Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 845–849
ISBN 978-83-60810-14-9
ISSN 1896-7094

Performance Evaluation of a Machine Learning Algorithm for Early Application Identification

Giacomo Verticale and Paolo Giacomazzi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Italy
{vertical,giacomaz}@elet.polimi.it

***Abstract*—The early identification of applications through the observation and fast analysis of the associated packet flows is a critical building block of intrusion detection and policy enforcement systems. The simple techniques currently used in practice, such as looking at the transport port numbers or at the application payload, are increasingly less effective for new applications using random port numbers and/or encryption. Therefore, there is increasing interest in machine learning techniques capable of identifying applications by examining features of the associated traffic process such as packet lengths and interarrival times. However, these techniques require that the classification algorithm is trained with examples of the traffic generated by the applications to be identified, possibly on the link where the the classifier will operate. In this paper we provide two new contributions. First, we apply the C4.5 decision tree algorithm to the problem of early application identification (i.e. looking at the first packets of the flow) and show that it has better performance than the algorithms proposed in the literature. Moreover, we evaluate the performance of the classifier when training is performed on a link different from the link where the classifier operates. This is an important issue, as a pre-trained portable classifier would greatly facilitate the deployment and management of the classification infrastructure.**

I. Introduction

THERE is a constant growth of new Internet applications using either random transport port numbers or reusing well-known ports registered to other applications. For example, peer-to-peer applications do not require the usage of well-known ports and some peer-to-peer applications, such as Skype, use port hopping. Moreover, recent applications frequently tunnel traffic through HTTP connections to seamlessly cross firewalls and NAT boxes. In these cases, the application generating a traffic flow cannot be identified by simply looking at the transport ports.

Therefore, there is a growing interest for alternative classification algorithms relying on features different from transport ports. The most widespread approach relies on the inspection of the packet payload and on the matching with signatures characteristics of the applications to be identified. This solution has the drawback of requiring computationally heavy elaborations that must be made at wire-speed; further it is not effective on cyphered traffic, such as that generated by applications secured through SSL.

Work on this paper has been supported by the Italian Ministry of Research project RECIPE

Other approaches use the information available in the non-cyphered IP packet header and complement these data with the analysis of statistical properties of the packet flow, considered as a random process. An example of this type of additional metrics is the packet interarrival time. With this approach, each traffic flow is associated with a set of features and, by examining the measured values of these features, a classifier yields the most likely application associated to the flow.

In [1] and [2] the authors use a Bayesian method to classify a traffic flow by using metrics such as flow duration, flow bandwidth, and statistics on packet sizes and interarrival times. The main drawback of this approach is that it is possible to classify only terminated flows. In [3] and [4], the authors introduce the idea of *Early Application Identification* and classify traffic flows by looking only at the lengths of the first packets in a flow; classification is performed by using clustering techniques such as K-means, Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM). The authors of [5] further extend the idea by including packet interarrival times in the feature set and by developing a new classification algorithm based on the Bayesian method. Finally, a preliminary comparison of general-purpose algorithms is given in [6], and the C4.5 algorithm [7] is indicated as one of the best performing. However, in [6] the authors study the classification of terminated flows and early application identification is not considered.

Using Machine Learning techniques for application identification still presents some challenges. In order to train the classifier it is necessary to collect a representative set of flow instances whose applications are known in advance. For example, by using offline payload inspection techniques to label the training data. The main issue with this approach is that the statistical properties of the traffic flows vary from link to link, therefore we expect that, if we train a classifier at a given link and operate it at another link, performance will be worse.

Another problem is that the flow must be observed for some time, in order to collect data enough to categorize it. In the case of early identification a trade-off must be found between speed and accuracy.

This paper presents several contributions along the line of assessing the capability of state-of-the-art Machine Learning techniques for fast traffic classification. The key contributions of the paper are the following.

First, we propose the C4.5 algorithm in the context of early application identification and we compare its accuracy to the results reported in [2], [4], and [5] and obtain superior results in terms of both true and false positives.

Second, we observe that the classification performance obtained by observing as few as 5 packets per direction is very similar to the performance obtained by observing all the packets in a flow, provided that suitable features, such as the actual packet lengths, are chosen for classification.

Third, we evaluate the performance of a classifier trained in a WAN link different than the one where it is operated. The performance is worse than using the classifier in the same link, but not to a large extent. Further, there are significant performance differences among protocols; in particular, the classification of HTTP protocol is very robust, whereas the Telnet and FTP traffic flows are more difficult to identify.

The paper is organized as follows. Section II explores the state-of-the-art of traffic classification using Machine Learning techniques. Section III discusses the data and the tools we use in the paper and explains the experimental setup. Section IV reports and discusses the results of our investigation. Finally, the last Section yields our concluding remarks.

II. Basic Concepts and Related Work

We identify two main application areas of Internet traffic classification: the separation of malicious traffic from good traffic and the identification of the application that generates the packets. Today, deep packet inspection boxes are used for both objectives. For example SNORT [8] for identification of malicious traffic and l7filter [9] for application identification. Both packages perform per-flow classification by comparing the payload of the first packets in the flow with a set of pre-configured signatures, which are generally human-made.

Deep packet inspection is not adequate when traffic is cyphered or when high-speed links are considered, therefore there is a growing research interest for traffic classification considering only packet lengths, interarrival times and any other information available in the protocol headers. One of the first proposals of application identification through the examination of the traffic process is made in [10], where the authors point out that RealAudio traffic exhibits a behavior significantly different from that of telnet, HTTP or FTP traffic, in particular in terms of flow duration and regularity of packet lengths and interarrival times. The authors, however, do not propose a specific classification algorithm.

A more recent work proposes to use clustering techniques for mapping a traffic flow to a QoS class [11]. The authors find that the most significant metrics are the average packet size and the flow duration. The main problem with these metrics is that classification can be performed only when the flow is already finished, limiting the practical utility of this technique.

The authors of [12] apply unsupervised Bayesian techniques to the problem of application identification and use a feature selection technique to to find the optimal set of attributes. The authors find that the most influential attributes are the mean and the variance of the packet lengths, the total amount of traffic in the flow, and the flow duration. Also the mean interarrival time has some influence. As for [11], not all these metrics are suitable for early application identification.

In [1], the authors propose a supervised Bayesian classifier and also use some TCP-specific metrics such as the count of all the packets with the PUSH bit set. However, only semantically complete TCP connections are considered.

Paper [13] proposes how to use behavior analysis to augment the efficacy of automatic classification techniques. Their BLINC framework considers attributes such as the social and functional role of the hosts, which require long observation times to elaborate and, therefore, making BLINC a valuable solution for off-line classification.

In [6], the authors compare the performance of five general purpose supervised Machine Learning algorithms, showing that the C4.5 tree based algorithm provides a good trade-off between classification accuracy and speed. The authors also consider two feature reduction schemes and identify two feature sets that show similar performance. In one feature set packet length statistics are predominant, while in the second feature set there is a balance of packet lengths and interarrival times. The authors of [6] have also developed a set of tools [14] which automates the process of collecting packet traffic, reconstructing the flows, computing metrics and invoking the Machine Learning algorithms.

The authors of [3] study the problem of Early Application Identification and show that it is enough to know the length of the first four or five packets to achieve promising classification results. The authors also compare three supervised clustering techniques.

Finally, the authors of [5] further refine the idea, by considering also packet interarrival times and developing a novel classification algorithm, but do not compare it to other state-of-the-art classification algorithms.

III. Experimental Setup

In our research work we have set up a laboratory to experiment in practice the classification algorithms. We have used publicly available software and data. Packet traces representative of WAN traffic have been retrieved from the NLANR traffic archive [15]. In particular we have used the `auckland-vi-20010611`, `auckland-vi-20010612`, and `nzix-ii-20000706`. The two `auckland` traces were collected at the same network link; in the paper, we will use the first trace to train the Machine Learning classifier, whereas the second trace will be used to assess the classification performance. We will refer to the first trace as *Samplepoint A (training)* and to the second trace as *Samplepoint A (test)*. The third trace was collected at a different link, and it will be referred to as *Samplepoint B (test)*.

In order to group packets in flows and to elaborate per-flow metrics, we have used the `NetMate`[16] software, extended with the `NetAI`[14] patch. This way, we obtain the features reported in Table I, which we will refer to as the two *Standard* sets. We have isolated the *Standard 2* features because they

TABLE I
THE *Standard* FEATURES

Standard 1 features
Min, max, mean and std. deviation of the packet lengths in the forward direction.
Min, max, mean and std. deviation of the packet lengths in the backward direction.
Min, max, mean and std. deviation of the packet interarrival times in the forward direction.
Min, max, mean and std. deviation of the packet interarrival times in the backward direction.
Transport protocol number.
Total number of TCP URG and PUSH flags in the forward direction.
Total number of TCP URG and PUSH flags in the backward direction.
Standard 2 features
Flow duration.
Total number of bytes and of packets in the forward direction.
Total number of bytes and of packets in the backward direction.

TABLE II
THE *Extended* FEATURES

Extended 1 features
Lengths of the first 3 packets in the forward direction
Lengths of the first 3 packets in the backward direction
Interarrival times of the first 3 packets in the forward direction
Interarrival times of the first 3 packets in the backward direction
Extended 2 features
Lengths of the first 5 packets in the forward direction
Lengths of the first 5 packets in the backward direction
Interarrival times of the first 5 packets in the forward direction
Interarrival times of the first 5 packets in the backward direction

are not meaningful when only the first packets of the flow are considered.

We have also implemented a new patch for `NetMate` to collect the feature sets proposed in [3] and [5], referred to as the *Extended* feature sets (Table II). These extended sets are the actual packet lengths and the interarrival times of the first packets of the flows. We have grouped the features in two sets of increasing size. The *Extended 1* feature set includes the packet lengths of the first three packets of a flow in each direction and the two interarrival times between them in each direction, for a total of 10 features. The *Extended 2* feature set includes all the packet lengths and interarrival times involving the first 5 packets in a flow.

As a concluding remark, we note that the `NetMate` software defines a flow as the packets belonging to a single TCP connection, in case TCP is the transport protocol. In case UDP is used, the flow is defined as all the packets with the same IP addresses and UDP port numbers and considered finished when no packets have arrived for 600 s. From all the flows available in the data sets, we have randomly chosen 4000 flows for each of the five application protocols considered in this paper. For some protocols there were fewer flows than required, so we used all the available ones. The total number of flows chosen in each data set is reported in Table III.

Then, we have processed the selected flows and flow features by using the `Weka` [17] machine learning suite to train the classifier and perform the validation tests. We have preliminarly evaluated the same algorithms as in [6] and concluded that the C4.5 algorithm [7] gives the best results, so we will show only the results obtained with that algorithm. In fact, the C4.5 algorithm well suits problems in which several attributes assume discrete values and when training data is noisy, which is common in large data sets. To ease a comparison, we performed our assessment by using the same 5 applications as in [6], i.e. FTP-data, Telnet, SMTP, DNS, and HTTP. For the training and for the validation we assumed that the flow category label is the server well-known port number.

TABLE III
SIZE OF THE DATA SETS

Data Set	Number of Flows	Flows with at least 5 packets per direction
Samplepoint A (training)	15606	15300
Samplepoint A (test)	15803	15545
Samplepoint B (test)	13335	12788

IV. EXPERIMENTAL RESULTS

In this section, we assess the performance of Machine Learning techniques for traffic classification and substantiate the following findings:

- the C4.5 algorithm is superior to the algorithms proposed in [2], [4], and [5];
- the classification accuracy obtained with five packets per direction is similar to the accuracy obtained observing all the packets in the flow even if the trained classifier is used on a different link, but only if the Extended Features are used;
- the performance loss measured when the trained classifier is used on a different link is mainly due to a few protocols, whereas other protocols, in particular HTTP, show minimal performance differences.

A. C4.5 Algorithm for Early Application Identification

As a first result, we show that the the state-of-the art C4.5 algorithm [7] improves the classification performance over the algorithm proposed in the literature. As performance metrics we use the True Positive Rate and the False Positive Rate, defined as:

$$\mathrm{TPR}(i) = \frac{e_i}{E_i}$$
$$\mathrm{FPR}(i) = \frac{\bar{e}_i}{e_i + \bar{e}_i}$$

where E_i is the number of instances of protocol i in the evaluation set, e_i is the number of instances of protocol i correctly classified as i, and $\overline{e_i}$ is the number of instances of other protocols incorrectly classified as i.

