KISS: Stochastic Packet Inspection Classifier for UDP Traffic

KISS: Stochastic Packet Inspection Classifier for UDP Traffic

Alessandro Finamore, *Student Member, IEEE*, Marco Mellia, *Senior Member, IEEE*, Michela Meo, *Member, IEEE*, and Dario Rossi, *Member, IEEE*

***Abstract*—This paper proposes KISS, a novel Internet classification engine. Motivated by the expected raise of UDP traffic, which stems from the momentum of Peer-to-Peer (P2P) streaming applications, we propose a novel classification framework that leverages on statistical characterization of payload. Statistical signatures are derived by the means of a Chi-Square (χ^2)-like test, which extracts the protocol "format," but ignores the protocol "semantic" and "synchronization" rules. The signatures feed a decision process based either on the geometric distance among samples, or on Support Vector Machines. KISS is very accurate, 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 traditional client–server protocols, VoIP, and both traditional and new P2P Internet applications. Results are astonishing. The average True Positive percentage is 99.6%, with the worst case equal to 98.1,% while results are almost perfect when dealing with new P2P streaming applications.**

***Index Terms*—Supervised learning algorithms, traffic classification.**

I. Introduction

LAST year 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) as for instance simple port-based classification.

As such, research on Internet traffic classification has gained significant attention, with a large number of proposals (see Section VI for an overview) that try to circumvent DPI limitations. Indeed, DPI is deemed to fail more and more due to: 1) proliferation of proprietary and evolving protocols; 2) embracement of strong encryption techniques; and 3) adoption of tunneling techniques [1], [2]. In previous proposals, UDP has usually been neglected in favor of applications running over TCP. Motivated by the rise 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.

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 the one generated by Skype [2]. In this paper, we push this intuition further, arguing that, due to the connectionless service of UDP, the very first bytes of the UDP payload of traffic streams are likely to carry some application layer protocol (L7-protocol) in which constant values, counters, random identifiers, etc., can be found. Recalling that a protocol specifies 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 results of the test are then used to compactly represent application fingerprints, which we call Chi-Square Signatures (pronounced as 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 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 or sampling, retransmission, or any possible strange packet arrival pattern since they are built over a statistical characterization of protocol format rather than over a deterministic description.
- They are suitable to both per-flow and per-endpoint classification.
- Their computational and memory requirements are very limited, so they are suitable for online classification.

However, KISS shares with DPI classifier the need to look at application layer messages. As a drawback, in case of encrypted payload, both approaches become ineffective.

After 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, from simple threshold-based heuristics

Manuscript received July 30, 2009; revised December 28, 2009; accepted February 09, 2010; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Kodialam. Date of publication April 05, 2010; date of current version October 15, 2010. This work was supported 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." A preliminary version of this paper appeared in [3].

A. Finamore, M. Mellia, and M. Meo are with the Politecnico di Torino, Torino 10129, Italy (e-mail: finamore@tlc.polito.it; mellia@tlc.polito.it; meo@tlc.polito.it).

D. Rossi is with ENST Telecom ParisTech, 75634 Paris, France (e-mail: dario.rossi@enst.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2010.2044046

[4] to Naive Bayesian classifiers [2], [5] to advanced statistical classification techniques [6]. In this paper, we compare a simple geometric decision process based on Euclidean distance to 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 performance of the proposed framework, we implemented KISS in Tstat [7], which we then use to derive the results presented in this paper. We test KISS on both testbed and real traffic traces, collected from an operative ISP network classifying traditional protocols (like DNS and RTP traffic), affirmed P2P protocols (like eMule, BitTorrent and Skype), and emerging P2P-TV applications (like PPLive, SopCast, Joost, TVants). KISS exhibits excellent performance, typically achieving more than 98.1% of True Positives when SVM is adopted. These astonishing results are due to both the accurate characterization of the KISS signatures and the precise classification of the SVMs.

The reminder of the paper is organized as follows. Section II introduces general concepts and specify the metrics chosen to evaluate KISS performance. Section III describes the KISS architecture, detailing both feature extraction and decision processes. Section IV describes the set of traces used in the experiments, and Section V reports a deep investigation of KISS, testing its performance and parameters in many different scenarios. An overview of other classification techniques is presented in Section VI, while Section VII concludes the paper.

II. General Framework

Before entering into the details of KISS, we briefly summarize the key ideas behind classification tools and the methodologies to test them and evaluate their performance.

A. Classifiers

Classifiers are defined by two main processes (see [6] and [8] for a more extended description):

- Feature extraction: the process of extracting the subset of information that summarizes a large set of data or samples;
- Decision process: the algorithm that assigns a suitable class to an observed sample.

Examples of features are specific strings in the payload (as in DPI), packet size, or amount of exchanged bytes. Potentially, any summary of a packet stream can be used, and its choice has a deep impact on the classifier performance. In our tool, features are defined from the statistical observation of the values taken by portions of the payload.

For the decision process, any machine learning technique can be adopted. In this paper, we focus on *supervised learning algorithms* [6], in which a training set composed of known traffic is used to build a model; the model is then used 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 "volume" into which samples of the considered class fall. During the classification process instead, the sample to be classified has to be labeled with the most likely class according to the volume it falls into. For example, assuming that there are two classes of objects, i.e., red and yellow apples, if the features of a sample place it in a volume dense of red apples, we are inclined to classify it as a red apple, too. However, defining the surface that delimits the volumes (to later take the decision) is tricky since training points can be spread out on the multidimensional space and complex surfaces must be described. In this paper, we consider both simple geometric decision process and SVM-based algorithm, which is considered to be among the most powerful supervised learning algorithms.

