

KISS: Stochastic Packet Inspection for UDP Traffic Classification

Alessandro Finamore*, Marco Mellia*, Michela Meo*

* Politecnico di Torino
Email: lastname@tlc.polito.it

Dario Rossi†

†ENST Telecom Paris
Email: dario.rossi@enst.fr

Abstract—This paper proposes KISS, a novel Internet classification engines. Motivated by the expected raise of UDP traffic, which stems from the momentum of P2P streaming applications, we propose a novel payload-based classification framework which leverages on statistical characterization of payload.

Statistical signatures are automatically inferred from training data, by the means of a Chi-Square like test, which extracts the protocol “format”, but ignores the protocol semantic and synchronization rules. The signatures feed a decision engine based either on a simple geometric decision process, or on Support Vector Machines. KISS is very efficient, and its signatures are intrinsically robust to packet sampling, reordering, and flow asymmetry, so that it can be used on almost any network.

KISS is tested in different scenarios, considering both data, VoIP, and traditional P2P Internet applications. Results are astonishing. The average True Positive percentage is 99.6%, with the worst case equal 98.7%. Less than 0.05% of False Positives are raised. But KISS is also proved to provide almost perfect results when facing new P2P streaming applications, such as Joost, PPLive, SopCast and TVants.

I. INTRODUCTION

Last years witnessed a very fast-paced deployment of new Internet applications, ignited by the introduction of the successful Peer-to-Peer (P2P) paradigm and fueled by the growth of Internet access rates. This entailed not only a deep change of the Internet application landscape, but also undermined the reliability of the traditional Internet traffic classification mechanisms, typically based on Deep Packet Inspection (DPI) such as simple port-based classification.

As such, research on Internet traffic classification has gained significant attention, with a large number of proposals (see Sec. V for an overview) that try to circumvent DPI limitations. Indeed, DPI classification is deemed to fail more and more due to: i) proliferation of proprietary and evolving protocols, ii) embracement of strong encryption techniques and iii) adoption of tunneling techniques [1], [2]. In previous proposals, UDP has usually been neglected in favor of applications running over TCP. Motivated by the raise of UDP traffic volume, which stems from the momentum of VoIP, streaming and P2P-TV applications that deeply rely on UDP at the transport layer, we propose a novel classification framework that explicitly targets long-lived UDP traffic flows.

This work was funded by the European Commission under the 7th Framework Programme Strep Project “NAPA-WINE” (Network Aware Peer-to-Peer Application under Wise Network) and by a Cisco Collaborative Research Initiative (CCRI) named “Protocol Oblivious (Behavioral) Internet Traffic Classification”.

The mechanism we propose is based on the idea of automatically identifying the application protocol “format”, by means of a statistical packet inspection. This already proved successful in assisting the identification of particularly tricky traffic such as Skype [2]. In this paper, we push this intuition further, arguing that, due to the connectionless semantic of UDP, the very first bytes of the UDP payload are likely to carry some application layer protocol (L7-protocol), in which constant values, counters, random identifiers, etc., can be found. A preliminary version of this paper appeared in [3]. Recalling that a protocol specifies the the rules governing the *format*, *semantics*, and *synchronization* of a communication, we propose to extract the L7-protocol *format* while ignoring the actual semantic and synchronization rules. This is achieved by statistically characterizing the frequencies of observed values in the UDP payload, by performing a test similar to the Pearson’s χ^2 test. The χ^2 values are then used to compactly represent application fingerprints, which we call Chi-Square Signatures - ChiSS (pronounced as in KISS).

While KISS fingerprints stem from packet inspection, they have several advantages over classical DPI signatures:

- They can be automatically derived, i.e., no cumbersome and tedious reverse engineering is required;
- they can be quickly updated, so that they are well apt to the context of fast-evolving Internet applications;
- they are easily portable across different network settings, since fingerprints depend solely on the L7-protocol format;
- they are robust to routing asymmetry, packet loss, retransmission, or any possible strange packet arrival pattern, since they build over a statistical characterization of protocol format rather than on a deterministic description; this includes packet sampling at the probe point;
- they are suitable to both per-flow and per-endpoint classification;
- their computational and memory requirements are very limited, so that they are suitable for on-line classification.

After that fingerprints have been extracted, proper classification must be achieved, i.e., individual items should be placed into the most likely class. A huge set of methodologies are available from the literature, that span from simple threshold based heuristics [4], to Naive Bayesian classifiers [2], [5], to advanced statistical classification techniques [6]. In this

paper, we compare simple geometric decision process based on Euclidean distance with Support Vector Machines (SVMs) [6], which are well known in the statistical classification field, but have been rarely exploited in the context of Internet traffic classification.

To prove the advantages of proposed framework, we implemented KISS in Tstat [11], which we then use to derive the results presented in this paper. We test KISS using both testbed traces, and real traffic traces, collected from an operative ISP network. To cross check the classification performance, we rely on a traditional payload-based DPI engine coupled with manual inspection, which we then use as ground truth in the classification. We test the performance of KISS in classifying both traditional applications (lik DNS and RTP traffic), affirmed P2P protocols (like eMule, Bittorrent and Skype) and emerging P2P-TV applications (like PPLive, SopCast, Joost, TVants). Our results show that KISS exhibits excellent performance, typically achieving more than **controllare cifre** 99% of true positives, and no more than 1% of false positives when SVM are adopted. These astonishing results are due to both the accurate characterization of the KISS signatures, and the precise classification of the SVMs.

II. GENERAL FRAMEWORK

Before entering into the details of KISS, we briefly summarize the two main processes that define any classifiers. We refer the reader to [?] for a more extended description:

- Feature extraction: this is the process of extracting the subset of information that summarize a large set of data, i.e., to describe samples.
- Decision process: this is the algorithm that assigns a suitable class to the sample.

Considering the decision process, any machine learning techniques can be adopted; in this paper we focus on supervised learning algorithms, in which a training set is made available to allow the classifier to first build a model, which is then used later on during the classification task. Given a geometric representation of features in a multidimensional space, during the training phase, labeled samples are used to identify and to define the “area” in which samples of the considered class falls into. During the classification process instead, the sample to be classified has to be labeled with the most likely class according to the area it falls into. For example, assuming that there are two classes of objects, i.e., red and yellow apples, if the observed sample features place it in an area dense of red apples, we are inclined to classify it as a red apple too. Defining the surface that delimits the areas (to later take the decision) is however tricky, since training points can be spread out on the multidimensional space, so that complex surfaces must be described. In this paper, we consider both simple geometric decision process, and SVM based algorithms, which are considered to be one of the most powerful supervised learning algorithms.

Once a classifier has been designed, its performance must be evaluated and proper metrics must be defined. Considering Internet traffic classifiers, assessing the performance of any

TABLE I
DEFINITION OF FALSE/TRUE POSITIVE AND FALSE/TRUE NEGATIVE

		Oracle Classification	
		True	False
Classification Result	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

classification engine is not a trivial task due to the difficulty to know the “ground truth”, i.e., what was the actual application that generated the traffic [12]. To solve this issue, an “oracle” is often invoked. Testing the classification engine by means of artificial traffic (e.g., by generating traffic in a testbed) solves the problem of knowing the ground truth (i.e., you are the oracle), but synthetic traces are hardly representative of real world traffic. Assessing the performance against traffic traces collected from operative networks is therefore mandatory. The major problem when dealing with real traffic traces is however finding a suitable oracle.

