# Stochastic Packet Inspection for TCP Traffic

Gianluca La Mantia, Dario Rossi
TELECOM ParisTech – INFRES Department
Paris, France
firstname.lastname@enst.fr

Alessandro Finamore, Marco Mellia, Michela Meo
Politecnico di Torino – DELEN Department
Torino, Italy
firstname.lastname@polito.it

*Abstract*—**In this paper, we extend the concept of Stochastic Packet Inspection (SPI) to support TCP traffic classification. SPI is a method based on the statistical fingerprint of the application-layer headers: by characterizing the frequencies of observed symbols, SPI can identify application protocol formats by automatically recognizing group of bits that take e.g., constant values, or random values, or are part of a counter. To correctly characterize symbol frequencies, SPI needs volumes of traffic to obtain statistically significant signatures. Earlier proposed for UDP traffic, SPI has to be modified to cope with the connection oriented service offered by TCP, in which application-layer headers are only found at the beginning of a TCP connection.**

**In this paper, we extend SPI to support TCP traffic, and analyze its performance on real network data. The key idea is to move the classification target from single flows to endpoints, which aggregates all traffic sent/received by the same IP address and TCP port pair. The first few packets of flows sent from (or destined to) the same endpoint are then aggregated to yield a single SPI signature. Results show that SPI is able to achieve remarkably good results, with an average true positive rate of about 98%.**

## I. Introduction

Despite the effort devoted to the task of Internet traffic classification yield to significant progress in the field [1]–[14], the ultimate and definitive solution is still far from being available. Deep Packet Inspection (DPI) is still regarded as the state of the art and deployed in practice, despite it is well known that the proliferation of proprietary and evolving protocols and the adoption of strong encryption techniques are deemed to make DPI ineffective.

Motivated by the expected raise of UDP traffic volume due the popularity of application such as P2P-VoIP and P2P-TV, we proposed in [11] a classification framework tailored to UDP traffic, based on Stochastic Packet Inspection (SPI). Considering Deep Packet Inspection (DPI), typically precise keywords are searched to identify a specific protocol. With a human analogy, one may try to recognize the foreign language of an overheard conversation by searching for known words from a small dictionary (e.g., "Thanks" for English language, "Merci" for French, "Grazie" for Italian and so on).

At opposite, considering SPI paradigm, the packet payload is statistically analyzed (e.g., by means of entropy measures, or Chi-Square tests) to automatically build protocol signatures. The intuition behind SPI is that an application-layer protocol can be identified by statistically characterizing the values observed in a stream of packets. Considering the previous analogy, this time we aim at recognizing the foreign language by considering only the cacophony of the conversation, e.g., observing the frequencies of the occurrence of symbols like "x", or "h", or "i". In other words, statistical characterization of application-headers lets the protocol format emerge, while ignoring the actual semantic.

The SPI approach has proven very effective in the classification of UDP traffic (i.e., over 98% true positive classification in the worst case, with negligible false positive events [11]). It is therefore interesting to assess whether SPI could also be used to handle TCP traffic classification as well. Extension of SPI to TCP traffic is indeed not straightforward: SPI typically needs volumes of traffic (e.g., several tens of packets), being based on statistical characterization.

Recalling that UDP offers a connectionless service, each segment has to carry the application-layer header. Moreover, possible message segmentation has to be handled at application layer. As a consequence, applications relying on UDP have to include headers in each UDP segment, which SPI techniques can then reliably extract and characterize.

On the contrary, TCP offers a connection oriented service, according to which the application stream of data may be segmented into several TCP segments, among which only the first ones may carry the application-layer header. This contrasts with the need for volumes of traffic to build statistically significant SPI signatures.

To extend SPI to TCP traffic, we propose to shift the classification target from a single flow to an aggregate of flows: more precisely, we consider TCP endpoint entities, that can be uniquely identified by the server IP-address and server TCP-port pair. In this case, rather than constructing signatures over several segments of a single TCP flow, we aggregate the first few segments of several flows originated from (or destined to) the same server endpoint into a single SPI signature. Our results show that TCP endpoint aggregation is an effective approach that yields reliable signatures: SPI classification achieves remarkable results considering TCP traffic, showing an average and worst-case true positive rate of 98% and 91% respectively considering most common applications.

