Early Classification of Network Traffic through Multi-classification

Alberto Dainotti, Antonio Pescapé, and Carlo Sansone

Department of Computer Engineering and Systems, Universitá di Napoli Federico II
{alberto,pescape,carlosan}@unina.it

Abstract. In this work we present and evaluate different automated combination techniques for traffic classification. We consider six intelligent combination algorithms applied to both traditional and more recent traffic classification techniques using either packet content or statistical properties of flows. Preliminary results show that, when selecting complementary classifiers, some combination algorithms allow a further improvement – in terms of classification accuracy – over already well-performing stand-alone classification techniques. Moreover, our experiments show that the positive impact of combination is particularly significant when there are *early-classification* constraints, that is, when the classification of a flow must be obtained in its early stage (e.g. first 1 – 4 packets) in order to perform network operations online.

1 Introduction

Traffic Classification gained a lot of attention from both the industrial and academic research communities because of its application in several contexts: traffic/user profiling, network provisioning and resource allocation, QoS, enforcement of security policies, etc. While significant progress has been made in this field, with development in several research directions, literature clearly shows that there is still no *perfect* technique achieving 100% accuracy when applied to the entire traffic observed on a network link [20].

Deep Packet Inspection (DPI) is still considered the most accurate approach, but because of (i) computational complexity, (ii) privacy issues, and (iii) lack of robustness to the increasing usage of encryption and obfuscation techniques, it is used today as a reference (*ground-truth*) in order to evaluate the accuracy of new experimental algorithms that should overcome these limitations. Most of these algorithms are based on the application of machine-learning classification techniques to traffic properties and, even if their accuracy never reaches 100%, it has been shown that they typically are more resistant to obfuscation attempts and applicable when encryption is in place [5, 30].

In [13] we proposed the implementation in a single classification platform of combination strategies able to collect the results of very different classification techniques. Moreover, literature in the machine-learning field and pattern recognition [22] has produced several combination algorithms for building *multi-classifier* systems able to achieve better accuracy than each stand-alone classifier composing them.

J. Domingo-Pascual, Y. Shavitt, and S. Uhlig (Eds.): TMA 2011, LNCS 6613, pp. 122–135, 2011.

In this work we apply several (6) of such algorithms to the problem of traffic classification, attempting the combination of classifiers (8) based on techniques known in the traffic classification field and we show preliminary results obtained from a real traffic trace. We show that in some cases it is possible to improve the overall classification accuracy over that of the best-performing classifier. Moreover, based on the observation that when a very limited quantity of information on each flow is available (which translates in less discriminating features) the accuracies of each stand-alone classifier decrease, we evaluate the improvement achieved by combining them under such conditions. Results show that the improvement is quite significant. This is important because several real-world applications of traffic classification as, for example, QoS, traffic shaping, and security policy enforcement, require *early-classification*, that is, the ability to generate a classification response when the flow is in its early stage (e.g. after 1 – 4 packets have been captured) [6] and thus could take real advantage from the use of the combination approaches here analyzed.

2 Related Work

A large amount of research work on traffic classification has been published in the past ten years, including several surveys and papers making comparisons among different techniques [20] [29] [10] [24]. All of them show *pros* and *cons* of different techniques and approaches as well as their inability to reach 100% classification accuracy. On the other side, research in the fields of machine-learning and pattern recognition has developed combination algorithms for classification problems that allow several improvements, included an increase in overall classification accuracy [22]. In the field of network traffic classification, a first rudimental combination approach to traffic classification was proposed in [26]: three different classification techniques are run in parallel (DPI, well-known ports and heuristic analysis), and a decision on the final classification response is taken only when there is a match between the results of two of them (otherwise the multi-classifier reports "unknown"). In [14] and [13], instead, we proposed the idea of combining multiple traffic classifiers using advanced combination strategies, inspired by research in the machine-learning and pattern-recognition fields related to multi-classification [22]. The approach of combining multiple classification techniques through specific algorithms to build a more accurate "multi-classifier", indeed, has been already used with success in other networking reasearch areas as network intrusion and anomaly detection [12]. As for traffic classification, concepts like *En-semble Learning* and *Co-training* have been introduced in [18], where a set of similar classifiers co-participate to learning, while an advanced combination of different traffic classification techniques has been shown in [9]. However, in that work, only variants of the *Dempster-Shafer* algorithm and a *majority vote* are taken into account, while in this paper we consider a more complete set of combination algorithms representative of the state of the art in multi-classification [22] – including those based on the Behavior Knowledge Space – plus we experiment on varying the composition of the pool of traffic classifiers.

