

Application Protocol Fingerprinting for Traffic Classification

Manuel Crotti, Maurizio Dusi, Alice Este, Francesco Gringoli, Luca Salgarelli

DEA, Università degli Studi di Brescia,
via Branze, 38, 25123 Brescia, Italy
E-mail: <lastname.firstname>@ing.unibs.it

Abstract—The proliferation of network protocols that try to bypass firewalls and application-level gateways by running on non-standard ports is making traditional techniques for the classification of network traffic ineffective. In this paper we present a classification mechanism based on statistical analysis of network traffic performed at the IP-level. The key idea behind our approach is a descriptive model of an application protocol that summarizes in a compact and efficient way its main IP-level statistical properties: we name this model *protocol fingerprint*. We introduce a simple algorithm that can quantitatively assess how much an unknown flow suits the basic properties captured by a protocol fingerprint: we then show that a *threshold-based* classification step can easily assign the analyzed flow to one of the known fingerprints and determine the application protocol carried by this flow. Preliminary experimental results show how this technique is effective in classifying network traffic.

I. INTRODUCTION

Accurate classification of network traffic has recently become a very popular research topic. The increasing interest is supported by both network security and management issues. On one hand, effective classification of traffic flows according to application layer protocols can be useful to counter network attacks and improve the reliability of Network Intrusion Detection Systems (NIDS). On the other hand, the need to allocate network resources in an efficient way makes traffic classification useful for the deployment of QoS and network management tools.

A very popular technique used in core router to assess the category of a flow relies on the analysis of its transport level fields [1], [2]. Although this method can work in real-time, it can be misled by applications running on arbitrary ports such as peer-to-peer: the results are hence too untrustworthy to be taken into account. Since at present time this is the only technique that can be used on core routers to deploy effective QoS mechanisms, flows must be classified at the network edge *before* they enter the backbone by means of other, more careful, methods.

Despite being very accurate, deep payload inspection methods require high computational power: they are, therefore, not compatible with the capacity of current and future high-speed networks and their use is generally limited to Network Intrusion Detection, e.g. many NIDS such as Bro and Snort [3], [4] adopt this method.

The accuracy and speed issues of these mechanisms are fostering the research towards a new generation of classifiers

that should base traffic analysis only on information collected at the network layer. The goal is the definition of effective classification methods that outperform the existing ones based on either port or payload analysis. Basically, these new classifiers carry out some statistical analysis and derive some features to characterize the traffic that is generated by a given application. This information is then exploited to classify unknown flows: it is worth noting that a well designed method could be simple and fast enough to be applicable to high-capacity links for real-time classification.

Our approach belongs to this class: starting from a collection of flows generated by the same protocol and collected at one site edge, we create a statistical model based on packet-size, inter-arrival time and arrival order of the IP packets. We then check if the features extracted from an unknown flow suit those described by the model that we call *protocol fingerprint*. To this end we introduce the concepts of *anomaly score* and *probability product* so that the generic flow can be compared to different fingerprints in order to establish the application protocol that generated it. Although at a preliminary stage of development, the results are promising and support the idea that a statistical classifier could be at least as reliable as payload-based classifiers: as we will show the ultimate advantages of the proposed technique are its lightweight and its robustness against the introduction of new application protocols or changes to existing ones.

The rest of the paper is organized as follows. In section II we discuss related work. In section III we deal with the description of our statistical model, while in section IV we present the classification algorithms used to test the model. Section V presents preliminary classification results and their analysis. Finally, section VI concludes the paper.

II. RELATED WORK

The use of statistical analysis as a method to model Internet traffic at the network layer is not new. The key idea is to derive statistical description of application protocols by choosing proper features: the identification of such features remains an open debate.

Under the umbrella of this research branch we can mention the pioneering studies of Paxson [5], [6]. In these works the author provides an empirical description of some application protocols considering random variables such as packet length and inter-arrival time; he then introduces analytical models

to describe these quantities, validating the idea that some variables could be effectively exploited to describe application protocols.