## II. Related work

Since port-based classification has become unreliable, three coarse classes of approaches have been proposed for Internet traffic classification [1]–[14]. Our work can be ascribed among **Payload-based** techniques, such as DPI [1]–[3], with an important difference. DPI techniques indeed inspect packet payload for the presence of known strings. The main idea of SPI is to give instead a statistical characterization of the observed values in the payload, automatically identifying constant, random, or periodic values. SPI is also different from **Statistical-based** classifications [4]–[8], which are based on the rationale that, being services extremely diverse (e.g., Web vs VoIP), so is the corresponding traffic (e.g., short bursts of large packets vs regular arrivals of short packets). Therefore, the classification can be based on statistical characterization of e.g. packet sizes, or inter-packet-gap, while completely ignoring actual payload values. Finally, **Behavioral-based** classification [9], [10], [12] targets a coarse-grained classification of Internet hosts on the basis of the transport layer traffic patterns they generate. For example, a Peer-to-Peer application generates and receives lot of connections, while an email client typically contacts only a single server, and a web client never receives connections.

## III. SPI Framework

### A. SPI Chunks

Statistical fingerprints can exploit a number of different metrics, such as for instance the Entropy measure, the Pearson's $\chi^2$ measure, the Kullback-Leibner divergence, etc. In the reminder of this paper, we evaluate the performance of SPI when its signatures are expressed using $\chi^2$ metric defined by a Pearson Chi-Square 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 frequencies of the observed values, out of $M$ total observations ($\sum O_k = M$). Let $E_k$ be the number of expected observations of $k$ for the theoretical distribution, $E_k = M \cdot p_k$ with $p_k$ the probability of value $k$. Given a $M$ large, the random variable $X$

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

represents the distance between the observed empirical and theoretical distributions. The distribution of $X$ can be approximated by a Chi-Square, or $\chi^2$, 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 SPI, we build a similar experiment analyzing the values taken by groups of bits having a fixed offset in the packet payload, called *chunks*. After a given number of packets (each giving an observation), the empirical distribution is collected and then compared to the *uniform distribution*, so to measure the amount of randomness of a chunk as an estimate of the source entropy. Notice that, by doing so, SPI is able to cope also with obfuscated or encrypted chunks [6].

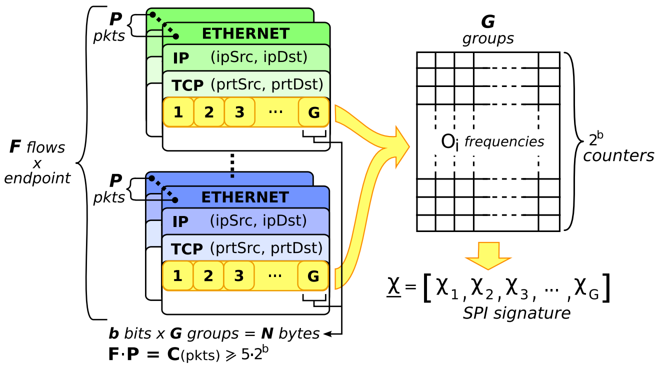

Fig. 1. SPI signatures of TCP traffic: $G$ groups of $b$-bits long chunks are extracted from the first $P$ packets of $F$ different flows (with $C = F \cdot P$) originated from (or destined) to the same endpoint $(IP, p)$.