Moreover, our contribution goes into a specific, and novel, direction by examining the impact of traffic classification under *early-classification* constraints. We pursue this target by evaluating the behavior of both the stand-alone classifiers and their combinations when trained and tested with discriminating features extracted only from a limited number of packets (from a single packet to the first ten packets). Several works have been presented that tackle the problem of early traffic classification [7] [11] [16], and they show the tradeoff between the amount of packets considered for extracting flow features and classification accuracy. In this work, for the first time we propose multi-classification as a way to improve accuracy while keeping the amount of information used for classification low.

3 Combination Algorithms

In many pattern recognition applications, achieving acceptable recognition rates is conditioned by the large pattern variability, whose distribution cannot be simply modeled. This affects the results at each stage of the recognition system so that, once it has been designed, its performance cannot be improved over a certain bound, despite the efforts in refining either the classification or the description method.

In the last years, some research groups concentrated the attention on a multiple classifier approach [8, 19, 21, 31]. The rationale of this approach lies in the assumption that, by suitably combining the results of a set of base classifiers, the obtained performance is better than that of any base classifier: it is claimed that the consensus of a set of classifiers may compensate for the weakness of a single classifier, while each classifier preserves its own strength [21]. The implementation of a multiple classifier system implies the definition of a *combiner* [22] for determining the most likely class a sample should be attributed to, considering the answers of the base classifiers.

Different combiners, independent of the adopted classification model, have been proposed in the literature [8, 22]. In the following we give a short introduction on the considered combiners. Since some traffic classifiers can be only seen as a *Type 1* classifier (i.e. a classifier that outputs just the most likely class), we considered only criteria that can be applied to classifiers that provide a crisp label as output. It is worth noting, in fact, that some well-known combination schemes (such as the *Decision Templates* proposed in [23]) cannot be applied to *Type 1* classifiers, since they require class probability outputs (i.e., the so-called *Type 3* classifiers[1]).

Before entering in details, it is worth recalling that some combiners make use of the so-called *confusion matrix* [31] for combining *Type 1* classifiers. The classification confusion matrix E^k is such that the generic element e^k_{ij} ($1 \leq i, j \leq m$, where m is the number of the classes) represents the percentage of samples belonging to the i-th class that the k-th classifier assigns to the j-th

[1] For the sake of completeness let us recall that *Type 2* classifiers operate at rank level, providing as output a subset of all the possible classes, with the alternatives ranked in order of plausibility of being the correct class.

class. Therefore, the value e_{ii}^k represents the percentage of samples belonging to the i-th class which are correctly classified by the k-th classifier. The values of the elements of E^k should be computed using a set of data (namely, a *validation set*) different from both the training and the test set.

1) Majority Voting (MV): each classifier votes for one class and the guess class is the one voted by the majority. If more classes obtain the same number of votes, the values e_{ii}^k are used for tie breaking, i.e. the vote of each classifier is weighted by the number representing the confidence degree of that classifier when it assigns a sample to the class it is voting for.

2) Weighted Majority Voting (WMV): in this case the confidence degree evaluated by means of the confusion matrices was used for weighting the votes given by each classifier. The combiner assigns each sample to the class C such that:

$$C = \arg\max_i \sum_k e_{ii}^k \cdot V_i^k \tag{1}$$

where V_i^k is 1 if the guess class of the k-th classifier is i and 0 otherwise.

3) Naïve Bayes (NB): the guess class is the one which maximizes the *a posteriori* probability. The probability that a sample belongs to the i-th class when the k-th classifier assigns it to the j-th class is assumed to be:

$$\frac{M_i \cdot e_{ij}^k}{\sum_{h=1}^{m} M_h \cdot e_{hj}^k} \tag{2}$$

being M_i the number of samples belonging to the i-th class. Applying the Bayes' formula and standing the assumption of the independence of the classifiers, it can be simply shown, starting from the results presented in [22], that the class C which maximizes the *a posteriori* probability is:

$$C = \arg\max_i M_i \cdot \prod_{k=1}^{N} e_{ij}^k \tag{3}$$

where N is the number of classifiers and j is the guess class provided by the k-th classifier.

