KISS: Stochastic Packet Inspection for UDP Traffic Classification *

Alessandro Finamore[1], Marco Mellia[1], Michela Meo[1], and Dario Rossi[2]

[1]Politecnico di Torino [2]TELECOM ParisTech
lastname@tlc.polito.it dario.rossi@enst.fr

Abstract. This paper proposes KISS, a new addition to the well populated and flavored world of Internet classification engines. Motivated by the expected raise of UDP traffic volume, which stems from the momentum of P2P streaming applications, we propose a novel statistical payload-based classification framework, targeted to UDP traffic.
Statistical signatures are automatically inferred from training data, by the means of a Chi-Square like test, which extracts the protocol "syntax", but ignores the protocol semantic and synchronization rules. The signatures feed a decision engine based 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 detected. KISS provides also excellent results when facing new P2P streaming applications, such as Joost, PPLive, SopCast and TVants.

1 Introduction

Last years witnessed a very fast-paced evolution of new Internet applications, ignited by the introduction of the very successful P2P networking 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

* This work was funded by the European Commission under the 7th Framework Programme Strep Project "NAPA-WINE" (Network Aware Peer-to-Peer Application over Wise Network)

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 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 expected raise of UDP traffic volume, which stems from the momentum of 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.

The mechanism we propose is based on the idea of identifying the application protocol "syntax", 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) information, and constant values, counters, random identifiers, etc., can be found. Recalling that a protocol specifies the rules governing the *syntax*, *semantics*, and *synchronization* of a communication, we propose to extract the L7-protocol *syntax* 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 KISS).

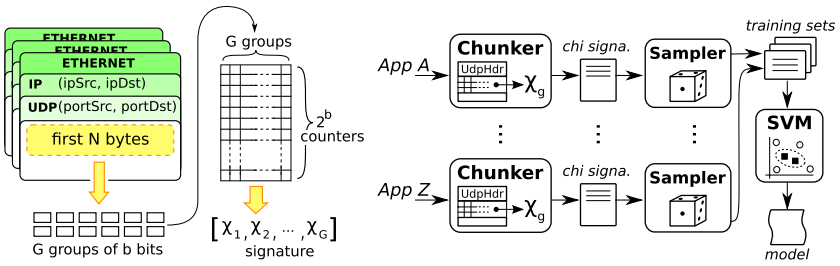
While KISS fingerprints stem from packet inspection, they have several advantages over classical deep-packet-inspection signatures: They can be automatically derived, i.e., no cumbersome and tedious reverse engineering is required; they can be quickly updated, so that they are well apt to the context of fast-evolving Internet applications; they are easily portable across different network settings, since fingerprints depend solely on the L7-protocol syntax; 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 syntax rather then 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 [3], to Naive Bayesian classifiers [2, 4], to advanced statistical classification techniques [5]. In this paper, we rely on Support Vector Machine (SVM) [5], which are well known in the statistical classification field, but have been rarely exploited in the context of Internet traffic classification. Indeed, in [6, 7] SVMs are used for P2P traffic classification using traditional transport layer information such as packet inter-arrival time, packet size, number of exchanged packets. SVMs are a set of supervised learning methods used for classification and regression. Given a geometric representation of computed features in a multidimensional space, the intuition suggests to assign a new sample to 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. 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. There is a large number of efficient algorithms and implementations already available. In particular, in this paper we adopted the `LIBSVM` library [8] implementation.

To prove the advantages of proposed framework, we implemented KISS in Tstat [9], 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 (like eMule, DNS, RTP traffic) and emerging P2P-TV applications (like PPLive, SopCast, Joost, TVants). Our results show that KISS exhibits excellent performance, typically achieving more than 99% of true positives, and no more than 1% of false positives in the worst case. These astonishing results are due to both the accurate characterization of the KISS signatures, and the precise classification of the SVMs.