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

	Oracle Classification: True	Oracle Classification: False
Positive	True Positive	False Positive
Negative	False Negative	True Negative

B. Testing Methodology

Once a classifier has been designed, its performance must be evaluated and proper metrics must be defined. Assessing the performance of Internet traffic classifiers is not a trivial task due to the difficulty in knowing the "ground truth," i.e., what was the actual application that generated the traffic [9]; for the ground truth, an "oracle" is needed. 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 (you are the oracle), but reduces the representativeness of the experiments since synthetic traces are hardly representative of real-world traffic. Assessing the performance against traffic traces collected from operative networks is therefore mandatory. To extract the ground truth from the real traces, we developed an *ad hoc* oracle, based on DPI mechanisms, and we manually tuned and checked those results. However, the oracle may still be fooled.

Classification accuracy is often reported in terms of False Positive (FP) and True Positive (TP), and the False Negative (FN) and True Negative (TN). A test is said to be "True" if the classification result 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 that 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 the following.

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

$$\%\mathrm{FP} = 100\frac{\mathrm{FP}}{\text{Total Number of Negative Samples}}.$$

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

$$\%\mathrm{FN} = 100 \frac{\mathrm{FN}}{\text{Total Number of Positive Samples}}.$$

- True Positive percentage (%TP) is $100 - \%\mathrm{FN}$.
- True Negative percentage (%TN) is $100 - \%\mathrm{FP}$.

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

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).

III. KISS

A. Feature Extraction

A traditional DPI classifier inspects packet payload looking for *deterministic patterns*, such as particular strings that are compared to those in a 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 be redefined manually.

The goal of KISS is instead to *automatically discover application-layer header format*, without caring about specific values of the header fields: We aim at automatically letting the protocol format emerge. Since UDP is a connectionless protocol, the first bytes of the payload of each UDP packet typically contain an application-layer protocol header whose fields can be 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 values observed in a sequence of packets. The process of the format extraction is achieved by using a simple χ^2-like test. The test originally 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 values, out of C total observations ($\sum_k O_k = C$). Let E_k be the number of expected observations of k for the theoretical distribution $E_k = C \cdot p_k$, with p_k the probability of value k. Given that C is large, the distribution of the random variable

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

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 χ^2 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.

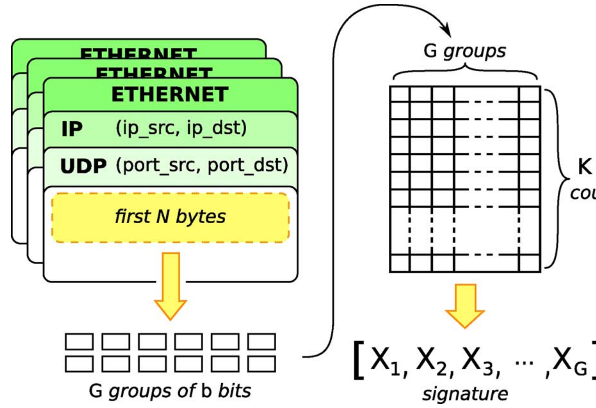

Fig. 1. Scheme of signature extraction process.

χ^2 signatures are built from *streams* of packets. The first N bytes of each packet 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

$$X_g = \sum_{i=0}^{2^b-1} \frac{\left(O_i^{(g)} - E_i^{(g)}\right)^2}{E_i^{(g)}} \tag{2}$$

and collect them in the vector

$$\overline{X} = [\, X_1,\, X_2, \ldots, X_G \,] \tag{3}$$

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 are created by describing the expected distribution of the protocols of interest. During the classification process then, the observed group g distribution $\{O_i^{(g)}\}$ must be compared to each of the $\{E_i^{(g)}\}$ in the database, for example using the χ^2 test to select the most likely distribution. However, this process ends up in a complex process in which (2) must be computed for each protocol of interest.

In addition to the high complexity, the comparison to reference distributions fails when the application protocol includes constant values that are randomly extracted for each flow. For example, consider a randomly extracted "flow ID" in group g. Consider two flows, one used for training and one for testing, generated by the same application. Let the training flow packets take the value 12 in group g. Let the test flow packets take instead the value 7 in the same group. Clearly, the comparison of the two observed distributions does not pass the χ^2 test, and the test flow is not correctly classified as using the same protocol as the training flow.

For the above motivations, we propose to simply check the distance between the observed values and a reference distribution, which we choose as the uniform distribution, i.e., $E_i^{(g)} = E = \frac{C}{2^b}$. In the previous example, the group randomness of the two flows has the same X_g values, which identify a "constant" value, independently of the actual value. In other terms, we use a χ^2-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 (2) evolves versus C, consider the case in which a deterministic group of bits is observed. Since for a deterministic group only one value is possible, the value of X_g becomes

$$X_g = \sum_{i=0}^{2^b-1} \frac{\left(O_i^{(g)} - E_i^{(g)}\right)^2}{E_i^{(g)}} \tag{4}$$

$$= \sum_{i=0}^{2^b-1} \frac{(O_i^{(g)} - E)^2}{E} \tag{5}$$

$$= \frac{(C-E)^2 + \left(2^b - 1\right) E^2}{E} = C\left(2^b - 1\right). \tag{6}$$

Thus, X_g linearly increases with C.

