We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

2

Automatic Internet Traffic Classification for Early Application Identification

Giacomo Verticale
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Italy

1. Introduction

The classification of Internet packet traffic aims at associating a sequence of packets (a *flow*) to the application that generated it. The identification of applications is useful for many purposes, such as the usage analysis of network links, the management of Quality of Service, and for blocking malicious traffic. The techniques commonly used to recognize the Internet applications are based on the inspection of the packet payload or on the usage of well-known transport protocol port numbers. However, the constant growth of new Internet applications and protocols that use random or non-standard port numbers or applications that use packet encryption requires much smarter techniques. For this reason several new studies are considering the use of the statistical features to assist the identification and classification process, performed through the implementation of machine learning techniques. This operation can be done offline or online. When performed online, it is often a requirement that it is performed early, i.e. by looking only at the first packets in a flow.

In the context of real-time and early traffic classification, we need a classifier working with as few packets as possible so as to introduce a small delay between the beginning of the packet flow and the availability of the classification result. On the other hand, the classification performance grows as the number of observed packets grows. Therefore, a trade-off between classification delay and classification performance must be found.

In this work, the features we consider for the classification of traffic flows are the sizes of the first n packets in the client-server direction, with n a given number. With these features, good results can be obtained by looking at as few as 5 packets in the flow. We also show that the C4.5 decision tree algorithm generally yields the best results, outperforming Support Vector Machines and clustering algorithms such as the Simple K-Means algorithm.

As a novel result, we also present a new set of features obtained by considering a packet flow in the context of the activity of the Internet host that generated them. When classifying a flow, we take into account some features obtained by collecting statistics on the connection generation process. This is to exploit the well-known result that different Internet applications show different degrees of burstiness and time correlation. For example, the email generation process is compatible to a Poisson process, whereas the request of web pages is not Poisson but, rather, has a power-law spectrum.

By considering these features, we greatly enhance the classification performance when very few packets in the flow are observed. In particular, we show that the classification perfor-

mance obtained with only $n = 3$ packets and the statistics on the connection generation process is similar to the performance obtained with $n = 5$ packets and no information on the connection process, therefore achieving a much shorter classification delay.
Section 2 gives a resume of the most significant work in the field and describe the various facets of the problem. In that section we also introduce the Modified Allan Variance, which is the mathematical tool that we use to measure the power-law exponent in the connection generation process. In Section 3 we describe the classification procedure and the traffic traces used for performance evaluation.
Section 4 discusses the experimental data and shows the evidence of power-law behavior of the traffic sources. In Section 5 we compare some machine learning algorithms proposed in the literature in order to select the most appropriate for the traffic classification problem. Specifically, we compare the C4.5 decision tree, the Support Vector Machines, and the Simple K-Means clustering algorithm.
In Section 6 we introduce the novel classification algorithms that exploit the per-source features and evaluate their performance in Section 7. Some conclusions are left for the final section.

2. Background Material

2.1 Related Work

Nguyen & Armitage (2008) identify three basic traffic classification approaches based on machine learning:

- clustering, based on unsupervised learning;
- classification, based on supervised learning;
- hybrid approaches, combining the best of both supervised and unsupervised techniques.

Roughan et al. (2004) propose the Nearest Neighbors (NN), Linear Discriminant Analysis (LDA) and the Quadratic Discriminant Analysis (QDA) algorithms to identify the QoS class of different applications. The authors identify a list of possible features calculated over the entire flow duration. In the reported results, the authors obtain a classification error value in the range of 2.5% to 12.6%, depending on whether three or seven QoS classes are used.
Moore & Zuev (2005) propose the application of Bayesian techniques to traffic classification. In particular they used the Naive Bayes technique with Kernel Estimation (NBKE) and the Fast Correlation-Based Filter (FCBF) methods with a set of 248 full-flow features, including the flow duration, packet inter-arrival time statistics, payload size statistics, and the Fourier transform of the packet inter-arrival time process. The reported results show an accuracy of approximately 98% for web-browsing traffic, 90% for bulk data transfer, 44% for service traffic, and 55% for P2P traffic.
Auld et al. (2007) extend the previous work by using a Bayesian neural network. The classification accuracy of this technique reaches 99%, when the training data and the test data are collected on the same day, and reaches 95% accuracy when the test data are collected eight months later than the training data.
Nguyen & Armitage (2006a;b) propose a new classification method that considers only the most recent n packets of the flow. The collected features are packet length statistics and packet inter-arrival time statistics. The obtained accuracy is about 98%, but the performance is poor if the classifier misses the beginning of a traffic flow. This work is further extended by proposing

the training of the classifier by using statistical features calculated over multiple short subflows extracted from the full flow. The approach does not result in significant improvements to the classifier performance.
Park et al. (2006a;b) use a Genetic Algorithm (GA) to select the best features. The authors compare three classifiers: the Naive Bayes with Kernel Estimation (NBKE), the C4.5 decision tree, and Reduced Error Pruning Tree (REPTree). The best classification results are obtained using the C4.5 classifier and calculating the features on the first 10 packets of the flow.
Crotti et al. (2007) propose a technique, called Protocol Fingerprinting, based on the packet lengths, inter-arrival times, and packet arrival order. By classifying three applications (HTTP, SMTP and POP3), the authors obtain a classification accuracy of more than 91%.
Verticale & Giacomazzi (2008) use the C4.5 decision tree algorithm to classify WAN traffic. The considered features are the lengths of the first 5 packets in both directions, and their inter-arrival times. The results show an accuracy between 92% and 99%.
We also review some fundamental results on the relation between different Internet applications and power-law spectra.
Leland et al. (1993) were among the first in studying the power-law spectrum in LAN packet traffic and concluded that its cause was the nature of the data transfer applications.
Paxson & Floyd (1995) identified power-law spectra at the packet level also in WAN traffic and also conducted some investigation on the connection level concluding that Telnet and FTP control connections were well-modeled as Poisson processes, while FTP data connections, NNTP, and SMTP were not.
Crovella & Bestavros (1997) measured web-browsing traffic by studying the sequence of file requests performed during each session, where a session is one execution of the web-browsing application, finding that the reason of power law lies in the long-tailed distributions of the requested files and of the users' "think-times".
Nuzman et al. (2002) analyzed the web-browsing-user activity at the connection level and at the session level, where a session is a group of connections from a given IP address. The authors conclude that sessions arrivals are Poisson, while power-law behavior is present at the connection level.
Verticale (2009) shows that evidence of power-law behavior in the connection generation process of web-browing users can be found even when the source activity is low or the observation window is short.

2.2 The Modified Allan Variance

The MAVAR (Modified Allan Variance) was originally conceived for frequency stability characterization of precision oscillators in the time domain (Allan & Barnes, 1981) and was originally conceived with the goal of discriminating noise types with power-law spectrum of kind $f^{-\alpha}$, recognized very commonly in frequency sources. Recently, Bregni & Jmoda (2008) proposed MAVAR as an analysis tool for Internet traffic. It has been demonstrated to feature superior accuracy in the estimation of the power-law exponent, α, coupled with good robustness against non stationarity in the data. Bregni & Jmoda (2008) and Bregni et al. (2008) successfully applied MAVAR to real internet traffic analysis, identifying fractional noise in experimental results, and to GSM telephone traffic proving its consistency to the Poisson model. We briefly recall some basic concepts.