In the following, we report results showing the False/True Positive percentages, and the False/True Negative percentages. A test is said “True” if the classification results and the oracle are in agreement. A test is said “False” on the contrary. The result of a test is “Positive” if the classifier accepts the sample as belonging to the specific class. On the contrary a test is “Negative”. For example, consider a flow. The oracle states that this flow is an eMule flow. If the flow is classified as an eMule flow, then we have a True Positive. If not, then we have a False Negative. Consider instead a flow which is not an eMule flow according to the oracle. If the flow is classified as an eMule flow, then we have a False Positive. If not, then we have a True Negative. Table I summarizes the definitions.

The corresponding percentages must be evaluated as

- False Positive Percentage (%FP) is the percentage of *negative* samples that were erroneously reported as being positive:

$$\%FP = 100 \frac{\text{Number of False Positives}}{\text{Total Number of Negative Samples}};$$

- False Negative percentage (%FN) the proportion of *positive* samples that were erroneously reported as negative:

$$\%FN = 100 \frac{\text{Number of False Negatives}}{\text{Total Number of Positive Samples}};$$

- True Positive Percentage (%TP) is 100-%FN;
- True Negative Percentage (%TN) is 100-%FP;

Indeed, if there are 100 eMule flows, and the classifier misses 10 of them, we have %FN=10% (%TP=90%). Similarly, if there are 500 NON eMule flows, and the classifier returns all of them as eMule, we have %FP=100% (%TN=0%).

Finally, results are often expressed by means of a *Confusion Matrix*. In the field of artificial intelligence, a confusion matrix is a visualization tool typically used in supervised learning. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

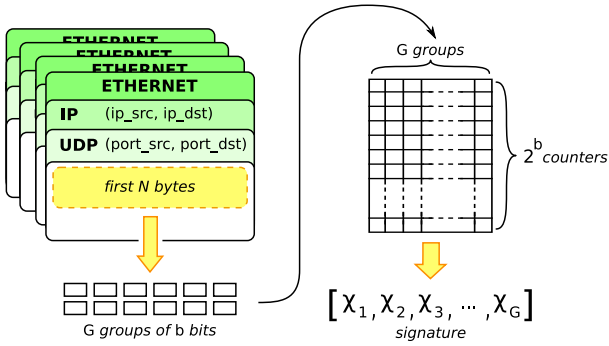


Fig. 1. Scheme of signature extraction process

A. Chi-Square Feature Extraction

A traditional DPI classifier inspects packet payload looking for *deterministic patterns*, such as particular strings which are compared to those in the signature database. The process of defining the signatures is a complex task that requires a deep knowledge of the protocols that need to be identified. As such, any changes in a protocol can invalidate the signature, which becomes outdated and must manually redefined.

The goal of the framework proposed in this paper is instead to *automatically discover application layer header format*, without caring about specific values of the header fields: we aim at automatically let the protocol format emerge. Since UDP is a connectionless protocol, the first bytes of the payload of each UDP packets typically contain application layer protocol header. In addition, the format of fields in protocol headers can be made of constant identifiers, counters, words from a small dictionary (message/protocol type, flags, etc), or truly random values coming from encryption or compression algorithms. These coarse classes of fields can be easily distinguished through a simple statistical characterization of the observed values seen in a sequence of packets. The process of the format extraction is achieved by using a simple Chi-Square statistical test.

The original Chi-Square statistical test estimates the goodness-of-fit between observed samples of a random variable and a given theoretical distribution. Assume that the possible outcomes of an experiment are K different values; and O_k are the empirical frequencies of the observed for values, out of M total observations ($\sum O_k = M$). Let E_k be the number of expected observations of k for the theoretical distribution, $E_k = M \cdot p_k$ with p_k the probability of value k . Given that M is large, the distribution of the random variable

$$X = \sum_{k=1}^K \frac{(O_k - E_k)^2}{E_k} \quad (1)$$

that represents the distance between the observed empirical and theoretical distributions, can be approximated by a Chi-Square, or χ^2 , distribution with $K - 1$ degrees of freedom. In the classical goodness of fit test, the values of X are compared with the typical values of a Chi-Square distributed random variable: the frequent occurrence of low probability values is

interpreted as an indication of a bad fitting. In KISS, we build a similar experiment analyzing the content of groups of bits taken from the packet payload we want to classify.

Chi-Square signatures are built from *streams* of packets directed to or originated from the same end-point. The first N bytes of the packets payload are divided into G groups of b consecutive bits each; a group g can take integer values in $[0, 2^b - 1]$. From packets of the same stream, we collect, for each group g , the number of observations of each value $i \in [0, 2^b - 1]$; denote it by $O_i^{(g)}$. We then define a window of C packets, in which we compute

$$\chi_g = \sum_{i=0}^{2^b-1} \frac{(O_i^{(g)} - E_i^{(g)})^2}{E_i^{(g)}} \quad (2)$$

and collect them in the vector

$$\bar{\chi} = [\chi_1, \chi_2, \dots, \chi_G] \quad (3)$$

which is the KISS signature. Fig. 1 shows a schematic representation of the KISS signature extraction.

One possibility to characterize a given protocol is to estimate the expected distribution $E_i^{(g)}$ for each group g , so that the set of signatures could be created by describing the expected distribution of protocols of interest. During the classification process then, the observed distribution $O_i^{(g)}$ should be compared to each of the $E_i^{(g)}$ in the database, so that the Chi-square test is used to select the most likely one. However, this process would end-up in a complex process, in which Eq.(eq:chi-teorico) must be computed for each protocol.

On the contrary, we propose to simply check the distance between the observed values and a reference distribution E_i , which we choose as a uniform distribution bits, i.e., $E_i^{(g)} = \frac{C}{2^b} = E$. In other terms, we use a Chi-Square like test to measure the randomness of groups of bits or as an implicit estimate of the source entropy.

To give the intuition of how Eq.(2) evolves versus C , consider the case in which a deterministic group of bits is observed. The value of χ_g^2 is

$$\begin{aligned} \chi_g^2 &= \sum_{i=0}^{2^b-1} \frac{(O_i^{(g)} - E_i)^2}{E_i} \quad (4) \\ &= \frac{(C - E)^2 + (2^b - 1) E^2}{E} = C(2^b - 1) \quad (5) \end{aligned}$$

Indeed, for a deterministic block, only one value for the block is possible, and χ_g^2 linearly increases with C .

In general, for a block in which b_0 bits are constant, it can be shown that

$$\chi_g^2 = C(2^{b_0} - 1) + 2\chi_{2^{b-b_0-1-1}}^2 \quad (6)$$

where $\chi_{2^{b-b_0-1-1}}^2$ is the Chi-Square with 2^{b-b_0-1-1} degrees of freedom.