2 General Framework

2.1 Chi-Square Signatures



(a) Signature extraction process. (b) KISS learning steps.

Fig. 1: Schematic representation of KISS internal structure.

The signature creation is inspired by the Chi-Square statistical test. The original 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 observations of value k, 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

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

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 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; we then check for the

distance between the observed values and uniformly distributed bits. 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.

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 $O_i^{(\mathrm{g})}$. We then define a window of C packets, in which we compute

$$\chi_g = \sum_{i=0}^{2^b-1} \frac{\left(O_i^{(\mathrm{g})} - E_i\right)^2}{E_i} \quad \text{with} \quad E_i = \frac{C}{2^b} \tag{2}$$

and collect them in the vector

$$\overline{\chi} = [\, \chi_1, \chi_2, \cdots, \chi_G \,] \tag{3}$$

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

The rationale behind KISS signatures is that they allow to automatically discover application layer message header without needing to care about specific values of the header fields: a measure of the degree of randomness is enough to let protocol syntax emerge and to allow the discrimination of different application protocols. Indeed, in the first bytes of the payload, UDP packets typically contain fields that 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 the operation in (2). For example, left plot in Fig. 2 reports the value of two 4-bit long groups belonging to two different traffic protocols, namely DNS and eMule, 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 random. In the extreme case of perfect determinism, it results $O_j^{(g)} = C, O_i^{(g)} = 0, \forall i \neq j$, from which

$$\chi_g = C\left(2^b - 1\right) \tag{4}$$

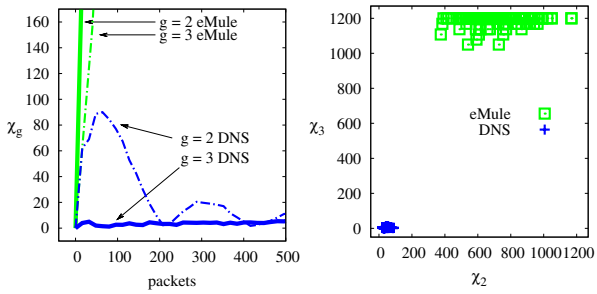


Fig. 2: Evolution in time (left) and dispersions in space (right) of χ^2 of two groups extracted from the second byte of UDP payloads.

The value of χ_g linearly grows with C. 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 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 $\overline{\chi}$ the fingerprint becomes extremely accurate. In order to have an intuition of this, 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.

Parameter setting The signature creation approach previously presented is based on a number of parameters. 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 protocol 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 significant is not too large, so that streams can be classified even if they are not too long, they are classified in short time and real-time 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 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) as Fig. 2 shows. The selection of which groups to include in $\overline{\chi}$ is then a complex task that is left out as future work.

Packet window size, $C = 80$. While we would like to keep the packet window as small as possible, the Chi-Square test 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 with respect to C is performed in Sec. 4.

2.2 KISS Model Generation

Since SVM is a supervised learning method, a training set must be used to allow the SVM to generate the model used later on during the classification task. To generate a KISS model we operate as sketched in Fig. 1b. We start by considering some streams that belong to a given set of applications we want to model. The streams could either be generated on purpose e.g., by running the applications; or they can be extracted from real traffic traces through some other reliable classification engine. Streams are then fed into a *chunker*, whose role is to derive the KISS signatures as in (3). This signature set is than randomly sampled by the *sampler* (according to a uniform distribution), so as to select the *training set*, whose size is 300 by default (the impact of this value

will be discussed in Sec. 4). The training set is then fed to the SVM learning phase after which the KISS model is produced.

As previously discussed, the SVM approach is based on the idea of mapping training samples so that samples of two different classes are displaced in compact areas separated by hyperplanes. The learning phase is the procedure of transforming the training vectors and of identifying the separation surface. In our case, the KISS signatures which belong to a $G-$dimensional space, will be mapped into a possibly infinite dimensional space during the training phase. The optimal transformation, once identified, will be used during the decision process: each classifying sample will be transformed so that it falls in a particular area in which only samples of a given class are present. Classification is then straightforward.

