.033	0.726	0.000	0.048	0.446	0.000	0.071	0.000	0.028
finger	2.288	1.753	7.881	34.621	0.000	33.458	0.211	3.404	12.341	0.179	1.683	0.000	0.000
http	0.000	0.000	2.422	16.261	99.967	3.391	0.000	0.000	0.000	0.000	0.000	0.000	1.183
kugoo	2.577	7.128	6.512	25.874	0.000	31.261	1.545	3.114	18.817	0.000	3.080	0.000	1.817
ares	0.000	0.000	3.950	0.000	0.000	0.000	96.760	0.000	2.663	0.000	0.000	0.000	0.000
msnmessenger	51.285	0.000	0.000	2.368	0.000	0.000	0.000	78.510	1.665	0.000	0.000	0.000	0.000
ntp	0.000	1.141	4.915	8.713	0.000	22.521	1.484	0.000	57.438	0.000	1.973	0.000	1.455
pop3	0.079	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	88.692	0.000	1.351	0.000
rtp	1.932	2.139	4.186	3.123	0.000	7.815	0.000	1.557	4.391	0.000	70.846	0.000	11.811
smtp	0.204	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3.424	0.000	73.005	0.000
ssl	0.542	1.200	3.131	3.378	0.000	0.000	0.000	0.000	1.056	0.000	21.383	0.232	82.949
Total	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Table 3: SVM model #3 precision for each traffic class. Dataset: *isp*, 5 segments per flow, trained with 15% flows of each class

the centroid based classification in some cases where a single SVM model could not outperform it. Other datasets behave in a similar way, as shown in figures 12 and 13.

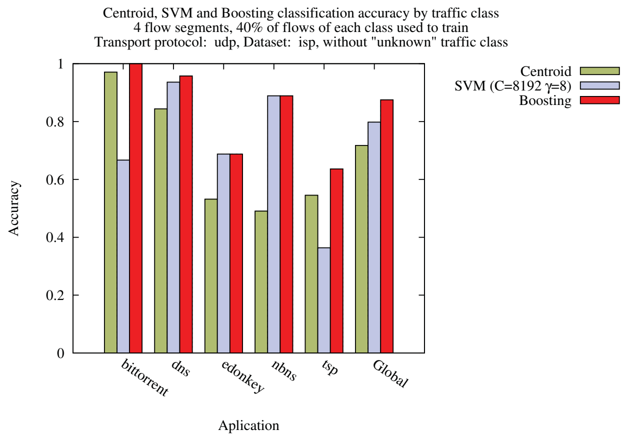

Figure 10: Centroid, SVM and boosting accuracy (UDP, *ISP*)

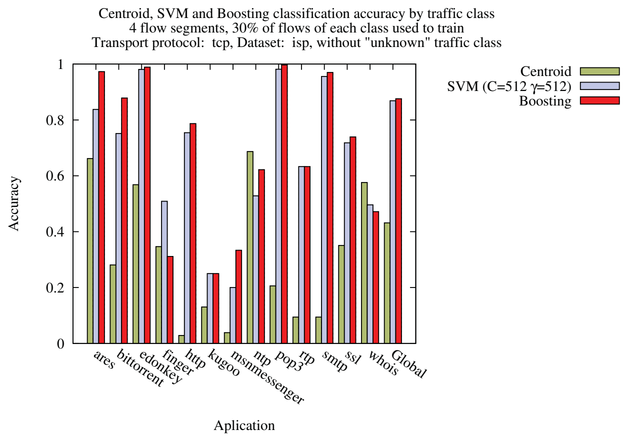

Figure 11: Centroid, SVM and boosting accuracy (TCP, *ISP*)

7. EFFECT OF "UNKNOWN" TRAFFIC

One prevalent point in the traffic classification area is how to consider the *unknown* traffic.

In the training phase of the machine learning methods described, we employ pre-classified traffic to identify some traffic classes. Therefore in the testing phase, each flow will be assigned to one of this pre-defined traffic classes.