Given an infinite sequence $\{x_k\}$ of samples of an input signal $x(t)$, evenly spaced in time with sampling period τ_0, MAVAR is defined as:

$$\mathrm{Mod}\sigma_y^2(\tau) = \frac{1}{2n^2\tau_0^2}\left\langle \left[\frac{1}{n}\sum_{j=1}^{n}\left(x_{j+2n} - 2x_{j+n} + x_j\right)\right]^2 \right\rangle \tag{1}$$

where $\tau = n\tau_0$ is the observation interval and the operator $\langle\cdot\rangle$ denotes infinite-time averaging. In practice, given a finite set of N samples over a measurement interval $T = (N-1)\tau_0$, the MAVAR can be computed using the ITU-T standard estimator (Bregni, 2002):

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

with $n = 1,2,\cdots,\lfloor N/3 \rfloor$.
We consider the random processes $x(t)$ with one-sided Power Spectral Density (PSD) modeled as:

$$S_x(f) = hf^{-\alpha}, \tag{3}$$

where α and h are the model parameters. Such random processes are commonly referred to as *power-law* processes. For these processes, the infinite-time average in (1) converges for $\alpha < 5$. The MAVAR obeys a simple power law of the observation interval τ (ideally asymptotically for $n \to \infty$, keeping constant $n\tau_0 = \tau$, in practice for $n > 4$):

$$\mathrm{Mod}\sigma_y^2(\tau) \simeq A_\mu \tau^\mu \tag{4}$$

where $\mu = \alpha - 3$ and A_μ is a constant.
Therefore, if $x(t)$ obeys (3), a log-log plot of the MAVAR ideally looks as a straight line, whose slope μ gives the exponent estimate $\alpha = \mu + 3$ of the power-law component. Bregni & Jmoda (2008) show these estimates to be accurate, therefore we choose this tool to analyze power laws in traffic traces.

3. Classification Procedure

Figure 1 shows the general architecture for traffic capture. Packets coming from a LAN to the Internet and vice versa are all copied to a PC, generally equipped with specialized hardware, which can either perform real-time classification or simply write to a disk a traffic trace, which is a copy of all the captured packets. In case the traffic trace is later made public, all the packets are anonymized by substituting their IP source and destination addresses and stripping the application payload.
In order to have repeatable experiments, in our research work we have used publicly available packet traces. The first trace, which we will refer to as *Naples*, contains traffic related to TCP port 80 generated and received by clients inside the network of University of Napoli "Federico II" reaching the outside world (*Network Tools and Traffic Traces*, 2004). The traces named *Auckland*, *Leipzig*, and *NZIX* contain a mixture of all traffic types and are available at the *NLANR PMA: Special Traces Archive* (2009) and the *WITS: Waikato Internet Traffic Storage* (2009). Table 1 contains the main parameters of the used traces.
Figure 2 shows the block diagram of the traffic classification procedure.

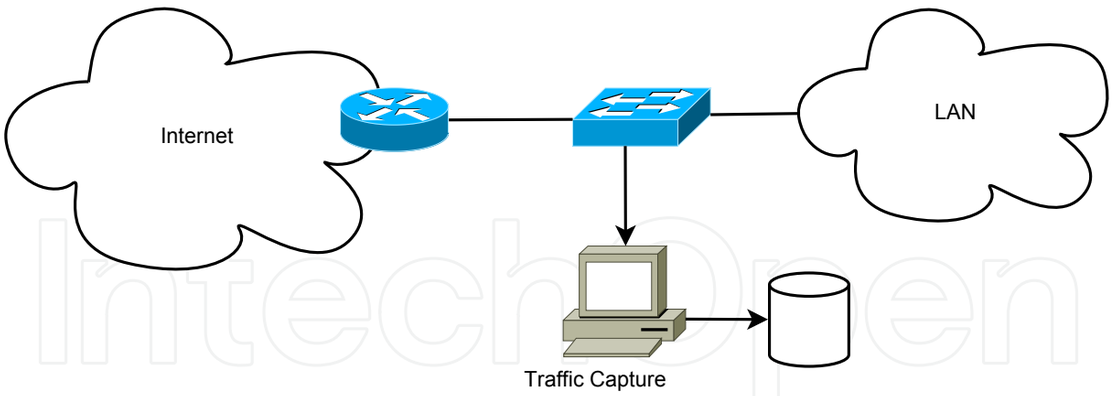


Fig. 1. Architecture of the Traffic Capture Environment.

Name	Length (hh:mm)	Date	Start Time (hh:mm)
Auckland (a)	24:00	June 11th, 2001	00:00
Auckland (b)	24:00	June 12th, 2001	00:00
Leipzig (a)	4:23	Feb. 21st, 2003	12:14
Leipzig (b)	4:24	Feb. 21st, 2003	16:37
Naples	1:00	June 14th, 2004	11:00
NZIX (a)	24:00	July 6th, 2000	00:00
NZIX (b)	24:00	July 7th, 2000	00:00

Table 1. Parameters of the Analyzed Traffic Traces

Given a packet trace, we use the *NetMate Meter* (2006) and *netAI, Network Traffic based Application Identification* (2006) tools to group packets in traffic flows and to elaborate the per-flow metrics. In case TCP is the transport protocol, a flow is defined as the set of packets belonging to a single TCP connection. In case UDP is used, a flow is defined as the set of packets with the same IP addresses and UDP port numbers. A UDP flow is considered finished when no packets have arrived for 600 s. If a packet with the same IP addresses and UDP port numbers arrives when the flow is considered finished, it is considered the first packet in a new flow between the same couple of hosts.
For each flow, we measure the lengths of the first n packets in the flow in the client-server direction. These data are the per-flow metrics that will be used in the following for classifying the traffic flows. We also collect the timestamp of the first packet in the flow, which we use as an indicator of the time of the connection request.
For the purpose of training the classifier, we also collect the destination port number for each flow. This number will be used as the data label for the purpose of validating the proposed classification technique. Of course, this approach is sub-optimal in the sense that the usage of well-known ports cannot be fully trusted. A better approach would be performing deep packet inspection in order to identify application signatures in the packet payload. However, this is not possible with public traces, which have been anonymized by stripping the payload. In the rest of the paper we will made the assumption that, in the considered traffic traces, well-known ports are a truthful indicator of the application that generated the packet flow.