Notice that the output of the training phase is the definition of a number of regions equal to the number of classes defined during the training phase, e.g., one for each protocol to be classified. 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 SVM training set must contain 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*, that represents the set of applications that we cannot classify or we are not interested in classifying.

2.3 Complexity

KISS has limited computational complexity. In terms of memory, 2^b counters are needed per group, leading to a total of $G \cdot 2^b$ counters for each tracked stream. Considering bytewise counters, $G = 24$ and $b = 4$, 384 Bytes are required for each flow, i.e., a Gigabyte of memory allows to track more than 2.7millions of streams.

The computation complexity of updating a $\overline{\chi}$ 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 (1). With 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 sustained.

Finally, the computational complexity of the SVM decision corresponds to some products between vectors, i.e., it has a complexity of $O(G \cdot M)$ multi-

plications, being M the number of classes. Using the LIBSVM library it takes around 100 μs to classify a signature from empirical measurements. 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-sized UDP packets, even with no code optimization or parallelization.

3 Testing Methodology

Assessing the performance of any 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 [10]. 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 (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. Even when considering real traffic traces, performance of the classifiers can be affected by the scenario. For example, corporate networks or backbone networks have very different traffic mixes. The major problem when dealing with real traffic traces is however finding a suitable oracle.

In this paper, we aim at assessing KISS performance in the most difficult scenario, whenever possible. Thus, 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 double check its results. Moreover, to prove KISS flexibility, we also 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.

3.1 Testing Datasets

Real Traffic Traces Real traffic traces (RealTrace) were collected from the network of FastWeb [11], 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 [12], 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 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 a dataset collected starting from the 26th of May 2006, and ending on the 4th of June 2006[1]. The trace contains 6455 millions UDP packets, 77.6 millions flows, 56368 endpoints. Among the most popular applications generating UDP traffic, we selected: i) eMule, ii) VoIP (over RTP), and iii) DNS protocol. Indeed, these three protocols account for more than 80% of UDP endpoints, corresponding to 95% of the flows, and 96% of the total UDP bitrate.

Testbed Traces Since we are 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 [13] 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 40 PCs running P2P-TV applications in 5 different Countries, at 11 different institutions. The dataset includes traces collected 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.

[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.

3.2 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 monodirectional *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).

As discussed in Sec. 2, 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, it may be expected that UDP flows and endpoints last for few packets. Left plot of Fig. 3 confirms this intuition. It reports the Cumulative Distribution Function (CDF) of the number of packets of an endpoint or flow. All incoming UDP traffic in RealTrace is considered to derive the CDF. The plot clearly shows that 40% of flows and endpoints have 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 93 .8% and 98.6% of the *bytes* carried by UDP, as shown by the right plot of Fig. 3, which reports the fraction of bytes carried by flows/endpoints with less than C packets. Thus, 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 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) there is no need to observe the first packets of a flow/endpoint, the classification engine can be started in any moment; ii) there is no need to observe bidirectional packet streams, thus, there is no restriction on routing asymmetry; and, finally, iii) not all the packets belonging to the same flow/endpoint must be exposed to the classifier; possible packet drop, reordering, sampling can be present.

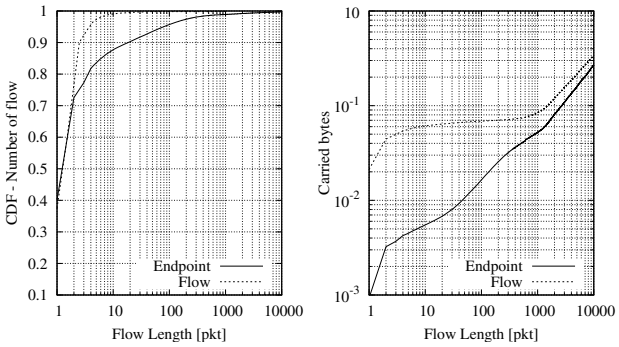


Fig. 3: CDF of the probability that a flow/endpoints has less than C packets (on the left), and fraction of the total traffic amount flows/endpoints with less than C packet carry (on the right).