4) Dempster-Shafer combiner ($D\text{-}S$) [31]: this criterion is based on the Dempster-Shafer theory [17]. According to it, we define for each classifier, the *belief* in every possible subset A of the set $\Theta = \{A_1, A_2, \ldots, A_m\}$. In our context A_i is a proposition representing the fact that a sample is assigned to the i-th class by the considered classifier. The belief $bel(.)$ is calculated from a function, called *basic probability assignment*, which is denoted $m(.)$, by using the following equation:

$$bel(A) = \sum_{B \subseteq A} m(B) \tag{4}$$

where B is any subset of A. Obviously, we have $bel(A_i) = m(A_i)$ and $bel(\Theta) = 1$. In our case, when the k-th expert votes for the i-th class, we consider $m(A_i) = e_{ii}^k$

and $m(\Theta) = 1 - e_{ii}^k$. The values $m(A)$ supplied by each expert are combined via the Dempster rule, and the values $bel(A_i)$ are calculated using equation (4). The estimated class is the one that maximizes the value of $bel(A_i)$.

5) Behavior-Knowledge Space (*BKS*) method: one of the main drawbacks of the previously described approaches lies in the fact that they require (in a more or less explicit way) the independent assumption of the combining classifiers. This assumption does not usually hold in real applications, especially when the number of classifiers to be combined grows. More recently, a combiner has been proposed in order to overcome such limitations. It derives the information needed to combine the classifiers from a knowledge space, which can concurrently record the decision of all the classifiers on a suitable set of samples. This means that this space records the behavior of all the classifiers on this set, and then it is called the *Behavior-Knowledge Space* [19]. So, a Behavior-Knowledge Space is a N-dimensional space where each dimension corresponds to the decision of a classifier. Given a sample to be assigned to one of m possible classes, the ensemble of the classifiers can in theory provide m^N different decisions. Each one of these decisions constitutes one *unit* of the BKS. In the learning phase each BKS *unit* can record m different values c_i, one for each class. Given a suitably chosen data set, each sample x of this set is classified by all the classifiers and the *unit* that corresponds to the particular classifiers' decision (called *focal unit*) is activated. It records the actual class of x, say j, by adding one to the value of c_j. At the end of this phase, each *unit* can calculate the best representative class associated to it, defined as the class that exhibits the highest value of c_i. It corresponds to the most likely class, given a classifiers' decision that activates that *unit*. In the operating mode, the BKS acts as a look-up table. For each sample x to be classified, the N decisions of the classifiers are collected and the corresponding *focal unit* is selected. Then x is assigned to the best representative class associated to its *focal unit*.

6) Wernecke's (*WER*) method: it is similar to BKS and aims at reducing overtraining. The difference is that in constructing the BKS table, Wernecke [27] considers the 95 percent confidence intervals of the frequencies in each unit. If there is overlap between the intervals, the prevailing class is not considered dominating enough for labeling the unit. In this case, the "least wrong" classifier among the N members of the pool is identified, by using the confusion matrices. This classifier is authorized to assign the class to that unit. To calculate the 95 percent confidence intervals (CI), we used the Normal approximation of the Binomial distribution, as described in [22].

7) Oracle (*ORA*): when dealing with the evaluation of a MCS, it is useful to consider the performance of the so-called "*Oracle*". The *Oracle* is the theoretic MCS that correctly classifies a sample if at least one of the base classifiers is able to provide the correct classification. It is evident that for a defined set of classifiers, the performance of the *Oracle* is an upper bound of all the MCS's obtainable from the same set of classifiers by using any combiner.

Table 1. Combination Algorithms

Label	Technique	Category	Training
NB	Naive Bayes	Bayesian	Confusion Matrix
MV	Majority Voting	Vote	Confusion Matrix
WMV	Weighted Majority Voting	Vote	Confusion Matrix
D-S	Dempster-Shafer	Dempster-Shafer	Confusion Matrix
BKS	BKS	Behavior Knowledge Space	BKS
WER	Wernecke	Behavior Knowledge Space	BKS&Confusion Matrix
ORA	Oracle	Oracle	*na*

4 The Tools Used

TIE[2] is a software platform for experimenting with and comparing traffic classification techniques. TIE allows the development of algorithms implementing different classification techniques as *classification plugins* (see Fig. 1) that are plugged into a unified framework, allowing their comparison and combination. We refer the reader to [13] as regards the TIE platform as well as the TIE-L7 classification plugin, which implements a DPI classifier using the techniques and signatures from the Linux L7-filter project [1] and that we used here to produce the ground truth. In the following, instead, we describe the new features we introduced in TIE in order to develop this work.

First of all, the above-mentioned combination strategies have been implemented in TIE's *decision combiner* (Fig. 1) and a set of support scripts have been developed in order to extract from the ground-truth (generated by TIE-L7) the confusion matrix and the BKS matrix needed for training the combiners. This information is reported into configuration files that are read at run time by the combiner selected by a command-line flag.