In [7] and [8] a first attempt of statistical classification is shown. The authors introduce a technique that successfully recognizes Real Audio and Internet Relay Chat flows from traffic aggregates, demonstrating the effectiveness of statistical approaches in the framework of QoS deployment. The authors of [9] and [10] suggest two clustering techniques that can be used to establish if a flow belongs to a certain class of protocols, where each class includes application protocols sharing a similar behavior. They show promising results despite only a few representative features are taken into account: total number of bytes, number of exchanges between endpoints and connection duration. A different approach is shown in [11], [12]: here the classification of class of protocols is based on a supervised machine learning technique. However, none of these works aims at a fine-grained, application oriented, classification.

The application of a clustering technique to the problem of traffic classification is proposed once again in [13]. In this work, the size and the direction of packets are used to create clusters in an n -dimensional space, where n represents the number of packets considered for each flow. The classification algorithm maps the flow under test in the n -dimensional space, and assigns it to the couple cluster-application protocol according to heuristics based on minimum distance criteria.

The authors of [14] suggest a promising approach for fine-grained protocol classification. Although their goals are similar to ours, their approach is different: the technique, in fact, relies on information about the hosts that originated the traffic and the networks that carried it to perform a classification that associates a host behavior pattern to one or more applications.

The work presented in this paper follows the approach we recently proposed in [15]: we use packet-size, inter-arrival time and arrival order to build application protocol fingerprints. In contrast to other works such as [16], our approach does not rely on clustering techniques and, working with higher-resolution binning of the involved variables, it leads to better results. Finally, being based on fingerprints, it requires a supervised training phase before any classification can take place.

III. PROTOCOL FINGERPRINTS

Atomic units for both training and classification phases are single flows as they are seen on the network edge where the algorithm is implemented. The statistical features to build the fingerprint of a protocol are, in fact, extracted from flows whose nature has been previously ascertained by a deep payload inspection mechanism. We can afford using such computationally-intensive mechanism because in this phase we only need to apply it to off-line traces. The same features are then extracted, during the classification phase, from unknown flows to determine their application protocols basing on the information inside the fingerprints. In this paper, we focus on TCP flows, in particular those produced by HTTP, SMTP and

POP3 applications. We leave as future work the evaluation of the applicability of the technique to a richer set of applications, as well as other transport layer protocols.

Here we refer to *flow* F as the unidirectional, ordered sequence of IP packets: we name the flow going from the client to the server F_{client} and F_{server} that going backward. From each flow we extract the ordered sequence of pairs $P_i = (s_i, \Delta t_i)$ that summarizes the size s_i of the i -th packet and the inter-arrival time between the i -th packet and the previous one. These quantities and the arrival order i represent the statistical features of our model. According to such definitions, any application session generates two flows that could be represented as follows:

$$F_{[client|server]} = \{P_i\}_{i=1}^{N_{[client|server]}}$$

where $N_{[client|server]} + 1$ is the number of packets that compose the flow: the sequence P_i , in fact, loses the first item since the first pair could be gathered only after the first two packets of the flow have been seen by the algorithm.

We chose these features following the work of Paxson [5], [6] that supports the idea that the application-layer machines that generate the flows impact the involved quantities in a manner that is unique for each application protocol: intuitively, a CHAT session would have a different behavior in terms of inter-arrival-time than a POP3 session, where no user interaction is involved. In its general form the presentation of our technique requires quite some space so the reader is referred to [15] for a more detailed description.

A. Protocol fingerprints

Starting from the pairs P_i we define the concept of *protocol fingerprint* as an efficient and compact way to represent the behavior of an application protocol. We first consider a training set of flows that we are sure belong to the same, known application protocol and estimate a vector of Probability Density Functions $P\vec{D}F$: clearly we will group together only flows of the same type, so we will deal with a couple of training set for each protocol, one set for F_{client} traffic and a different one for F_{server} . Similarly at the end of the training phase we will have two $P\vec{D}F$ for each application protocol. Here each component PDF_i of the vector represents the empirical joint density of the size s and inter-arrival time Δt referred to the i -th pairs, collected from those flows of a given training set that are at least $i + 1$ packets long. The length L of vectors $P\vec{D}F$ should depend on the longest flow used during the training phase but we limited it to a maximum value common to all protocols: as we will see L should be chosen to optimize the classification results.