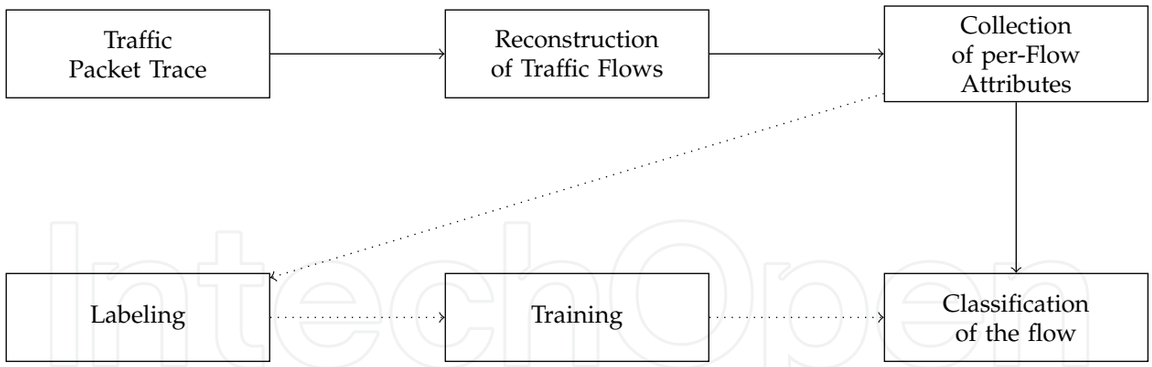


Fig. 2. Block diagram of classification procedure.

The collected data are then passed to the *R* software (R Development Core Team, 2008) to collect the per-source metrics, to train the classifier, and to perform the cross-validation tests. In particular we used the *Weka* (Witten & Frank, 2000) and the *libsvm* (Chang & Lin, 2001) libraries. From the timestamps of the first packets in each flow, we obtain the discrete sequence $x_i^p(k)$, which counts the connection requests from the i-th client, associated to the p-th transport port, in the k-th time interval. Each interval is long $\tau_0 = 1$ s. Each time a new connection request arrives, the sequence $x_i^p(k)$ is updated and we compute the metrics in Table 2.

Metric	Definition
Coefficient of Variation	the ratio between the standard deviation of $x_i^p(k)$ and its mean
Skewness	the standardized third moment of $x_i^p(k)$
Kurtosis	the standardized fourth moment of $x_i^p(k)$
Power-law exponent	the exponent α of the power-law component in the Power Spectral Density of $x_i^p(k)$

Table 2. Per-source metrics.

4. The Power-law Exponent

In this section, we present some results on the power-law behavior of the connection request process by commenting the measurements on the Naples traffic trace, which contains only web-browsing traffic, and the Auckland(a) traffic trace, which contains a mix a different traffic types.

Figure 3 shows the three sequences $x_1^{80}(k)$, $x_2^{80}(k)$, and $x^{80}(k)$. The first sequence is obtained by considering only connections from a single IP address, which we call *Client 1*. Similarly, the second sequence is obtained considering connections from *Client 2*. Finally, the third sequence is obtained considering all the connections in the trace. The total traffic trace is one-hour long and the two clients considered are active for all the duration of the measurement. Neither the aggregated connection arrival process nor the single clients show evident non stationarity.

As discussed in Section 2.2, the slope of Modσ^2 vs τ in the log-log plot can be used as a measure of the power-law exponent. In order to avoid border effects and poor confidence

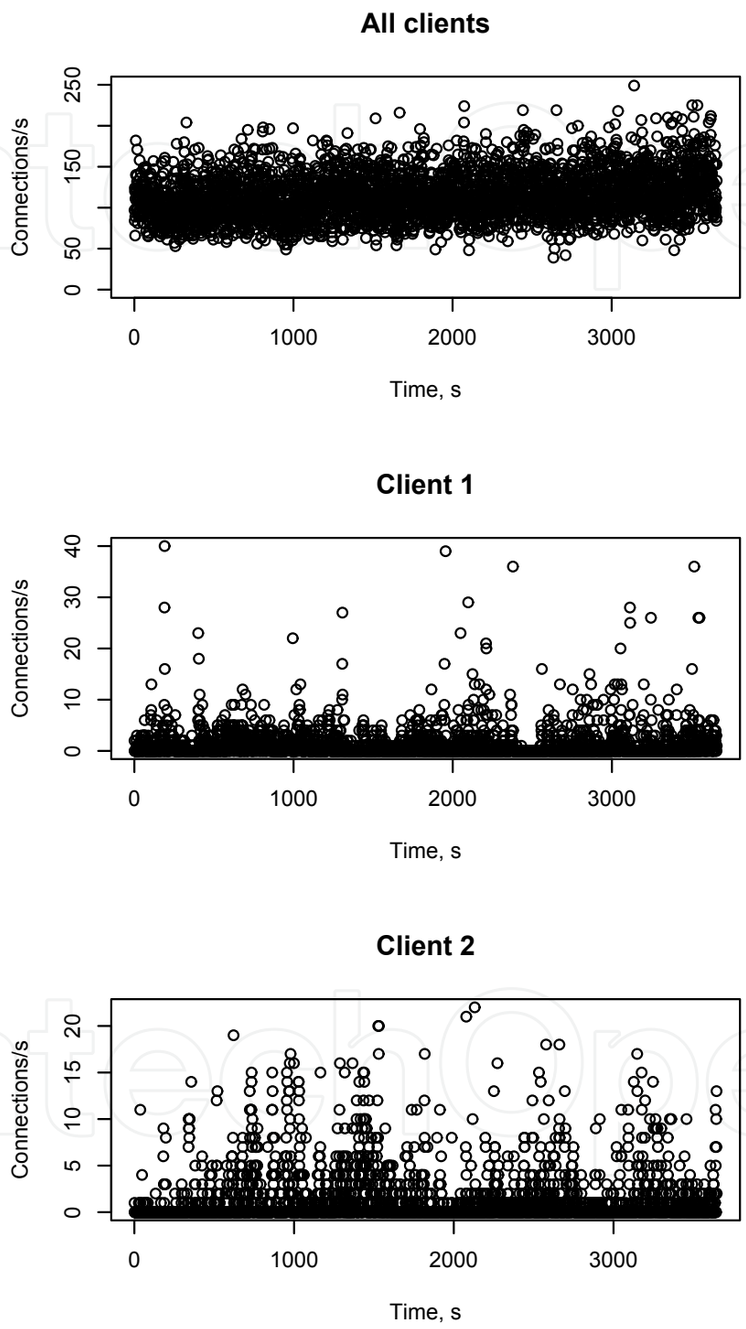


Fig. 3. Connection requests per second in the Naples traffic trace.