Moreover, in order to be able to rapidly test different machine-learning approaches to traffic classification we used the WEKA tool[3] that already implements a large number of machine-learning classification techniques. We plan to implement some of such techniques as TIE classification plugins, but in order to study and test a relevant number of machine-learning approaches we implemented a "bypass" mechanism in TIE which is structured in three phases:

- A new option allows, for each flow, to dump the corresponding classification features extracted by TIE (e.g. first ten packet sizes, flow duration, etc.) along with the ground truth label assigned by TIE-L7. Such information is dumped in a file in the *arff* format used by WEKA.
- The arff file is split in the *training* and *test sets* that are used to train and test various WEKA classifiers, whose classification output is in arff format too.
- A new TIE classification plugin is able to read the output of a WEKA classifier and use it to take the same classification decision for each flow. Multiple instances of such plugin can be loaded in order to support the output of several "WEKA" classifiers at the same time.

[2] `http://tie.comics.unina.it`

[3] `http://www.cs.waikato.ac.nz/ml/weka`

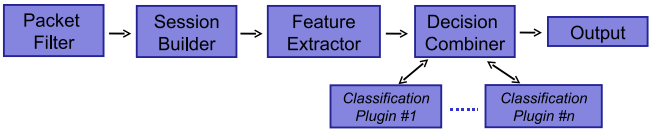

Fig. 1. TIE overall architecture

In this way TIE has a common view of both WEKA classifiers and TIE classification plugins: all these classifiers are seen as TIE plugins. This approach allowed us to easily test several classification approaches and to combine several of them plus pre-existing TIE classification plugins not based on machine-learning techniques (e.g. port-based and a novel lightweight payload inspection technique). In addition, based on the results of our studies on multi-classification we can later implement in TIE only the best performing classifiers.

Finally, in order to study the behavior of the classifiers and of the multi-classifier systems built on them, we introduced the option in TIE to generate a different file of features (in arff format) depending on the number of packets for each flow that can be used for extracting features. This option affects also the *native* TIE classification plugins that acquire the features directly by TIE's *feature extractor* (Fig. 1).

5 Data Set and Stand-Alone Classifiers

For the experimental results shown in this paper we used the traffic trace described in Table 2, in which we considered flows bidirectionally (*biflows* in the following) [13]. Each biflow has been labeled by running TIE with the TIE-L7 plugin in its default configuration, i.e. for each biflow a maximum of 10 packets and of 4096 bytes are examined.

Table 2. Details of the observed traffic trace

Site	Date	Size	Pkts	Biflows
Campus Network of the University of Napoli	Oct 3rd 2009	59 GB	80M	1M

From such dataset we then removed all the biflows labeled as UNKNOWN (about 167,000) and all the biflows that summed to less than 500 for their corresponding application label. Table 3 shows the traffic breakdown obtained[4]. This set was then split in three subsets in the following percentages:

- 20% classifiers *training set*
- 40% classifiers & combiners *validation set*
- 40% classifiers & combiners *test set*

[4] QQ is an instant messaging application.

Table 3. Traffic breakdown of the observed trace (after filtering out unknown biflows and applications with less than 500 biflows)

Application	Percentage of biflows
BITTORRENT	12.76
SMTP	0.78
SKYPE2SKYPE	43.86
POP	0.24
HTTP	16.3
SOULSEEK	1.06
NBNS	0.14
QQ	0.2
DNS	4.08
SSL	0.21
RTP	1.16
EDONKEY	19.21

Table 4. Stand-alone classifiers

Label	Technique	Category	Features
J48	J48 Decision Tree	Machine Learning	PS, IPT
K-NN	K-Nearest Neighbor	Machine Learning	PS, IPT
R-TR	Random Tree	Machine Learning	L4 Protocol, Biflow duration & size, PS & IPT statistics
RIP	Ripper	Machine Learning	L4 Protocol, Biflow duration & size, PS & IPT statistics
MLP	Multi Layer Perceptron	Machine Learning	PS
NBAY	Naive Bayes	Machine Learning	PS
PL	PortLoad	Payload Inspection	Payload
PORT	Port	Port	Ports

We considered eight different traffic classifiers, summarized in Table 4. The first five are based on Machine-Learning approaches common in literature both in terms of algorithms and discriminating features [28, 3, 25, 4]. As regards the features, in the same table *PS* stands for Payload Size, while *IPT* means Inter-Packet Time [15]. The J-48, K-NN, MLP, and NBAY classifiers consider the vectors of the first 10 PS and IPT, whereas the R-TR and RIP classifiers use statistics of PS and IPT as their average and standard deviation. The latter classifiers also take into account the transport-level protocol of the biflow, the biflow duration (in milliseconds) and size (in bytes). The *PortLoad* classifier, instead, is a light-weight payload inspection approach, recently presented in [2], that overcomes some of the problems of DPI, as computational complexity and invasiveness, at the expense of a reduced accuracy. PortLoad only uses the first 32 bytes of transport-level payload from the first packet (carrying payload) seen in each direction. Finally, we also considered the traditional traffic classifier based on transport-level protocol ports.