For the time being, let us consider a single traffic stream. More precisely, consider an arbitrary payload chunk $\mathrm{x}$ of $b$ consecutive bits, and observe the values taken by the chunk over a stream of $C$ packets: due to the fact that the chunk is $b$-bits long, we have that $\mathrm{x}$ can take values in $[0, 2^b - 1]$. Denoting with $O_i^{(\mathrm{x})}$ the number of times that chunk $\mathrm{x}$ takes a value $i \in [0, 2^b - 1]$, we have:

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

Intuitively, $\chi_{\mathrm{x}}$ achieves low values ($\simeq 0$) whenever the chunk under observation has a random behavior (e.g., due to obfuscation, encryption, compression, etc.). In case of deterministic behavior (e.g., a constant identifier, address, etc.), we have that $\chi_{\mathrm{Det}} = (2^b - 1)C$, which is also the maximum value that $\chi_{\mathrm{x}}$can take. For convenience, we renormalize the $\chi_{\mathrm{x}}$ as:

$$\omega_x = \frac{\chi_{\mathrm{x}}}{(2^b - 1)C} \tag{3}$$

Chunks have therefore $\omega_x \in [0, 1]$.

### B. SPI Signatures

SPI signatures are then build by aggregating together the $\omega$ values of several chunks. As outlined in Fig. 1, the first $N$ bytes of the payload (i.e., the application protocol header) are divided into $G$ groups of $b$ consecutive bits each. $C$ observations of each chunks (i.e., $C$ packets) are necessary to form the SPI signature.

In the case of UDP, all $C$ packets identified by the same tuple $(IP_{src}, UDP_{src}, IP_{dst}, UDP_{dst})$ belong to the same flow. Conversely, in case of TCP, the segmentation introduced at the transport layer reduces the chances that a TCP segment carries the application protocol header. However, we can expect that the very first few segments of each flow carries information that are specific to each protocol (e.g., as in behavioral classification approaches [7], [8] that exploit the size and arrival time of the first few packets of a flow).

We therefore consider a TCP endpoint, that is uniquely individuated by the server IP address and TCP port pair

$(IP, p)$. We assume to be at the edge of the network, where all the endpoint traffic transits, and separately consider the two traffic directions, i.e., the traffic directed to, and the traffic originated from the endpoint $(IP, p)$. As outlined in Fig. 1, signatures are computed by observing the $G$ groups of chunks over the first $P$ packets of $F$ consecutive flows originated from (or destined to) the same endpoint, where $P$ and $F$ satisfies $C = P \cdot F$:

$$\underline{\hat{\omega}} = \{\omega_1, \omega_2, \dots, \omega_C\} \tag{4}$$

The rationale behind SPI signatures $\underline{\hat{\omega}}$ is that they allow to automatically discover application layer message header format without needing to care about specific values of the header fields (e.g., known keywords). Indeed, we expect application header to contain fields such as constant identifiers, counters, words from a small dictionary (message type, commands, 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). While randomness test provides only a coarse classification over individual groups, by jointly considering a set of $G$ groups as in (4) the fingerprint becomes extremely accurate. Notice indeed that the *position and length* of the different fields within the application protocol header will likely be different from protocol to protocol.

### C. SPI Parameter Selection

SPI signatures depend on a number of parameters, some of which are tied to the extension of SPI to TCP endpoint classification (such as $P$,$F$), whereas others (such as $b, C$ and $N$) pertain to the $\omega$ metric. Here we report guidelines on their selection, and refer the reader to [11], [15] for a more detailed sensitivity analysis.

**Bits per group** ($b = 4$). The choice of $b = 4$ trade-offs opposite needs. On the one hand, $b$ should be as closest as possible to typical length of protocol fields (e.g., $b$ should be 4 or 8 or a multiple of 8). On the other hand, $b$ should be small enough to enable statistically significant test over the smallest possible windows $C$, to allow live classification if possible.

**Packet window** ($C = 80$). While we would like to keep the packet window as small as possible, the $\chi^2$ 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 $C = 80$. Sensitivity to $C$ is evaluated in [11].

**Number of bytes per packet** ($N = 12$). In general, classification accuracy increases with the number of bytes per packet. However, complexity of the classification increases also with the $N$, in terms of both memory and computational complexity. As a convenient trade-off we choose $N = 12$. Given $b = 4$, this value corresponds to $G = 24$ groups for each signature. Notice that, as can be seen from Fig. 2, a fewer number of bytes and chunks may be sufficient to successfully discriminate different protocols.