The size s of the captured IP packets composing the collected traces is discrete and ranges in the set $[40, 1500]$ bytes: traffic was, in fact, captured on an Ethernet link. The variable Δt , instead, depends on the clock resolution of the capture device: in the case of Tcpcap running on off-the-shelf-hardware, the minimum not null value of Δt is 10^{-7} seconds. We limited the maximum value to 10^3 seconds for practical reasons since it is not convenient to wait so long.

While s is binned linearly, Δt is binned on a logarithmic time axis with step 10^{-2} .

Each resulting PDF_i is a 1461x1001 matrix: each cell expresses the probability that the i -th packet of a flow of the described protocol takes those particular values of s_i and Δt_i .

Since we perform an empirical estimation of PDF_i , i.e., based on a finite number of flows, a Gaussian filtering is introduced. This should provide a more realistic description of the protocol: it could happen, in fact, that a generic flow not used during the training phase but being generated by this protocol is not properly described by the model due to “noise” factors such as network congestion. In this case a variation of the round trip time could produce local oscillation of Δt and one of the packets of the flow could fall in a zero valued cell of the matrix even if the surrounding cells are populated with non zero values: a filtering should smooth matrix values and solve this issue. Since we prefer to deal with density estimation also after the filtering, matrices are rescaled so that they still sum up to 1. The Gaussian window holds a key role in the design of the fingerprint and we expect (and preliminary verify) that as the number of flows used in the training phase increase, the filter size can be reduced.

We then move to the concept of protocol *fingerprint* \vec{M} that consist of the two filtered PDF vectors evaluated on the training sets of that protocol.

B. Building accurate protocol fingerprints

The model assumes that a fingerprint is built starting from a set of flows belonging to the same and known application protocol. Generally, any technique that performs the classification based on a supervised approach relies on this assumption; however as reported in [14], this is not an easy task.

In this work, we adopted a payload-based pattern-matching technique to select the flows composing the training sets: even if the technique is generally time-consuming and computationally inefficient, it needs to be seldom performed, i.e. the first time the fingerprint is created and every time the fingerprint is updated. Behind the computational matter, it is worth noting that even a payload-based analysis cannot guarantee an error free training set: for instance, if one is looking for HTTP flows to build a model for web application, a blind payload based capture could incorrectly include in the set plenty of flows generated by peer-to-peer applications that use HTTP to tunnel their own traffic.

Although this issue should require a deeper investigation, we report in this paper a preliminary discussion about the consequence of “polluted” fingerprints: those flows not strictly belonging to the protocol under fingerprinting can distort the actual statistical description and increase the false-positive ratio during the classification phase. However, since the presence of “impurities” in the training set could not be completely eliminated, one should investigate on how to reduce their weight in the protocol fingerprint.

IV. CLASSIFICATION ALGORITHM

To test the effectiveness of the statistical protocol description introduced in section III-A we implemented two

classification algorithms: although they adopt different decision criteria, they both test how much an unknown flow is statistically conforming to the available fingerprints, or if it is the case, if none of them captures the behavior of the flow.

Since we want to evaluate how classification performance changes as we switch the algorithm, we used the same set of fingerprints \vec{M} during the two experiments: in the first we consider the product of probability as decision criteria; the second, instead, is based on anomaly score as described in our previous work [15].

A. Probability product algorithm

Given an unknown flow F we want to check how much it matches the description \vec{M}^j of protocol j . In the following N represents the flow length and L the length of the fingerprint vectors. We transform the N long sequence $P_i = (s_i, \Delta t_i)$ associated to the flow into a new L long one: the coefficients Z_i^j of the new sequence depend on $M_i^j(P_i)$, the probability density functions evaluated in P_i , and are weighted by $P^j(N \geq i)$, the probability that flows generated by the j -th protocol are at least i packets long. If the flow is shorter than L we compute the remaining coefficients Z_i^j for $N < i \leq L$ by considering the probability that flows generated by this protocol are shorter than i packets. When $N > L$ we simply discard exceeding coefficients P_i , $i > L$. According to this definition we have L coefficients for every unknown flow:

$$Z_i^j = \begin{cases} M_i^j(P_i) \cdot P^j(N \geq i) & N \geq i \\ P^j(N < i) & N < i \end{cases} \quad (1)$$

The probability function P^j is known since the *pdf* of the flow length is computed during the training phase, analyzing the length of every flow in the training set. We consider next the contribution of all pairs by evaluating the product

$$V(F, \vec{M}^j) = \prod_{i=1}^L Z_i^j \quad (2)$$

Low values of $V(F, \vec{M}^j)$ are expected if the flow does not fit the protocol behavior description \vec{M}^j .