Table 5 shows the classification accuracy (i.e. percentage of correctly classified biflows) of every stand-alone classifier for each application and over the entire test set. The different performance of the classifiers for every application, and in particular the best accuracy score for each of them (printed in bold font), show that they have some complementarities. Moreover, the *Port* classifier has a very low overall score, which in general would suggest to avoid its use in a multi-classifier system, but we considered it because it reaches very high accuracy values for some specific applications. Finally, the last column contains the accuracies that would be obtained by the oracle, that is, by selecting for

Table 5. Classification accuracy – per-application and overall – of stand-alone classifiers (best values are in bold font) and oracle

	Classifier								
Class	J48	K-NN	R-TR	RIP	MLP	NBAY	PL	PORT	ORACLE
Bittorrent	98.8	97.4	**98.9**	98.6	55.1	79.9	7.7	21.0	99.9
SMTP	95.1	92.9	93.8	**96.0**	90.6	69.2	8.2	96.3	99.4
Skype2Skype	98.8	97.2	96.5	**99.2**	94.6	31.8	98.7	0	99.7
POP	96.0	95.0	98.7	93.9	0	79.6	29.2	**100**	100
HTTP	99.5	98.9	**99.6**	99.3	94.3	63.3	99.1	47.7	100
Soulseek	**98.6**	96.8	98.3	98.1	93	97.7	0	0	99.9
NBNS	78.4	75.9	79.9	**80.4**	9	0	0	0	85.4
QQ	0	0.7	**2.5**	0	0	0	0	0	3.2
DNS	93.6	92.6	95.3	94.4	51.1	86.2	**100**	99.7	100
SSL	96.1	93.1	95.2	93.7	69.5	68.2	**99.1**	0	99.6
RTP	**84.0**	74.1	64.5	77.3	0	41.5	0	0	92.2
EDonkey	93.0	91.7	**93.3**	91.5	72	16.1	92.9	0.1	95.7
overall	**97.2**	95.9	96.3	97.0	82.3	43.7	83.7	15.6	98.8

each biflow the correct response when this is given by at least one of the stand-alone classifiers. The overall accuracy obtained by the oracle (98.8%) shows that the combination of these classifiers can theoretically bring an improvement with respect to the best standalone classifier (97.2%).

6 Experimental Evaluation of Combiners

We experimented the combination of the stand-alone classifiers from the previous section using the algorithms explained in Section 4. When combining the classifiers we experimented with different pools of them, as shown in Table 6, where the overall accuracies for each pool and combiner are reported. The values show that in general it is indeed possible to gather an improvement through combination, as suggested theoretically by the oracle, but this improvement depends not only on the combiner adopted, but also on the choice of the classifiers. The port-based classifier has in general a negative impact on the performance of the multi-classifier system, the same happens for the Naive Bayes classifier. This behavior can be easily explained by looking at their rather low performance as stand-alone classifiers (Table 5). In particular, the performance of the MV and the WMV combiners dramatically depend on the weak performance of the Naive Bayes classifier, since the worst pools for these combiners are those in which this classifier is present. This can be explained by considering that the worst performance of the Naive Bayes classifier happen on the classes in which also the MLP classifier performs quite bad, so lowering the performance of voting-based combiners. On the contrary, the D-S combiner and the multinomial approach followed the BKS methods and are able to cope with such a situation. Finally, since the independent assumption of the base classifier does not always hold, the Naive Bayes combiner does not perform very well on the average.