**Number of packets per flow** ($P = 5$). The segmentation imposed by TCP yields an upper bound on $P$, the maximum number of packets at the beginning of a flow carrying application header at the beginning of the payload. As far as the number of packets per flow is concerned, we employ $P = 5$ which was observed to be a good value in [7], [8]. Sensitivity analysis to $P$ is provided in [15].

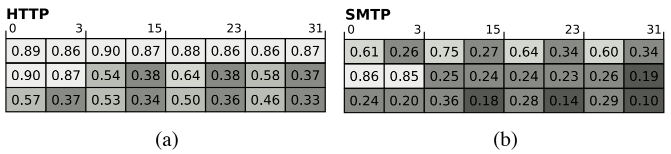

Fig. 2. Example of SPI signatures of HTTP (a) and SMTP (b) protocols, server endpoint is the destination.

**Number of flows per endpoint** ($F$=$C/P$=16). Constraints on $C$ and $P$ yield a lower bound on $F = 16$, the minimum number of flows to observe before an endpoint classification decision can be taken. This translates into a constraint on the classification timeliness, i.e., how fast and frequently classification can be taken, since the start of $F$ different flows have to be observed prior that a classification decision can be taken. Notice however that, the more active the endpoint, the quicker the identification (which is beneficial since operators are interested in classifying volumes of traffic, and should pose no problem in discriminating between active endpoints such as server vs P2P).

### D. Example of SPI signature

An example of SPI signatures for two different protocols, namely HTTP and SMTP is given in Fig. 2. It is derived considering segments of client requests directed toward the server endpoint. Average SPI signatures over 100 different endpoints are reported. In the example, parameters are set to their default values as stated above. The classical header representation is adopted, representing chunks in network order from left to right, top to bottom. Four bytes are reported on each row (i.e., 8 chunks) and, for reference, bit offsets are reported at the top. Each chunk reports the $\omega$ value, which is also visually represented with different scale of gray. Lighter colors correspond to higher values of $\omega$, suggesting deterministic fields, while darker colors correspond to low values of $\omega$, hinting to random fields. First of all, comparison of Fig. 2-(a) and Fig. 2-(b) confirms that, though the randomness test provides only a coarse classification over individual groups, and expressive fingerprints can be built by considering the whole set of $G$ chunks. This allows to clearly differentiate between protocols.

To grasp the SPI signatures expressiveness, let us first consider the case of the Web service, implemented over the simple and stateless HTTP protocol, whose SPI signature is reported in Fig. 2-(a). In the HTTP case, requests directed toward the server often begins with "`GET /`": the high occurrence of this 5-characters string translates into the first 10 chunks to be almost deterministic (high $\omega$ values). Variability of the first chunks is due to the fact that server can receive other HTTP methods than GET (e.g., POST, HEAD, PUT). Variability of subsequent chunks is instead tied to the different resources that can be specified after the method (e.g., URL in case of GET, parameters in case of POST, etc.).

Interestingly, HTTP uses an ASCII alphabet, which translates also into a reduced set of values chunks can take. Given a byte, since we use $b = 4$ bits long chunks, an ASCII encoded character is splitted into two chunks, corresponding to the most and least significant part of the byte respectively. The most significant chunk shows higher determinism ($\omega \simeq 0.6$), while least significant chunk shows higher randomness ($\omega \simeq 0.3$). For example, consider the ASCII uppercase letters $\{$A,…,Z$\}$ which take hexadecimal values in $\{0\times41,\dots,0\times54\}$. The most significant bits of a character fall into a chunk that takes only values of 4 and 5. Conversely, least significant bits falls into a chunk that takes any possible values from 0 to 15. This leads to different $\omega$ values, i.e., a different randomness. In Fig. 2-(a), the impact of ASCII encoding can be appreciated by observing the alternation of lighter and darker chunks.