To provide an example of the evolution of χ_g , left plot in Fig. 2 reports the value of two 4-bit long groups belonging to two different traffic protocols, namely DNS and eMule,

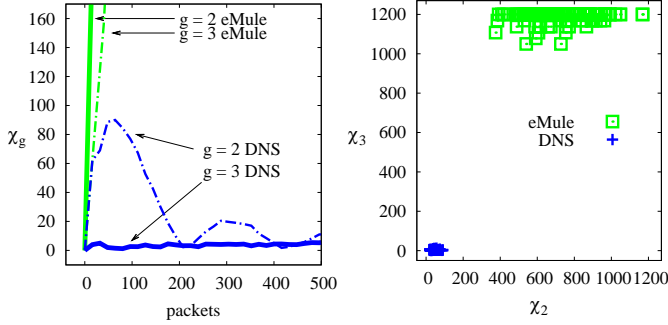


Fig. 2. Evolution in time (left) and dispersions in space (right) of χ^2 of two groups extracted from the second byte of UDP payloads.

versus the number of collected samples C . The steep lines corresponding to groups taken from an eMule stream refer to fields that are almost constant. In this case, the longer the experiment is (larger C), the larger the distance from the uniform distribution is, i.e., the bits are far from being uniformly distributed. In the same plot, observe the lines referring to DNS traffic. The lowest one has a very slow increase with C , its behavior is almost perfectly random, the values of χ_3 being compatible with those of a Chi-Square distribution. The bouncing line, instead, corresponds to the typical behavior of a counter. The computation (2) over consecutive bits of a counter cyclically varies from very low values (when all the values have been seen the same number of times) to large values. The periodicity of this behavior depends on the group position inside the counter.

While randomness provides a coarse classification over individual groups, by jointly considering a set of G groups through the vector $\bar{\chi}$ the fingerprint becomes extremely accurate. Observe right plot in Fig. 2. In this case, we consider $C = 80$ packets of a stream. Points in the figure are plotted using (χ_2, χ_3) as coordinates; each point corresponds to a different stream. Points obtained from DNS streams are displaced in the low left corner of the plot; points from eMule are spread in the top part of the plot. Intuitively, different protocols fall in different areas that are clearly identified and easily separable: a simple straight horizontal line can effectively separate the two regions considering this simple example.

1) *Parameter setting*: The signature creation approach previously presented is based on a number of parameters whose setting may be critical. These are the criteria we used to set them:

Bits per group, $b = 4$. The choice of b should trade-off two opposite needs. From the one hand, we would like b to be the closest as possible to typical length of protocol fields; since protocols dialogs are usually based on words whose length is multiple of the byte or, sometimes, is half of a byte, b should be 4 or 8 or a multiple of 8. From the other hand, b should be small enough to allow that the packet window C over which the Chi-Square test is statistically significant is not too large, so that streams can be classified even if they are not

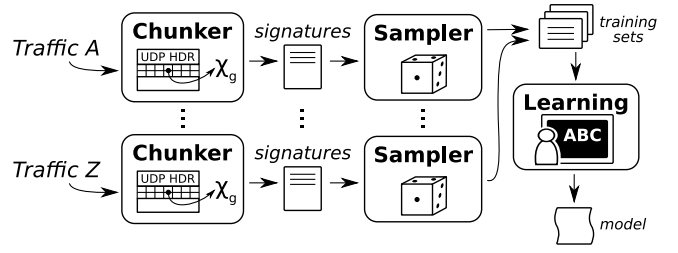


Fig. 3. Schematic representation of KISS learning steps.

too long, they are classified in short time and live classification is possible. Thus, we chose $b = 4$.

Number of bytes per packet, $N = 12$. In general, classification accuracy increases with the number of considered bytes per packet. However, complexity of the classification tool increases also with the N , in terms of both memory and computational complexity. As a convenient trade-off we choose $N = 12$. Given $b = 4$ this values corresponds to $G = 24$ groups. Another reason to choose $N = 12$ bytes is that, this way, we collect 20 bytes of the IP packet payload (12 bytes + 8 bytes of the UDP header) that is the minimum size of the TCP header and the typical value used by measurement tools. Notice that the optimal value of N depends from the targeted applications. For example, DNS and eMule can be clearly identified by only considering (χ_2, χ_3) (see Fig. 2). However, when considering different protocols, possibly more and different groups must be considered. The selection of which is the best set of groups to include in $\bar{\chi}$ is then a complex task that is left out as future work.

Packet window, $C = 80$. While we would like to keep the packet window as small as possible, the estimation of the observed distribution is considered to be statistically significant if the number of samples for each value is at least 5. Having chosen $b = 4$, in order to have $E_i = C/2^b$ equal to 5, we need C to be equal to about 80. A sensitivity of KISS accuracy versus C is performed in Sec. IV.

B. KISS Decision Process

KISS is based on supervised machine learning decision process. During the training phase, we operate as sketched in Fig. 3. We start by considering some streams that belong to the set of applications we want to model. Streams are then fed into a *chunker*, whose role is to derive the KISS signatures as in (3). The signature set is randomly sampled by the *sampler*, so as to select the *training set*, whose size will be discussed in Sec. IV. The training set is then fed to the learning system after which the KISS model is produced. In this paper, we investigate two different leaning systems, the first based on euclidean distance, and the second based on Support Vector Machines (SVM).

1) *Euclidean Decision Process*: In the case a simple Euclidean model is adopted, the KISS model produced during the training phase corresponds to a set of hyper-spheres, one for each protocol, that defines the areas in which samples of each class are expected to fall. The classification process is then

straightforward: a point that falls inside a sphere is classified according to the protocol associated to that sphere, while if doesn't belong to any sphere it's assumed to be of an unknown protocol.

For a given class A , the representative hyper-sphere is fully defined by its center $\hat{\chi}(A)$ and its radius $\rho(A)$. $\hat{\chi}(A)$ is simply computed using the arithmetic mean of each class A sample in the training set, while the identification of the radius is more complex. Indeed, the hyper-sphere should be big enough to include all the points of the training, but it has to be small enough to avoid to include samples of other classes. Using machine learning terminology, one wants to maximize the *True Positive* ratio while minimizing the *False Positive* ratio.

Formally, the following equation can be used to state the problem:

$$\rho(A) = \arg \max_{\rho} \left(TP(\rho) - FP(\rho) \right) \quad (7)$$

Notice that $TP(\rho)$ is computed considering sample of class A , while $FP(\rho)$ is computed considering samples of all other classes of the training set.

2) *SVM Decision Process*: SVM are a set of supervised learning methods used for classification and regression [6]. The key idea of SVM is to displace the training samples (by means of a transformation from the original N-dimensional space to a possibly infinite-dimensional space) so that samples belonging to different classes can be separated by the simplest surface, i.e., an hyper-plane. This makes SVMs very powerful:

- They are robust to the training set size and composition;
- Their computational and memory requirements are very limited during the classification phase, even if the training phase can be computationally expensive;
- They exhibit a very high discriminating power, so that they typically achieve very high classification accuracy;
- There is a large number of efficient algorithms and implementations already available. In particular, in this paper we adopted the LIBSVM library [9] implementation.