The pool of classifiers achieving the best results is reported in Table 6 in bold fonts, using 6 classifiers out of the 8 tested, and closely followed by the second

Table 6. Classification accuracy of each combiner for different pools of classifiers combined (best or close to the best values are reported in bold font)

Pool of classifiers								Combiner					
J48	K-NN	R-TR	RIP	MLP	NBAY	PL	PORT	NB	MV	WMV	D-S	BKS	WER
X		X	X					54.1	96.3	96.3	96.2	**97.7**	**97.7**
X		X	X			X		55.2	96.4	96.2	96.6	**97.8**	**97.8**
X	X	X	X	X				53.5	90.7	90.7	**96.7**	96.0	96.1
X	X	X	X	X	X			80.1	72.0	72.2	96.7	**97.3**	**97.3**
X	**X**	**X**	**X**	**X**		**X**		93.5	90.8	91.0	97.0	**97.9**	**97.9**
X	X	X	X	X	X	X		80.9	72.0	72.2	97.0	**97.7**	**97.7**
X	X	X	X	X		X	X	93.6	90.5	90.8	97.1	**97.7**	**97.7**
X	X	X	X	X	X	X	X	54.6	72.8	71.2	97.1	**97.4**	**97.4**

pool in the table that includes only 4 classifiers. As for the combiners, the same table shows that the best accuracies (percentages in bold fonts) are achieved by the combiners based on the Behavior Knowledge Space (BKS and Wernecke), with the highest score of 97.9% overall accuracy. This value should be interpreted by considering the highest overall accuracy achieved by a stand-alone classifier (97.2%) and the maximum theoretically possible combination improvement set by the oracle (98.8%): an improvement equal to 43% of the maximum achievable.

We then focused our experiments on the context of early classification. This subject has been previously investigated in literature because of its important applicative characteristics, being early classification indispensable to perform online classification of traffic flows: a new traffic flow is observed on a link and the system must identify as soon as possible the application associated to it (e.g. in order to apply a security policy to the flow). In such case, therefore, classification cannot be performed with all the flow information available, and literature [6,2] has shown that there is indeed a trade-off between the ability of classifying a flow using only its first packets and the classification accuracy. In our experimental analysis we investigated the benefits of multi-classification in this context. We therefore repeated all the training and testing of the stand-alone classifiers previously considered with a variable amount of information available, that is by varying the number of packets for each flow from which the discriminant features were extracted. We also repeated the combination experiments varying the number of packets and considering the *J48,R-TR,RIP,PL* pool of classifiers. We chose this pool because its overall accuracy values are very close to those of the best pool but a reduced number of classifiers is used. Moreover, all the classifiers from this pool use algorithms with a small computational complexity. This is particularly relevant in the context of online classification.

Table 7 shows the performance of the stand-alone classifiers when 1 to 10 packets are used to extract classification features. The PortLoad classifier is based on a technique that uses at most 2 packets, therefore accuracies related to more than 2 packets are all equal, whereas the port-based classifier needs a single packet to perform classification. The best accuracy value for each number of available packets is reported in bold font. The results in the table confirm the impact of reducing the amount of available information on classification accuracy, as suggested by the literature. Moreover, the values in this table can

Table 7. Classification accuracy of each stand-alone classifier depending on the number of packets used for the feature extraction (the highest accuracy for each column is reported in bold font)

	Number of packets observed for each biflow									
Classifier	1	2	3	4	5	6	7	8	9	10
J48	62.1	**94.6**	**95.9**	96.0	**96.8**	**97.1**	**97.2**	**97.2**	**97.2**	**97.2**
K-NN	62.4	91.5	92.8	95.0	94.9	94.9	95.4	95.7	95.6	95.9
R-TR	72.7	93.4	93.6	94.9	95.3	96.8	96.0	96.0	96.1	96.2
RIP	69.5	93.7	94.7	**96.2**	96.1	96.5	96.7	96.9	96.9	96.9
MLP	43.5	71.7	81.0	82.3	82.3	82.3	82.3	82.3	82.3	82.3
NBAY	31.5	39.9	42.6	43.7	43.7	43.7	43.7	43.7	43.7	43.7
PL	**76.2**	83.7	83.7	83.7	83.7	83.7	83.7	83.7	83.7	83.7
PORT	15.6	15.6	15.6	15.6	15.6	15.6	15.6	15.6	15.6	15.6

Table 8. Classification accuracy when varying the number of available packets. Pool of combined classifiers: *J48,R-TR,RIP,PL*.

	Number of packets observed for each biflow									
Combination	1	2	3	4	5	6	7	8	9	10
MV	57.8	93.9	94.4	95.6	95.9	96.2	96.3	96.3	96.4	96.4
D-S	83.1	96.0	96.9	97.0	97.4	97.4	96.4	96.5	96.5	96.5
BKS	97.0	98.4	98.3	98.4	98.4	98.4	98.4	98.4	98.4	98.4
WER	97.0	98.3	98.2	98.4	98.4	98.4	98.4	98.4	98.4	98.4