Let us now consider the SMTP protocol signature reported in Fig. 2-(b). Recall that an SMTP client contacts a server with the typical sequence of commands EHLO, MAIL, RCPT, DATA. Notice that these commands are 4-characters long (which correspond to 8-chunks) and, with the exception of the DATA command, are followed by a space character and some parameters of variable length. Since several commands are used during the same session, there is a larger number of observed symbols, which therefore decrease $\omega$ of corresponding chunks. Also in the SMTP protocol case, commands are encoded using ASCII alphabet, causing a higher $\omega$ value for most significant chunks than for least significant chunks.[1] The highly probable space character at the 5th byte causes the 9th and 10th chunks to take deterministic values, as the high $\omega$ value observed in such position shows. Chunks corresponding to characters after the 5th position may contain any symbol of the ASCII alphabet, (e.g., angle brackets to enclose mail addresses, etc.) or user data, which then decrease the $\omega$ values of corresponding chunks.

## E. Decision process

After SPI signature have been computed for some known TCP protocols, classification implies to label samples according to the most similar signature. We resort to a supervised machine approach, in which the decision algorithm is first trained using a set of labeled samples, which are characterized by the $\underline{\hat{\omega}}$ features as from Eq. (4). After the training phase, the decision algorithm is then used to classify samples. In this paper, we rely on state of the art technique known in the literature as Support Vector Machines (SVM) [16]. SVM has only recently been applied to the context of Internet traffic classification [11]–[13], but it is considered among the most powerful supervised algorithm. Due to lack of space, we refer the reader to [16] for a good tutorial.

[1]The higher variability of the first 8 chunks is also due to other possible commands (e.g., VRFY), the presence of old clients (e.g., HELO instead of EHLO), clients using lower case letters, etc.

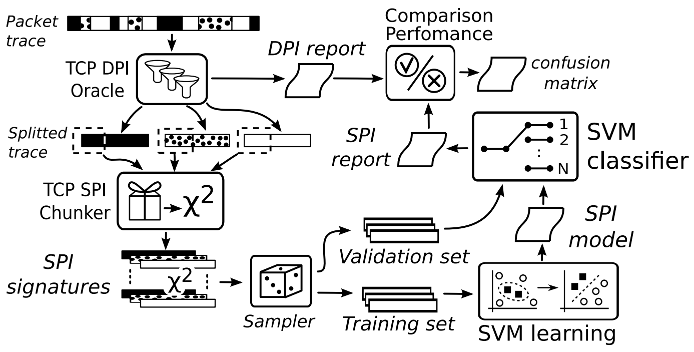

Fig. 3. Classification Workflow

TABLE I
AMOUNT OF BYTES, PACKETS, FLOWS, TCP ENDPOINTS AND SPI SIGNATURES IN THE CONSIDERED DATASET.

| Protocol | Bytes $[\cdot 10^9]$ | Packets $[\cdot 10^6]$ | Flows $[\cdot 10^3]$ | TCP Endpoints | SPI Signatures |
|---|---|---|---|---|---|
| HTTP | 343.67 | 507.08 | 6531.19 | 177 | 114222 |
| FTP | 0.04 | 0.65 | 19.39 | 21 | 229 |
| IMAP | 0.73 | 1.34 | 2.49 | 10 | 66 |
| POP | 3.40 | 7.74 | 156.39 | 25 | 3551 |
| Skype | 1.95 | 20.38 | 145.22 | 322 | 2752 |
| SMTP | 61.00 | 126.61 | 4917.20 | 56 | 83677 |
| SSH | 8.84 | 19.47 | 31.64 | 141 | 304 |
| Other | 453.83 | 744.53 | 13400.98 | 1512 | 46773 |
| *Total* | *873.46* | *1427.80* | *25204.5* | *2246* | *251574* |

## IV. CLASSIFICATION RESULTS

### A. Workflow

The overall workflow of SPI classification and validation is depicted in Fig. 3. As usually done in the literature, SPI performance is validated against the ground truth provided by an oracle. The oracle is used to split the packet trace file into different sub-traces, one for each protocol. For each sub-trace, we then compute the SPI signatures for each TCP endpoint. A subset of these signatures, uniformly selected at random, is used to train the SVM. As a result, SVM produces a "model" that is used during the classification process.