Finally, notice that the output of the training phase is a definition of a number of regions equal to the number of classes defined during the training phase, e.g., one for each protocol that is offered during the training phase. This implies that a sample will then always be classified as belonging to any of the known classes. Considering traffic classification, an additional region is needed to classify all samples that do not belong to any of the given protocols, i.e., to represent the "other" protocols. Thus, the training set must contains two types of signatures: i) the ones referring to traffic generated by the applications to classify; ii) the ones representing all the remaining traffic, which we refer to as *Background*. It represents the set of applications that we cannot classify or are not interested in classifying.

III. TESTING METHODOLOGY

In this paper, we aim at assessing KISS performance in the most difficult scenario, whenever possible. We consider real traffic traces, collected from an operative, totally uncontrolled

network. We had to develop an ad-hoc oracle, that is based on DPI mechanisms, and to manually tune it and to double check its performance. To prove the KISS flexibility, we test its capability in detecting P2P-TV traffic, namely, PPLive, Joost, SopCast and TVants which deeply rely on UDP at the transport layer. In this second case, we use both real traffic traces, and, since real traffic traces contain no P2P-TV traffic, large testbed traces.

A. Testing Datasets

Real Traffic Traces: Real traffic traces (RealTrace) were collected from the network of FastWeb [13], an ISP provider that is the main broadband telecommunication company in Italy. The FastWeb network offers telecommunication services to more than 5 millions of users. Based on a full IP architecture, FastWeb offers converged services, in which data, native VoIP [14], and IPTV services share a single broadband connection. The FastWeb network is a very heterogeneous scenario, in which users are free to use the network without any restrictions. It therefore represents a very demanding scenario considering traffic classification. A probe node based on high-end PC running Linux has been installed in a PoP located in Turin, in which more than 500 users are connected, using more than 2000 different IP addresses (e.g., VoIP phones, set-top-boxes, PCs, etc.). All packets entering/leaving the PoP have been captured. The measurements presented in this paper refer to two dataset that we will call RealTrace-I and RealTrace-II¹.

Both traces contains many popular applications generating UDP traffic, in particular we selected: i) eMule, ii) VoIP (over RTP), and iii) DNS protocols. Indeed, these three protocols account for more than 80% of UDP endpoints, corresponding to 95% of the flows, and to more than 96% of the total UDP bitrate.

Napawine Traces: Since we are also interested in evaluating the performance of KISS when dealing with new protocols, we selected, as case study, P2P-TV applications. Indeed P2P-TV systems have been recently introduced and they are starting to became popular. In addition, they rely on proprietary design and protocols, they preferentially use UDP as transport protocol, and they are expected to offer a large amount of traffic to the network. Among the available P2P-TV applications, we selected PPLive, Joost, SopCast and TVants. However, none of the selected applications was available at the time of real traffic trace collection. Therefore, we are forced to rely on Testbed P2P-TV Traces (P2Ptrace) to assess the performance of KISS. We used as P2Ptrace dataset the set of traces collected in the context of Napa-Wine [15] project, in which a large scale experiment was organized to observe the performance of the above mentioned P2P-TV applications. The resulting dataset consists of packet level traces collected from more than 45 PCs running P2P-TV applications in 5 different Countries, at 11 different institutions. The dataset includes traces collected

¹Due to a NDA, we are not allowed to show results referring to more recent traces. Nonetheless, we can affirm that this trace is representative of typical KISS performance.

TABLE II
DESCRIPTION OF THE DATA TRACES CONSIDERED

	Bytes	Packets	Flows	Endpnts	Time period
RealTrace I	53.13G	321M	18.25M	1.72M	22h, May '06
RealTrace II	31.33G	133M	5.25M	1.02M	12h, Jun '07
Napawine	10.25M	14M	132K	48.5K	3h, Apr '08
Skype	3.7G	24.7M	966	559	96h, May '06

from PCs in Campus LANs, Corporate networks with restrictive policies, home ADSL connections, so that both nodes with public and private IP addressing are present. We are therefore confident that the heterogeneity of the P2Ptrace dataset is representative of a wide range of different scenarios.

Skype Traces: In the test, we also use the public available dataset for the Skype traffic [10]. The dataset contains both Skype traffic identified in [2] and traces collected in a controlled environment using two PCs (a sender and a receiver) running different versions of Skype and different operating systems such as Windows, Linux and Pocket-PC.

B. Classification Scenario

We consider the scenario in which a network provider or administrator is interested in knowing the traffic that is going to or coming from a set of internal hosts. We consider

- a *monidirectional flow*: all packets coming from the same source IP address and UDP port and going to the same destination IP address and UDP port;
- an *endpoint*: all packets having the same IP destination (source) address and UDP destination (source) port.

Depending on the application, one can be interested in identifying a single flow (as in the case of a VoIP stream), or in detecting the endpoint and therefore all packets sent/received from it (as in the case of a P2P application).

Tab. II summarized the previously described datasets. In particular, it reports the total amount of bytes, packets, flows and endpoints for each dataset, and the collection time and duration of each trace. In the following, we briefly detail each dataset.

As discussed in Sec. II, the packet windows size C plays an important role in the KISS design, and it may affect the applicability of KISS. Indeed, given the connectionless characteristic of UDP, one expects that UDP flows and endpoints last for few packets. Left plot of Fig. 4 confirms this intuition. It reports the Cumulative Distribution Function (CDF) of the probability that a flow and endpoint lasts for less than C packets. All incoming UDP traffic in RealTrace is considered to derive the CDF. The plot clearly show that 40% of flows and endpoints has only 1 packet, while only 0.2% of flows and 5% of endpoints have at least 80 packets. However, these flows/endpoints respectively account for more than 93.8% and 98.6% of the *bytes* carried by UDP, as shown by the right plot of Fig. 4, which reports the fraction of bytes carried by flows/endpoints with less than C packets. This clearly shows that, while KISS is not suitable for the classification of short lived UDP flows/endpoints, it can however successfully target

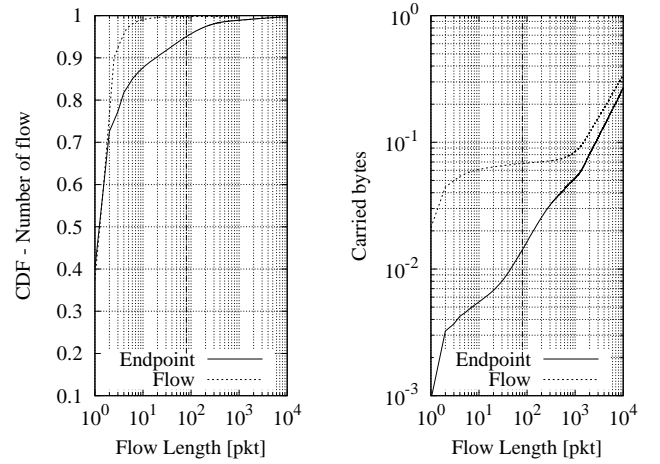


Fig. 4. CDF of the probability that a flow/endpoints has less than C packets on the left, and fraction of the total traffic amount flows/endpoints with less than C packet carry on the right. The vertical line is in correspondence of 80 packets

the small fraction of them that generate the majority of the traffic, i.e., long-lived flows.