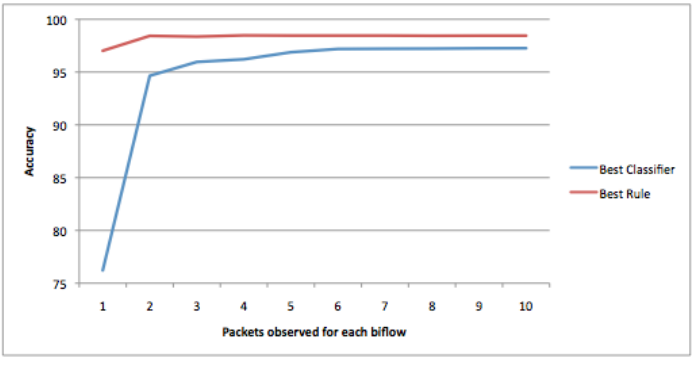

Fig. 2. Classification accuracy of the best-performing stand-alone classifier (blue line) *vs* the multi-classifier (red line)

be compared with the results of multi-classification reported in Table 8. Here, to reduce the large amount of experiments, we limited our tests to only four combiners (including the best two methods). The overall accuracy values show that in the case of early-classification the impact of multi-classification is rather significant, this is also visible in Figure 2 where we plotted for each number of available packets both the highest accuracy achieved by stand-alone classifiers (blue line) and the highest accuracy achieved by the combiners (red line): for 1 packet the combination brings an improvement of about +21% overall accuracy, while for 2 packets it is of about +4%. Such large improvements suggest that

multi-classification may be an effective strategy for the implementation of more accurate traffic classifiers able to work online in the context of early classification.

7 Conclusion

In this work we have presented and evaluated different combination techniques for traffic classification, including the BKS-based algorithms, which were not previously proposed in the traffic classification field. Moreover, for the first time we proposed the use of multi-classification in the context of early traffic classification. The preliminary experimental results here presented show several findings:

- The combination of stand-alone classifiers that present complementarities can improve the overall classification accuracy.
- The combiners based on the Behavior Knowledge Space look more promising than the others with respect to traffic classification. This behavior can be due to the fact that in our case the independent assumption of the combining classifiers does not hold. Moreover, the availability of a significant amount of training data does not cause BKS overtraining (which is one of the main drawbacks of this method).
- Even if literature has shown that the transport-level port is still a useful classification feature, combiners cannot effectively exploit the (small) discriminating power of a port-based traffic classifier. On the contrary, classifiers based on (light-weight) payload inspection complement very well with machine-learning classifiers.
- The positive impact on overall accuracy of combination is particularly significant in the context of early classification. With very strict requirements (e.g. 1 or maximum 2 packets per biflow) the performance decrease in terms of classification accuracy of the stand-alone classifiers can be almost entirely compensated by their combination.

As future work, we plan to extend this investigation to traffic traces from different links. Moreover we will focus furthermore on the exploitation of multi-classification in the context of early-classification, investigating in detail also computational complexity and timing-related issues. For this purpose we will also implement the machine-learning classifiers that were best performing as TIE plugins.

Acknowledgments

The research activities presented in this paper have been partially funded by Accanto Systems and by LATINO project of the FARO programme jointly financed by the Compagnia di San Paolo and by the Polo delle Scienze e delle Tecnologie of the University of Napoli Federico II. The authors would like to thank Antonio Quintavalle for his support to experimental activities.