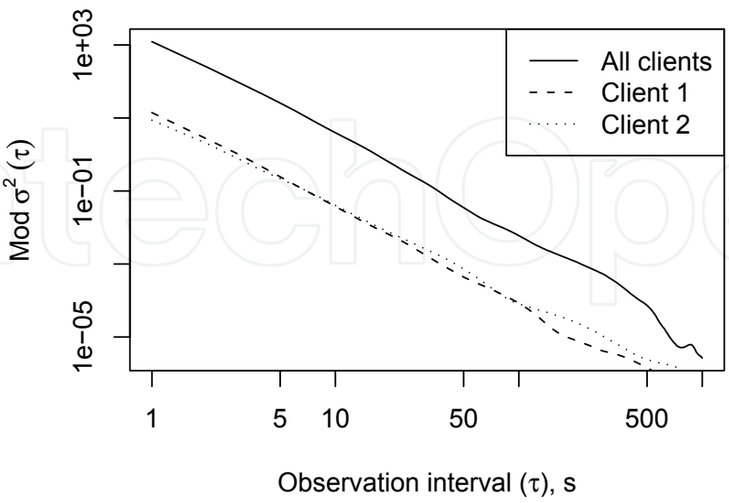


Fig. 4. MAVAR computed on the sequence of connection requests from two random clients and from all the clients in the Naples traffic trace.

in the values of Modσ^2, we calculate α by considering only the range $4\tau_0 \leq \tau < 0.3 \max k\tau_0$, as suggested in (Bregni & Jmoda, 2008). Figure 4 shows the MAVAR calculated on the three sequences. In the considered range of τ, the three curves in Figure 4 have a similar slope, corresponding to the values of $\alpha_1 = 0.28$ and $\alpha_2 = 0.35$ for clients 1 and 2 respectively, and $\alpha = 0.24$ for the aggregated process. These data confirm our expectations that the sum of sequences showing power-law behavior also shows power-law behavior.

We have considered so far only TCP connection requests to servers listening on port number 80, which is the well-known port for HTTP data traffic. We expect that traffic using different application protocols shows a different time-correlation behavior. With reference to the *Auckland* traffic trace, we have extracted the per-client connection request sequence $x_i^p(k)$ considering only requests for servers listening on the TCP ports 25, 80, 110, and 443, which are the well-known ports for SMTP, HTTP, POP3, and HTTPS. We have also considered requests for servers listening on either TCP or UDP port 53, which is the well-known port for DNS requests.

Figure 5 shows the estimate m_α for the various destination ports, obtained by averaging the value of α measured for the clients with at least 50 connection requests in the observation window. The figure also shows 95% confidence intervals for the mean. From the observation of Figure 5, we also notice that the confidence intervals for the estimate of the power-law exponent of the email application traffic includes $\alpha = 0$ both for port 25 and 110, therefore showing no evidence of power-law behavior. Instead, the estimates for web requests, both on insecure (port 80) and on secure connections (port 443) have overlapping confidence intervals not including $\alpha = 0$. Then we conclude that these processes come from similar populations and show evidence of power-law behavior. Finally, the confidence interval for DNS requests does not include $\alpha = 0$ and does not overlap with web traffic, allowing us to conclude that,

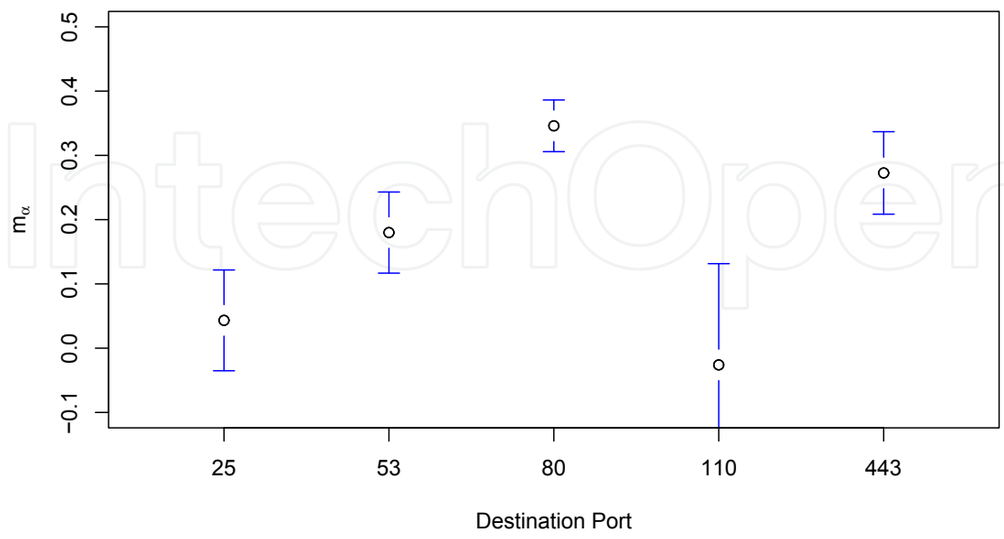


Fig. 5. Estimated power-law exponent of the connection requests process for different destination port numbers in the Auckland traffic trace. Estimations are averaged over all the clients and 95% confidence intervals are shown.

from the point of view of time-correlation, the DNS request process shows evidence of power-law behavior and comes from a different population than web traffic.

5. Comparison of Learning Algorithms

In this section, we compare three algorithms proposed for the classification of traffic flows. In order to choose the classification algorithm to be used in the hybrid schemes discussed later, we performed a set of experiments by training the classifiers using the Auckland(a), NZIX(a), and Leipzig(a) traffic traces and testing the performance by classifying the Auckland(b), NZIX(b), and Leipzig(b) traffic traces, respectively.
To ease a comparison, we performed our assessment by using the same 5 applications as in (Williams et al., 2006), i.e. FTP-data, Telnet, SMTP, DNS (both over UDP and over TCP), and HTTP. In all the experiments, traffic flows are classified by considering only the first 5 packets in the client server direction. The performance metric we consider is the error rate, calculated as the ratio between the misclassified instances to the total instances in the data set. We consider two supervised learning algorithms namely the C4.5 Decision Tree and the Support Vector Machines (SVM), and an unsupervised technique, namely the Simple K-means.
For the SVM, we considered the polynomial kernel with degrees $d = 2$ and $d = 3$ and the RBF kernel. In the polynomial case we normalized attributes in the range $[0,1]$, while in the RBF case we normalized attributes in the range $[-1,1]$, as suggested in (Abe, 2005).
To choose the cost parameter we performed a 10-fold cross validation on the Auckland(a) traffic trace and obtained the best results with the following configurations: polynomial kernel with degree $d = 2$ and cost $C = 10^6$; RBF kernel with exponent $\gamma = 4$ and cost $C = 10^3$.

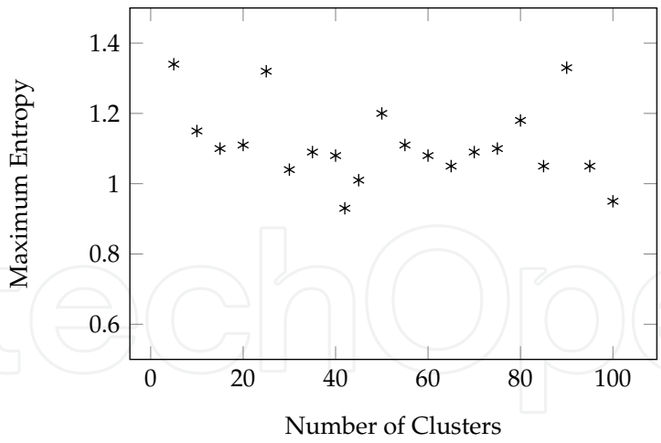


Fig. 6. Maximum entropy of clusters in the simple k-means clustering.