We assume packets belonging to the same flow/endpoint are exposed to the KISS engine, so that after digesting C packets, a classification decision is taken, and a new observation window begins. Therefore several classification decisions will be eventually taken considering a single flow or endpoint. Notice that i) no assumption about observing the first set of packets is stated; ii) there is no need to observe bidirectional streams of packet; and iii) not all packets belonging to the same flow/endpoint must be exposed to the classifier; possible packet drop, reordering, sampling can be present.

Due to space limitations, we report results from the RealTrace I and II datasets considering all flows/endpoints whose destination IP address belongs to any of the IP addresses used by hosts inside the FastWeb PoP (i.e., we are looking at identifying the traffic entering the PoP). Considering the P2Ptrace and Skype dataset, we consider all endpoints whose source IP address belongs to any of the PCs taking part to the experiments (i.e., in this case we classify the traffic sent from a P2P-TV application). This results are representative of the performance of KISS when considering other possibilities.

C. The Oracle Definition

To obtain the ground truth, we developed a DPI classifier that was explicitly designed. It was implemented in Tstat [11], and its performance were manually fine tuned and double checked. In particular, DPI rules can be summarized as follows:

- **DNS: we rely on simple port classification, since UDP port 53 was only used by the DNS system during 2006.**
- **RTP: we rely on the state machine described in [14] to identify RTP flows. It combines a DPI signature and correlates the value of the fields in consecutive packets (e.g., to check the validity of the counters).**

- eMule: a DPI classifier based on [16], [17] has been developed and adapted to the considered scenario². We have also developed an heuristic based on the analysis of packets size that identify obfuscated communications so we have a set of rules that can identify all the possibles eMule application and dialects.

Since the oracle itself can be unreliable, manual inspection and pinpointing of suspect cases are detailed in the performance results.

IV. RESULTS

A. Real Traffic Traces

We first report results considering a small subset of the RealTrace-I dataset, corresponding to the first 50GBytes of data. The oracle is used to split the trace into 4 sub traces: each sub trace includes only packets classified as the same protocol, i.e., RTP, eMule, DNS and Background traffic only. Each trace is fed to the KISS classifier, so that signatures are evaluated. Both SVM and Euclidean decision processes are trained using 300 signatures for each class, and the remaining signatures are used to assess the performance of KISS. Recall that a signature is generated every C samples, so that a flow/endpoint can be classified several times (i.e., every C packets).

Tab. III summarizes the results. Each row corresponds to a sub trace that was classified according to the oracle. The second column reports the total number of signatures extracted from each trace while the remaining report the percentage number of true positive and false negative for the two models.

The SVM results are astonishing: the True Positives are always higher than 99%. The performance of the euclidean classifier are more variable, e.g., it performs very well for RTP but the accuracy decreases when considering eMule and DNS protocols. This is related to the adoption of an hyper-sphere as an approximation of the separation surface between classes. To this extent, Fig. 5 reports Eq.7 an examples of optimization for RTP, eMule and DNS. For RTP, any choice of $\rho(RTP) \in [12.2, 28.0]$ allows to almost perfectly identify RTP traffic. On the contrary, eMule class is not well represented by the hyper-sphere surface, so that any choice of $\rho(eMule)$ trades between TP% and FP%. Similar reasoning apply for DNS traffic. This shows that a simple decision process based on Euclidean distance is hard to design, while the adoption of SVM allows to avoid this problem. This conclusion is supported by other tests we performed, not reported here due to lack of space. Therefore, in the following we will consider only the SVM classifier and we will investigate how KISS performance are affected by parameter setting and considered scenario.

B. Signature Robustness

We are interested in quantifying the KISS robustness to a training set which is independent from the test set. We thus

²The eMule client used by FastWeb users has been optimized to exploit FastWeb network architecture. This entailed a modification to the KAD protocol, called KADu. Off-the-shelf DPI signatures have been then adapted to cope with the modified protocol.

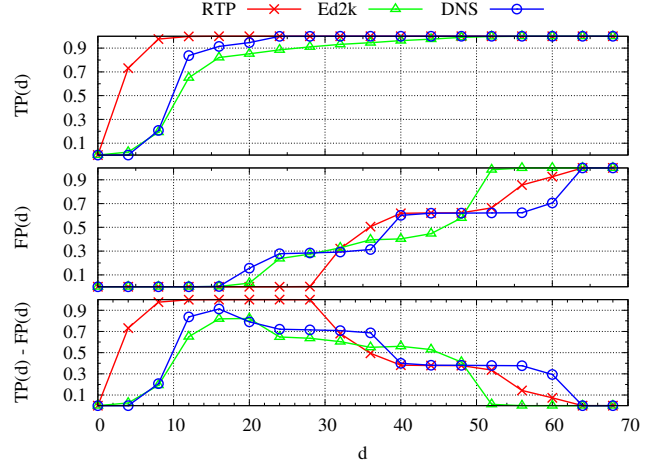


Fig. 5. Euclidean Decision Process: True Positive - False Positive evolution of versus ρ for RTP, eMule and DNS classes.

TABLE III
EUCLIDE AND SVM PERFORMANCE ON REALTRAFFIC TRACES

	Tot.	Euclide		SVM	
		tp%	fn%	tp%	fn%
RTP	8386	99.9	0.1	99.9	0.1
eMule	1527	84.3	15.7	99.3	0.7
DNS	8245	91.3	8.7	99.9	0.1
Backg	2579	99.1	0.9	99.6	0.4

perform an experiment in which the SVM is trained using samples extracted from the initial part of the RealTrace I. A 9 hour long subset of the RealTrace I is considered, and the training set includes 300 samples randomly extracted from the first 30 minutes only. Results are reported in Fig. 6. Only False Positive percentages are reported, since the %TP is always higher than 99%. The plot confirms the intuition that the characterization of the Background traffic may be a problem, since there are peaks that clearly show that the SVM is fooled by the sudden appearance of unknown protocols that were not described by the training set.

Investigating further, we indeed noticed that the high percentage of Background traffic classified as RTP traffic is due a single endpoint which is receiving traffic with the same “format” of RTP protocol. However, the DPI based oracle did not classify this endpoint as RTP, since a mismatch in the RTP version field is present: it takes a values of 1 instead of 2. Apart from this difference, all other protocol fields are in perfect agreement with the RTP standard. Moreover, all packets received by this endpoint are 200B long, which is typical of VoIP streams using the ITU-T G.711 encoder [14]. We then claim that this is an *actual* RTP flow, but the DPI oracle was fooled by the wrong version value! On the contrary, KISS correctly classified this flow as a RTP flow.

Similarly, investigating the samples that are misclassified as DNS (e.g., from 15:30 to 16:00) we notice that a single endpoint (listening to port number 9940) is the only responsible. We manually inspected this traffic, and verified that

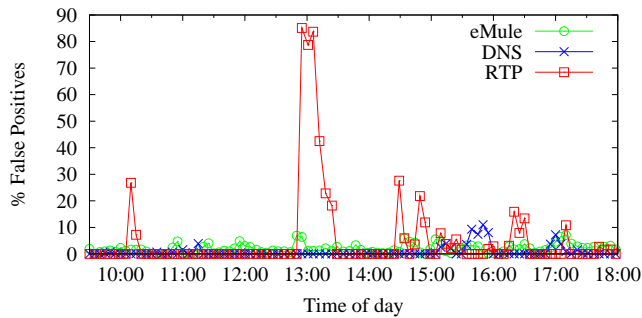


Fig. 6. False Positive percentage variation versus time. Background in the training set.