When using this classification methods into an on-line identification system, each new flow will be labeled according to the already known classes. In a real scenario, some traffic may be not assignable to one of the known classes. This may be because we faced a new application not known previously or not included in the training dataset.

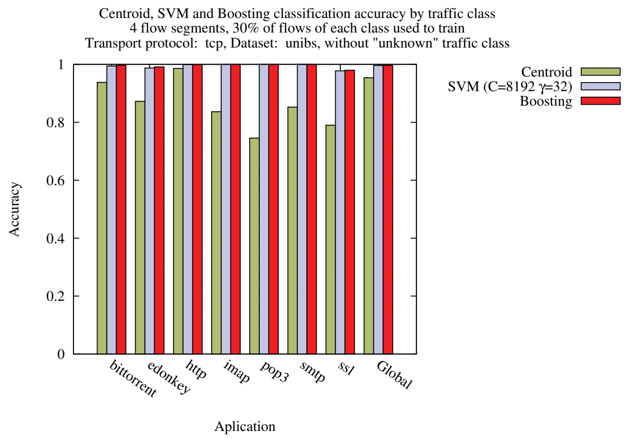

Figure 12: Centroid, SVM and boosting accuracy (TCP, *unibs*)

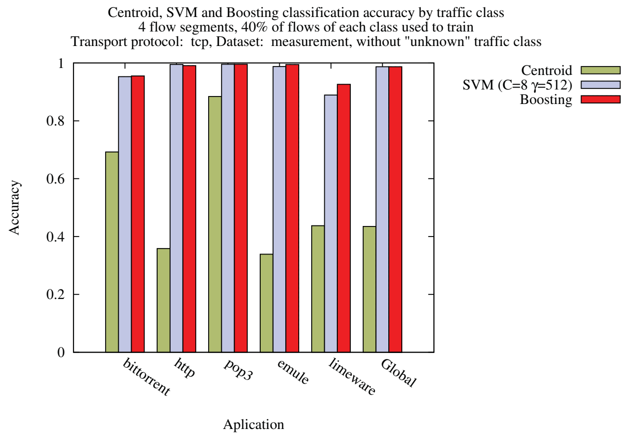

Figure 13: Centroid, SVM and boosting accuracy (TCP, *measurement*)

Some methods has been proposed to identify unknown traffic and proceed accordingly, typically comparing some kind of distance to the known classes to decide the presence of an unknown flow. A common approach when considering the segment sizes as a feature for traffic classification, is to train a fictitious traffic class with segment sizes distributed uniformly into the valid segment sizes range. For ethernet, this leads to the $[-1500, +1500]$ range [26, 36].

In this work we did not address this particulary point, but we analyzed the influence of including an unknown traffic class. Our implemented approach is to consider real traffic previously labeled as unknown to train a real *unknown traffic class*. Two of the datasets analyzed include unknown flows that could not be classified by the identification methods used. We use this traffic to train a new traffic class which represents the *unknown* traffic and which is much near to reality than using the uniform distributed segment sizes proposed in the literature.

When analyzing the accuracy variation including or not the *unknown* traffic class, we identify some classes which clearly are being confused with unknown traffic. The accuracy classification of most relevant traffic classes remains unaltered by the inclusion of this *unknown* traffic class. This may be attributed to some errors in the ground truth assignment procedure.