	C4.5	SVM (Polynomial)	SVM (RBF)	Simple K-means
Auckland	0.8%	7.8%	4.3%	11%
Leipzig	0.6%	3.6%	4.3%	12%
NZIX	0.5%	1.9%	0.2%	7%

Table 3. Error rate for three traffic traces with the different classification techniques.

For the Simple K-Means, we tried different values for the number of clusters. Since the algorithm could not perfectly separate the labeled instances, we labeled each cluster with the most common label. To choose the number of clusters, we performed a 10-fold cross validation on the Auckland(a) traffic trace. For several possible choices for the number of clusters, we computed the entropy of each cluster. In Figure 6 we plot the entropy of the cluster that has the maximum entropy versus the number of clusters. The figure does not show a clear dependency of the maximum entropy on the number of clusters, so we decided to use 42 clusters, because, in the figure, it corresponds to a minimum.
Table 3 reports the measured error rate for the selected classifiers in the three experiments. Comparing the experiments we do not see a clear winner. With the Auckland and Leipzig traces, C4.5 performs better, while SVM with RBF kernel yields the best results with the NZIX trace. In the Leipzig case, however, the SVM with RBF kernel perform worse than the SVM with polynomial kernel. The Simple K-means technique always shows the highest error rate. Since the C4.5 classifier seems to give the best results overall, in the following we will consider this classifier as the basis for the hybrid technique.

6. The Hybrid Classification Technique

As discussed in Section 4, the statistical indexes computed on the connection-generation process depend on the application that generated the packet flow. Therefore, we introduce a new classifier capable of exploiting those indexes. The block diagram of this new classifier, which we will refer to as the *hybrid classifier*, is shown in Figure 7.

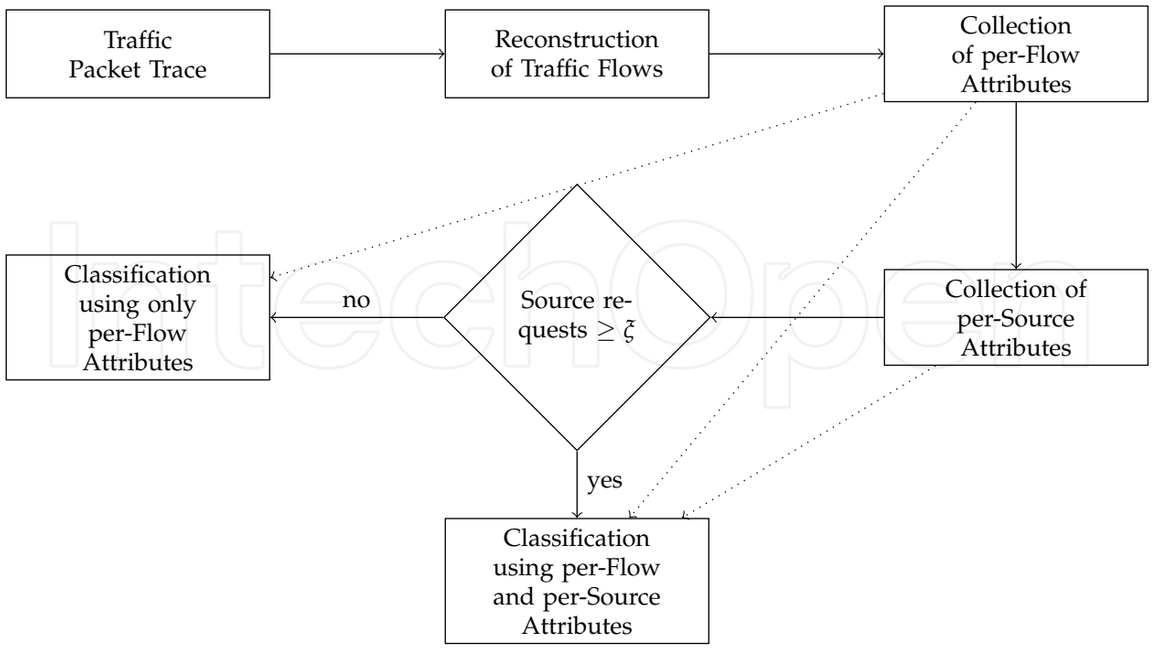


Fig. 7. Block diagram of the hybrid classifier.

As usual, we capture the packets from the communication link and reconstruct the TCP connections. We also collect the per-flow features, which comprise the length of the first n packets in the flow. In addition, we maintain running statistics on the connection generation process. For each pair (IP source, destination port number), we calculate the per-source attributes discussed in Section 3 and listed in Table 2. It is worth noting that all these attributes do not require to keep in memory the whole list of the connection request arrival times, because they can be updated with a recurrence formula each time a new connection request arrives. As discussed in Section 4, when a given IP source has generated only a few requests, the statistical indexes have a large error, so we do not consider them for the purpose of traffic classification. Instead, when the IP source has generated many connection requests, the statistical indexes show better confidence, so we use them for classification. In order to choose whether the indexes are significant or not, we compare the total number of connections that the source has generated to a given threshold, ξ, which is a system parameter. If the source has generated fewer than ξ connections, we perform classification of the traffic flow by using only the flow attributes (i.e. the sizes of the first packets). Otherwise, if the source has generated more than ξ connections, we perform classification by using both the flow attributes and the source attributes (i.e. the statistical indexes). The same rule applies to training data. Labeled flows generated by IP sources that, up to that flow, have generated fewer requests than ξ, are used to train the classifier using only flow attributes. On the other hand, the labeled flows generated by IP sources that have generated more than ξ requests are used to train the classifier using both the per-flow and the per-source attributes. In both cases, the used classifier is a C4.5 decision tree.

The number of the packets to consider for classification is a critical parameter. The more packets are considered, the less the classification error. However, collecting the required number of

packets requires time, during which the flow remains unclassified. It would be better to perform classification as soon as possible. In this work, we consider the scenario in which only the packets from the client to the server are available. In this scenario, we have observed that the hit ratio does not grow significantly if more than 5 packets are considered. This is consistent to results in (Bernaille et al., 2006). However, we will show that the average time needed to collect 5 packets is usually in the order of the hundreds of ms, depending on the network configuration. On the other hand, if classification were performed considering only the first 3 packets per flow, the time required would drop significantly. Classification performance, however, would be much worse.

In this work, we propose a hybrid classification technique that aims at achieving good classification performance but requiring as few packets as possible. In order to evaluate the performance of the hybrid classifier, we consider the following configurations.