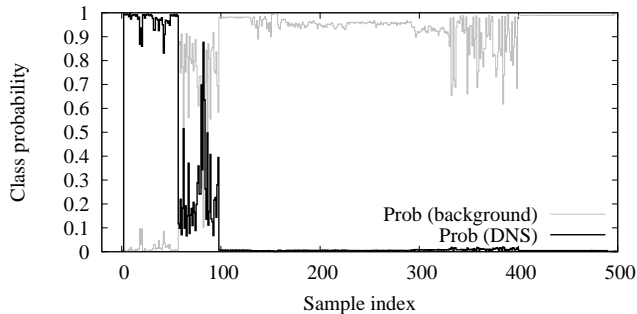


Fig. 7. Example of an endpoint that causes false positives. Different classification windows over time.

it cannot be a DNS endpoint, so that the oracle is reliable. Interestingly, no samples of this endpoint is included in the training set of Background traffic. Since the SVM is always forced to classify the sample as one of the four possible classes, it sometimes resolves to classify it as DNS rather than Background. Considering this endpoint, Fig. 7 shows the probability the SVM evaluates that it is a Background or DNS sample versus time. It can be seen that some uncertainty is present. Repeating the experiment by including this sample signatures in the Background training set, KISS correctly classifies it.

Similar conclusions can be drawn investigating the eMule False Positives. We noticed that they all correspond to endpoints listening to port number 3374, possibly related to the Xbox-Live protocol, which is sometimes confused by the SVM as eMule protocol, since the SVM is “under-trained”. Also in this case, by adding some samples of these endpoints, no more FPs are detected.

We can conclude that KISS shows excellent performance, since in all cases the True Positive percentages are higher than 99%. The training of the SVM is robust considering the signature of known protocols, but it can suffer when the Background training set is small or does not include all protocols that may be present in the considered network scenario. This leaves room for improving the performance of KISS by carefully selecting the training set samples, but it is

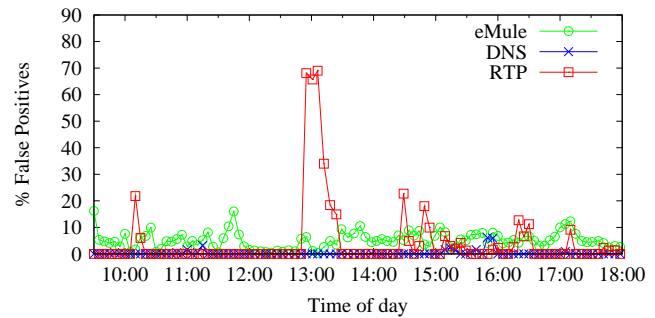


Fig. 8. False Positive percentage variation versus time. Aggregate in the training set.

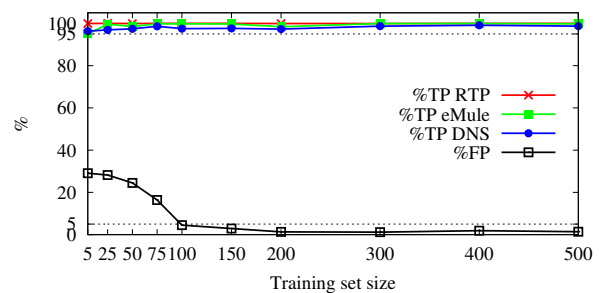


Fig. 9. Classification accuracy versus different training set size.

out of the scope of this paper.

C. Training with the Aggregate

A possible weakness of KISS is that the SVM must be trained with the Background traffic, i.e., with actual traffic extracted from the network the classifier is used in. Moreover, the background traffic must be deprived from the traffic that should be classified. While the first assumption does not pose particular issues, the extraction of “pure” background traffic is very questionable. A possible solution to this issue is to use, during the SVM learning phase, the whole *Aggregate* of traffic as “background” traffic. This poses some problems, since samples of a given class may be part of the *Aggregate* traffic as well.

Fig. 8 shows results obtained by running KISS in the scenario previously described, but using the *Aggregate* trace to train the SVM for the Background traffic. Also in this case the True Positive percentage remains higher than 99% (results are not plotted for the sake of brevity). Considering the False Positives, apart from the RTP endpoint that the oracle misclassifies, we observe an increased percentage of False Positives being classified as eMule (with an average %FP=4.5%). Nonetheless, results are very good.

D. Parameter Sensitivity

Among the parameters that are part of KISS, the number of samples C to evaluate the signature is the most critical one. Indeed, as discussed in Sec. II, to have a good estimate of the

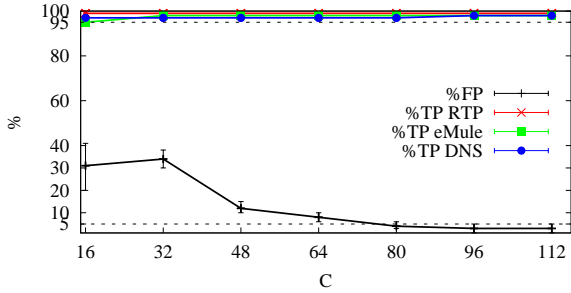


Fig. 10. Classification accuracy versus C .

TABLE IV
CONFUSION MATRIX CONSIDERING P2P-TV APPLICATIONS

	Tot.	Joost	PPLive	SopCast	TVants	Aggr.
Joost	33514	98.1	-	-	-	1.9
PPLive	84452	-	100.0	-	-	-
SopCast	84473	-	-	99.9	-	0.1
TVants	27184	-	-	-	100.0	-
Aggr.	1.2M	0.3	-	-	-	99.7

observed frequencies, at least 5 samples for each value should be collected (in case a uniform distribution is considered). This leads to $C \geq 80$. However, since in KISS we are not performing a real Chi-square test, we are interested in observing the classification accuracy of KISS when reducing the number of observation and therefore allowing an earlier classification. Fig. 10 reports the True Positive percentages of well-known protocols, and the False Positive percentages, without distinguishing among protocols. Confidence intervals are evaluated over 250 different RealTrace substraces each comprising more than 100 samples. The Figure clearly shows that the %TP is almost not affected by the number of samples that are considered to evaluate the observed frequencies in Eq.(1). Indeed, the format of the considered protocols is very different and the SVM has little problem in distinguishing them even if C is small. However, the %FP is much more sensible to the C value, and only for $C > 80$ it goes below 5%.

Similarly, it is interesting to observe how performance changes with training sets of different size. Results are plotted in Fig. 9, which reports the %TP and %FP for increasing training set size. The plot shows that KISS is able to correctly classify RTP, DNS and eMule traffic with excellent %TP, (average %TP>95%) even with 5 samples training sets. Also in this case, more problematic is the correct classification of the Background traffic, since the False Positive percentage goes below 5% only when the training set comprises at least 100 samples. The intuition behind this is that the Background traffic is far more heterogeneous with respect to traffic of a given protocol, and a larger number of samples are required to describe it.