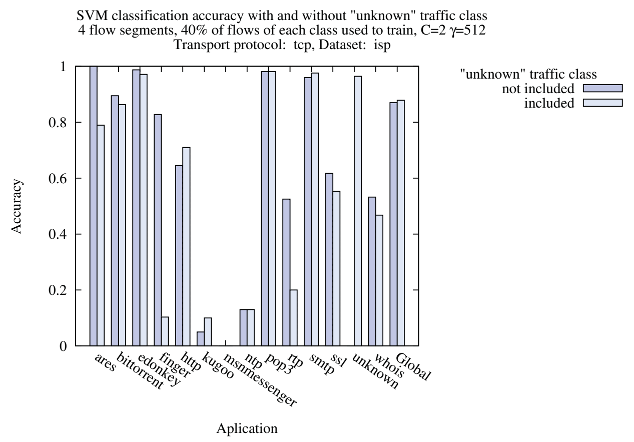

Figure 14: Impact of the unknown traffic class in SVM classification accuracy (TCP, *ISP*)

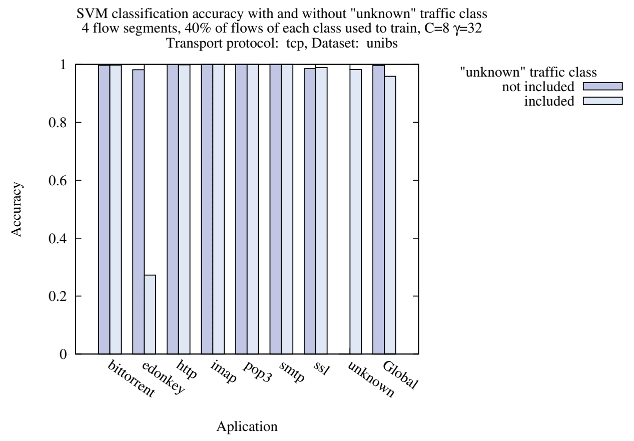

Figure 15: Impact of the unknown traffic class in SVM classification accuracy (TCP, *unibs*)

8. CONCLUSIONS

The results obtained in this work confirms that the data length of the first segments exchanged by a transport flow is a effective statistical feature for traffic classification. Moreover it was shown that only the data length of the first 3 o 4 exchanged segments is enough to obtain accurate results.

This result is valid not only for TCP traffic as reported previously in [14] and [26], but also for UDP traffic.

The proposed clustering method based on SVM classification yields more accurate results than the classical centroid based classification approach. Also the inclusion of the boosting technique together with a weighted voting scheme, improves the accuracy of poorly classified traffic classes without accuracy detriment of the other classes.

For future work we would like to analyze the computational cost of SVM classification, and the penalty added by the inclusion of the boosting technique. Also it will be interesting to analyze the classification accuracy considering only one way of the flow, something very common on a large ISP with multiples network connections. The unknown traffic analysis would be also an interest point to address in future works.

9. ACKNOWLEDGMENTS

We would like to thank ANTEL (Administración Nacional de Telecomunicaciones, Uruguay) for providing the *ISP* dataset and The Telecommunication Networks group @ UniBS, specially Luca Salgarelli, for providing the *unibs* called dataset.

10. REFERENCES

[1] H.-C. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *ACM CoNEXT 2008*. [Online]. Available: http://www.caida.org/publications/papers/2008/classification_demystified/

[2] A. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," in *Proceedings of the Passive y Active Measurement Workshop (PAM2005)*, March/Apri 2005, 2.

[3] S. Valenti, D. Rossi, M. Meo, M. Mellia, and P. Bermolen, "A behavioral classification framework for p2p-tv applications," TELECOM ParisTech (France), Politecnico di Torino (Italy), Tech. Rep. WP3.1, January 2009.

[4] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 313–326.

[5] 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*. New York, NY, USA: ACM, 2004, pp. 512–521.

[6] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, April 2006. [Online]. Available: http://dx.doi.org/10.1145/1129582.1129589

[7] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*. New York, NY, USA: ACM, 2006, pp. 1–12.

[8] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, 2007.

[9] N.-F. Huang, G.-Y. Jai, and H.-C. Chao, "Early identifying application traffic with application characteristics," May 2008, pp. 5788–5792.

[10] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques." in *PAM '04*, 2004, pp. 205–214.