Due to space limitations, we report results from the RealTrace dataset considering all traffic directed to endpoints whose 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 dataset, we consider all traffic originated by endpoints whose 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.

3.3 The Oracle Definition

To obtain the ground truth, we developed a DPI classifier that was explicitly designed. It was implemented in Tstat [9], an 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 used only by the DNS service during 2006.
- RTP: we rely on the state machine described in [12] 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).

Table 1: Definition of false/true positive and false/true negative

		Oracle Classification	
		Positive	Negative
Classif. Result	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

– eMule: a DPI classifier based on [14, 15] has been developed and adapted to the considered scenario[2].

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

4 Results

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 KISS classifies the flow 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 KISS classifies this flow as an eMule flow, then we have a False Positive. If not, then we have a True Negative. Table 1 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:

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

[2] The 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.

Table 2: Confusion matrix considering the RealTrace case

	Tot.	RTP	eMule	DNS	Back.
RTP	8389	99.9	0.05	-	0.05
eMule	7167	-	99.9	-	0.1
DNS	4491	-	-	98.7	1.3
Background	1477	-	-	-	100.0

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

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

– 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 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 row of the matrix represents how the instances of a given class are actually classified in the possible classes. 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).

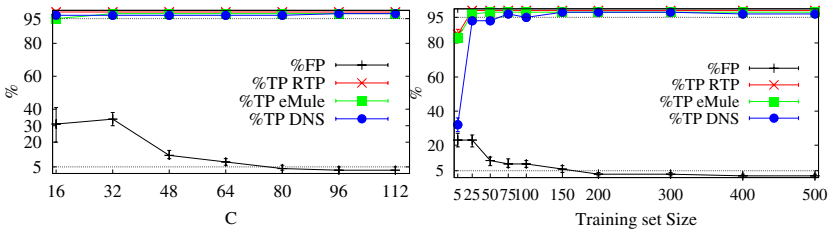
4.1 Real Traffic Traces

We first report results considering a small subset of the RealTrace 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. The SVM is trained using 300 signatures for each class, and the remaining signatures are used to assess the performance of the KISS classifier. 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. 2 reports the confusion matrix. Each row corresponds to a sub trace as classified according to the oracle. The first column reports the total number of samples in each class, the other columns report percentages. Values on the main diagonal correspond to %TP, while other values details the %FN and %FP.

Results are astonishing. The average True Positive percentage is 99.6%, with the worst %TP equal 98.7%, since 1.3% DNS endpoints are misclassified as Background (58 over 4491 tests). %FP=0.05%: all samples in the Background class has been correctly classified, while 5 RTP instances have been misclassified as eMule.

4.2 Parameter Sensitivity



(a) Classif. accuracy versus C. (b) Classif. accuracy versus train set size.

Fig. 4: Parameter sensitivity of KISS model.

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. 2, to have a good estimate of the 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 observations, thus allowing an earlier classification. Fig. 4a 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 subtraces each comprising more than 100 samples. Accuracy of 95% is reported. The Figure

clearly shows that the %TP is almost not affected by the number of samples that are considered. Indeed, the syntax 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%.

It is also interesting to observe how performance changes with training sets of different size. Results are plotted in Fig. 4b, which reports the %TP and %FP for increasing training set sizes. The plot shows that KISS correctly classifies RTP, DNS and eMule traffic with excellent %TP, (average %TP>95%) even with 25 samples training sets. Also in this case, the correct classification of the Background traffic is more problematic, since the False Positive percentage goes below 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 describe it.