E. P2P-TV traffic traces

To prove the KISS flexibility, we explore its ability to identify traffic generated by P2P-TV applications. Since these are

novel applications, they follow a proprietary and closed design, and they might exploit obfuscation and encryption techniques, the design and engineering of a DPI mechanism would be daunting and extremely expensive. On the contrary, training KISS to identify P2P-TV traffic is quite straightforward: For each considered application, a packet trace is captured by simply running the application, and then used to train the SVM. Similarly, Aggregate traffic is used during the training phase (we used the RealTrace I as Background traffic). The total amount of time required to complete this task is less than 6 hours, including the time to generate the traces on the testbed.

To test the ability of KISS to classify P2P-TV traffic, all traces from the P2PTrace dataset are used to evaluate the True Positive percentages. The RealTrace I is instead used to evaluate the False Positive percentage, since we assume no P2P-TV traffic could be present during 2006.

Results are summarized in Tab. IV, which reports percentages averaged over more than 1.2 millions of tests. Also in this case, results are amazing. KISS is able to correctly classify more than 98.1% of samples as True Positives in the worst case, and only 0.3% of False Positives are present.

F. Training with many classes

All the results reported evaluate a KISS model considering only 3 or 4 classes at time. It is interesting to analyze the performance of the classifier with a larger number of target protocols. Using Realtrace-II, P2P-TV testbed and the Skype datasets we create a KISS model including ten different classes, including the aggregate background traffic. Each class has been characterized with 300 signatures randomly chosen from the initial part of each dataset. Tab. V reports the confusion matrix of the classification results. As before, labels on the lines represents the ground truth. The first column reports the total number of signature extracted from each class while the other columns show the agreement between the ground truth and KISS. Again, results are impressive: KISS always achieves more than 99% of true positives, with less than 10% of false positives from the background class. Further analyses revealed that 7.59% of the false eMule samples are related to a single endpoint, which generates lots of short flows directed to a high number of different destinations. Unfortunately, we were not able to identify which actual protocol was used. After the adding of some sample of this endpoint in the training set of the background class, all eMule false positives disappeared. For what concern the 2.67% of samples identified as RTP, more than the 90% of them is generated by only two endpoints that use a mix of RTP protocol with version number 1 and 2 as previously discussed in Sec. IV-B.

G. Complexity

KISS has limited computational complexity. In terms of memory, 2^b counters are needed per group, giving a total of $G \cdot 2^b$ counters for each tracked stream. Considering bitwise counters, $G = 24$ and $b = 4$, 384 Bytes are required for each

TABLE V
CONFUSION MATRIX CONSIDERING P2P-TV, REAL-TRACES AND SKYPE TRAFFIC

	Tot.	Bittorrent	eMule	RTCP	RTP	DNS	Skype	SopCast	TVAnts	PPLive	Backg
Bittorrent	1268	100	-	-	-	-	-	-	-	-	-
eMule	57255	0.02	99.15	-	-	0.03	-	-	-	-	0.80
Rtcp	2407	-	-	99.96	-	-	-	-	-	-	0.04
Rtp	585647	-	-	-	99.79	-	-	-	-	-	0.21
Dns	2707	0.46	-	-	-	99.54	-	-	-	-	-
Skype	46600	-	-	-	-	-	99.61	-	-	-	0.39
Sopcast	83460	-	-	-	-	-	-	99.95	-	-	0.05
Tvants	25748	-	-	-	-	-	-	-	99.69	-	0.73
PPLive	27278	-	-	-	-	-	-	-	-	99.24	0.76
Backg	84273	0.27	7.59	-	2.67	0.22	-	-	-	-	89.25

flow, i.e., a Gigabyte of memory allows to track more than 2.7millions of streams.

The computation complexity of updating a $\bar{\chi}$ signature involves $G = 24$ increments for each packet. Once every $C = 80$ packets, the signature is computed. The cost of this computation is $O(G \cdot 2^b)$ multiplications, see Eq.(1). Considering a 1GHz CPU, and minimum sized packets of 40 Bytes, a load of about $(1/(24+(24*16/80)))*(40*8)Gbps=11.2Gbps$ can be theoretically sustained.

The computational complexity of the SVM decision corresponds to some products between vectors, i.e., it has a complexity of $O(G \cdot M)$ multiplications, being M the number of classes. Using the LIBSVM library it takes around 100 μs to classify a signature from empirical measurements on an linux system with an Intel(R) Core(TM)2 T8300@2.40GHz. Considering a single UDP flow, KISS can roughly classify $8 \cdot 10^5$ packets/s; thus, on-line classification is possible for a 256Mbps stream of minimum-size UDP packets, even with no code optimization or parallelization. The euclidean classifiers instead takes 300 μs but given the formulation of the distance computed there is space for optimization using parallelized operations.

V. RELATED WORK

Since **port-based** classification [1] has become unreliable, a number of different solutions and methodologies have been proposed to classify Internet Traffic [4], [5], [12], [18], [19], [20], [21], [22], [23], [24], [25], [26]. Classification engines can be coarsely divided into three categories, each of them exploiting different ideas.

Payload-based techniques [12], [19], [20], [21] inspect the content of packets looking for distinctive signatures that allow to recognize a given application. All DPI techniques fall in this class.

Statistical-based classification [5], [22], [23], [24], [27], [28] is based on the rationale that, since the nature of the services is extremely diverse (e.g., Web vs VoIP), the corresponding generated traffic is very diverse as well (e.g., short-lived bursts of big packets versus long-lived, constant bitrate flows of small packets). This class of work stems from the characterization and modeling research field, which started from pioneering work [29]. Initial work in this area

focused on the offline traffic classification, exploring which flow properties and which classification technique was best suited to discriminate traffic flows according to the different classes of applications [5], [22], [23], [24]. More recently, [27], [28] addressed the problem of “early” classification of individual applications, basing solely on information such as the size, direction (and inter-packet gap in case of [28]) of the very first packets of each flow: the initial handshake phase of different applications is distinctive and can be used as protocol fingerprint (e.g., SMTP handshake is different from HTTP one).

Finally, **Behavioral-based** classification [4], [25], [26] target the classification of Internet hosts on the sole basis of the transport layer traffic patterns they generate (e.g., P2P hosts contacts many different hosts typically using a single port, whereas a Web server is contacted by different clients with multiple parallel connections).

Our work aims at fine-grained classification of Internet traffic. As such, we consider work targeted to host identification [4], [25], [26] or to coarse-grained identification [5], [22], [23] to be not suited as a comparison for our purposed. Moreover, this work aims at filling a gap in the current Internet classification spectrum, specifically addressing UDP traffic classification. Since UDP is a connectionless protocol, we argue that approach such as [27], [28] cannot be applied as no handshake can be reliably identified in this case. Indeed, even the notion of “flow” is fuzzy considering UDP streams.

Works closest to ours are those that belong to the payload-based class. However, our work is very different from [12], [19], since the definition of application signatures does not rely on any reverse engineering of the applications. Instead, our approach is more similar in spirit to [20], [21], in which authors automate the extraction of signatures from the application payload. Both [20], [21] rely on signatures extracted from the beginning of each data stream (more specifically, the first 64–256 bytes). We remove this assumption, so that the classification can start at any point in a flow. This is an important difference, since, for example, it opens the door to adopt packet sampling to cope with the ever increasing link data rate.

Another significant difference consists in the technique used to express the payload fingerprint: [20] uses discrete byte

encoding, whereas the framework of [21] proposes the use of different models of increasing complexity. A latter difference consists in the technique explored to perform the classification: indeed, in this work we use Support Vector Machines which, to the best of our knowledge, has not yet been deeply tested in the context Internet traffic classification.

VI. CONCLUSIONS

We presented KISS, a novel classifier that couples the stochastic description of application protocols with the powerful discrimination power of Support Vector Machines. Signatures are extracted from a traffic stream by the means of Chi-square like test that allows application protocol format to emerge, while ignoring protocol synchronization and semantic rules. A SVM is then used to classify the extracted signatures, leading to exceptional performance.

KISS has been tested in different scenarios, considering both data, VoIP, and traditional P2P Internet applications. Results are astonishing. The average True Positive percentage is 99.6%, with the worst case equal 98.7%. Less than 0.05% of False Positives are detected. But KISS has also been proved to provide almost perfect results when facing new P2P streaming applications, such as Joost, PPLive, SopCast and TVants. Moreover, KISS is very robust to internal parameter setting and it is efficient both considering memory and computational requirements.

REFERENCES

- [1] T. Karagiannis, A. Broido, N. Brownlee, K.C. Claffy, M. Faloutsos, "Is P2P dying or just hiding?," *IEEE GLOBECOM '04*, Vol.3, No., pp. 1532-1538, November 2004.
- [2] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, P. Tofaneli, "Revealing Skype Traffic: when Randomness Plays with You," *ACM SIGCOMM*, Kyoto, JP, August 2007.
- [3] A. Finamore, M. Mellia, M. Meo, D. Rossi, "KISS: Stochastic Packet Inspection", *1st Traffic Monitoring and Analysis (TMA) Workshop*, Aachen, 11 May 2009
- [4] T. Karagiannis, K. Papagiannaki, M. Faloutsos "BLINC: multilevel traffic classification in the dark," *ACM SIGCOMM Computer Communication Review*, Vol. 35, No. 4, pp. 229-240, 2005.
- [5] A. W. Moore, D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS*, Banff, Canada, pp. 50-60, June 2005.
- [6] N. Cristianini, J. Shawe-Taylor, "An introduction to support Vector Machines and other kernel-based learning methods," *Cambridge University Press*, New York, NY, 1999.
- [7] R. Wang, Y. Liu, Y. Yang, X. Zhou, "Solving the App-Level Classification Problem of P2P Traffic Via Optimized Support Vector Machines," *Sixth International Conference on Intelligent System Design (ISDA 2006)*, Vol. 2, pp. 534-539, October 2006.
- [8] R. Wang, Y. Liu, Y. Yang, X. Zhou, "Solving P2P Traffic Identification Problems Via Optimized Support Vector Machines," *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA '07)*, pag. 165-171, May 2007.
- [9] C.C. Chang, C.J. Lin, "LIBSVM: A library for support vector machines," Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2001.
- [10] Skype Testbed Traces. Available at <http://tstat.tlc.polito.it/traces-skype.shtml>
- [11] M. Mellia, R. Lo Cigno, F. Neri, "Measuring IP and TCP behavior on edge nodes with Tstat," *Computer Networks*, Vol.47, No.1, pp. 1-21, January 2005.
- [12] A.W. Moore, K. Papagiannaki, "Toward the Accurate Identification of Network Applications," *In Passive and Active Measurement (PAM'05)*, Boston, MA, USA, March/April 2005.
- [13] "FastWeb Company Information", <http://company.fastweb.it>, 2006.
- [14] R. Birke, M. Mellia, M. Petracca, D. Rossi, "Understanding VoIP from Backbone Measurements", *IEEE INFOCOM 2007*, Anchorage, Ak, May 2007.
- [15] E. Leonardi, M. Mellia, A. Horvart, L. Muscariello, S. Niccolini, D. Rossi, "Building a Cooperative P2P-TV Application over a Wise Network: the Approach of the European FP-7 STREP NAPA-WINE", *IEEE Communications Magazine*, Vol. 46, pp. 20-211, April 2008.
- [16] "IPP2P home page", <http://www.ipp2p.org/>.
- [17] Y. Kulbak, D. Bickson, "The eMule protocol specification," *Technical Report Leibniz Center TR-2005-03*, School of Computer Science and Engineering, The Hebrew University, 2005.
- [18] C. Dewes, A. Wichmann, A. Feldmann, "An analysis of Internet chat systems," *3rd ACM SIGCOMM Internet Measurement Conference (IMC'03)*, Miami Beach, FL, pp.51-64, October 2003.
- [19] S. Sen, O. Spatscheck, D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," *13th International Conference on World Wide Web (WWW'04)*, pp. 512-521, New York, NY, May 2004.
- [20] P. Haffner, S. Sen, O. Spatscheck, D. Wang, "ACAS: automated construction of application signatures," *ACM SIGCOMM Workshop on Mining Network Data (Minenet'05)*, pp. 197-202, Philadelphia, PA, August 2005.
- [21] J. Ma, K. Levchenko, C. Kreibich, S. Savage, G.M. Voelker, "Unexpected means of protocol inference," *6th ACM SIGCOMM Internet Measurement Conference (IMC'06)*, pp. 313-326, Rio de Janeiro, BR, October 2006.
- [22] A. McGregor, M. Hall1, P. Lorier1, J. Brunskill, "Flow Clustering Using Machine Learning Techniques", *PAM 2004*, Antibes Juan-les-Pins, Fr., pp. 205-214, April 2004.
- [23] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," *4th ACM SIGCOMM Internet Measurement Conference (IMC'04)*, Taormina, IT, pp. 135-148, October 2004.
- [24] J. Eрман, M. Arlitta, I. Cohen, C. Williamson, "Offline/realtime traffic classification using semi-supervised learning", *Performance Evaluation*, Vol. 64, No. 9-12, pp. 1194-1213, October 2007.
- [25] T. Karagiannis, A. Broido, M. Faloutsos, K. Claffy, "Transport layer identification of P2P traffic," *4th ACM SIGCOMM Internet Measurement Conference (IMC'04)*, Taormina, IT, pp. 121-134, October 2004.
- [26] K. Xu, Z. Zhang, S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," *ACM SIGCOMM 2005*, Philadelphia, PA, pp. 169-180, August 2005.
- [27] L. Bernalle, R. Teixeira, K. Salamatian, "Early Application Identification," *Conference on Future Networking Technologies (CoNEXT'06)*, Lisboa, PT, December 2006.
- [28] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, "Traffic Classification through Simple Statistical Fingerprinting," *ACM SIGCOMM Computer Communication Review*, Vol. 37, No. 1, pp.5-16, January 2007.
- [29] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 4, pp. 316-336, August 1994.