As an unknown flow comes to the classifier, eq. 2 is calculated for each protocol fingerprint at disposal, and the flow is *marked* as being generated by protocol j if the corresponding value $V(F, \vec{M}^j)$ is the greatest. Eventually, this maximum is compared to a threshold value T : if greater, the flow is *labeled* with that protocol. An outline of the algorithm is shown in figure 1(a).

B. Anomaly score algorithm

We introduce another classification algorithm to decide if a generic flow belongs at least to one fingerprint. It is based on the vector \vec{A} whose components are given by

$$A_i(P_i, M_i^j) = \frac{1}{\max(\varepsilon, M_i^j(P_i))}, \quad (3)$$

where M_i^j is the i -th probability density function of protocol j and ε is a small positive quantity useful to let the expression be

```

1: for ( $j = 1; j \leq \#\vec{M}; j++$ ):
2:   compute  $V(F, \vec{M}^j)$  as in eq.2
3:   if  $\max_j\{V(F, \vec{M}^j)\} \geq T$ :
4:      $F \in \text{argmax}_j\{V(F, \vec{M}^j)\}$ 
5:   else:
6:      $F \in \text{unknown}$ 

```

(a) Product of probability algorithm.

```

1: for ( $j = 1; j \leq \#\vec{M}; j++$ ):
2:   compute  $S_n(F, \vec{M}^j)$  as in eq.4
3:    $X_n^j = S_n(F, \vec{M}^j)/T_n^j$ 
4:   if  $\min_j\{X_n^j\} \leq 1$ :
5:      $F \in \text{argmin}_j\{X_n^j\}$ 
6:   else:
7:      $F \in \text{unknown}$ 

```

(b) Anomaly score algorithm.

Fig. 1. Classification algorithm.

always finite: by construction we have $1 \leq A_i(P_i, M_i^j) \leq \varepsilon^{-1}$. The algorithm performs this calculation for each pair of the flow: after n pairs have been seen, it computes the *anomaly score* S_n of flow F versus \vec{M}^j as follows:

$$S_n(F, \vec{M}^j) = \frac{\left[\sum_{i=1}^n A_i(P_i, M_i^j) / n \right] - A_{min}}{A_{max} - A_{min}}, \quad (4)$$

where n is bounded between 1 and N_{sects} , the minimum between the number of pairs composing F and L ; $A_{min, max}$ are the allowed extreme values of A , i.e. 1 and ε^{-1} respectively. This leads to $0 \leq S_n(F, \vec{M}^j) \leq 1$. Since the algorithm produces a fresh value of the anomaly score every time a new pair is seen, a classification verdict can be emitted in real-time.

For each protocol fingerprint \vec{M}^j , the classifier computes the anomaly score $S_n(F, \vec{M}^j)$ and normalizes it to a previously calculated threshold value T_n^j that is introduced in the next section. The flow is then *labeled* with the protocol which gives the smallest result if the corresponding S_n does not exceed the threshold, otherwise it is marked as “unknown”. Figure 1(b) describes an outline of the classification algorithm.

1) *Threshold values*: The anomaly score threshold is useful to establish if a flow fits or not the behavior of a fingerprint. Since the fingerprint summarizes the statistics of the generic application protocol j , we consider as threshold value the quantity T_n^j composed of the sum of the mean μ and standard deviation σ of the anomaly scores of the flows used to build the fingerprint \vec{M}^j . Since, as we stated before, we can stop the classification at the n -th pair, we need a threshold for any possible n :

$$T_n^j = \mu \left\{ S_n(F, \vec{M}^j) \right\} + \sigma \left\{ S_n(F, \vec{M}^j) \right\}, \quad (5)$$

In this case, F is the set of flows used to build the fingerprint having at least n packets: note that, as for many other quantities in this paper, we need to differentiate the threshold depending on the direction of the flow (server or client). Therefore we have a couple of threshold for every fingerprint, one built on F_{client} flows and another built on F_{server} .

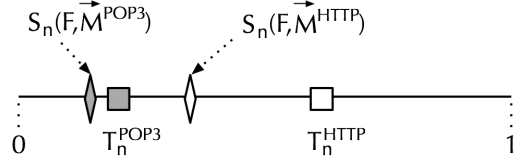


Fig. 2. Example: use of thresholds in the classification algorithm.

In the classification phase, for each unknown flow we compute as many S_n as the available fingerprints. To get the minimum and decide if the flow belongs to a fingerprint, each $S_n(F, \vec{M}^j)$ needs to be normalized by the corresponding T_n^j value. This is a key operation and it is better explained with an example. Figure 2 shows the hypothetical values of anomaly score of a given flow against the HTTP and POP3 protocol fingerprints, together with the respective threshold value. In the example, even though the flow score against HTTP is higher than its score against POP3, the former is farther than the latter, in relative terms, from the corresponding threshold. Therefore, it makes sense to classify the flow as HTTP rather than POP3. As we will see, experimental results support this intuition.

V. EXPERIMENTAL ANALYSIS

In this section we first characterize the data set used to verify the effectiveness of our classification method and then we present some experimental results. The results were obtained by considering only F_{client} flows, since we found out that they give better results than F_{server} .

A. Experimental setup

The data set is composed of several traffic traces collected at the edge router of our University. Our network topology comprises various layer-2 segments routed through a Linux-based dual-processor box and includes about a thousand workstations with different operating systems. All the traffic traces used during testing were gathered on the 24Mb/s link connecting the campus network with our Internet Service Provider.

The data set is pre-classified by applying a payload-based pattern-matching mechanism to all the data set, according to the patterns reported in [14], [17], to select and separate the flows that were generated by the protocols we considered. The data set is split into two parts: the training set and the evaluation set, each corresponding to the traffic of one week.

The training set is used to build the fingerprints and is composed of HTTP, POP3 and SMTP traffic: from collected traces, we randomly chose about twenty thousand flows for each application protocol. The selection of these three specific protocols is mainly due to the fact that they currently represent the most widespread protocols in our network. Another reason is that the payload-based techniques used to pre-classify the flows are reliable on these three protocols. The evaluation of the technique applied to other protocols is left as future work.

The evaluation set is only used for independent validation of the classification method. The evaluation set is composed of about ten thousand flows for each of the three considered

protocols, plus about five thousand flows of other protocols. We name OTHER the class of these flows, since it is composed of traffic not belonging to any of the fingerprinted protocols. To ensure the correctness of the evaluation set, in addition to running each flow through a pattern-matching mechanism, we analyzed by hand several flows.

B. Performance parameters

Here we introduce the definitions of the performance parameters used to evaluate our classifier. The symbol p represents a generic protocol. p can be one of {POP3, HTTP, SMTP, OTHER}.

- E_p is the number of flows generated by protocol p in the evaluation set, selected by the pattern-matching/by-hand pre-classification procedure. For example, with the data presented in the previous section, the following would hold: $E_{HTTP} = E_{POP3} = E_{SMTP} = 10000$.
- $E_{p|p}$ is the number of flows of the evaluation set correctly classified as protocol p by our classifier.
- $E_{w|p}$ is the number of flows of the evaluation set of protocol p incorrectly classified as protocol w by our classifier.

We can write the following identity:

$$E_p = E_{p|p} + \sum_{i \neq p} E_{i|p}. \quad (6)$$

- H_p is the hit-ratio, defined as the percentage of flows of the evaluation set of protocol p correctly classified:

$$H_p = \frac{E_{p|p}}{E_p}. \quad (7)$$

- F_p is the false-positive ratio, defined as the percentage of flows not belonging to protocol p that are incorrectly classified as p . If n is the number of classes, inclusive of the OTHER class, and all the protocols are equally probable:

$$F_p = \frac{1}{n-1} \sum_{\substack{i=1 \\ i \neq p}}^n \frac{E_{p|i}}{E_i}. \quad (8)$$

- *Classification accuracy* H is the total percentage of correct classifications:

$$H = \frac{1}{n} \sum_{i=1}^n H_i. \quad (9)$$

C. Sensitivity to the Gaussian window parameters

As stated in section III-A, any matrix in a protocol fingerprint can be modeled as a mixture of Gaussian windows: each window, centered at a sample point (a cell of the matrix), is a $N \times N$ matrix and is characterized by the parameter α . The values inside this matrix are given by the following equation

$$f[h, k] = \exp \left\{ -\frac{\alpha^2}{2} \left[\left(\frac{h - N/2}{N/2} \right)^2 + \left(\frac{k - N/2}{N/2} \right)^2 \right] \right\} \quad (10)$$

where $0 \leq h, k \leq N$ and $\alpha \geq 2$.

The window parameters need to satisfy some requirements. If the support width (matrix size N) is too large, then the fingerprints will be smoothed too much. On the contrary, if the support is too small, then the fingerprints will be made by sharp pulses centered at training samples and will be characterized by too much statistical variability.

The parameters of the Gaussian window are estimated by maximizing the classification accuracy H defined in eq. 9. When using the approach based on probability product, the best classification results are obtained for $N = 55$ and $\alpha = 18$ and are shown in table I. It is worth noting that the value of the threshold was also chosen to optimize this outcome: to this end we set $T = 10^{-15}$.

Each column of the table indicates the percentages of F_{client} flows assigned by the classifier to each class. When using the approach based on anomaly score, the optimum parameters are $N = 37$ and $\alpha = 1$, as shown in table II. In this case the classifier was instructed to take its decision after evaluating the 4th F_{client} packet.

	HTTP	POP3	SMTP	Unknown
HTTP	88.56%	0.01%	0.00%	11.43%
POP3	0.00%	97.95%	1.09%	0.96%
SMTP	0.03%	0.57%	98.75%	0.65%
OTHER	6.69%	2.62%	0.33%	90.36%

TABLE I

CLASSIFICATION RESULTS WITH OPTIMUM PARAMETERS OF THE GAUSSIAN WINDOW USING THE APPROACH BASED ON PROBABILITY PRODUCT, WITH THRESHOLD $T = 10^{-15}$.

	HTTP	POP3	SMTP	Unknown
HTTP	91.28%	0.02%	0.00%	8.70%
POP3	0.01%	95.19%	3.87%	0.93%
SMTP	0.03%	3.15%	95.78%	1.03%
OTHER	4.76%	4.27%	0.23%	90.74%

TABLE II

CLASSIFICATION RESULTS WITH OPTIMUM PARAMETERS OF THE GAUSSIAN WINDOW, USING THE APPROACH BASED ON ANOMALY SCORE.

D. Sensitivity to the accuracy of training set

The pre-classification phase, based on patterns described in [14], [17], has a strong influence on the accuracy of the training set. If the fingerprints are built using training sets that do not reflect the real statistical properties of protocols, the classification results deteriorate. There are two possible issues with the pre-classification phase: the first is due to a misbehavior of the pre-classifier that incorrectly drops some flows belonging to the protocol under consideration. This happens with more than 20% of the POP3 flows tested. The second issue is related to the inclusion of ‘‘polluting’’ flows in the training set: in the following we examine the latter, showing some experimental results.

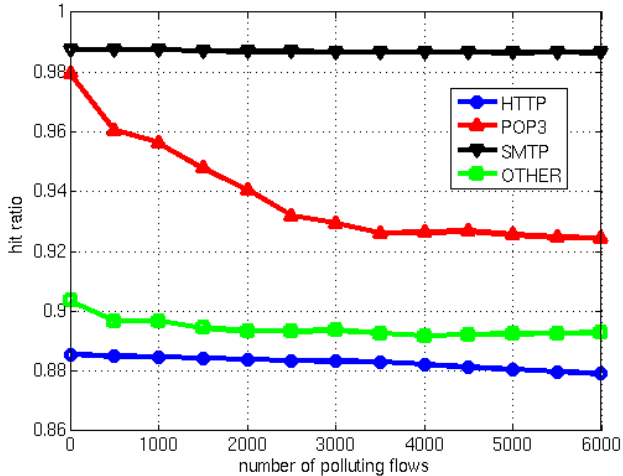


Fig. 3. Hit ratio H_p when some FTP flows are inserted in the training set of HTTP protocol, using the classification technique based on probability product, with threshold $T = 10^{-15}$.

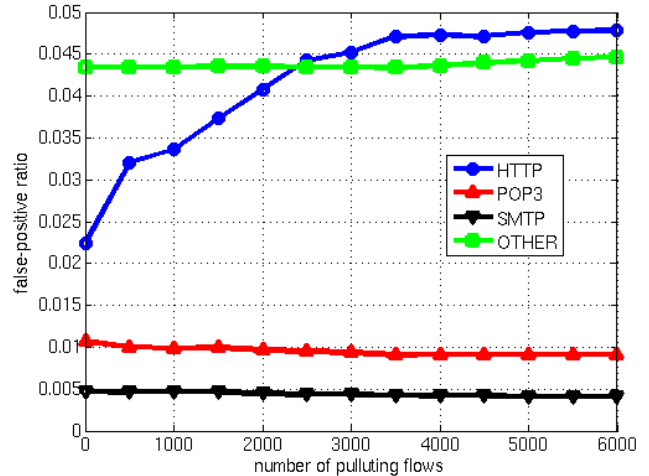


Fig. 4. False-positive ratio F_p when some FTP flows are inserted in the training set of HTTP protocol, using the classification technique based on probability product, with threshold $T = 10^{-15}$.

After inserting a growing number of polluting flows in a training set, we try to use the resulting fingerprint. We evaluate, for example, the experimental results obtained by inserting some FTP flows into the training set of HTTP. Figures 3 and 4 show the hit-ratio H_p and the false-positive ratio F_p of the four classes tested using the probability product algorithm. The hit-ratio of POP3 decreases from 97.95% to 92.45% when the 23% of the HTTP training set is incorrect.

On the contrary, when we classify through the method based on anomaly score, the performance gets worse in a more noticeable way, as the training set accuracy decreases (figures 5 and 6). The hit-ratio of the POP3 decreases from 95.19% to 36.08% when the 23% of the HTTP training set is composed of FTP flows. The greater fluctuation of the results is caused by the alterations of the threshold value, defined in the paragraph IV-B.1. The polluting flows modify the actual statistical distribution of the fingerprint, leading to higher values of the parameters μ and σ .

The example reported above shows the importance of building a correct training set to produce accurate fingerprints. Nevertheless, the algorithm based on probability product is more robust than the one based on anomaly score against this form of noise.

VI. CONCLUSIONS

In this paper we have proposed a statistical traffic classification mechanism. The key idea at the basis of the technique is the concept of fingerprint: by using only three network-layer properties, i.e., size, inter-arrival time and arrival order, we have suggested a compact and efficient way to represent the behavior of an application protocol at the IP-level. The usefulness of protocol fingerprints was tested with two simple classification algorithms. The results, albeit preliminary, seem promising and show that the technique works well on three popular application protocols. Since the classification is

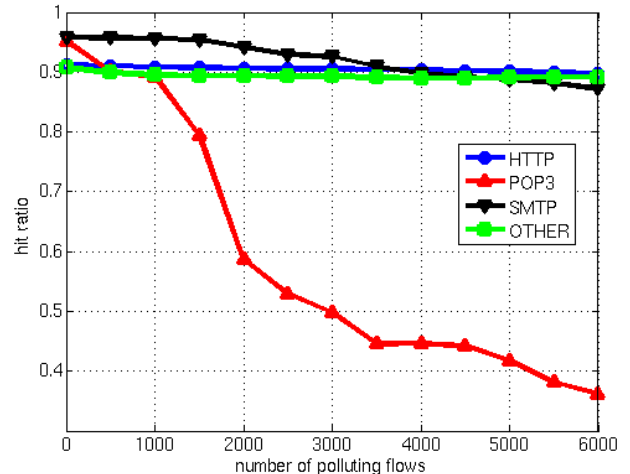


Fig. 5. Hit ratio H_p when some FTP flows are inserted in the training set of HTTP protocol, using the classification technique based on anomaly score.

performed using quantities that are readily available on every router, and the proposed algorithms are not computationally demanding, we think the technique is an effective solution for real-time classification and can be suitable for high-capacity links.

Our work is continuing in several directions. A first natural step will be the extension of the number of protocols being fingerprinted, to validate the model in a more general scenario. We have also discussed how to build accurate fingerprints, showing the importance of Gaussian filtering operation to counter what we called network noise and the effects of an incorrect training set pre-selection. This aspect is worthy of a more detailed study, since many supervised techniques rely on the correctness of training phase. Several improvements to our algorithms are also possible. Currently, we are working

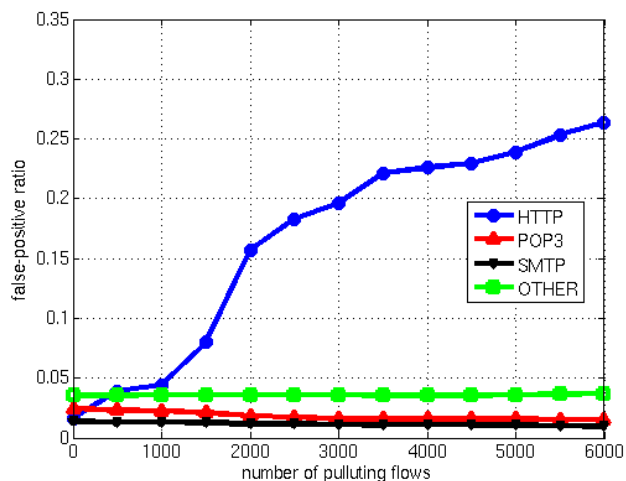


Fig. 6. False-positive ratio F_p when some FTP flows are inserted in the training set of HTTP protocol, using the classification technique based on anomaly score.

on the definition of more elaborate classification metrics that combine the information carried by F_{client} and F_{server} flows to enhance the ability of our classifier in the decision process.

REFERENCES

- [1] Pankaj Gupta and Nick McKeown. Packet classification on multiple fields. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 147–160, New York, NY, USA, 1999. ACM Press.
- [2] Florin Baboescu and George Varghese. Scalable packet classification. *IEEE/ACM Trans. Netw.*, 13(1):2–14, 2005.
- [3] M. Roesch. SNORT: Lightweight Intrusion Detection for Networks. In *LISA '99: Proceedings of the 13th USENIX Conference on Systems Administration*, pages 229–238, Seattle, WA, USA, November 1999.
- [4] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, 1999.
- [5] V. Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Trans. Netw.*, 2(4):316–336, 1994.
- [6] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Trans. Netw.*, 3(3):226–244, 1995.
- [7] A. Mena and J. Heidemann. An Empirical Study of Real Audio Traffic. In *Proceedings of the IEEE Infocom*, pages 101–110, Tel-Aviv, Israel, March 2000.
- [8] C. Dewes, A. Wichmann, and A. Feldmann. An analysis of Internet chat systems. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 51–64, Miami Beach, FL, USA, October 2003.
- [9] F. Hernández-Campos, F. Donelson Smith, K. Jeffay, and A. B. Nobel. Statistical Clustering of Internet Communications Patterns. In *Computing Science and Statistics*, volume 35, July 2003.
- [10] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow Clustering Using Machine Learning Techniques. In *Proceedings of the 5th Passive and Active Measurement Workshop (PAM 2004)*, pages 205–214, Antibes Juan-les-Pins, France, March 2004.
- [11] A. W. Moore and K. Papagiannaki. Toward the Accurate Identification of Network Applications. In *Proceedings of the 6th Passive and Active Measurement Workshop (PAM 2005)*, pages 41–54, October 2005.
- [12] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 50–60, Banff, Alberta, Canada, June 2005.
- [13] L. Bernaille, R. Teixeira, and K. Salamatian. Early Application Identification. In *The 2nd ADETTI/ISCTE CoNEXT Conference*, Lisboa, Portugal, December 2006.

- [14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *SIGCOMM'05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 229–240, Philadelphia, PA, USA, August 2005.
- [15] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic Classification through Simple Statistical Fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37(1):7–16, January 2007.
- [16] C. Trivedi, H. J. Trussel, A. Nilsson, and M-Y. Chow. Implicit Traffic Classification for Service Differentiation. Technical report, ITC Specialist Seminar, Wurzburg, Germany, July 2002.
- [17] L7 Filter. <http://l7-filter.sourceforge.net>.