Network Protocol Verification by a Classifier Selection Ensemble

Francesco Gargiulo[1], Ludmila I. Kuncheva[2], and Carlo Sansone[1]

[1] Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II, Via Claudio, 21 I-80125 Napoli, Italy
{francesco.grg,carlosan}@unina.it
[2] School of Computer Science, University of Bangor, UK
l.i.kuncheva@bangor.ac.uk

Abstract. Classical approaches for network traffic classification are based on port analysis and packet inspection. Recent studies indicate that network protocols can be recognised more accurately using the flow statistics of the TCP connection. We propose a classifier selection ensemble for a fast and accurate verification of network protocols. Using the requested port number, the classifier selector directs the decision to an ensemble member responsible for this port. The chosen ensemble member ramifies the decision further using the "sign pattern" of the first four packets. Finally, a decision tree classifier labels the flow as 'accepted' or 'rejected' using the sizes of the first four packets. The ensemble has modular architecture which allows further modules to be individually trained and added. The classifiers were cross-tested using designated training and testing data of network traffic traces from three institutions. The results show that accuracy need not be sacrificed for speed of classification, and that the protocol classification is robust from one network to another.

1 Introduction

Network traffic classification is important for ensuring quality of service (QoS), security, optimal priority assignment, and general traffic management. Fast and accurate catching of an inadequate application protocol is imperative when security is concerned. Ideally, a smart firewall would block such protocols at their onset. The basic traffic unit we consider in this paper is termed a **flow**. We define a flow as a bi-directional ordered sequence of packets with the same IP addresses and TCP port numbers. A flow is either accepted by the classifier as one of the valid protocols or rejected as unknown or known but non-allowed. Usually, to label a flow, the information of all packets is needed. For this information to be extracted, the flow must have entered the network, so the classification would come when it may be too late to decline service.

The main approaches to traffic classification are port-based, payload-based and statistical.

The **port-based** approach uses the assumption that each port is associated with one protocol [13]. The protocol classification is made simply by reading the port number. However, this method is not suitable for networks with dynamic port allocation. For such networks an undesirable *application* may be directed through a port conventionally

associated with an acceptable application. Thus the classification of applications cannot be done from the port number only.

Passing a non-allowed protocol through an accepted port may also be a result of malicious activity. Another problem with the port-based approach is that it will not prevent 'tunnelling', i.e., protocol X embedded within and disguised as protocol Y, where Y an accepted protocol for the network while X might not be [3, 5].

The **payload-based** approach looks at the content of the packets [15]. The protocol verification is more accurate but requires more computational resources. An adverse issue associated with the payload-based approach is related to the privacy of the content. Also, when the traffic is encrypted the approach will not work [9].

The **statistical** approach takes characteristics of the flow as the input features, e.g., number of packets; their length; minimum, maximum and average length, etc. Various statistical classifiers have been tried on the extracted features, e.g., Bayesian networks [1], Support Vector Machines [12], Gaussian Mixture Modelling and Decision Trees [6]. Statistical tests such as goodness of fit, χ^2 and Kolmogorov-Smirnov have also been tried [7] to single out anomalies such as incidents of `skype` application within an `http` protocol. Extracting discriminative features is a major focus of the works on statistical traffic classification [1, 12]. It is worth noting that most of the statistical approaches proposed so far [1, 6, 16] need to be retrained when the number of allowed protocols varies.

The main problem of both the payload-based and the statistical approaches is that traffic flows can only be classified once they have passed through the system completely. This limits their applicability for online classification. Anyway, it has been recently shown that accurate classification can be achieved using only the sizes and the directions of the first few packets of the TCP connection [2, 6].

The requirement for operational speed brings in the idea of classifier *selection* ensemble where only one of a set of 'experts' has to make a decision [11]. The ensemble consists of member classifiers (experts) and an 'oracle' that authorises one of the classifiers to pass its decision as the ensemble decision. The oracle may have pre-defined regions of competence for the classifiers [14] or dynamically allocated regions [17]. With dynamic competence allocation the suggested labels for the object of interest $\mathbf{x}$ are further analysed using past data. The classifier whose predicted label has been most accurate for the neighbourhood of $\mathbf{x}$ is chosen to produce the ensemble decision. While dynamic allocation has been found to be very successful, it requires that all ensemble members classify $\mathbf{x}$. Besides, past data needs to be stored and searched through. Since we are aiming at a fast classification, we propose to use pre-defined competence regions and train a bespoke classifier for each region. We propose to use the port number (pretend protocol name) as the oracle determining the regions of competence. The directions and sizes of the first four packets of the TCP flow are then used as the features in a further 2-stage classifier. The features and the modular architecture were chosen so that the classification is both fast and accurate, and new modules can be trained and added to the system without re-training any already trained part.