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

$$X_g = C\left(2^{b_0} - 1\right) + 2^{b_0} \chi_A^2 \tag{7}$$

where χ_A^2 is the Chi-square with A degrees of freedom. In this case, $A = 2^{b-b_0} - 1$.

To provide an example of the evolution of X_g, the left plot in Fig. 2 reports the value of two 4-bit-long groups belonging to two streams of two different traffic protocols, namely DNS and eMule, versus the number of collected packets C. The steep lines corresponding to groups taken from the eMule stream refer to fields that are almost constants. 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, and its behavior is almost perfectly random, the values of X_3 being compatible with those of a χ^2 distribution. The bouncing line, instead, corresponds to the typical behavior of a counter. In fact, (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 $\overline{X}$, the fingerprint becomes extremely accurate. Observe the right plot in Fig. 2. Each point in the figure corresponds to a different stream. A window of $C = 80$ packets is used to derive the signatures using the couple of features (X_2, X_3) as coordinates. Points obtained from DNS streams are displaced in the lower left corner of the plot; points from eMule are spread in the top part of the plot. Notice also that signatures of the same protocol class are not identical. This is due to both the behavior of each application and to different implementations of the same protocol. For example, some eMule clients can be downloading, uploading, or waiting, therefore exchanging different types of messages. Similarly, different implementations of a DNS server can use different random number generators to extract the query identifier. It is the scope of the decision process to define the areas where points of the same protocols are expected. 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 example. However, when several protocols are considered, more complex surfaces have to be found.

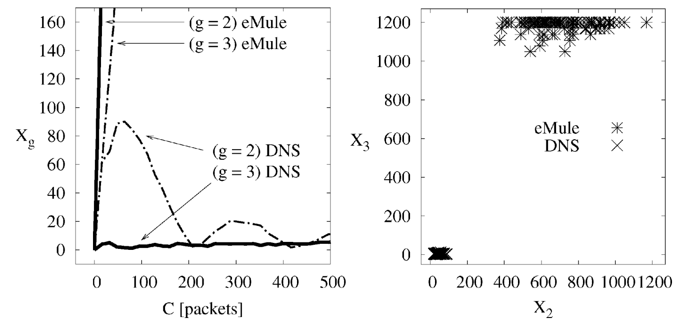

Fig. 2. (left) Evolution in time and (right) dispersions in space of X of two groups of 4 bits extracted from the second byte of UDP payloads.

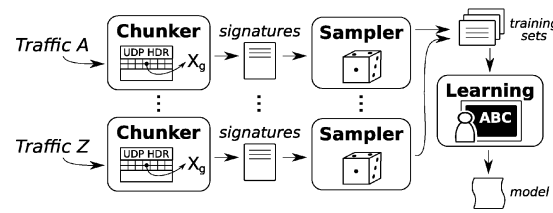

Fig. 3. Schematic representation of KISS learning steps.

B. 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 Section V-E. The training set is then fed to the learning system, after which the KISS model is produced. In this paper, we investigate two different learning systems, the first based on Euclidean distance and the second based on SVMs.

1) Euclidean Decision Process: A simple Euclidean distance is used for the decision process. A set of hyper-spheres, one for each protocol, is identified to define 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 a point that does not fall into any sphere is assumed to be of an *unknown* protocol.

For a given class A, the representative hyper-sphere is fully defined by its center $\bar{X}(A)$ and its radius $\rho(A)$. $\bar{X}(A)$ is simply computed, component by component, as the arithmetic mean of each signature in the training set of class A. 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:

$$\rho(A) = \arg\max_{\rho} \left(\%\mathrm{TP}(\rho) - \%\mathrm{FP}(\rho)\right). \tag{8}$$

Recall that $\%\mathrm{TP}(\rho)$ is computed considering samples of class A, while $\%\mathrm{FP}(\rho)$ 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. 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., a hyper-plane. SVMs exhibit a number of advantages.

- 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 [10] implementation.

Finally, notice that the output of the SVM 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 one 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 *unknown* protocols. Thus, the training set must contain two types of signatures: 1) the ones referring to traffic generated by the applications to classify; 2) the ones representing all the remaining traffic. We refer to this second class as *background* since it represents the set of applications that we cannot classify or are not interested in classifying.

IV. Testing Data Set

We aim at assessing KISS performance in the most difficult scenario, whenever possible. For most of the results we show, we consider real traffic traces, collected from an operative, totally uncontrolled network. In addition, to evaluate the performance of KISS when dealing with new protocols, we also selected, as a case study, P2P-TV applications. Indeed, P2P-TV systems have been recently introduced, and they are starting to became popular. These applications 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; thus, their classification is going to be more and more important.