[11] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 50–60, June 2005. [Online]. Available: http://dx.doi.org/10.1145/1071690.1064220

[12] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based*

Learning Methods. Cambridge University Press, March 2000. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0521780195

[13] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, August 1997. [Online]. Available: http://portal.acm.org/citation.cfm?id=261540.261549

[14] G. Gómez Sena and P. Belzarena, "Early traffic classification using support vector machines," in *Proceedings of the 5th International Latin American Networking Conference*, ser. LANC '09. New York, NY, USA: ACM, 2009, pp. 60–66. [Online]. Available: http://doi.acm.org/10.1145/1636682.1636693

[15] 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, pp. 12–18, October 2009. [Online]. Available: http://doi.acm.org/10.1145/1629607.1629610

[16] 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*, ser. TMA '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 64–74. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01645-5_8

[17] S. Lee, S. Lee, H. Kim, D. Barman, C.-K. Kim, T. T. Kwon, and Y. Choi, "NeTraMark: A network traffic classification benchMark," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, Jan. 2011. [Online]. Available: http://popeye.snu.ac.kr/\~{}sclee/NeTraMark/doc/sclee_ccr_2011.pdf

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[19] "Unibs: Data sharing, available traces with associated ground truth." [Online]. Available: http://www.ing.unibs.it/ntw/tools/traces/index.php

[20] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the validation of traffic classification algorithms," ser. PAM'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 72–81. [Online]. Available: http://portal.acm.org/citation.cfm?id=1791949.1791960

[21] "Application layer packet classifier for linux (l7-filter)." [Online]. Available: http://17-filter.sourceforge.net/

[22] "The perl programming language." [Online]. Available: http://www.perl.org/

[23] Z. Li, R. Yuan, and X. Guan, "Accurate classification of the internet traffic based on the svm method," in *ICC*, 2007, pp. 1373–1378.

[24] Y. xiang Yang, R. Wang, Y. Liu, S. zhen Li, and X. yong Zhou, "Solving p2p traffic identification problems via optimized support vector machines," in *AICCSA*, 2007, pp. 165–171.

[25] Y. Liu, H. Liu, H. Zhang, and X. Luan, "The internet traffic classification an online svm approach," in *Information Networking, 2008. ICOIN 2008. International Conference on*, 2008, pp. 1 –5.

[26] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for tcp traffic classification," *Computer Networks and Isdn Systems*, vol. 53, pp. 2476–2490, 2009.

[27] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995. [Online]. Available: http://portal.acm.org/citation.cfm?id=211359

[28] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[29] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. Berkeley and Los Angeles: University of California Press, 1951, pp. 481–492.

[30] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[31] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," Taipei, Tech. Rep., 2003. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[32] L. Cheng, J. Zhang, J. Yang, and J. Ma, "An improved adaboost algorithm based on adaptive weight adjusting," in *Proceedings of the 3rd international conference on Advanced Data Mining and Applications*, ser. ADMA '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 625–632. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73871-8_60

[33] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Y. Bang, "Constructing support vector machine ensemble," *Pattern Recognition*, vol. 36, no. 12, pp. 2757 – 2767, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-49202D3-3/2/fca47360665ecf4dd30f1b4b0f01151e

[34] X. Zhang and F. Ren, "Improving svm learning accuracy with adaboost," in *Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 03*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 221–225. [Online]. Available: http://portal.acm.org/citation.cfm?id=1473245.1473612

[35] X. Zan, J. Han, J. Zhang, Q. Zheng, and C. Han, "A boosting approach for intrusion detection," *Journal of Electronics (China)*, vol. 24, pp. 369–373, 2007, 10.1007/s11767-005-0201-z. [Online]. Available: http://dx.doi.org/10.1007/s11767-005-0201-z

[36] P. Bermolen, M. Mellia, M. Meo, D. Rossi, and S. Valenti, "Abacus: Accurate behavioral classification of p2p-tv traffic," *Computer Networks*, vol. 55, no. 6, pp. 1394 – 1411, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610003713