References

1. L7-filter, Application Layer Packet Classifier for Linux, `http://l7-filter.sourceforge.net`
2. Aceto, G., Dainotti, A., de Donato, W., Pescapé, A.: PortLoad: taking the best of two worlds in traffic classification. In: IEEE INFOCOM 2010 - WiP Track (March 2010)
3. Alshammari, R., Zincir-Heywood, A.N.: Machine learning based encrypted traffic classification: identifying ssh and skype. In: CISDA 2009: Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, pp. 289–296. IEEE Press, Piscataway (2009)
4. Auld, T., Moore, A.W., Gull, S.F.: Bayesian neural networks for internet traffic classification. IEEE Transactions on Neural Networks 18(1), 223–239 (2007)
5. Bernaille, L., Teixeira, R.: Early recognition of encrypted applications. In: Uhlig, S., Papagiannaki, K., Bonaventure, O. (eds.) PAM 2007. LNCS, vol. 4427, pp. 165–175. Springer, Heidelberg (2007)
6. Bernaille, L., Teixeira, R., Akodjenou, I., Soule, A., Salamatian, K.: Traffic classification on the fly. ACM SIGCOMM CCR 36(2), 23–26 (2006)
7. Bernaille, L., Teixeira, R., Salamatian, K.: Early Application Identification. In: ACM CoNEXT (December 2006)
8. Bloch, I.: Information combination operators for data fusion: a comparative review. IEEE Trans. System Man and Cybernetics, Part A 26(1), 52–76 (1996)
9. Callado, A., Kelner, J., Sadok, D., Kamienski, C.A., Fernandes, S.: Better network traffic identification through the independent combination of techniques. Journal of Network and Computer Applications 33(4), 433–446 (2010)
10. Callado, A., Szabó, C.K.G., Gero, B.P., Kelner, J., Fernandes, S., Sadok, D.: A Survey on Internet Traffic Identification. IEEE Communications Surveys & Tutorials 11(3) (July 2009)
11. Carela-Español, V., Barlet-Ros, P., Solé-Simó, M., Dainotti, A., de Donato, W., Pescapé, A.: K-dimensional trees for continuous traffic classification. In: Ricciato, F., Mellia, M., Biersack, E. (eds.) TMA 2010. LNCS, vol. 6003, pp. 141–154. Springer, Heidelberg (2010)
12. Corona, I., Giacinto, G., Mazzariello, C., Roli, F., Sansone, C.: Information fusion for computer security: State of the art and open issues. Information Fusion 10(4), 274–284 (2009)
13. Dainotti, A., de Donato, W., Pescapé, A.: Tie: A community-oriented traffic classification platform. In: Papadopouli, M., Owezarski, P., Pras, A. (eds.) TMA 2009. LNCS, vol. 5537, pp. 64–74. Springer, Heidelberg (2009)
14. Dainotti, A., de Donato, W., Pescapè, A., Ventre, G.: Tie: A community-oriented traffic classification platform. In: Technical Report TR-DIS-102008-TIE, Dipartimento di Informatica e Sistemistica, Universitá degli Studi di Napoli Federico II (October 2008)
15. Dainotti, A., Pescapè, A., Ventre, G.: A packet-level characterization of network traffic. In: CAMAD, pp. 38–45. IEEE, Los Alamitos (2006)
16. Gómez Sena, G., Belzarena, P.: Early traffic classification using support vector machines. In: LANC 2009: Proceedings of the 5th International Latin American Networking Conference, pp. 60–66. ACM, New York (2009)
17. Gordon, J., Shortliffe, E.: The dempster-shafer theory of evidence. In: Buchanan, B.G., Shortliffe, E. (eds.) Rule-Based Expert Systems, pp. 272–292. Addison-Wesley, Reading (1984)

18. He, H., Che, C., Ma, F., Zhang, J., Luo, X.: Traffic classification using en-semble learning and co-training. In: AIC 2008: Proceedings of the 8th Conference on Applied Informatics and Communications, pp. 458–463. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point (2008)
19. Huang, Y.S., Suen, C.Y.: A method of combining multiple experts for the recognition of unconstrained handwritten numerals. IEEE Trans. Pattern Analysis and Machine Intelligence 17(1), 90–94 (1995)
20. Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K.: Internet traffic classification demystified: myths, caveats, and the best practices. In: CoNEXT 2008: Proceedings of the 2008 ACM CoNEXT Conference, pp. 1–12. ACM, New York (2008)
21. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Analysis and Machine Intelligence 20(2), 226–239 (1998)
22. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
23. Kuncheva, L.I., Bezdek, J.C., Duin, R.P.W.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recognition 34(2), 299–314 (2001)
24. Nguyen, T.T., Armitage, G.: A Survey of Techniques for Internet Traffic Classification using Machine Learning. IEEE Communications Surveys and Tutorials (2008) (to appear)
25. Park, J., Tyan, H.R., Kuo, C.C.J.: Ga-based internet traffic classification technique for qos provisioning. In: International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 251–254 (2006)
26. Szabo, G., Szabo, I., Orincsay, D.: Accurate traffic classification, pp. 1–8 (June 2007)
27. Wernecke, K.D.: A coupling procedure for discrimination of mixed data. Biometrics 48, 497–506 (1992)
28. Williams, N., Zander, S., Armitage, G.: Evaluating machine learning algorithms for automated network application identification. Tech. Rep. 060401B, CAIA, Swinburne Univ. (April 2006)
29. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. ACM SIGCOMM CCR 36(5), 7–15 (2006)
30. Wright, C.V., Monrose, F., Masson, G.M.: On inferring application protocol behaviors in encrypted network traffic. Journal of Machine Learning Research 7, 2745–2769 (2006)
31. Xu, L., Krzyzak, A., Suen, C.Y.: Method of combining multiple classifiers and their application to handwritten numeral recognition. IEEE Trans. Syst. Man Cybernetics 22(3), 418–435 (1992)