Signatures that have not been used for training purpose constitute the validation set: the SVM model is applied to this set, and SVM decisions are then compared against oracle labels to evaluate the correctness of the classification results.

Notice that the SVM training phase *partitions* the signature space into a number of regions equal to the number of protocols offered during the training: this implies that a sample will *always* be classified as belonging to any of the known classes. Thus, an additional label is needed for all samples that do not belong to any of the above protocols: in the following, we refer to these protocol as the "Other" set, comprising the applications that we cannot classify or are not interested in classifying.

### B. Dataset

Unfortunately, due to privacy issues, the scientific community lacks a common reference dataset used to benchmark the different proposal [1]–[12] although valuable effort in that direction is going on [13], [14]. For this reason, we evaluate the SPI performance using a traffic trace collected during may 2008 at the egress router of Politecnico di Torino network. The traces correspond to a one week long dataset, in which about 7000 internal hosts exchange data with more than 3 million different hosts in the Internet. Details concerning the traffic volume, in terms of the number bytes, packets, flows, endpoints and signatures, are given in Tab. I. The table reports the total traffic volume, and the breakdown across the most common application protocols considered in this work, namely HTTP, FTP, IMAP, POP, Skype, SMTP, SSH, Other.

In this paper we focus only on *internal endpoints,* i.e., servers whose IP address is internal to our LAN. Recall that we need to observe several flows involving a single endpoint to gather a single signature, and thus take a classification decision. In case of external endpoints, this means that several of our internal hosts have to contact the same endpoint to collect enough packets to compute the signature. While this is not an issue for popular external server and protocols (e.g., popular Web sites), however it limits the number of protocols we could use considering the dataset we use in this paper.

Our dataset includes more than 250000 signatures, that refer to about 2250 endpoints. As expected, Web service constitutes the bulk of traffic, while a fairly large amount of incoming SMTP traffic is present. The protocols we consider account for about one half of the traffic (in terms of bytes, packets and flows), yielding to a large fraction of the traffic to be labeled as “others”, which therefore includes all P2P traffic.

Concerning the number of available signatures, notice that each internal endpoint has to be contacted by $F$ different hosts of at least $P$ packets to compute the signature. The number of signatures per protocol depends on the arrival pattern as well as on the flow length as well.

## C. DPI Oracle

As already pointed out in [2], the definition of a reliable DPI oracle is a daunting task, that we have to carry on due to the lack of a labeled dataset. Except for the Skype protocol, for which we resort to [6], we devise a two-stages DPI oracle, defined as follows.

- **Port filter**: The first phase only involves TCP port number. We consider only those flows whose TCP destination port correspond to the corresponding service well-known port, i.e., 80 for HTTP, 22 for SSH, and so on. By doing so, we forcibly miss some endpoint. For example, HTTP servers running on port 8080 or on other non-standard ports end-up in the “other” protocol sub-trace. However, this choice yields to a conservative evaluation of the classification performance results.
- **Protocol syntax check**: The second phase involves application protocol check, that is done using the Wireshark tool. Wireshark is a well-know sniffer which is able to *parse* the headers of known protocols. In case during the parsing Wireshark fails to identify the protocol, we move the flow to the sub-trace containing all the other protocols since it is syntactically wrong.

TABLE II
CLASSIFICATION PERFORMANCE FOR TRAFFIC DIRECTED TO (TOP) OR ORIGINATED FROM (BOTTOM) THE SERVER-SIDE ENDPOINT.

| DST | HTTP | FTP | SMTP | IMAP | Skype | SSH | POP | Other |
|---|---|---|---|---|---|---|---|---|
| HTTP | 94.94 | 0 | 0.06 | 2.58 | 0 | 0 | 0 | 2.39 |
| FTP | 0 | 98.59 | 0 | 0 | 0 | 0 | 0.03 | 0 |
| SMTP | 0 | 0 | 99.86 | 0 | 0 | 0 | 0 | 0 |
| IMAP | 0.02 | 0 | 0 | 90.97 | 0 | 0 | 0 | 0 |
| Skype | 0.01 | 0 | 0 | 0 | 100 | 0 | 0 | 0.05 |
| SSH | 0.05 | 0 | 0 | 0 | 0 | 100 | 0 | 0.03 |
| POP | 0.01 | 1.31 | 0.02 | 2.9 | 0 | 0 | 99.94 | 0 |
| Other | 4.97 | 0.1 | 0.06 | 3.55 | 0 | 0 | 0.03 | 97.53 |

| SRC | HTTP | FTP | SMTP | IMAP | Skype | SSH | POP | Other |
|---|---|---|---|---|---|---|---|---|
| HTTP | 91.63 | 0 | 0.07 | 1.54 | 0 | 0 | 0 | 13.99 |
| FTP | 0.35 | 98.98 | 0.02 | 0 | 0 | 0 | 0 | 1.05 |
| SMTP | 0 | 0.03 | 99.45 | 0 | 0 | 0 | 0.03 | 0.03 |
| IMAP | 0 | 0 | 0 | 58.08 | 0 | 0 | 0 | 0 |
| Skype | 0.01 | 0 | 0 | 0 | 100 | 0 | 0 | 0.03 |
| SSH | 0.15 | 0 | 0 | 0 | 0 | 100 | 0 | 0.05 |
| POP | 0 | 0 | 0 | 0 | 0 | 0 | 99.59 | 0 |
| Other | 7.86 | 0.99 | 0.46 | 40.38 | 0 | 0 | 0.38 | 84.85 |

## D. Performance evaluation

Evaluation of classification performance is conducted over the entire dataset, by comparing the SVM labels to the DPI oracle labels for each signature.

Results reported in this section refer to a test in which the training set containing 5000 signatures, proportionally balanced across protocols. Each test is repeated 10 times, by randomizing the training set at each execution, and validating the model on the remaining signatures. Average results over all 10 iterations are reported in the following.

In particular, 1800 training signature are used to describe the “other” protocol set, since this set comprises possibly several protocols and its proper description requires that such protocols are well represented in the training set. A sensitivity analysis to the training set size is not reported due to lack of space. Readers are referred to [15], which shows that, even considering only 35 signatures per each of the known protocols the classification results are minimally compromised. This is a consequence of the discriminative power of SVM, whose performance are known to be highly robust even in presence of few learning samples.

Tab. II summarizes the results. A *confusion matrix* representation is used, in which each column corresponds to a sub-trace filtered by the DPI oracle, which is fed to a trained SVM, whose output labels are reported on each row. Thus, diagonal elements of the confusion matrix account for True Positive classification (i.e., a protocol labeled as $X$ by DPI is also labeled as $X$ by SVM). Conversely, cells outside the diagonal refer to misclassified signatures: a protocol labeled as $X$ by DPI is labeled as $Y$ by SVM; this decision accounts for False Positive classification of $Y$ and False Negative classification of $X$.

Results considering the two different traffic directions are reported. Top (bottom) portion of the table reports the case where traffic is destined to (originated from) the internal server endpoints. Notice that, although classification results are very good in both cases, best results are obtained when traffic is destined to the server endpoints. This is visible for HTTP, IMAP and Other protocols. The intuition behind this is that the client protocol requests are easier to characterize than the server replies, which can be more variable. For example, HTTP requests use limited set of protocol keyword as discussed in Sec. III-D, while server answers can be much more different.

Focusing on traffic destined to the server, we gather that true positive rate classification always exceeds 90.97%, with an average of about 97.62%. Compared to the UDP classification results presented in [11] which yielded a 98% true positive rate *in the worst case*, the classification performance of TCP traffic decreases. This is somehow expected: in the UDP case, application protocol headers are present in each segment, yielding to very reliable SPI signatures; in the TCP case, the TCP connection oriented service and segmentation algorithms affect the SPI signatures, that are possibly computed over both application protocol headers and actual data carried in the first 5 TCP flow segments.

## V. Discussion and conclusions

This paper focused on the classification of TCP endpoints by means of Stochastic Packet Inspection. Even though SPI achieves remarkably good results, (average and worst case true positive rate of about 98% and 91% respectively), there is room for improvement, especially when compared to the results achieved by SPI for UDP traffic.

Two possible directions could be undertaken to improve SPI performance. The first implies to find an optimal value for $P$, which clearly depends on the length of the application protocol keywords. However, it is likely that there is no single value of $P$ that is optimal for all protocols, as already observed in [8]. A second direction could be instead of using SPI signatures based on the Predictive Entropy: in this case, the SPI signatures would statistically encode an *expected sequence* (rather than an *expected frequence*) of chunks, yielding to more robust signatures.

Finally, we are currently testing the SPI classifier to include also other classes of traffic, and in particular Peer-to-Peer traffic. Preliminary results are very promising, and shows that SPI has excellent performance also for those kind of traffic.

## Acknowledgment

The French researchers' work has been funded by the Celtic TRANS project.

## References

[1] S. Sen, O. Spatscheck, D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures", *In WWW'04*, New York, NY, US, May 2004.

[2] AẆ. Moore, K. Papagiannaki, "Toward the Accurate Identification of Network Applications", *In PAM'05* Boston, MA, USA, March 2005.

[3] J. Ma, K. Levchenko, C. Kreibich, S. Savage, G.M. Voelker, "Unexpected Means of Protocol Inference", *ACM IMC'06,* Brazil, Oct. 2006.

[4] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, "Class-of-Service Mapping for QoS: a Statistical Signature-based Approach to IP Traffic Classification", *In ACM IMC'04*, Taormina, Italy, October 2004.

[5] A.W. Moore, D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques", *In ACM SIGMETRICS '05*, Banff, June 2005.

[6] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, P. Tofanelli, "Revealing Skype Traffic: when Randomness Plays with You", *In ACM SIGCOMM'07*, Kyoto, Japan, August 2007.

[7] L. Bernaille, R. Teixeira, K. Salamatian, "Early Application Identification," *In CoNEXT'06*, Lisboa, PT, December 2006.

[8] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, "Traffic Classification Through Simple Statistical Fingerprinting", *ACM Computer Communication Review*, Vol. 37, No. 1, pp.5-16, January 2007.

[9] T. Karagiannis, K. Papagiannaki, M. Faloutsos "BLINC: Multilevel Traffic Classification in the Dark", *In ACM SIGCOMM'05,* Philadelphia, PA, August 2005.

[10] K. Xu, Z. Zhang, S. Bhattacharyya, "Profiling Internet Backbone Traffic: Behavior Models and Applications", *ACM SIGCOMM'05*, Philadelphia, PA, August 2005.

[11] A. Finamore, M. Mellia, M. Meo and D. Rossi, "KISS: Stochastic Packet Inspection," *In Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537,* May 2009.

[12] S. Valenti, D. Rossi, M. Meo, M.Mellia and P. Bermolen, "Accurate and Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets", *In Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537,* May 2009.

[13] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K. Y. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," *In CoNEXT'08*, Madrid, Dec. 2008.

[14] G. Szabo, D. Orincsay, S. Malomsoky, I. Szabo, "On the Validation of Traffic Classification Algorithms," *In PAM'08*, Cleveland, Ohio, USA, April 2008.

[15] G. La Mantia, "Statistical classification of TCP traffic with Support Vector Machines", *M.Sc thesis,* 2008.

[16] N. Cristianini, J. Shawe-Taylor, "An introduction to Support Vector Machines and Other Kernel-based Learning Methods", *Cambridge University Press*, New York, NY, 1999.