4.3 Signature Robustness

In the previous test, the SVM has been trained using samples extracted from the set of instances to be classified. It is therefore interesting to observe what happens if the training set is extracted from a subset that does not include all instances. We perform an experiment in which the SVM is trained using samples extracted from the initial part of the RealTrace. In particular, a 9 hour long subset of the RealTrace is considered, and the training set includes 300 samples extracted from the first 30 minutes. Results are reported in Fig. 5. Only False Positive percentages are reported, since the %TP was always higher than 99%. The plot confirms 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 "syntax" of RTP protocol. However, the DPI oracle did not classify this endpoint as RTP, since the RTP version field had value of 1 instead of 2. Apart from this difference, all other fields are in perfect agreement with the RTP standard. Moreover, all packets received by this endpoint are 200Bytes long, which is typical of VoIP streams using the ITU-T G.711 encoder [12]. We can then claim that this is an *actual* RTP flow, but the DPI

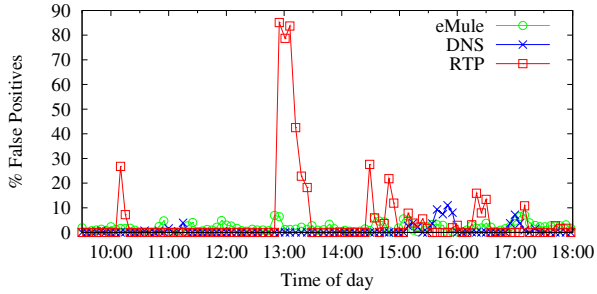


Fig. 5: False Positive percentage variation versus time. Background in the training set.

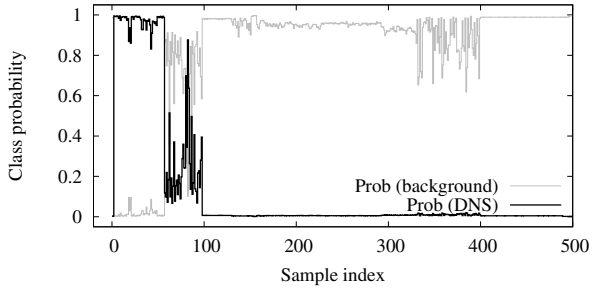


Fig. 6: Example of an endpoint with false positives. Different classification windows in time.

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 responsible of the higher %FP. We manually inspected this traffic, and verified that it cannot be a DNS endpoint; this time the oracle is reliable. Interestingly, this endpoint is not comprised in the Background training set. 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. 6 shows the probability estimated by the SVM that a given sample is Background or DNS, versus the sample index (or, equiv-

Table 3: 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

alently, 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 has 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 out of the scope of this paper.

4.4 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. Those traces are then used to train the SVM. Similarly, background traffic (in the aggregate form) is used during the training phase (we used the RealTrace as Background traffic). The total amount of time required to complete this task is less than 5 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 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. 3, which reports percentages averaged over more that 1.3 millions of tests. Also in this case, results are amazing. KISS is able to correctly classify 98.1% of samples as True Positives in the worst case, and only 0.3% of False Positives are present.

5 Conclusions

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

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.Tofanelli, "Revealing Skype Traffic: when Randomness Plays with You," *ACM SIGCOMM*, Kyoto, JP, August 2007.
3. 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.
4. A. W. Moore, D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS*, Banff, Canada, pp. 50-60, June 2005.

5. N. Cristianini, J. Shawe-Taylor, "An introduction to support Vector Machines and other kernel-based learning methods," *Cambridge University Press*, New York, NY, 1999.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. "FastWeb Company Information", `http://company.fastweb.it`, 2006.
12. R.Birke, M.Mellia, M.Petracca, D.Rossi, "Understanding VoIP from Backbone Measurements", *IEEE INFOCOM 2007*, Anchorage, Ak, May 2007.
13. 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.
14. "IPP2P home page", `http://www.ipp2p.org/`.
15. 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.