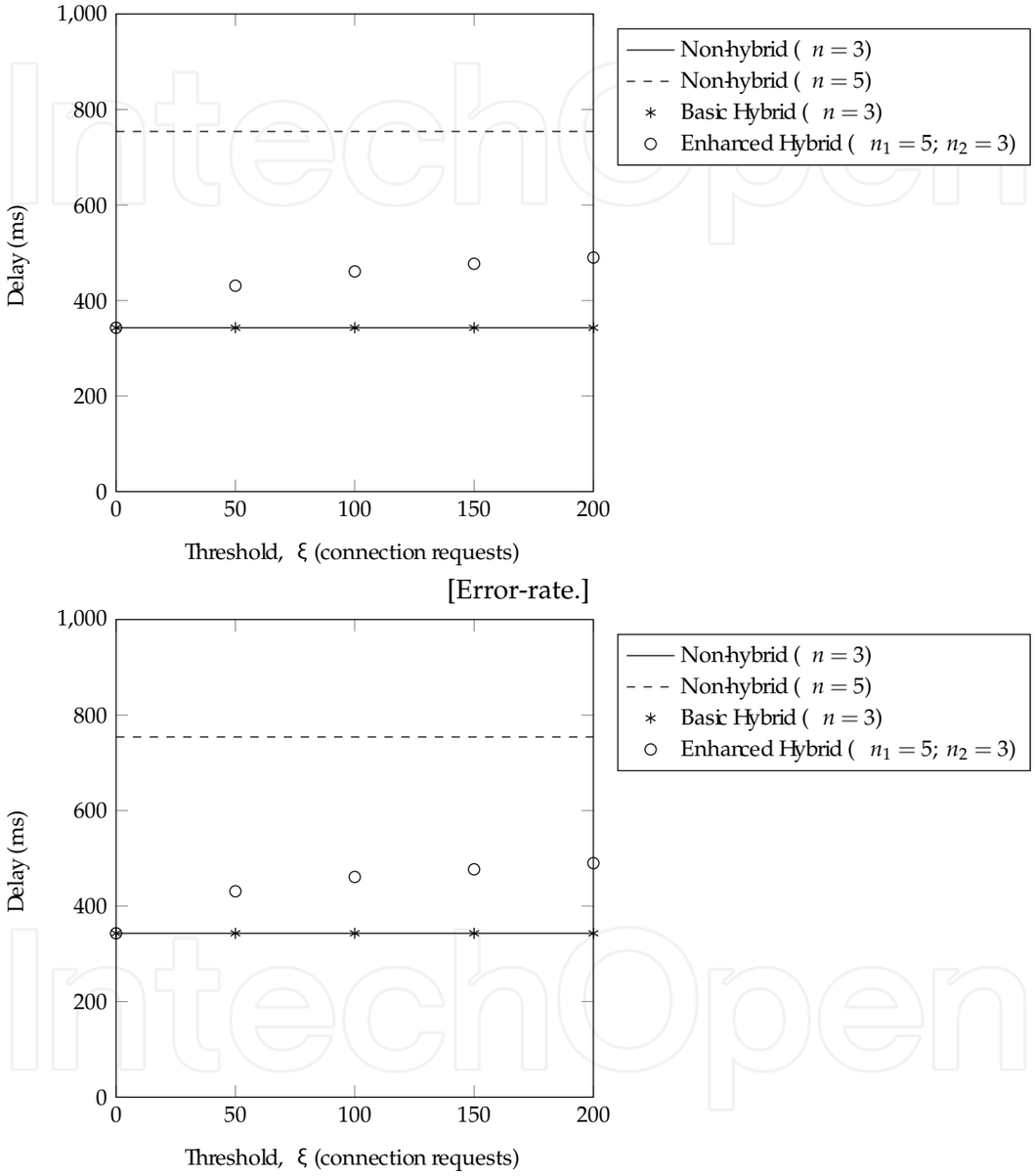
The first two configurations, which we will refer to as *non-hybrid* perform classification by using only the packets sizes. For each flow, the first n packets are collected and then their sizes are fed to the classifier. The time required to collect the required data corresponds to the time required to collect exactly n packets. If the flow contains fewer packets, then classification can be performed only when the flow is over. We consider the cases where either $n = 3$ or $n = 5$ packets.

The third configuration, which we will refer to as *basic hybrid classifier* splits the incoming flows in two sets, depending on the IP source activity, as explained above. Then, the first n packets are collected and classification is performed by using the packet sizes and, possibly, the source statistical indexes. Since the source indexes are available at the flow beginning, exploitation of these features introduces no delay. Therefore the basic hybrid classifier is appealing because it yields a better hit ratio than the non-hybrid classifier using the same number of packets, n. In this chapter, we consider the case where $n = 3$.

Finally, we consider the *enhanced hybrid classifier*. Similarly to the basic configuration, this classifier splits the incoming flows in two sets depending on the IP source activity. However, the number of packets collected for each flow depends on the set. For the flows coming from low activity sources, the classifier waits for n_1 packets, whereas, for the flows coming from high activity sources, the classifier waits for n_2 packets. Since this second classifier already has valuable information for performing classification, it needs fewer packets, therefore $n_1 > n_2$. This way, the result of classification is obtained more quickly for those flows coming from high activity sources and for which other data are available. We consider the case where $n_1 = 5$ and $n_2 = 3$. Since the decision of which set each flow belongs to depends on the threshold ξ, if the threshold is low, then the classification is quicker, but the hit ratio is lower because the statistical indexes are less reliable. On the other hand, if the threshold is higher, then classification is slower, but more precise. At the extrema, if $\xi = 0$, the performance converges to that of the basic hybrid classifier; as ξ goes to infinity, performance converges to that of the non-hybrid classifier with $n = n_1$.

7. Numerical Results

In this Section, we evaluate the performance of the proposed traffic classification techniques. The first set of experiments is a validation using the *NZIX* traffic traces. The classifier is trained using the NZIX(a) trace and the tests are performed using the NZIX(b) trace. Figure 8(a) shows the error rate obtained with the different techniques. The best results are obtained with the non-hybrid classification considering the first $n = 5$ packets in the flow, which results in a percentage of misclassified flows of about 1.8%. The non-hybrid classifier does not use any



[Error-rate.]

[Feature collection delay.]

Fig. 8. Classification performance. Training with the NZIX(a) traffic trace and tests with the NZIX(b) traffic trace.

