Using per-Source Measurements to Improve Performance of Internet Traffic Classification

Stefano Bregni, *Senior Member, IEEE*, Diego Lucerna, *Student Member, IEEE*
Cristina Rottondi, Giacomo Verticale, *Member, IEEE*

Politecnico di Milano, Dept. of Electronics and Information, Piazza Leonardo Da Vinci 32, 20133 Milano, ITALY
Tel.: +39-02-2399.3503 – Fax: +39-02-2399.3413 – E-mail: {bregni, lucerna, vertical}@elet.polimi.it

***Abstract* — Obfuscated and encrypted protocols hinder traffic classification by classical techniques such as port analysis or deep packet inspection. Therefore, there is growing interest for classification algorithms based on statistical analysis of the length of the first packets of flows. Most classifiers proposed in literature are based on machine learning techniques and consider each flow independently of previous source activity (per-flow analysis). In this paper, we propose to use specific per-source information to improve classification accuracy: the sequence of starting times of flows generated by single sources may be analyzed along time to estimate peculiar statistical parameters, in our case the exponent α of the power law $f^{-\alpha}$ that approximates the PSD of their counting process. In our method, this measurement is used to train a classifier in addition to the lengths of the first packets of the flows. In our experiments, considering this additional per-source information yielded the same accuracy as using only per-flow data, but observing fewer packets in each flow and thus allowing a quicker response. For the proposed classifier, we report performance evaluation results obtained on sets of Internet traffic traces collected in three sites.**

***Index Terms* — Communication system traffic, Internet, long-range dependence, traffic measurement (communication).**

I. Introduction

The goal of Internet traffic classification is associating a sequence of packets between two hosts on a transport port pair (i.e., a *flow*) to the source application. The identification of applications may be useful, for example, for link usage analysis, for management of Quality of Service (QoS) or for blocking traffic flows not conforming to local policies.

Common techniques used for Internet traffic classification are based on the packet payload inspection or on well-known transport protocol port numbers. However, newer Internet applications and protocols may use random or non-standard port numbers, or employ packet encryption or traffic obfuscation, making these techniques ineffective. Therefore, recent studies consider using statistical analysis to assist traffic identification and classification by way of machine-learning techniques.

A comprehensive survey of machine learning (ML) traffic classification techniques is provided in [1] and [2]. Several papers studied the performance of various classification algorithms, based on statistical analysis of single flows.

The Nearest Neighbours (NN), Linear Discriminate Analysis (LDA) and the Quadratic Discriminant Analysis (QDA) algorithms were proposed in [3] to identify the QoS class of different applications. The authors identified a list of possible features, calculated over the entire flow duration, and claimed to obtain a classification error ranging from 2.5% to 12.6%.

The application of a Bayesian neural network was proposed in [4]. The classification accuracy reached 99% when the training and the test data were collected on the same day and 95% when the test data were collected eight months after the training data. A Protocol Fingerprinting technique was proposed in [5], based on observing packet lengths, inter-arrival times, and packet arrival order. By classifying three applications (HTTP, SMTP, POP3), an accuracy greater than 91% was obtained was obtained by observing as few as 4 or 5 packets.

A decision tree algorithm to classify Internet traffic was proposed in [6]. The traffic features considered are the lengths of the first 5 packets in both directions and their inter-arrival times. Accuracy between 92% and 99% was achieved.

Traffic classification can be done offline or in real time. When in real time, classification is required as earliest as possible, i.e. by looking at as few packets as possible, in order to have the smallest delay to get the classification result. On the other hand, the classification accuracy improves with the number of observed packets. Thus, a trade-off between classification delay and accuracy must be sought.

Our work is motivated by the consideration that different flows, originated by the same host, are likely to be started by the same application or by a limited number of applications running on the same host. A vast literature on traffic measurement reports that traffic statistical characteristics depend on the generating application [7][8]. Therefore, observing the source activity along time can reveal some peculiar behaviour.

Moreover, it has been shown that traffic data series often exhibit Long-Range Dependence (LRD), i.e. an asymptotic power-law decrease of their Power Spectral Density (PSD) as $\sim f^{-\alpha}$ (for $f \to 0$) or, equivalently, of their autocovariance.

Among early works, power-law PSD was identified in LAN packet traffic in [7]. Authors concluded that was caused by the nature of the data transfer applications.

Power-law PSD at packet level was identified also in WAN traffic [9]. Authors conducted some investigation also at connection level, concluding that Telnet and FTP control connections were well-modelled as Poisson processes, while FTP data connections, NNTP, and SMTP were not.

Web-browsing traffic was studied in [10], by measuring the sequence of file requests performed during each session (i.e., one execution of the web-browsing application), finding that the reason of the power law lies in the long-tailed distributions of the requested files and of the users' "think-times".

978-1-4244-7173-7/10/$26.00 © 2010 IEEE

We decided to analyze the sequence of start times of flows generated by single sources towards specific destination ports, estimating the exponent α of the power law $f^{-\alpha}$ that approximates the PSD of their counting process. In our method, this measurement is used to train a classifier in addition to the lengths of first packets of each flow.

In our experiments, considering this additional per-source information has yielded the same accuracy as using only per-flow data, but observing fewer packets in each flow and thus allowing a quicker response. Indeed, using this per-source information enhances classification performance, but requires no additional delay for data processing.

It is worth noting that, if the source exhibits low activity, this per-source statistics may be not accurate. Therefore, we propose a novel method (*tandem classifier*), which takes into account also per-source information only when this is based on enough data. Otherwise, only packet lengths are considered.

The paper is organized as follows. Section II introduces basic concepts on the estimation of the power-law PSD of LRD series using the Modified Allan VARiance (MAVAR) and on the Support Vector Machines (SVMs) and Random Forest (RF) classification techniques. Section III presents the proposed tandem classifier and details what per-source information is used to enhance classification performance. Section IV reports performance evaluation results of the proposed tandem classifier obtained on sets of Internet traffic traces collected in three sites. Section V draws some conclusions.

II. Overview of Background Concepts

In this section, we summarize our technique to estimate the α parameter of time series supposed with PSD $\sim f^{-\alpha}$, based on MAVAR, and two traffic classification methods, viz. Support Vector Machines (SVM) and Random Forests (RF).

A. Estimating the α Parameter of LRD Series Using MAVAR

For estimating the α parameter of traffic series supposed with PSD $\sim f^{-\alpha}$, we use the Modified Allan Variance (MAVAR), due to its superior spectral sensitivity and accuracy in LRD parameter estimation, coupled with excellent robustness against data non-stationarity (e.g., drift and steps) [11]. MAVAR is a well-known time-domain quantity, originally conceived in 1981 for frequency stability characterization of oscillators [12]−[16]. Here, only its main properties will be recalled.

Given a finite set of N samples $\{x_k\}$ of a signal $x(t)$, evenly spaced by sampling period τ_0, MAVAR can be estimated using the ITU-T standard estimator [16][17]:

$$\mathrm{Mod}\,\sigma_y^2(\tau) = \frac{\sum_{j=1}^{N-3n+1}\left[\sum_{i=j}^{n+j-1}\left(x_{i+2n}-2x_{i+n}+x_i\right)\right]^2}{2n^4\tau_0^2\left(N-3n+1\right)} \qquad (1)$$

where the observation interval is $\tau = n\tau_0$ and $n = 1, 2, \ldots, \lfloor N/3 \rfloor$.

MAVAR is a kind of variance of the second difference of input data, including an internal average over n adjacent samples. A recursive algorithm for fast computation of this estimator exists [16], which cuts the number of operations needed to compute it for *all* values of n to $O(N^2)$ instead of $O(N^3)$, or to $O(N \log N)$ if MAVAR($n\tau_0$) is computed only for $n = 2^k$ ($k = 0, 1, \ldots, \lfloor \log_2 N \rfloor$) as in wavelet logscale diagrams [11].

Fractional noise with one-sided PSD $\sim 1/f^{\alpha}$, for $0 \le \alpha \le 4$, has been revealed in practical measurements of various phenomena, such as phase noise of precision oscillators [16] and Internet traffic. Under this power-law model of input data and in the whole range of MAVAR convergence $0 \le \alpha < 5$, MAVAR itself is found following the simple power law (ideally asymptotically for $n \to \infty$, $n\tau_0 = \tau$, but in practice for $n > 4$)

$$\mathrm{Mod}\,\sigma_y^2(\tau) \sim A_\mu \tau^\mu \,, \quad \mu = -3 + \alpha \qquad (2).$$

Therefore, the slope μ of Mod $\sigma_y^2(\tau)$ in a log-log plot can be estimated (e.g., by linear regression) to yield the exponent $\alpha = 3 + \mu$ of the fractional noise that is dominant in input data. This estimate of α was demonstrated to be very accurate, unbiased and robust against non-stationarity in data analyzed (viz. drifts, periodic trends and steps) [11].

B. The SVM and RF Classification Algorithms

Each flow is characterized by a set of per-flow features (the size of the first packets) and possibly of per-source features (the α parameter estimated from past source history). We adopted two state-of-the-art supervised machine-learning algorithms to map instances of traffic flows into traffic classes, viz. Support Vector Machines (SVMs) and Random Forest (RF).

SVMs are a class of supervised learning technique initially conceived for binary classification, but later extended to deal with multi-class classification. They have good generalization ability and are known to deal well with outliers, both in the training data and in the data to classify [18].

In this paper, we used the implementation of SVM provided by the *libsvm* library [19]. In particular, we used the C-SVC algorithm with an RBF kernel and, for the multiclass extension, the pairwise SVM technique. In order to select the machine parameters C and σ, we performed a grid search and, for each experiment, we performed a 10-fold cross validation on a random subset of the data. Then, we picked the parameters that yielded the least error rate.

RFs are a class of decision algorithms based on a wide set of simple classification trees, each trained on a random subset of the data and of the features. The final classification decision is taken by majority vote. These classifiers are easy to train and are known to be very robust to unbiased noise in the data. In our work, we used the *R* package *RandomForest* [20] and trained forests made of 500 trees, which are large enough to minimize the classification error on training data.

III. Traffic Classification Methodology

In this section, we outline our technique for traffic classification, highlighting what per-flow and per-source attributes are taken into account and how per-source features are used to improve classification performance.

A. Statistical Analysis of Flow Starting Time Series

If the transport protocol is TCP, a flow is defined as the set of packets belonging to a single TCP connection. In case of UDP, a flow is defined as the set of packets with same source

and destination pairs of IP addresses and UDP port numbers and with no interruptions longer than threshold T_{idle}. If a packet with same port/address pairs arrives after a silence longer than T_{idle}, the previous flow is considered finished and the packet is considered the first packet in a new flow with same port/address pairs. In this paper, we set T_{idle} = 600 s, which is the default value used by the connection tracking algorithm of the widely used *iptables* packet filter [21].

We consider flows in the direction from the client, which initiates the flow, to the server. On each flow, we record the lengths of the first n packets (the per-flow data used for classifying traffic flows). We also record the timestamp of the first packet in each flow, indicating the flow starting time.

From the timestamps of the first packet of each flow, we get the discrete traffic sequences $x_{i,p}(k)$ (for $k = 1, ..., N$), which count the flows started from the i source, associated to the p transport port, in the k-th time interval. In our measurements, we set the time interval to $\tau_0 = 1$ s, while data acquisition lasted from 30 minutes to a few hours ($N = 10^2 \div 10^4$).

Each time a new flow begins, the sequence $x_{i,p}(k)$ is updated and its new statistical values are computed, namely:

- the α parameter, assuming that its PSD can be approximated as $\sim f^{-\alpha}$, as defined in Sec. II.A and omitting the last octave of MAVAR($n\tau_0$) points ($\alpha = 0$ with Poisson traffic);
- the Index of Dispersion for Counts (*IDC*)

$$IDC = \frac{\sigma_x^2}{\mu_x} \tag{3}$$

  where μ_x and σ_x^2 are the estimated values of the mean and variance of the sequence (*IDC* = 1 with Poisson traffic);
- the skewness

$$SKEW = \frac{1}{N} = \sum_{k=1}^{N} \left(\frac{x_k - \mu_x}{\sigma_x} \right)^3 \tag{4}$$

  where N is the total number of time intervals ($SKEW = \lambda^{-1/2}$ with Poisson traffic, where λ is the mean flow start rate);
- the total number of flows X since the beginning of the measurement period.

B. The Tandem Classifier

The scheme of the proposed *tandem classifier* is shown in Fig. 1. Whenever the first packet of a new flow is observed, the relevant per-source statistical values ($\alpha_{i,p}$, $IDC_{i,p}$, $SKEW_{i,p}$, $X_{i,p}$) are updated and a preliminary decision is made (*per-source meter* and *pre-classifier* blocks).

If the source IP address has generated less than ξ flows towards that port since the beginning, the classification capability of the per-source parameters is considered not sufficient. Then, the flow is classified as soon as n_1 packets are observed from client to server (*N1 classifier*). The values considered for classification are the lengths $l_1, l_2, ..., l_{n1}$ of the collected packets. If the flow finishes before n_1 packets are observed, classification is performed considering the missing packets as having length -1 (we call this *post-mortem* classification).

If at least ξ flows have been generated, the measured per-source parameters are considered trustable for classification.

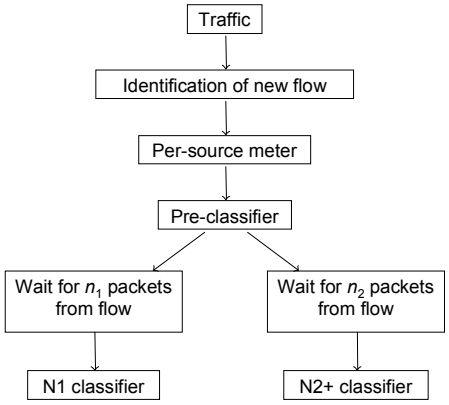

Fig. 1: Block diagram of the tandem classifier.

Again, the flow is classified as soon as n_2 packets are observed from client to server (*N2+ classifier*), considering the lengths $l_1, l_2, ..., l_{n2}$ of the collected packets *together with the per-source parameters* $\alpha_{i,p}$, $IDC_{i,p}$, $SKEW_{i,p}$, $X_{i,p}$ collected upon flow pre-classification. If the flow ends before n_2 packets are observed, *post-mortem* classification is performed considering missing packets as having length -1.

The threshold ξ is a parameter of the tandem classifier and can be adjusted empirically. In our experiments, as reported in Section IV, values in range 50 to 100 yielded a good balance between classification delay and accuracy.

The N1 and N2+ classifiers are trained separately, based on the available labelled traffic data. We extract 1000 labelled flows per class and, for each flow, we count the number of flows previously generated by the same source. If this is smaller than ξ, we use the flow to train the N1 classifier. Otherwise, we calculate the per-source parameters and use the flow to train the N2+ classifier. For both classifiers, we consider using either the SVM or the RF algorithm.

Classifiers similar to N1 are common in literature, while N2+-type classifiers have not been considered yet, to the best of our knowledge. However, using the N2+ classifier alone would result in bad performance, because per-source data are not trustworthy when the source is young. Our tandem classifier uses the per-source information when based on enough data or, alternatively, exploits only the standard flow attributes when the confidence of per-source information is low.

By choosing $n_2 < n_1$, the tandem classifier decides by observing, on the average, fewer packets per flow than the N1 classifier alone, resulting in shorter classification delays and keeping fewer flows in a pending-decision state. If ξ is low, the number of required packets per flow approaches n_2. However, high classification error rate is expected, because inaccurate estimates are used. Conversely, if ξ is large, the number or required packets approaches n_1, but fewer errors are expected.

IV. Evaluation of Classification Accuracy

We compared the classification accuracy of the tandem classifier, with $n_1 = 5$ and $n_2 = 3$, to the performance of the N1 classifier alone, with either $n_1 = 3$ or $n_1 = 5$. Results presented in this section show that, by choosing ξ appropriately, the tandem classifier achieves better classification accuracy than the

N1 classifier alone with $n_1 = 3$, while it achieves similar accuracy as the one with $n_1 = 5$ but with shorter delay. Using fewer than 3 packets would not give information on TCP flows, because only the connection setup would be considered, while considering more than 5 would not give significant performance improvements [5].

A. Traffic Data Used and Experimental Procedure

Given a packet traffic trace, we use *NetMate Meter* [22] and *netAI (Network Traffic based Application Identification)* [23] to group packets in traffic flows and to calculate per-flow metrics. For training the classifier, we also collect the destination port number for each flow under the assumption that, in the considered traffic traces, well-known ports are a truthful indicator of the application that generated the packet flow.

Following [24], we discriminated 8 different applications based on port numbers in the couples of traffic traces (training trace and test trace) named Auckland-VI, Leipzig-II and Nzix-II [25], as listed in Tables 1 and 2. For each application (class), we sampled randomly 1000 flows from the training trace and 1000 flows from the test trace (if less flows are available in traces for some application, we considered all those available). Then two sub-traces are created. We trained the classifiers on the first sub-trace and tested them on the second one. This procedure was repeated 10 times and results were averaged.

B. Results of the Experiments

Among the various tests that we executed on all traffic traces, we report the results obtained on the Auckland traces.

In Figs. 2 and 3, we plot the classification error rate versus the threshold ξ for the proposed classifiers based on RF and SVM, respectively. On this data set, the RF and SVM N1 classifiers with $n_1 = 3$ packets perform poorly (error rate about 47%). On the other hand, with $n_1 = 5$, the RF and SVM N1 classifiers achieve an error rate of about 12.5%.

Table 1: Port numbers and protocol associated to applications (traffic classes) in the Auckland-VI, NZIX-II and Leipzig-II traces.

Application	Port/Protocol
FTP	21/tcp
Telnet	23/tcp
SMTP	25/tcp
DNS (tcp)	53/tcp
DNS (udp)	53/udp
HTTP	80/tcp
AOL	5190/tcp
Half-Life	27015/udp

Table 2: Recording times of traces used for training and testing, respectively.

Trace	Date	Duration
Auckland-VI	11 June 2001 12 June 2001	from 12:00 to 18:00 from 12:00 to 18:00
Nzix-II	20 July 2000 20 July 2000	from 06:00 to 12:00 from 12:00 to 18:00
Leipzig-II	21 December 2003 21 December 2003	from 12:14 to 15:00 from 15:00 to 21:00

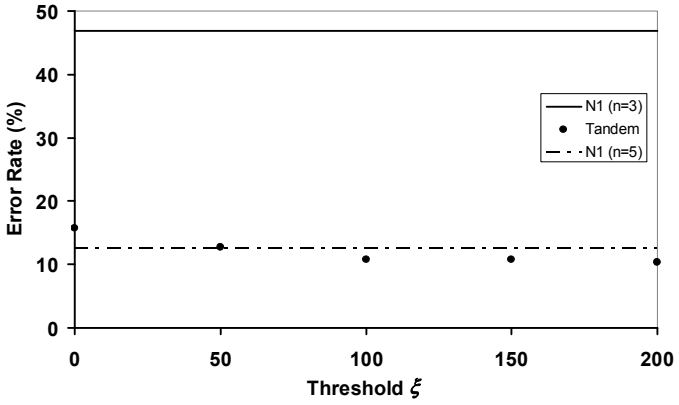

Fig. 2: Classification accuracy on Auckland traces (RF methods).

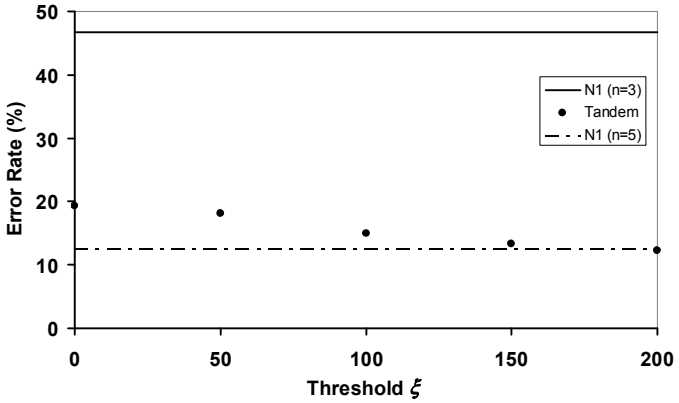

Fig. 3: Classification accuracy on Auckland traces (SVM methods).

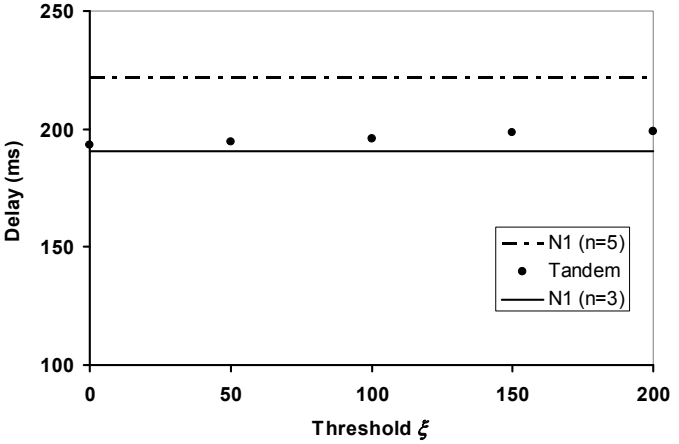

Fig. 4: Classification delay on Auckland traces.

In Fig. 4, we show the average time (delay) needed by classifiers, to collect the number of packets required, versus the threshold ξ. The N1 classifiers with $n_1 = 5$ take their decision after, on the average, 222 ms. Conversely, the N1 classifiers with $n_1 = 3$ need only 190 ms delay.

The tandem technique strikes a balance between the error rate and the delay. With $\xi = 0$, most of the flows are classified considering $n_2 = 3$ packets plus the per-source attributes (if they can be computed). With larger ξ, fewer flows take advantage of the per-source attributes, so more flows are classified

using the N1 scheme with $n_1 = 5$ packets. On the other hand, the per-source attributes are more reliable and the classification accuracy is better. Figs. 2 and 3 show that the error rate drops to as low as 14.9% and 10.8%, using SVM or RF respectively, with $\xi = 100$.

With larger threshold ξ, the drawback is that more flows are classified using more packets and thus the classification delay is larger. Fig. 4 shows that, when $\xi = 100$, the classification delay is about 195 ms. This delay is only 5 ms longer than the delay of the N1 classifiers with $n_1 = 3$, which have much worse classification accuracy, and is 27 ms lower than the delay of the N1 schemes with $n_1 = 5$. Moreover, the SVM tandem approach exhibits a classification error rate that is only 2% greater than the N1 ($n_1 = 5$) counterpart when threshold is $\xi > 100$, while the RF tandem scheme yields even better results than the corresponding N1 approach ($n_1 = 5$).

The same tests were carried out also on the NZIX and Leipzig traces. The behavior of the classification error rate and delay versus the threshold ξ, for all classifiers considered, resulted similar as on Auckland traces. For the sake of conciseness, in Tables 3 and 4 we report the classification error rate and delay obtained with setting the threshold as $\xi = 100$.

V. Conclusions

In this paper, we proposed to collect per-source information to improve the performance of Internet traffic classification, otherwise based on per-flow statistical analysis. We focus on the problem of early application identification, which requires finding a balance between the classification accuracy and the number of packets required by the classifier.

In our method, we record the sequence of starting times of flows from each source and we estimate the exponent α of the power law $f^{-\alpha}$ that approximates the PSD of their counting process. Then, we use this value to train SVM and RF classifiers, in addition to the length of first packets of each flow.

We tested our method on three traffic traces containing a mix of traffic from different applications. In our experiments, considering this additional per-source information has yielded the same accuracy as using only per-flow data, but observing fewer packets in each flow and thus allowing with shorter classification delay.

Table 3: Classification accuracy and delay on *NZIX* traces ($\xi = 100$).

Classifier	**Error Rate**		**Delay**
	SVM	RF	
N1 ($n_1 = 3$)	12.29%	12.89%	229 ms
Tandem	6.22%	2.59%	344 ms
N1 ($n_1 = 5$)	3.14%	2.25%	529 ms

Table 4: Classification accuracy and delay on Leipzig traces ($\xi = 100$).

Classifier	**Error Rate**		**Delay**
	SVM	RF	
N1 ($n_1 = 3$)	15.89%	19.35%	143 ms
Tandem	9.13%	7.14%	202 ms
N1 ($n_1 = 5$)	5.76%	5.05%	346 ms

Finally, we note that using this per-source information during training enhances classification performance, but requires no additional delay for data processing on classification.

References

[1] T. Nguyen, G. Armitage, "A Survey of Techniques for Internet Traffic Classification Using Machine Learning", *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 2008.

[2] Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K. 2008. "Internet traffic classification demystified: myths, caveats, and the best practices". In *Proceedings of the 2008 ACM CoNEXT Conference*, Madrid, Spain, December 09 - 12, 2008.

[3] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, "Class-of-Service Mapping for QoS: a Statistical Signature-Based Approach to IP Traffic Classification", *Proc. of the 4th ACM SIGCOMM Conference on Internet Measurement*, Taormina, Italy, 2004.

[4] T. Auld, A. Moore, S. Gull, "Bayesian Neural Networks for Internet Traffic Classification", *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 223–239, Jan. 2007.

[5] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, "Traffic Classification through Simple Statistical Fingerprinting," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, 2007.

[6] G. Verticale, P. Giacomazzi, "Performance Evaluation of a Machine Learning Algorithm for Early Application Identification", *Proc. of International Multiconference on Computer Science and Information Technology*, 2008, Mragowo, Poland, Oct. 2008.

[7] W. E. Leland, M. S. Taqqu, W. Willinger, D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Trans. Networking*, vol. 1, no. 2, pp. 1–15, Feb. 1994.

[8] Detlef Sass, Simon Hauger, Martin Kohn, "Architecture and scalability of a high-speed traffic measurement platform with a highly flexible packet classification", Elsevier Computer Networks, April 2009.

[9] V. Paxson, S. Floyd, "Wide-Area Traffic: the Failure of Poisson Modeling", *IEEE/ACM Trans. Networking,* vol. 3, no. 6, pp. 226-244, 1995.

[10] M. Crovella, A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Trans. Networking,* vol. 5, no. 6, pp. 835–846, Dec 1997.

[11] S. Bregni, L. Jmoda, "Accurate Estimation of the Hurst Parameter of Long-Range Dependent Traffic Using Modified Allan and Hadamard Variances", *IEEE Trans. Commun.*, vol. 56, no. 11, pp. 1900–1906, Nov. 2008. Extended version: http://home.dei.polimi.it/bregni/public.htm.

[12] D. W. Allan, J. A. Barnes, "A Modified Allan Variance with Increased Oscillator Characterization Ability", *Proc. 35th Annual Freq. Contr. Symp.*, 1981.

[13] P. Lesage, T. Ayi, "Characterization of Frequency Stability: Analysis of the Modified Allan Variance and Properties of Its Estimate", *IEEE Trans. Instrum. Meas.*, vol. 33, no. 4, pp. 332-336, Dec. 1984.

[14] L. G. Bernier, "Theoretical Analysis of the Modified Allan Variance", *Proc. 41st Annual Freq. Contr. Symp.*, 1987.

[15] D. B. Sullivan, D. W. Allan, D. A. Howe, F. L. Walls, Eds., *Characterization of Clocks and Oscillators*, NIST Tech. Note 1337, March 1990.

[16] S. Bregni, "Chapter 5 - Characterization and Modelling of Clocks", in *Synchronization of Digital Telecommunications Networks*. Chichester, UK: John Wiley & Sons, 2002, pp. 203-281.

[17] ITU-T Rec. G.810 "Definitions and Terminology for Synchronisation Networks", Geneva, 1996-2003.

[18] S. Abe, *Support Vector Machines for Pattern Classification*. London, UK: Springer-Verlag, 2005.

[19] C.-C Chang, C.-J. Lin, *LIBSVM: a library for support vector machines,* 2001. URL: http://www.csie.ntu.edu.tw/cjlin/libsvm

[20] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, Springer, Oct. 2001.

[21] The netfilter core team. The netfilter.org "iptables" project. Available: http://www.netfilter.org/projects/iptables/index.html. Retr. on 27-7-2010

[22] C. Schmoll, S. Zander, *NetMate Meter 0.9.3*, 2005. Available: http://sourceforge.net/projects/netmate-meter/.

[23] S. Zander, N. Williams, *netAI Network Traffic based Application Identification*, 2006. URL: http://caia.swin.edu.au/urp/dstc/netai/.

[24] S. Zander, T. Nguyen, G. Armitage, "Automated Traffic Classification and Application Identification Using Machine Learning", *Proc. of IEEE Conf. on Local Computer Networks*, Sidney, Australia, 2005.

[25] *Waikato Internet Traffic Storage (WITS)*. Http://www.wand.net.nz/wits/