A. Classification Objects

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. In this context, we define a classification entity as:

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

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

B. Testing Data Sets

1) Real Traffic Traces: Real traffic traces (RealTrace) were collected from the network of FastWeb [12], an ISP provider that is the main broadband telecommunication company in Italy, offering converged services, in which data, native VoIP [13], 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 restriction. It therefore represents a very challenging scenario for traffic classification. A probe node based on high-end PC running Linux has been installed in a PoP located in Turin, Italy, 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 datasets that we call RealTrace-I and RealTrace-II , collected in 2006 and 2007.[1]

Both traces contain many popular applications generating UDP traffic, in particular we selected: 1) eMule and Bittorrent; 2) VoIP (over RTP); and 3) 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 volume. In the remaining traffic, nearly 2% of flows are related to BitTorrent accounting for less than 1% of bytes. Skype communications instead present the typical mice/elephant behavior since a negligible number of flows account for more than 1% of the total volume in both traces. Being dated back to 2006 and 2007, no P2P-TV traffic is present.

2) P2P-TV Traces: To assess the performance of KISS with P2P-TV traffic, we selected, among the available P2P-TV applications, PPLive, Joost, SopCast, and TVants. Since none of the selected applications was available at the time of real traffic trace collection, we are forced to rely on testbed P2P-TV traces, called P2Ptrace, to assess the performance of KISS. This dataset of traces has been collected in the context of the Napa-Wine [14] 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

[1]Due 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

	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
P2Ptrace	10.25G	14M	132K	48.5K	3h, Apr '08
Skype	3.7G	24.7M	966	559	96h, May '06

collected from more than 45 PCs running P2P-TV applications in five different countries at 11 different institutions. The data set includes traces collected from PCs in campus LANs, corporate networks with restrictive policies, and home ADSL connections, so both nodes with public and private IP addressing are present. We are therefore confident that the heterogeneity of the P2Ptrace data set is representative of a wide range of different scenarios.

3) Skype Traces: In the tests, we also use the public available data set for the Skype traffic [11]. The data set contains both Skype traffic identified in [2] and traces collected in a controlled environment using PCs running different versions of Skype and different operating systems such as Windows, Linux, and Pocket-PC.

Table II summarizes the previously described data sets and reports the total amount of bytes, packets, flows, and endpoints for each data set and the collection time and duration of each trace.

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 are possibly taken for a single flow or endpoint. In this paper, we consider independent classifications, so the same flow/endpoint can be classified differently at each window. Notice that some reconciliation algorithm can be easily designed to increase the accuracy of the classification by considering the set of classifications involving the same flow or endpoint, e.g., adopting a majority criterion. We leave this issue to future work.

Notice that: 1) no assumption about observing the first set of packets is stated; 2) there is no need to observe bidirectional streams of packets; and 3) not all packets belonging to the same flow/endpoint must be exposed to the classifier; possible packet drop, reordering, and sampling can be present.

C. Oracle Definition

To obtain the ground truth from an *aggregated* trace, i.e., a traffic trace with a mixture of communications of different protocols, we developed a DPI classifier that was explicitly designed. It was implemented in Tstat [7], and its performance was manually fine-tuned and double-checked. In particular, DPI rules can be summarized as follows.

- DNS: We rely on simple port-based classification and a manual inspection as to identify only the flows with respect to DNS RFC1035.
- RTP/RTCP: We rely on the state machine described in [13]. It combines a DPI signature and correlates the value of the fields in consecutive packets (e.g., to check the validity of the counters).

TABLE III
EUCLIDEAN AND SVM PERFORMANCE ON REAL TRAFFIC TRACES

		Euclidean		SVM	
	Tot.	%TP	%FP	%TP	%FP
RTP	8386	99.9	-	99.9	-
eMule	1527	84.3	0.12	99.3	-
DNS	8245	91.3	-	99.9	0.01
Backg	2579	99.1	5.31	99.6	0.15

- eMule/BitTorrent: We developed a DPI classifier based on [15] and [16], adapting it to the considered scenario.
- Skype: We rely on the Bayesian framework described in [2].

All the aggregated traffic that does not match any of the rules is placed in a subtrace called *Background* since it represents all the *Unknown* protocols.

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

V. RESULTS

A. Real Traffic Traces

We first report results considering a small subset of the RealTrace-I data set, corresponding to the first 1 h of traffic. The oracle is used to split the trace into 4 subtraces: Each subtrace includes only packets classified as belonging to 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 a flow/endpoint can be classified several times (i.e., every C packets).

Table III summarizes the results. Each row corresponds to a class of traffic according to the oracle. The second column reports the total number of signatures extracted from each subtrace, while the remaining columns report the percentages of True Positives and False Positives for both Euclidean and SVM decision process.

The SVM results are astonishing: The True Positives are always higher than 99%, while False Positives are negligible. The performance of the Euclidean classifier is 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 a hyper-sphere as an approximation of the separation surface between classes. To this extent, Fig. 4 reports (8) as an example 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 applies 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 for the sake of brevity. Therefore, in the following

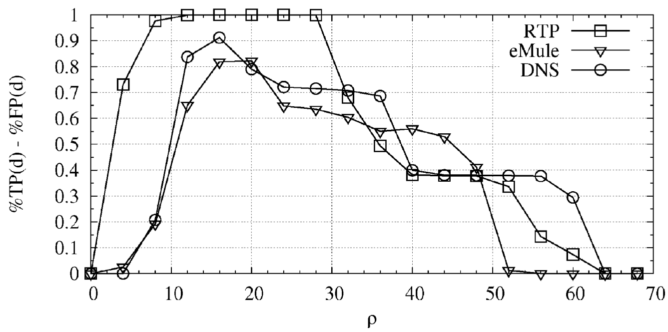

Fig. 4. Euclidean decision process: True Positive–False Positive evolution versus ρ for RTP, eMule, and DNS classes.

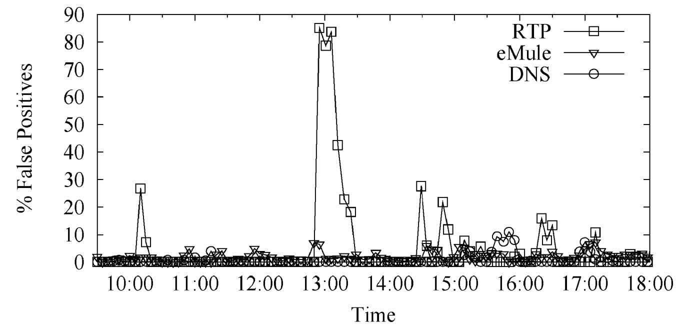

Fig. 5. False Positive percentage variation versus time. Background in the training set.

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

we will consider only the SVM classifier, and we will investigate how KISS performance is affected by parameters setting and different scenarios.

B. P2P-TV Traffic Traces

To prove the KISS flexibility, we explore its ability to identify traffic generated by P2P-TV applications. The design and engineering of a DPI mechanism for proprietary and closed P2P-TV applications would be daunting and extremely expensive. On the contrary, training KISS is quite straightforward: Apacket trace is captured by simply running the target application, and then it is used to train the SVM. RealTrace-I instead is used as Background class. In this way, all traces from the P2Ptrace data set are used to evaluate %TP, while RealTrace-I is instead used to evaluate the %FP since we assume no P2P-TV traffic could be present during 2006. The total amount of time required to complete this task is less than 6 h.

Results are summarized in Table IV, which reports percentages computed over more than 1.2 millions tests. Labels on the rows represents the ground truth. 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.

C. Signature Robustness

We are first interested in quantifying KISS robustness with respect to a training set independent from the test set. We thus perform an experiment in which the SVM is trained using samples extracted from the initial part of the RealTrace-I. A 9-h-long subset of RealTrace-I is considered, but the training set includes samples extracted from the first 30 min only. As in the experiment in Section V-A, only RTP, eMule, DNS, and Background are considered in the SVM model. Results are reported in Fig. 5 showing only Background False Positive percentages 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 included in the training set.

Investigating further, we notice that the high percentage of Background traffic classified as RTP traffic is due a single endpoint that 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 header is present: The RTP version field takes a value of 1 instead of 2. Apart from this difference, all other fields are in perfect agreement with the RTP standard as in RFC3550. Moreover, all packets received by this endpoint have 172 B of UDP payload, which is typical of VoIP streams using the ITU-T G.711 encoder [13] used in the FastWeb network. 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 classifies 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 UDP port number 9940) is responsible for this behavior. We manually inspected this traffic and verified that it cannot be a DNS endpoint, so the oracle is reliable. Interestingly, no sample 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 possible classes, it resolves to classify it as DNS rather than Background. Considering this endpoint only, Fig. 6 shows the probability that the SVM evaluates it as a Background or DNS sample versus time. It can be seen that some uncertainty is present. Repeating the experiment by including some of these endpoint signatures in the Background training set, KISS correctly classifies it. This is an example of "undertraining" of SVM.

Similar conclusions can be drawn investigating the eMule False Positives. They all correspond to endpoints listening to UDP port number 3074, possibly related to the Xbox-Live protocol, which is sometimes confused by the SVM as eMule traffic since the SVM is "undertrained." Also in this case, by adding some samples of these endpoints to the training set, no FP is detected.

We can conclude that KISS shows excellent performance since the True Positive percentages are higher than 99% in all cases. 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

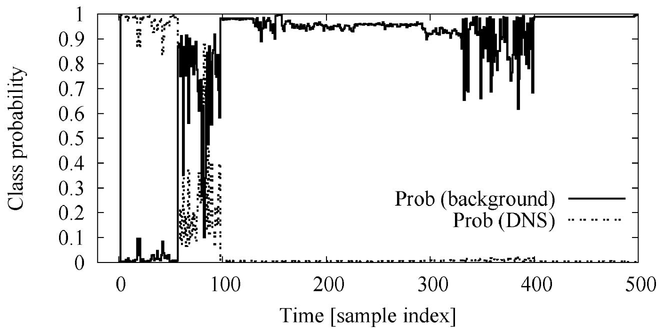

Fig. 6. Example of an endpoint that causes False Positives. Different classification windows over time.

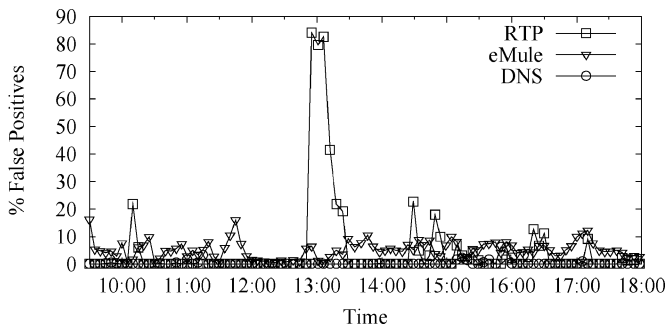

Fig. 7. False Positive percentage variation versus time. Aggregate in the training set.