The rest of the paper is organised as follows. Section 2 describes the proposed system and Section 3 shows the experimental results. Our conclusions and future plans are given in Section 4.

2 A Classifier Selection Ensemble for Network Traffic Classification

We propose the following classifier selection ensemble. Each port number has a classifier trained to verify that the traffic through that port follows the expected protocol. Thus the classifier selector only checks the port number and directs the flow to the respective classifier. If the port number is not one of the pre-defined set the flow is rejected.

2.1 The Features

Following [1],[6] and [7], we propose to use only the first four packets and record the following features:

- x_0, the pretend name of the flow guessed from the port number;
- x_1, x_2, x_3, x_4, the directions of the first four packets, $x_i \in \{0, 1\}$, where 0 means that the packet is transferred from server to client, and 1, from client to server;
- s_1, s_2, s_3, s_4, the payload sizes of the first four packets, where s_i are positive integers. As in [6], we leave off packets without payload because they are mostly used to exchange connection state information.

The generic architecture of the classifier ensemble is shown in Figure 1.

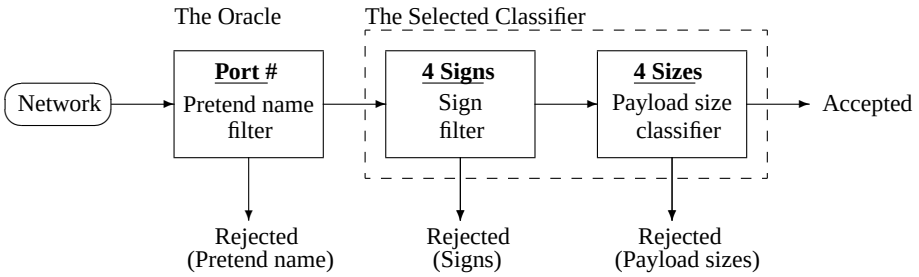


Fig. 1. The generic classifier ensemble architecture. Only the selected ensemble member is shown. Each ensemble member is implemented as a cascade classifier with 2 stages.

2.2 Classifier selector: The pretend name

The port information is often neglected in classifying network traffic flow [6]. In this study we use this information in two ways. First, we label as unknown all flows whose port number (pretend name) does not appear in a pre-set list. Second, the pretend name is used to branch out the classification to a bespoke classifier. This partitions the feature space on the value of x_0, thereby reducing a multi-class problem to a two-class problem: match versus mismatch of the pretend name.

2.3 Ensemble classifier – Stage 1: Sign filter

To illustrate the consecutive steps of the system design we use a data set consisting of network traffic traces at the University of Brescia, Italy [6]. The data set is divided into training (58 478 instances) and testing (75 163 instances). The known protocols in the training data are: `pop3`, `smtp`, `http`, `msn`, `ftp` and `BitTorrent`. The testing data contains an additional class named 'unknown'. Table 1 shows a summary of the training data.

Table 1. Summary of the network traffic data (training) from the University of Brescia, Italy

Signs	Protocol and port number					
	`pop3`	`ftp`	`smtp`	`msn`	`BitTorr`	`http`
1 2 3 4	110	21	25	1863	6881	80
0 0 0 0	0	138	16	0	0	3
0 0 0 1	1	75	55	0	0	0
0 0 1 0	21	216	543	0	0	0
0 0 1 1	0	0	4	0	1	0
0 1 0 0	749	21	604	1	0	0
0 1 0 1	18823	5845	18186	0	1	0
0 1 1 0	17	1	18	0	1	0
0 1 1 1	0	0	1	0	0	0
1 0 0 0	0	0	0	328	23	5348
1 0 0 1	0	0	0	30	520	240
1 0 1 0	0	0	0	660	3609	826
1 0 1 1	0	0	0	4	753	12
1 1 0 0	0	0	0	1	8	427
1 1 0 1	0	0	0	0	87	76
1 1 1 0	0	0	0	0	9	108
1 1 1 1	0	0	0	0	45	23

The table shows that groups of protocols can be distinguished by the signs of the first four packets. For example, protocols `msn` (1863), `BitTorrent` (6881) and `http` (80) hardly ever begin with a packet from sever to client ($x_1 = 0$). The 7 exceptions in the table (out of 13 144 flows) may be thought of as recording mistakes. The distribution of the data suggests that the four signs can be used to filter out very quickly protocols that clearly do not match their pretend name. This constitutes the second stage of the cascade classifier, called the sign filter. A rejection threshold p_r is chosen next. The occurrences if each protocol (a column in Table 1) are scaled to form a probability distribution across the 16 sign combinations. All values with likelihood less than the chosen threshold are treated as outliers. Thus for each protocol, there are "impossible" sign combinations which make up the filter for that pretend name. For example, with threshold $p_r = 0.02$, the "allowed" combination of signs for the `http` protocol (80) are 1000, 1001, 1010, and 1100. All other protocols will be rejected by the sign filter.

2.4 Ensemble classifier – Stage 2: Decision tree classifier using payload sizes

A separate classifier is trained for each sign combination that passes through the sign filter. For example, consider a protocol with pretend name `pop3` (110) and sign pattern 0100. Figure 2 shows a scatterplot of the data of network traffic traces from the Lawrence Berkeley National Laboratory (LBNL) with sign pattern 0100. The data points from `pop3` form a distinctive oblong cluster, away from the '+'–shaped cluster of the other protocols with the same sign pattern. [3]

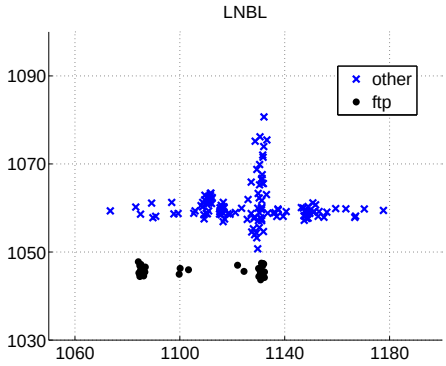


Fig. 2. Scatterplot of the LBNL training data with sign pattern 0100 in the plane spanned by the first two size features, s_1 and s_2.

Figure 3, on the other hand, displays the same scatterplot for the training data from the University of Brescia (UNIBS) and the Cooperative Association for Internet Data Analysis (CAIDA). The `ftp` protocol is also shown because it is present in these two data sets.

The figures show that:

1. Protocols `pop3` and `ftp` are very close to one another. To build a good classifier, both protocols should be present in the training data.
2. The classes have intricate irregular shapes (UNIBS and CAIDA data) which suggests that a decision tree classifier may fare well for this problem.
3. The class 'other' is different from one data set to another. The geometrical configuration of this class will depend on what protocols are accepted in the network. Note that class `ftp` in Figure 3 is, in fact, part of class 'other'.
4. Curiously, even the same class (`pop3`) has different appearances for the three different networks. This means that an ensemble has to be trained individually for each network. Hence an ensemble trained on the UNIBS data cannot be expected to be overly accurate on LBNL and CAIDA data, and vice versa.

[3] Since the payload size is a discrete variable, multiple points may share both coordinates (x, y). A small random noise is added to all data so that the points move slightly off (x, y) in a random way. This will create an impression of the density of the data.

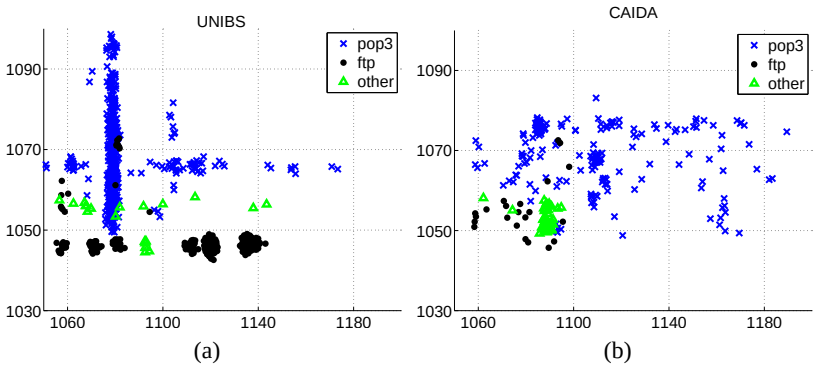


Fig. 3. Scatterplot of the UNIBS and CAIDA training data with sign pattern 0100 in the plane spanned by the first two size features, s_1 and s_2.

Our ensemble differs from the stereotype in that a flow can be classified as "unaccepted" at each stage: the combiner (classifier selector), the sign filter and the decision tree classifier. This speeds up the decision process, which is important for online traffic classification.

3 Experimental Evaluation

A summary of the content of the three data sets used in this study is given in Table 2.

Table 2. Protocols and number of flows in the three data sets

		UNIBS		CAIDA		LBNL	
Protocol	Port	Training	Testing	Training	Testing	Training	Testing
`pop3`	110	19611	19940	9591	2386	1172	1426
`smtp`	25	19427	19480	11831	20722	20825	1304
`http`	80	7063	4928	5930	12459	81984	38228
`ftp`	21	6296	14458	1652	16202	–	–
`BitTorrent`	6881	5057	7412	–	–	–	–
`msn`	1863	1024	1033	–	–	–	–
`netbios-ssn`	139	–	–	4575	10113	–	–
`https`	443	–	–	25427	7896	18013	3283
`oms`	4662	–	–	–	–	1716	2491
`imap4`	993	–	–	–	–	7677	422
other		–	7912	–	1423	–	4584

For the experimental evaluation we chose the three protocols that are common to all three data sets, `http`, `smtp` and `pop3`. A pilot experimental study on the UNIBS training data, using Weka [8], reinforced our choice of the decision tree classifier. Fur-

ther to that, we carried out the following sets of experiments, where A, B, C refer to the three protocols and a, b, c refer to the three data sets:

(1). Train a classifier for a protocol with pretend name A using training data set a. To do this, assume that all flows have pretend name A so as to form a training data set with class labels 'A' and 'other'. Identify the sign patterns relevant for protocol A. Filter the data for each sign pattern. Using this data, train a decision tree classifier to distinguish between the two classes.

(2). Test the classifier on testing data sets b and c.

(3). Repeat steps (1) and (2) for protocols B and C

In this imaginary scenario, all traffic takes the pretend identity of the protocol in question. In reality, the likelihood of class 'other' will be much smaller. Therefore the classification accuracy achieved in the experiments is a pessimistic estimate of the accuracy expected during operation. Because of the specific experimental set-up, a direct comparison with classification accuracies obtained elsewhere may be misleading.

The cross-data classification accuracies are shown in Table 3.

Table 3. Classification accuracy of the ensemble member classifiers (cross-data training and testing).

	`pop3`			`smtp`			`http`		
	UNIBS	LBNL	CAIDA	UNIBS	LBNL	CAIDA	UNIBS	LBNL	CAIDA
UNIBS	95.69	99.32	75.57	98.75	99.01	96.67	97.93	88.14	97.75
LBNL	78.99	99.11	76.53	81.46	99.17	95.39	99.29	95.49	94.78
CAIDA	83.42	99.21	99.83	80.44	99.34	99.54	98.50	93.68	97.94

Figure 4 gives the plot of Sensitivity versus $1-$Specificity for the protocol `pop3`.[4] The different markers correspond to the sources of the training data. Two of the anomalies with a substantial slip in the classification accuracy are indicated. The reasons for the inadequate classification can be illustrated with the findings in Figures 2 and 3. The presentation of the `pop3` protocol is very different from one network to another. In addition, the classification of `pop3` is further impaired by its similarity to `ftp`. The two marked points are for ensembles trained on UNIBS and LBNL and tested on CAIDA. The `pop3` protocol have similar appearance in the UNIBS and LBNL data and a different, more scattered, appearance in the CAIDA data (Figure 3 (b)). Thus the ensembles trained on UNIBS and LBNL data are ill-equipped to classify the version of `pop3` in CAIDA.

Figure 5 plots Sensitivity versus $1-$Specificity for protocols `smtp` and `http`. There are two inaccurate classifiers for `smtp`. This time the mismatch is between the `smtp` traffic in the UNIBS and LBNL data. The two points that lie closer to the diagonal line in subplot (a) are the cross-testing UNIBS-LBNL and LBNL-UNIBS.

[4] Sensitivity is the proportion of positives detected out of all positives (proportion correctly verified protocols). Specificity is the proportion of true positives out of all classified as positives (proportion of true protocols out of all non-rejected protocols).

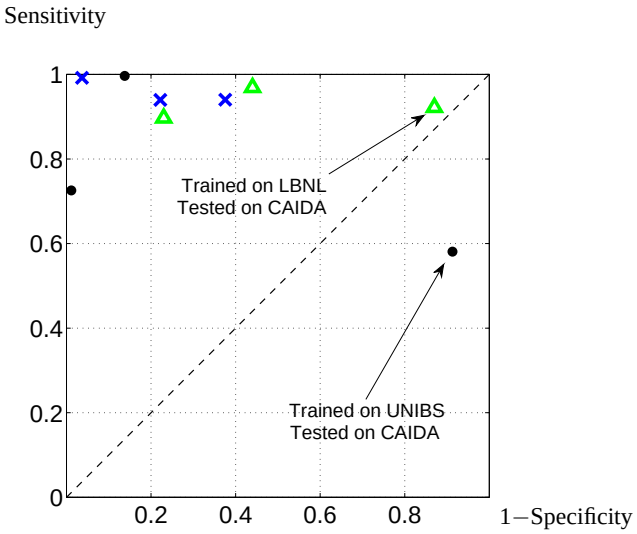
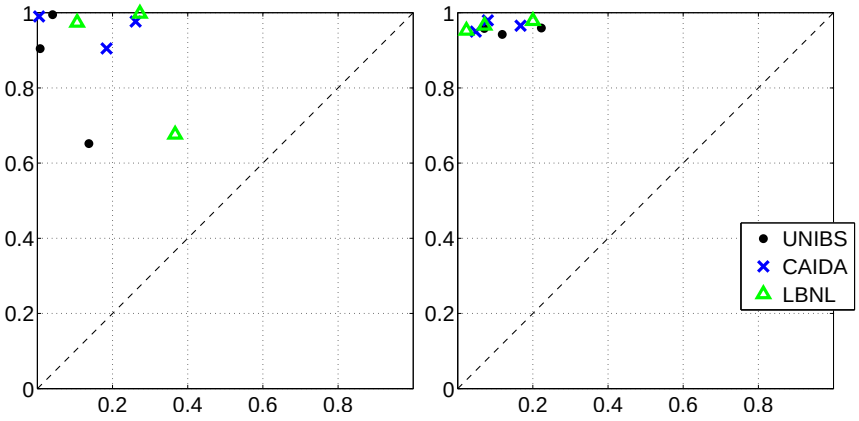


Fig. 4. Sensitivity-Specificity plot for protocol `pop3`. The markers indicate the training data source.

The experimental results show that the ensemble members have accuracies comparable to those in the state-of-the-art literature on traffic classification [1, 6, 15]. The high accuracy of the cross-data experiment, with a few exceptions discussed earlier, indicates that the statistical approach to traffic classification is robust across networks, so universal solutions can be sought. This reflects the fact that the network protocols have standard definitions, and the features we are using are not affected by differences in network configurations, traffic intensity or delays.

4 Conclusion

We propose a classifier selection ensemble for network traffic verification. upon receiving the first four packets of a flow, the classifier selector directs the decision to an ensemble member based upon the requested port number. The classifier responsible for this port number ramifies the decision further using the "sign pattern" of the four packets. A decision tree classifier labels the flow as 'accepted' or 'rejected'. The 'accepted' class is the protocol conventionally associated with the requested port number. The flow can be classed as 'rejected' at every stage of the ensemble classification: the oracle rejects non-allowed port requests, the sign filter rejects flows whose sign patterns are highly unlikely, and, finally, the decision tree classifier is responsible for the fine discrimination based on the sizes of the first four packets. Without looking at the payloads, it is difficult to detect "tunnelling" behaviours where an unaccepted protocol is wrapped and carried within an accepted one. In our system, tunnelling may pass through the port and the sign filters but then land as an outlier in the space of the payload sizes. Our system is expected to reject the protocol at this final stage.



(a) smtp (b) http

Fig. 5. Sensitivity versus 1−Specificity for protocols `smtp` and `http`.

The classifiers were cross-tested using designated training and testing data of network traffic traces from three institutions: UNIBS, LBNL and CAIDA. The results show the robustness of the statistical approach to traffic classification.

While the ensemble accuracy is comparable to that reported in the literature [1, 6, 15] the proposed ensemble has the following advantages:

– Compared to classifiers that use the whole flow, our protocol verification is quick, as the sequence of decisions is based on the port number and the directions and sizes of the first four packets of a flow. Should operational speed permit it, the ensemble can be used online; a flow can be stopped before the application is processed by the network.
– The proposed ensemble needs only two parameters. First, we must choose a threshold for selecting the valid sign patterns (Here we used 2%. All sign patterns with likelihood higher than the threshold will merit separate decision tree classifiers. Flows with unlikely sign patterns are rejected.) The second parameter is the level of pruning for the decision tree classifiers.
– The structure of the ensemble is modular. Classifiers can be trained and added without disturbing the rest of the ensemble. For example, if a new port joins the list of allowed ports, an ensemble member can be trained separately for this port. Also, if the traffic changes, e.g., by allowing a new application through an existing protocol, and an unlikely sign pattern starts appearing more often, a separate classifier can be trained for this sign pattern and added to the ensemble.

One interesting future research direction comes from the fact that network traffic changes by definition, and so would the class descriptions (‘accepted’ and ‘rejected’) [10]. To respond to these changes, the ensemble should be further developed so as to cope with concept drift. Moreover, we are planning to implement our protocol verification system in an online platform such as the one described in [4].

References

1. T. Auld, A.W. Moore, and S.F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Trans. on Neural Networks*, 18(1):223–239, Jan. 2007.
2. L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2006. ACM.
3. M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Detecting http tunnels with statistical mechanisms. In *Proc. IEEE International Conference on Communications ICC '07*, pages 6162–6168, 2007.
4. A. Dainotti, W. de Donato, A. Pescapè, and G. Ventre. Tie: a community-oriented traffic classification platform. Technical Report TR-DIS-10-2008, Dipartimento di Informatica e Sistemistica, University of Napoli Federico II, 2008.
5. M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Detection of encrypted tunnels across network boundaries. In *Proc. IEEE International Conference on Communications ICC '08*, pages 1738–1744, 19–23 May 2008.
6. A. Este, F. Gargiulo, F. Gringoli, L. Salgarelli, and C. Sansone. Pattern recognition approaches for classifying ip flows. In N. da Vitoria Lobo, T. Kasparis, F. Roli, J.T.-Y Kwok, M. Georgiopoulos, G.C. Anagnostopoulos, and M. Loog, editors, *SSPR/SPR*, volume 5342 of *Lecture Notes in Computer Science*, pages 885–895. Springer, 2008.
7. E.P. Freire, A. Ziviani, and R.M. Salles. On metrics to distinguish skype flows from http traffic. In *Proc. Latin American Network Operations and Management Symposium LANOMS 2007*, pages 57–66, 2007.
8. S.R. Garner. Weka: The waikato environment for knowledge analysis. In *Proc. of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.
9. R. Holanda Filho, M.F. Fontenelle do Carmo, J. Maia, and G.P. Siqueira. An internet traffic classification methodology based on statistical discriminators. In *Proc. IEEE Network Operations and Management Symposium NOMS 2008*, pages 907–910, 2008.
10. L.I. Kuncheva. Classifier ensembles for changing environments. In F. Roli, J. Kittler, and T. Windeatt, editors, *Multiple Classifier Systems*, volume 3077 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.
11. L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
12. Z. Li, R. Yuan, and X. Guan. Traffic classification - towards accurate real time network applications. In *HCI (4)*, pages 67–76, 2007.
13. D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy. The coralreef software suite as a tool for system and network administrators. In *LISA '01: Proceedings of the 15th USENIX conference on System administration*, pages 133–144, Berkeley, CA, USA, 2001. USENIX Association.
14. L. A. Rastrigin and R. H. Erenstein. *Method of Collective Recognition*. Energoizdat, Moscow, 1981. (In Russian).
15. F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus. Lightweight, payload-based traffic classification: An experimental evaluation. In *Proc. IEEE International Conference on Communications ICC '08*, pages 5869–5875, 2008.
16. N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, 2006.
17. K. Woods, W.P. Kegelmeyer, and K.W. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):405–410, 1997.