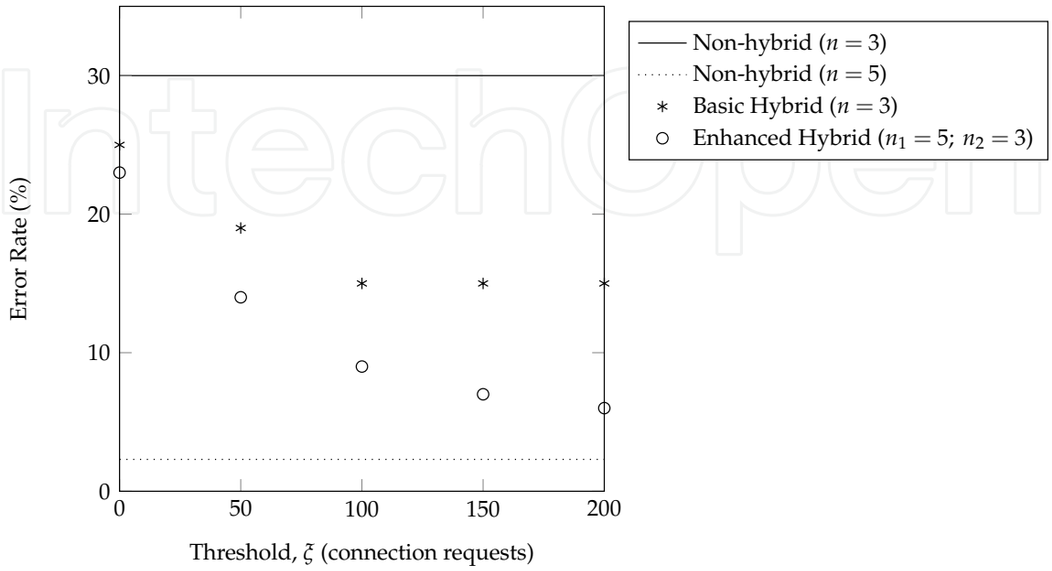
per-source attribute, so the results are independent of the threshold ξ and Figure 8(a) shows this result as an horizontal line. On the other hand, the worst results are obtained with the non-hybrid classifier using the first $n = 3$ packets in the flow. This classifier results in an error rate of about 8%. We discuss these results by comparing the achieved classification performance to the delay between the beginning of the flow and the time to obtain the classifier output. In Figure 8(b), we show the average time necessary to collect the number of packets required by the classification algorithm. The non-hybrid algorithm with $n = 5$, which shows the best classification performance, gives an answer after, on average, 750 ms. Conversely, the non-hybrid classification technique with $n = 3$ only requires half of that time, giving its output after only 350 ms.

The hybrid techniques try to achieve good classification performance, while requiring fewer packets. The Basic Hybrid technique only requires $n = 3$ packets, so it yields the same delay as the non-hybrid technique with $n = 3$, the classification error, however, is much lower, ranging from 5.2% to about 6.3% depending on the threshold ξ. The threshold controls the minimum number of connection requests necessary to have confidence in the per-source attributes and use them in classifying the flows coming from that source. Therefore, the Basic Hybrid technique reduces the classification error by 2% yielding no increase in the classification delay. The classification performance is influenced by the threshold, but not to a great extent.

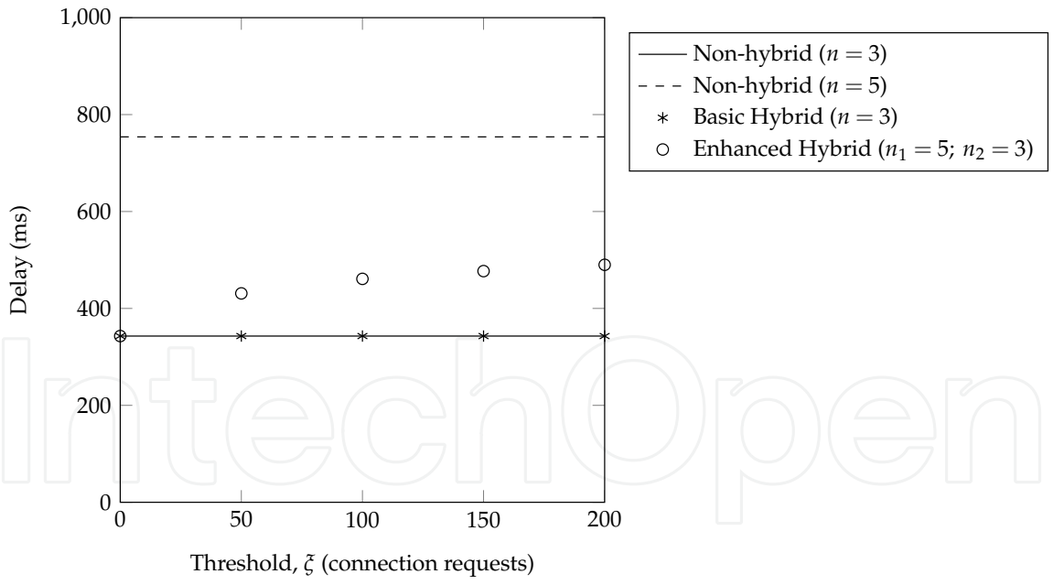
The Enhanced Hybrid technique tries to strike a balance between the error rate and the delay. In Figure 8(a), we plot the classification error rate versus the threshold. When the threshold is 0, most of the flows are classified considering $n_2 = 3$ packets plus the per-source attributes, so the classification performance converges to the classification performance of the Basic Hybrid technique. Independently of the threshold, some flows cannot take advantage of the per-source attributes because these attributes cannot be computed, for example because the source must have been active for at least some time interval in order to compute the power law exponent α; therefore, the results for the Basic and the Enhanced techniques do not coincide. As the threshold grows, fewer flows take advantage of the per-source attributes and are classified using the non hybrid scheme with $n_1 = 5$ packets. On the other hand, the per-source attributes are more reliable and the classification performance is better. Figure 8(a) shows that the error rate drops to as low as 2.5% when $\xi = 200$ connection requests.

The drawback is that, as the threshold increases, more and more flows are classified using more packets and the delay increases. Figure 8(b) shows that, when $\xi = 200$, the classification delay is about 500 ms. This delay is about 150 ms more than the delay obtained with the Basic scheme, which has a much worse classification performance, and is 250 ms less than the delay of the non hybrid scheme with $n = 5$, which only yields slightly better results.

Figure 9 shows a similar experiment with the *Auckland* data set. The classifier is trained with the Auckland(a) traffic trace and the tests are performed on the Auckland(b) traffic trace. In Figure 9(a) we plot the error rate versus the threshold, ξ. With this data set, the non hybrid technique with $n = 3$ packets performs poorly, with an error rate of about 30%. Instead, if $n = 5$ packets are considered, the error rate drops to about 2.5%, which is similar to the error rate obtained with the *NZIX* data set. Figure 9(b) shows that the average delay required to collect $n = 3$ and $n = 5$ packets is similar, being 200 ms and 235 ms, respectively. In this scenario the hybrid techniques are less appealing, because the delay difference is limited. However, these techniques yield some advantage also in this scenario. In Figure 9(a) we observe that, as the threshold increases, both the Basic and the Enhanced schemes show better classification performance. The Basic Hybrid Classifier settles at an error rate of about 15% when the threshold is larger or equal to 100 connection requests. Larger values do not seem to give better results.



[Error-rate.]



[Feature collection delay.]

Fig. 9. Classification performance. Training with the Auckland(a) traffic trace and tests with the Auckland(b) traffic trace.

Therefore the Basic scheme halves the error rate without increasing the delay. The Enhanced scheme shows even better results: with $\xi = 100$, the Enhanced Hybrid Classifier has an error rate of 9%, which drops to 6% when $\xi = 200$. From 9(b) we observe that the Enhanced classifier shows a delay of about 215 ms for $\xi = 100$ and only slightly more for $\xi = 200$. This delay is halfway between the delay of the non hybrid classifier with $n = 3$ and with $n = 5$.

8. Conclusions

In this work, we report experimental results about the classification of Internet traffic by examining the packet flow in the client-server direction. We focus on the problem of early application identification, which requires to find a balance between the classification accuracy and the number of packets required by the classifier.