In Table IV, we compare the True Positive Rates as reported in [2], [4], [5], as well as the test results obtained using the C4.5 classifier trained on the *Samplepoint A (training)* data and tested on the *Samplepoint A (test)* data. In our experiment we used the *Standard-1* plus *Extended-2* feature sets. In Table V, we make a similar comparison considering the False Positive Rates. In reporting results from other papers, we have considered only the classes considered in at least two papers.

TABLE IV
COMPARISON OF THE TRUE POSITIVE RATES

Protocol	[2]	[4]	[5]	Our solution
HTTP	89.2%	96.2%	91.8%	99.7%
SMTP	*97.2%	90.1%	94.5%	98.6%
POP3	*97.2%	93.4%	94.6%	N/A
FTP	97.9%	92.4%	N/A	94.8%

*In [2] STMP and POP3 form a single class

TABLE V
COMPARISON OF THE FALSE POSITIVE RATES

Protocol	[2]	[4]	[5]	Our solution
HTTP	10.4%	1.3%	6.4%	0.1%
SMTP	*2.3%	0.1%	3.1%	1.4%
POP3	*2.3%	0.7%	3.1%	N/A
FTP	1.8%	0.4%	N/A	0.5%

*In [2] STMP and POP3 form a single class

Our solution shows higher true positive rates than the other solutions for HTTP and SMTP and is inferior to [2] only for the FTP class, where, however, our solution shows a much lower FPR. Our proposal shows false positive rates superior to [2] and [5]. Compared to [4], our solution performs similarly for the FTP class and worse only for the SMTP class, where our solution scores a much higher TPR.

Finding a good trade-off between TPR and FPR is a common challenge when using Machine Learning techniques. In the context of internet traffic classifications for monitoring and security purposes, it is important to achieve a low FPR. From Tables IV and V, we conclude that the C4.5 classifier shows an FPR performance comparable to the best other solution ([4]), but with a higher TPR. Therefore, in the rest of the paper we will consider only the C4.5 algorithm.

B. Effect of the Observation Horizon and of the Feature Sets

Table VI reports the percentage of incorrectly classified traffic flows for different observation horizons and different feature sets. Results are given for three data sets. The column labeled *Samplepoint A(training)* contains the percentage of flows in the training set equivocated by the classifier, which gives us a lower bound on the classification error that we expect at run time. In all the cases, when we consider at least the first 5 packets per direction, the residual error is lower than 0.3%, and is about 0.5% if we consider only the first 3 packets per direction.

In Table VI the column labeled *Samplepoint A (test)* gives the observed percentage of incorrectly classified packets when the trained classifier is used on a different data set collected on the same WAN network link. Again, when we consider the first 5 or more packets per direction, the error does not change and settles at about 1.5%. On the other hand, if we consider only the first 3 packets, the error is more than 3.5%. This trend is in line with [3] and [5], which indicate in 4 or 5 packets the ideal observation horizon.

The last column gives the classification error when the trained classifier is used with a data set collected on a completely different WAN network link. There are no data

TABLE VI
INCORRECTLY CLASSIFIED INSTANCES USING THE *Standard* AND *Extended* FEATURES

		Samplepoint		
Packets per direction	Feature set	A (training)	A (test)	B (test)
3	std1	0.48%	3.52%	13.54%
3	std1 + ext1	0.44%	3.22%	13.06%
5	std1	0.27%	1.55%	6.01%
5	std1 + ext2	0.22%	1.80%	4.10%
10	std1	0.21%	1.66%	6.01%
all	std1 + std2	0.21%	1.45%	4.45%

TABLE VII
ACCURACY BY CLASS WHEN USING THE *Standard 1* AND *Extended 2* FEATURES

	Samplepoint A		Samplepoint B	
Class	TPR	FPR	TPR	FPR
DNS	100%	0.4%	99%	0.1%
FTP (data)	95%	0.5%	77%	1.4%
Telnet	92%	0.1%	84%	0.4%
SMTP	98%	1.4%	95%	3.0%
HTTP	99%	0.1%	99%	0.2%

in the literature about this problem, but we expect that the different statistics of the collected features hamper the work of the classifier. Results show that the error grows more than 3.5 times, if we consider only the standard set and 3, 5, or 10 packets. Slightly better results are obtained if all the packets of the flow are considered, with an error about 4.5%, which is only 3 times larger than in the *Samplepoint A(test)* case. On the other hand, the Extended Set greatly improves the robustness of the classifier, bringing the error from about 6% down to about 4%, which is about 2.5 times the result obtained in the *Samplepoint A(test)* case and is better than the result obtained with the standard set and observing all the packets in the flow. Further, these apparently non promising results should not make us abandon the idea of using this kind of classifiers because, as we show later, the vast majority of the incorrectly classified packets come from a limited amount of protocols.

C. Per-protocol Classification Accuracy

Finally, Table VII shows the classification performance obtained on test data, broke down by protocol. All the results are obtained considering 5 packets per direction and the *Standard* plus the *Extended 2* metrics. As in the experiment above, the classifier was trained with the *Samplepoint A (training)* dataset and operated with the *Samplepoint A (test)* and *B (test)* datasets.

As already observed in [3] and [5], there are significant performance differences among protocols. Our results, however, shed light on what protocols fingerprints are most likely to be preserved from link to link.

In *Samplepoint A*, the true positives are always reasonably good, with only FTP and Telnet equal to or below 95%. Porting the classifier to *Samplepoint B* worsens the performance of these two protocols, whose TPR drops below 85%, while the other protocols maintain their good performance. Regarding the false positives, the only class with FPR larger than 1% is

SMTP in the *Samplepoint A*, whereas, in the *Samplepoint B* case, both SMTP and FTP have a large FPR.

At the light of these results, we remark that, when porting a trained classifier, those protocols that had a lower TPR tend to decrease their TPR and those protocols that had a higher FPR tend to increase it. However, there can be anomalies, for example in our experiment DNS and FTP had similar FPR in the first link and very different FPR in the second.

V. Conclusions

In our research work we have experimented, with a thorough laboratory activity, the behavior of machine learning algorithms for the classification of applications by examining the traffic flow process. We have concentrated our attention to the early application identification, therefore, we have examined extended feature sets including lengths and interarrival times of the first few packets of flows and we have matched their performance against that of standard feature sets requiring the examination of the entire traffic flow.

We have examined the behavior of the C4.5 decision tree algorithm with extended feature sets and we have determined, as a novel result, that this algorithm performs very well for early application identification.

We have proceeded by studying a problem so far left open in the related research, that is, the portability of a trained Machine Learning classifier on a link different from that used for training. The possibility of porting pre-trained classifier would be very appealing for practical implementations. We have determined that the performance of a trained classifier moved to another link worsens, but the degradation seems to be concentrated on specific protocols such as FTP and Telnet, while other protocols such as HTTP and DNS are recognized effectively even by a moved classifier.

Our current work concentrates on devising feature sets capable of improving the portability of trained classifiers to different links.

References

[1] 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*. New York, NY, USA: ACM, 2005, pp. 50–60.

[2] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *Neural Networks, IEEE Transactions on*, vol. 18, no. 1, pp. 223–239, Jan. 2007.

[3] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, 2006.

[4] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *The 2nd ADETTI/ISCTE CoNEXT Conference*, Dec. 2006.

[5] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, 2007.

[6] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, 2006.

[7] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[8] "SNORT," http://www.snort.org/.

[9] "L7filter: Application Layer Packet Classifier for Linux," http://l7-filter.sourceforge.net/.

[10] J. Mena, A.; Heidemann, "An empirical study of real audio traffic," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 101–110 vol.1, 2000.

[11] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 135–148.

[12] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," *lcn*, vol. 0, pp. 250–257, 2005.

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

[14] "netAI, Network Traffic based Application Identification," http://caia.swin.edu.au/urp/dstc/netai/.

[15] "NLANR traces," http://pma.nlanr.net/Special/.

[16] "NetMate Meter," http://sourceforge.net/projects/netmate-meter/.

[17] I. H. Witten and E. Frank, *Data mining: practical machine learning tools and techniques with Java implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000.