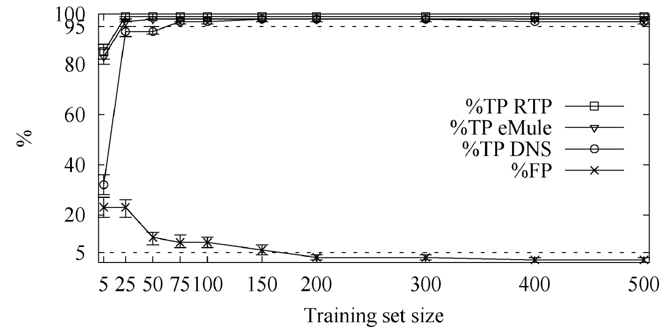

Fig. 8. Classification accuracy versus training set size.

room for improving the performance of KISS by carefully selecting the training set samples. Notice that the accuracy of any supervised machine learning decision process is strongly affected by the coverage and accuracy of the training set. Intuitively, a limited or outdated training set performs worse than an updated one. A discussion of the training set size and its impact on performance is presented in Section V.

D. 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 representing the *Unknown* protocols. While the adoption of actual traffic does not pose particular issues, the extraction of "pure" Background is very questionable. A possible solution to this issue is to use, during the SVM learning phase, the whole *Aggregate* of traffic as *Unknown* traffic. This poses some problems, since samples of a given class may be part of the Aggregate traffic as well.

Fig. 7 shows results obtained by running KISS in the scenario previously described, but using the Aggregate trace to train the SVM for the Unknown traffic. Also in this case, the True Positive percentage remains higher than 99% (results are not plotted for the sake of brevity). Considering FP, apart from the RTP endpoint that the oracle misclassifies, we observe an increased percentage of samples being classified as eMule (with an average $\%\mathrm{FP} = 4.5\%$). Nonetheless, results remain very good.

E. Training Set Size

Similarly, it is interesting to observe how performance changes with training sets of different size. Results are plotted in Fig. 8, which reports the %TP and %FP for increasing training set size with confidence intervals evaluated over 10 independent tests and accuracy $\alpha = 0.5$. The plot shows that KISS classifies RTP, DNS, and eMule correctly starting from a training set size of only 25 samples (worst case is $\%\mathrm{TP} > 91.73\%$ for DNS), but at least 75 samples are needed to obtain excellent results. Also in this case, the correct classification of the Background traffic is more problematic since the False Positive percentage is smaller than 5% only when the training set comprises at least 200 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 accurately describe it.

F. Training With Many Classes

All the results reported so far consider only three or four protocols. 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 data sets, we create a KISS model including nine different classes, plus one for the Background traffic. Each class has been characterized with 300 signatures randomly chosen from the initial portion of each trace. Table V reports the confusion matrix of the classification result. As before, labels on the rows represents the ground truth. The first column reports the total number of signatures, while the other columns show the agreement between the ground truth and KISS classification. 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 analysis 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 samples of this endpoint in the background training set, all eMule False Positives disappeared. For what concerns the 2.67% of samples identified as RTP, more than the 90% of them is generated by only two endpoints that use a RTP protocol with a wrong version number as previously discussed in Section V-C.

G. Parameter Selection and Tuning

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.

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

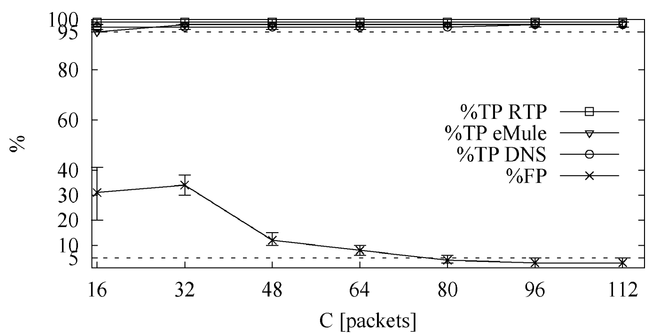

Fig. 9. Classification accuracy versus signature packet window C.

1) Bits Per Group, $b = 4$: The choice of b should trade off two opposite needs. On the one hand, we would like b to be the closest as possible to typical length of protocol fields; since protocol dialogs are usually based on words whose lengths are multiples of the byte or, sometimes, half a byte, b should be 4 or 8 or a multiple of 8. On the other hand, b should be small enough to allow that the packet window C over which the χ^2 test is statistically significant is not too large. Besides, in this case, streams can be classified even if they are not too long; they are classified in short time, and live classification is possible. Thus, we chose $b = 4$.

2) 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 N, in terms of both memory and computational complexity. As a convenient tradeoff, 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 on the targeted applications. For example, DNS and eMule can be clearly identified by only considering (X_2, X_3) as earlier showed in 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 the signature $\overline{X}$ is then a complex task that is left out as future work.

3) 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. However, since in KISS we are not performing a real χ^2 test, we are interested in the impact of smaller values of C, which would allow an earlier classification. Fig. 9 reports the True Positive percentages of well-known protocols and the False Positive percentages, without distinguishing among protocols. Confidence intervals with an accuracy $\alpha = 0.5$ are evaluated over 250 different subtraces from RealTrace-I, 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 (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 sensitive to C, and only for $C > 80$ it goes below 5%.

H. Coverage

The packet window size C plays an important role in 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. Fig. 10 confirms this intuition reporting the cumulative distribution function (CDF) of flow length for both flow/endpoint packets and bytes. All incoming UDP traffic in RealTrace-I is considered to derive the CDF. The left plot clearly shows 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 reported in the right plot. This clearly shows that while KISS is not suitable for the classification of short-lived UDP flows/endpoints, it can however successfully target the small fraction of them that generates the majority of the traffic, i.e., long-lived flows.

I. 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 byte-wise counters, $G = 24$ and $b = 4$, 384 B are required for each flow, i.e., a gigabyte of memory allows to track more than 2.7 M of streams.

The computation complexity of updating a $\overline{X}$ 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 (2). The computational complexity of the SVM decision corresponds to some products

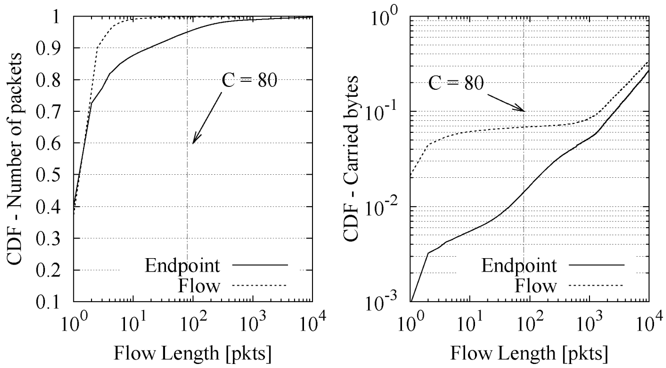

Fig. 10. CDF of the flow/endpoints length in (left) packets and (right) bytes. The vertical line is in correspondence of 80 packets.

between vectors, i.e., it has a complexity of $O(G \cdot M)$ multiplications, M being the number of classes. Using the LIBSVM library, it takes around 100 μs to classify a signature from empirical measurements on a Linux system with an Intel Core2 T8300 @ 2.40 GHz. Considering a single UDP flow, KISS can roughly classify $8 \cdot 10^5$ packets/s; thus, online classification is possible for a 256-Mb/s stream of minimum-size UDP packets, even with no code optimization or parallelization.

VI. 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], [9], [17]–[25]. Classification engines can be coarsely divided into three categories, each of them exploiting different ideas. For a good survey, see [8], while [26] is a complementary work for the survey.

Payload-based techniques [9], [18]–[20] inspect the content of packets looking for distinctive signatures that allow to recognize a given application. All DPI techniques fall in this class.

Machine-learning-based classification [5], [21]–[23], [27], [28] relies on the rationale that since the nature of the services is extremely diverse (e.g., Web versus 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], [21]–[23]. More recently, [27] and [28] addressed the problem of "early" classification of individual applications, basing solely on information such as the size and direction (and interpacket gap in the 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], [24], [25] targets the classification of Internet hosts on the sole basis of the transport-layer traffic patterns they generate (e.g., P2P hosts contact 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], [24], [25] or coarse-grained identification [5], [21], [22] to not be suited as a comparison for our purpose. 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 an approach such as [27] or [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 [9] and [18] 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 [19] and [20], in which authors automate the extraction of signatures from the application payload. Both [19] and [20] rely on signatures extracted from the beginning of each data stream (more specifically, the first 64–256 B). 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:[19] uses discrete byte encoding, whereas the framework of [20] proposes the use of different models of increasing complexity. Another difference consists in the technique explored to perform the classification. Indeed, in this work we use Support Vector Machines (SVMs) which, to the best of our knowledge, have not yet been deeply tested in the context of Internet traffic classification.

VII. Conclusion

We presented KISS, a novel classifier explicitly targeting UDP traffic that couples the stochastic description of application protocols with the discrimination power of SVMs. Signatures are extracted from a traffic stream by the means of χ^2-like test that allows application protocol format to emerge while ignoring protocol synchronization and semantic rules. A decision process based on SVM is then used to classify the extracted signatures, leading to exceptional performance.

Performance of KISS has been tested in different scenarios, considering both data, VoIP, traditional P2P applications, and novel P2PTV systems. Results are astonishing. The average True Positive percentage is 99.6%, and less than 1% of False Positives are typically detected. Moreover, KISS is very robust to internal parameter setting, and it is efficient considering both memory and computational requirements.

References

[1] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos, "Is P2P dying or just hiding?," in *Proc. IEEE GLOBECOM*, Nov. 2004, vol. 3, pp. 1532–1538.

[2] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype traffic: When randomness plays with you," in *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007, pp. 37–48.

[3] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "KISS: Stochastic packet inspection," in *Proc. 1st TMA Workshop*, Aachen, Germany, May 11, 2009, pp. 117–125.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," *Comput. Commun. Rev.*, vol. 35, no. 4, pp. 229–240, 2005.

[5] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proc. ACM SIGMETRICS*, Banff, Canada, Jun. 2005, pp. 50–60.
[6] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. New York: Cambridge Univ. Press, 1999.
[7] M. Mellia, R. Lo Cigno, and F. Neri, "Measuring IP and TCP behavior on edge nodes with Tstat," *Comput. Netw.*, vol. 47, no. 1, pp. 1–21, Jan. 2005.
[8] T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
[9] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. PAM*, Boston, MA, Mar./Apr. 2005, pp. 41–54.
[10] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," 2001 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[11] "Skype testbed traces," [Online]. Available: http://tstat.tlc.polito.it/traces-skype.shtml
[12] "FastWeb company information," 2006 [Online]. Available: http://company.fastweb.it
[13] R. Birke, M. Mellia, M. Petracca, and D. Rossi, "Understanding VoIP from backbone measurements," in *Proc. IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 2027–2035.
[14] E. Leonardi, M. Mellia, A. Horvart, L. Muscariello, S. Niccolini, and D. Rossi, "Building a cooperative P2P-TV application over a wise network: The approach of the European FP-7 STREP NAPA-WINE," *IEEE Commun. Mag.*, vol. 46, no. 4, pp. 20–211, Apr. 2008.
[15] "IPP2P home page," [Online]. Available: http://www.ipp2p.org/
[16] Y. Kulbak and D. Bickson, "The eMule protocol specification," School of Computer Science and Engineering, The Hebrew University, Tech. Rep. Leibniz Center TR-2005-03, 2005.
[17] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of Internet chat systems," in *Proc. 3rd ACM SIGCOMM IMC*, Miami Beach, FL, Oct. 2003, pp. 51–64.
[18] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th Int. Conf. WWW*, New York, May 2004, pp. 512–521.
[19] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: Automated construction of application signatures," in *Proc. ACM SIGCOMM Workshop Minenet*, Philadelphia, PA, Aug. 2005, pp. 197–202.
[20] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *Proc. 6th ACM SIGCOMM IMC*, Rio de Janeiro, Brazil, Oct. 2006, pp. 313–326.
[21] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proc. PAM*, Antibes Juan-les-Pins, France, Apr. 2004, pp. 205–214.
[22] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification," in *Proc. 4th ACM SIGCOMM IMC*, Taormina, Italy, Oct. 2004, pp. 135–148.
[23] J. Erman, M. Arlitta, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Perform. Eval.*, vol. 64, no. 9–12, pp. 1194–1213, Oct. 2007.
[24] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proc. 4th ACM SIGCOMM IMC*, Taormina, Italy, Oct. 2004, pp. 121–134.
[25] K. Xu, Z. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: Behavior models and applications," in *Proc. ACM SIGCOMM*, Philadelphia, PA, Aug. 2005, pp. 169–180.
[26] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. CoNEXT*, Madrid, Spain, Dec. 2008, Article No. 11.
[27] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proc. CoNEXT*, Lisboa, Portugal, Dec. 2006, Article No. 6.
[28] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, Jan. 2007.
[29] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Trans. Netw.*, vol. 2, no. 4, pp. 316–336, Aug. 1994.

Alessandro Finamore (S'09) received the M.Sc. degree in computer engineering at Politecnico di Torino, Torino, Italy, in 2008. He has been a Ph.D. student with the Telecommunication Network Group at Politecnico di Torino since 2009.

From 2008 to 2009, he was a research assistant at Politecnico di Torino. His research interests are traffic classification and computer programming.

Marco Mellia (SM'08) received the Ph.D. degree in telecommunications engineering from Politecnico di Torino, Torino, Italy, in 2001.

In 1999, he was with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Since April 2001, he has been with the Electrical Engineering Department, Politecnico di Torino. He has coauthored over 140 papers published in international journals and conferences, and he participated in the program committees of several conferences including IEEE INFOCOM and ACM SIGCOMM. His research interests are in the fields of traffic measurement, P2P applications, and energy-aware network design.

Michela Meo (M'02) received the Laurea degree in electronic engineering and the Ph.D. degree in electronic and telecommunication engineering from Politecnico di Torino, Torino, Italy, in 1993 and 1997, respectively.

Since November 1999, she has been an Assistant Professor with Politecnico di Torino. She coauthored more than 100 papers, about 40 of which are in international journals. She edited six special issues of international journals, including *Mobile Networks and Applications*, *Performance Evaluation*, and *Computer Networks*. Her research interests are in the field of performance evaluation and modeling, traffic classification and characterization, peer-to-peer, green networking.

Dr. Meo was Program Co-Chair of two ACM MSWiM conferences; General Chair of another ACM MSWiM conference; Program Co-Chair of IEEE QoS-IP, IEEE MoVeNet 2007, and IEEE ISCC 2009; and a program committee member of about 50 international conferences, including ACM SIGMETRICS, IEEE INFOCOM, IEEE ICC, and IEEE GLOBECOM.

Dario Rossi (M'02) received the Ph.D. degree in electronic and telecommunication engineering from Politecnico di Torino, Torino, Italy, in 2005.

From 2003 to 2004, he was with the Computer Science Department, University of California, Berkeley. Since October 2006, he has been an Associate Professor with Telecom ParisTech, Paris, France. He has coauthored over 40 papers in leading conferences and journals. He participated in the program committees of several conferences like IEEE ICC, IEEE IPCCC, and IEEE GLOBECOM. His research interests include P2P networks, Internet traffic measurement, and sensor and vehicular networks.