The contribution of this work is twofold. First, we compare the performance of some well-known supervised and unsupervised classification techniques, namely the C4.5 decision tree, the Support Vector Machines, and the Simple K-Means. We performed validation tests on three traffic traces containing a mix of traffic from different applications and concluded that the C4.5 decision tree algorithm has the best performance overall, even if the SVMs follow closely. The unsupervised technique always yields the worst performance.

Second, we introduce a new classification scheme based on the observation that the connection generation process from a given traffic source is influenced by the application generating the requests. In particular, we show that, in experimental data, the Power Spectral Density of such processes often shows a power-law behavior. Therefore, we propose to use the measured power-law exponent of the traffic source as an additional feature in the classification of a traffic flow. This new feature comes at no additional delay, because its computation is based on the timestamps of the initial packets of past flows.

By using this feature we were able to significantly reduce the classification error rate in all the considered scenarios. Further, we also propose an enhanced scheme in which we perform classification using the first 5 packets in a flow for low-activity sources and the first 3 packets in flow for high-activity sources. By using this scheme, we obtain a low error rate and, at the same time, we have low average classification delay.

There are some possible future directions for this research. In this work, we did not consider the problem of training a classifier on a data set collected on a given link and used on a different link. We expect that the classification error rate increases, but that the per-flow features still yield an increased accuracy, because the connection request process mainly depends on the application and is weakly dependent on the specific network context. In order to study the portability of a classifier it is necessary to use traces captured at different sites but in the same day and at the same hour. This is because traffic patterns evolve over time. Another possible future work is the study of the temporal evolution of the connection generation process.

Acknowledgements

This work has been partially funded by the Italian Research Ministry (MIUR) PRIN 2006 project RECIPE.

9. References

Abe, S. (2005). *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Allan, D. & Barnes, J. (1981). A modified Allan variance with increased oscillator characterization ability, *Thirty Fifth Annual Frequency Control Symposium. 1981* pp. 470–475.

Auld, T., Moore, A. & Gull, S. (2007). Bayesian neural networks for internet traffic classification, *Neural Networks, IEEE Transactions on* **18**(1): 223–239.

Bernaille, L., Teixeira, R. & Salamatian, K. (2006). Early application identification, *The 2nd ADETTI/ISCTE CoNEXT Conference*.

Bregni, S. (2002). *Time and Frequency Measurement Techniques in Telecommunications*, Wiley, pp. 305–375.

Bregni, S., Cioffi, R. & Decina, M. (2008). An empirical study on time-correlation of GSM telephone traffic, *Wireless Communications, IEEE Transactions on* **7**(9): 3428–3435.

Bregni, S. & Jmoda, L. (2008). Accurate estimation of the Hurst parameter of long-range dependent traffic using modified Allan and Hadamard variances, *Communications, IEEE Transactions on* **56**(11): 1900–1906.

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

Crotti, M., Dusi, M., Gringoli, F. & Salgarelli, L. (2007). Traffic classification through simple statistical fingerprinting, *SIGCOMM Comput. Commun. Rev.* **37**(1): 5–16.

Crovella, M. & Bestavros, A. (1997). Self-similarity in World Wide Web traffic: evidence and possible causes, *Networking, IEEE/ACM Transactions on* **5**(6): 835–846.

Leland, W. E., Taqq, M. S., Willinger, W. & Wilson, D. V. (1993). On the self-similar nature of Ethernet traffic, *in* D. P. Sidhu (ed.), *ACM SIGCOMM*, San Francisco, California, pp. 183–193.

Moore, A. W. & Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques, *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ACM, New York, NY, USA, pp. 50–60.

netAI, Network Traffic based Application Identification (2006).
URL: *http://caia.swin.edu.au/urp/dstc/netai/*

NetMate Meter (2006).
URL: *http://sourceforge.net/projects/netmate-meter/*

Network Tools and Traffic Traces (2004).
URL: *http://www.grid.unina.it/Traffic/Traces/ttraces.php*

Nguyen, T. & Armitage, G. (2006a). Synthetic sub-flow pairs for timely and stable IP traffic identification, *Proc. Australian Telecommunication Networks and Application Conference*.

Nguyen, T. & Armitage, G. (2006b). Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world IP networks, *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pp. 369–376.

Nguyen, T. & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning, *Communications Surveys & Tutorials, IEEE* **10**(4): 56–76.

NLANR PMA: Special Traces Archive (2009).
URL: *http://www.nlanr.net/*

Nuzman, C., Saniee, I., Sweldens, W. & Weiss, A. (2002). A compound model for TCP connection arrivals for LAN and WAN applications, *Elsevier Science Computer Networks* **40**(3): 319–337.

Park, J., Tyan, H.-R. & Kuo, C.-C. (2006a). Internet traffic classification for scalable QOS provision, *Multimedia and Expo, 2006 IEEE International Conference on*, pp. 1221–1224.

Park, J., Tyan, H.-R. & Kuo, C.-C. J. (2006b). GA-based internet traffic classification technique for QoS provisioning, *Intelligent Information Hiding and Multimedia Signal Processing, International Conference on* **0**: 251–254.

Paxson, V. & Floyd, S. (1995). Wide area traffic: the failure of Poisson modeling, *IEEE/ACM Trans. Netw.* **3**: 226–244.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL: *http://www.R-project.org*

Roughan, M., Sen, S., Spatscheck, O. & Duffield, N. (2004). Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification, *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ACM, New York, NY, USA, pp. 135–148.

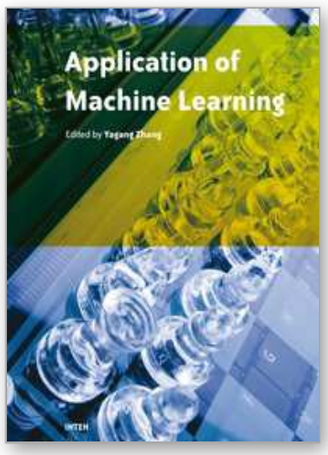
Verticale, G. (2009). An empirical study of self-similarity in the per-user-connection arrival process, *Telecommunications, 2009. AICT '09. Fifth Advanced International Conference on*, pp. 101–106.

Verticale, G. & Giacomazzi, P. (2008). Performance evaluation of a machine learning algorithm for early application identification, *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pp. 845–849.

Williams, N., Zander, S. & Armitage, G. (2006). A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification, *SIGCOMM Comput. Commun. Rev.* **36**(5): 5–16.

WITS: Waikato Internet Traffic Storage (2009).
URL: *http://www.wand.net.nz/wits/*

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



Application of Machine Learning
Edited by Yagang Zhang

ISBN 978-953-307-035-3
Hard cover, 280 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The goal of this book is to present the latest applications of machine learning, which mainly include: speech recognition, traffic and fault classification, surface quality prediction in laser machining, network security and bioinformatics, enterprise credit risk evaluation, and so on. This book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides them with a good introduction to many application researches of machine learning, and it is also the source of useful bibliographical information.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen