This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

Early Identification of Peer-To-Peer Traffic

Béla Hullár*, Sándor Laki†
*Dept. of Physics of Complex Systems
†Dept. of Information Systems
Eötvös Loránd University, Budapest, Hungary
Email: {hullarb, lakis}@elte.hu

András György
Machine Learning Research Group
MTA SZTAKI Institute for Computer Science
and Control, Budapest, Hungary
Email: gya@szit.bme.hu

***Abstract*—To manage and monitor their networks in a proper way, network operators are often interested in identifying the applications generating the traffic traveling through their networks, and doing it as fast (i.e., from as few packets) as possible. State-of-the-art packet-based traffic classification methods are either based on the costly inspection of the payload of several packets of each flow or on basic flow statistics that do not take into account the packet content. In this paper we consider the intermediate approach of analyzing only the first few bytes of the first (or first few) packets of each flow. We propose automatic, machine-learning-based methods achieving quite good early classification performance on real traffic traces generated from a diverse set of applications (including several versions of P2P TV and file sharing), while requiring only limited computational and memory resources.**

I. Introduction

The assurance of quality of service in IP-based networks, especially, the Internet, is one of the most important challenges in today's network developments. To ensure quality of service, network operators have to analyze the traffic traveling through their networks. This analysis can lead to more efficient resource allocation as well as can help planning the expansion and development of the network. This is the reason why traffic classification is in the forefront of research on next generation networks.

A decade ago traffic classification was very simple, since applications used their well defined port numbers assigned by IANA and the network operators just looked at the transport layer port number for identification. Since then the Internet has became the largest existing entertainment medium in the world. Many new applications have appeared from voice over IP (VoIP) and IP TV to peer-to-peer (P2P) file sharing systems that have a revolutionary effect on people's everyday life. Nowadays these applications are responsible for the majority of network traffic. To by-pass firewalls and routers, or to hide their presence, they use dynamic port numbers or well-known trusted ports (such as HTTP or SMTP ports), limiting the achievable performance of port-number-based classification.

Current state-of-the-art methods for traffic classification are based on the analysis of (i) packet payloads generated by the application; (ii) some network layer or transport layer features; (iii) host-behavior, or some mixtures of the above. Approach (i), called deep packet inspection (DPI) aims at identifying typical protocol patterns in the messages of different applications. These methods work reasonably well in practice, but their disadvantages are well-known: The performed pattern matching is resource-intensive, requiring both computational time for scanning packet payloads and memory for storing large databases of application (or protocol) specific signatures. Moreover, keeping the signature databases up-to-date is a cumbersome task usually requiring human expertise, since new network-based applications are born day-by-day and the current ones evolve constantly. To eliminate these drawbacks, statistical payload inspection methods were proposed recently [1], [2] that learn relevant protocol patterns automatically from traffic traces (labeled with the corresponding applications).

In some situations packet content may not be available due to resource limitations or encryption. Then the less resource intensive approaches (ii) and (iii) may be applied. In the former, machine learning methods (such as decision trees, Bayes classifiers, SVM, k-NN, etc.) are used to train classifiers to estimate the traffic type based on some features extracted from some network or transport layer data, such as the number of packets in the flow, the average packet size, the count of Push flags, inter-arrival time statistics, etc. In the last few years, numerous feature sets and machine learning algorithms were examined in the literature, see, e.g., the thorough review [3] or the systematic comparison of existing methods in [4]. However, the results are not general in the sense that the efficiency of different feature sets highly depends on the generating applications as well as the location where the flow data was captured [5]. Finally, approach (iii) is based on the analysis of host behavior instead of individual network connections.The basic idea behind this solution is that different applications or application groups have different communication habits, for example, a client application has few outgoing connections, a server has numerous incoming connections, while P2P applications have many connections in both directions, giving rise to methods analyzing the connection patterns of host interactions [6] or utilizing the assumption that similar hosts tend to be related to one another [7].

An important observation is that classification algorithms analyzing packet content outperform others in case of unencrypted traffic, while the rest are much less resource intensive, hence are much cheaper. Furthermore, identifying the application type as early as possible is very important from several aspects (from security to improved quality of service) allowing the operator much faster reactions. To this end, the first few bytes of each flow are used for making decisions in [2], while [8] performs classification from some simple descriptors of the first four packets of each flow.

In this paper we tackle the above problems, and, extending and improving the work of [2], demonstrate that automated

978-1-61284-233-2/11/$26.00 ©2011 IEEE

traffic classification based on the first few bytes of the first (or first few) packet(s) of a connection is possible with high accuracy. In addition, our approach requires far less computational and memory resources than usual DPIs with comparable performance, and no human expertise is needed in the development (training) of the method. It is worth noting that [9] and [4] have also found that analyzing only the first 16 bytes of a P2P connection is usually sufficient for a well designed DPI to identify P2P traffic (except for Gnutella).

The performance of our traffic classification algorithm is measured on a diverse data set recorded in our lab over LAN, WiFi and 3G links. Although this method can be criticized of generating artificial user behavior and, hence, traffic patterns, we believe that the content of the first few packets is not really affected (classification performance was measured on traffic traces collected in a similar fashion also in, e.g., [10]). Furthermore, several papers, such as [11] consider the potential disadvantages of using DPI tools as the source of ground truth, since such engines may misclassify or simply ignore traffic flows (this observation is also supported by our measurements in Section IV, although the best DPIs examined provide very good results for most traffic types considered).

The rest of the paper is organized as follows. Automatic payload-based traffic classification methods together with our new algorithms are introduced in Section II. Section III describes our data collection methodology. A comprehensive performance analysis of our algorithms is presented in Section IV. Finally, conclusions are drawn and future work is outlined in Section V.

II. Payload-Based Traffic Classification and an Algorithm for Early Identification

Since the majority of network traffic is still unencrypted and the protocol-format-based identification works well in practice, it is worth exploiting the potential of the payload analysis with advanced techniques. A very promising method in the literature is the KISS algorithm [1] that uses χ^2-statistics as features computed from the first few bytes of the first several packets of each flow. The efficiency of this solution is demonstrated on different applications, such as P2PTV clients, where, by combining the payload-based classification with other techniques, based on, e.g., host similarities, they achieve 99% accuracy in flow classification. The main disadvantage of this approach is that many packets (80) are required for the feature computation to achieve the above results (and, accordingly, the measurements are restricted to flows that generate more than 80 packets), so the method is not applicable for early classification.

Simple statistical methods are considered in [2] for traffic classification from the first 64 bytes of each flow. The beginning of the payload is analyzed based on a stationary memoryless model and a first-order Markov model over the bytes, respectively, as well as using the graph-based common substring method. These methods show promising results on a test set containing a few basic protocols, such as HTTP, DNS or NTP.

In this paper we extend the latter approach in several ways: we apply more involved models, such as context trees or random forests, analyze only the first few bytes of a few packets per flow, and consider more recent applications, such as P2PTV and file sharing systems.

A. Early classification algorithms

We examine the problem of classifying traffic flows from very limited information: the first N bytes of the payload of the first n packets of each flow, where both N and n are small. Combining the data obtained from the analyzed packets, each flow can be described by nN bytes (called feature vector), and our goal is to determine the generating application from this data only. In order to do so, we consider a few classification algorithms (including ones based on zero- and first-order Markov modeling mentioned above). All of these approaches work in the following, automated way: A sample of flows generated from the traffic classes to be identified is collected and each flow is labeled with its true class (i.e., the application generating the flow). Then the classification methods are trained on this sample (called the training data). Finally, the trained classifiers are used to estimate the traffic class from the feature vectors.

In the following we briefly overview the applied techniques including Markov-model-based classification methods of different orders as well as random forests: In the case of Markovian methods we build a stochastic (Markov) model for each traffic class i during the training that provides an estimate $p_i(\mathbf{x})$ of the conditional probability that we observe a feature vector $\mathbf{x} = (x_1, \ldots, x_{nN})$ if the flow was generated from class i. Then a maximum likelihood decision is applied based on the models: that is, a flow with a feature vector $\mathbf{x}$ is classified to $\operatorname{argmax}_i p_i(\mathbf{x})$. The estimates $p_i(\mathbf{x})$ are obtained in different ways for different orders:

KT-ESTIMATOR: This is a zero-order model where the bytes in each feature vector are assumed to be independent and identically distributed. The distribution is estimated by the Krichevsky-Trofimov estimator [12] instead of the empirical frequencies to robustly handle the case when a byte value appears in the test data that is not present in the training data. More formally, if n_i and $n_i(x)$ denote the total number of bytes and the number of times byte x appears in the training data for class i, respectively, then the conditional probability $p_i(x)$ is estimated by $p_i(x) = (n_i(x) + 1/2)/(n_i + 128)$, and $p_i(\mathbf{x}) = \prod_{j=1}^{nN} p_i(x_j)$.

MARKOV: Here we apply a first order Markov-model (similar to [2]) where the distribution of the first byte $p_{i,1}(x_1)$ as well as the transition probabilities $p_i(x_{j+1}|x_j)$ are estimated by the corresponding empirical frequencies in the training data for class i. The conditional probability of a vector $\mathbf{x}$ is then estimated as $p_i(x) = p_{i,1}(x_1) \prod_{j=1}^{nN-1} p_i(x_{j+1}|x_j)$.

MARKOVKT: This method is very similar to the previous one except that we apply the Krichevsky-Trofimov estimate in all cases instead of empirical frequencies.

CONTEXT TREE WEIGHTING METHOD (CTW): CTW is a state-of-the-art lossless data compression algorithm [13] that

produces a variable-order Markov model which can be used to obtain the conditional probability of a feature vector $\mathbf{x}$ for each traffic class i. In the experiments the maximum depth of the tree was set to 5 (i.e., the method can produce an at most fifth-order model). The variability in the memory of the model can decrease complexity as well as improve prediction accuracy if the number of data points containing certain byte patterns is not sufficiently large.

RANDOM FOREST (RF) is a state-of-the-art classification method [14] that averages the estimate of many decision trees. Each individual decision tree is built on a bootstrap sample of the original data, separating randomly selected leafs according to feature values that provide the best split among a given number of randomly selected features until each leaf contains only one feature point (in our case the features are the bytes of the feature vector $\mathbf{x}$). Using several decision trees at the same time can improve the classification accuracy significantly and may make the method more robust against noise.

III. DATA COLLECTION

Data collection and labeling are crucial parts of traffic classification studies. A generally prevailing method in the literature is to collect (unlabeled) traffic traces from different sources (network operators, campus networks, testbeds, etc.), and then to label each flow with the generating protocol or application based on some DPI tools or other heuristics. The obvious advantage of this method is that the resulting labeled traces are real. On the other hand, the result of the labeling phase may be unreliable due to the shortcomings of DPI engines, and significant portion of the applications/protocols may be dropped out from the analysis because of its small incidence or because the applied labeling method is not able to recognize it (see, e.g., [11]).

According to the above considerations, we used an active data collection technique: traffic traces were captured in a fully controlled environment and the flows were marked with the real applications by a modified kernel module; this methodology results in an unquestionable source for the ground truth (similar active data collection can be found in, e.g., [10]).

In this work we concentrate on P2P protocols, as they are less studied than traditional protocols (like HTTP, FTP, etc.), play an increasing role in the Internet, and have become responsible for a large portion of Internet traffic recently. We have generated and recorded traffic of different P2P applications in two main groups: P2PTV clients and P2P file-sharing clients. The P2PTV group contains four clients with different proprietary protocols (PPLive, PPStream, SopCast, TVUPlayer), while the group of file-sharing applications consists of three client applications with different protocols (BitLord as a BitTorrent client, Emule as an eDonkey client and LimeWire as a Gnutella client). To be able to analyze the performance of our algorithms on basically different traffic types, traces of other applications, such as HTTP, Skype, gaming, encrypted torrent, etc., have also been collected.

Two data sets are used during the evaluation: a smaller set, WIRELESS, containing traces with full payload, recorded in wireless environments (WiFi and 3G) in July 2010, and a larger set, LAN, recorded over high-speed LAN connection during an earlier experiment in November 2009, containing only the first 16 bytes of the payload of each packet. A detailed description of the recorded traffic, showing the number of flows and the amount of traffic carried, is given in Table I.

Application Class	WIRELESS		LAN	
	Flows	Bytes	Flows	Bytes
PPLive	0	0	5100	2100 Mb
PPStream	6900	538 Mb	5700	3545 Mb
SopCast	1700	608 Mb	48000	24855 Mb
TVUPlayer	3250	1365 Mb	2800	1872 Mb
BitLord	1900	2253 Mb	26500	23923 Mb
Emule	8700	540 Mb	59500	6662 Mb
LimeWire	2100	2585 Mb	2500	1443 Mb
others	0	0	30000	42000Mb

Table I
THE AMOUNT OF TRAFFIC IN WIRELESS AND LAN.

In the next section we demonstrate the feasibility of our early traffic classification approach on the combined data set composed of WIRELESS and LAN (without the *others* class), while refinements of our methods, concerning the number of packets and the amount of payload to be considered are going to be tested only on WIRELESS. We also use the combined data set (together with the *others* class) to test how the algorithms deal with unknown traffic types and asymmetric routing, while robustness to changing network environments will be examined using the WIRELESS data as training and the LAN data as test set.

IV. EXPERIMENTS

In this section we present experimental results analyzing the performance of our traffic classification methods. We used two main metrics: the true positive (TP) and the false positive (FP) ratio. The TP ratio is the proportion of successfully classified samples to all samples, while the FP ratio is the proportion of samples falsely classified to a given class to all samples that do not belong to that class. Both metrics are usually computed for both flows and bytes. Each experiment was repeated ten times on randomly selected training and test data (of a given size).

A. A feasibility test: classification from the first 16 bytes of the first packet of each flow

To demonstrate the feasibility of our approach and improvements over the memoryless and first-order Markovian methods of [2], we tested the proposed methods on the large combined data set of WIRELESS and LAN, using the first 16 bytes of the first packet of each flow. The TP and FP ratios (using ten-fold cross-validation) are shown in Figure 1. One can observe that higher order models usually perform better. The only exception is the simple first-order MARKOV model (also used in [2]) whose performance is degraded by the inadequate handling of patterns not observed in the training data, which is corrected by using the Krichevsky-Trofimov estimator in MARKOVKT. The RF method achieved a remarkable $98-99\%$ average TP ratio (in bytes), and the CTW and MARKOVKT methods also achieved very good average TP ratios of about $95-96\%$.

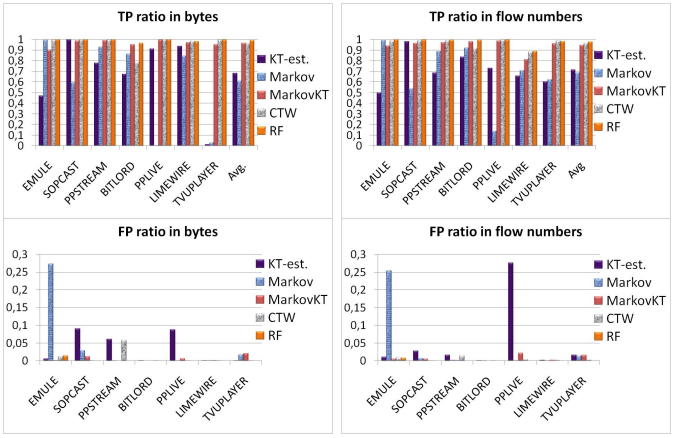

Figure 1. Comparison of the different machine learning methods on the aggregated data set.

The performance is also attractive if we consider the FP ratio, as the higher order methods achieved FP ratios far below 5%, with the exception of CTW in case of PPStream with 6%. The classification results on LimeWire worth some considerations: as the UDP version of this application starts with a 16-byte random string, its identification is clearly impossible from our data, while the TCP versions use some hand-shake mechanism, which is identifiable. The error rates in identifying this application are proportional to the share of UDP traces in our data.

We also compared our algorithms to different publicly available DPI methods. Since the latter are developed to identify traffic types from full payload, we performed these tests on the WIRELESS data set. We believe that this comparison also provides some important insights, although it is not fair from either side: (i) the DPI algorithms are designed to solve the harder problem of recognizing more traffic types, and it is likely that the performance of our algorithms would degrade if the number of traffic types were increased; (ii) the DPI algorithms can use more information than ours.

We used two popular DPI tools, OpenDPI [15] and Tstat [16], as well as the payload-based classifier of [9], called Coral in this paper, which served as the source of ground truth in the large-scale experiments of [4]. Coral applies host level identification if the payload inspection fails; to make a fairer comparison, we also tested the tool without this component, solely relying on payload data (Coral*). The results are given in Figure 2 for both flow and byte accuracy. Because of space limitations, we only provide comparison to our best algorithm, RF (based on the first 16 bytes of the first packet of each flow), whose ten-fold cross-validation performance is also reported. Moreover, in case of the DPI algorithms we only provide results for the traffic classes they support.

The results show that the performance of RF is usually at least as good as those of the DPI methods in TP ratio, while the DPIs are usually better in FP ratio (the full results are omitted due to space limitations).

As a side effect of our (admittedly limited) measurements one can see that a careful test of DPI tools is needed on a given data set if they are to be used as the source of ground truth for real traffic traces.

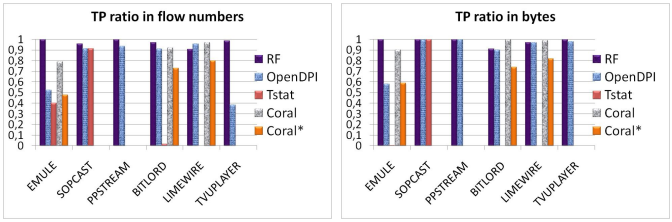

Figure 2. The performance of the Random Forest classifier using the first 16 bytes of the first packet of each flow, compared to some DPI tools.

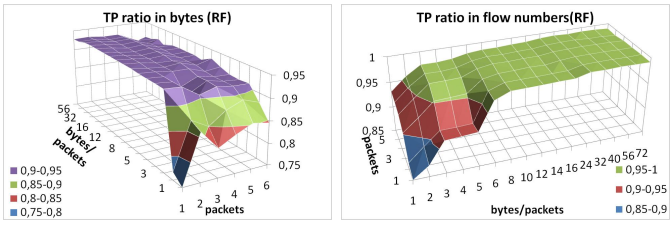

Figure 3. The TP ratio of the Random Forest classifier as a function of the number of packets n and bytes N used ($N = 1, \ldots, 72$ $n = 1, \ldots, 6$).

B. *Bytes and packets: how much data is needed?*

In this section we examine how the accuracy of our classifiers depends on the amount of data used. We test how the results change with the number of packets and bytes used from each flow, as well as how fast the algorithms learn, that is, how many training flows are needed to achieve good performance. Since in these experiments we often use payload beyond the first 16 bytes of each packet, we have run the tests on the WIRELESS data set. Due to space limitations, results are only provided for our best algorithm, RF (our other methods show similar characteristics).

Figure 3 shows the performance of the RF classifier (using ten-fold cross-validation) as a function of the number of packets n and number of bytes N used from each packet. The experiments were run for $n = 1, \ldots, 6$ and $N = 1, \ldots, 72$, the latter being justified by the assumption that protocol specific headers are usually contained at the beginning of each packet. One can see that increasing the number of bytes N sharply improves the performance at the beginning, which flattens out around 8 bytes, while increasing the number of packets may actually worsen the performance in the latter region, which may be attributed to the increased feature-dimension of the problem that eventually leads to overfitting.

One can also see from the above experiment that using the first packet of each flow is sufficient (this choice also has the advantage that it seems pretty much network independent: the content of the first packet usually cannot depend on the underlying network infrastructure). To be able to calibrate the necessary amount of data to be collected from different traffic classes, we tested how many training flows are needed to achieve good performance (as a function of the number of bytes used from the first packet). The classification accuracy of the RF algorithm is given in Figure 4 for different training set sizes: 50, 100, 200, and 400 flows from each class were used

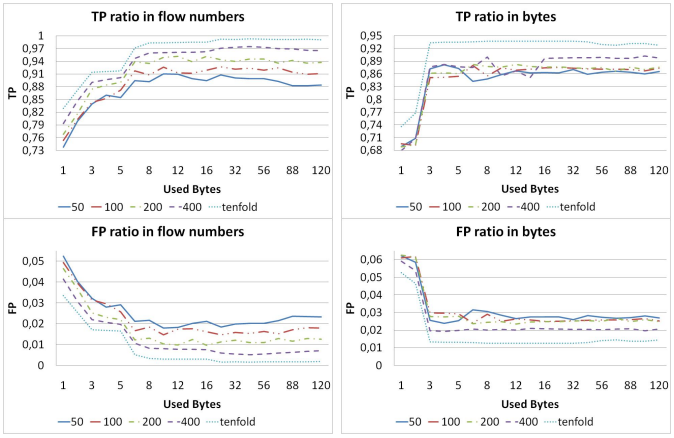

Figure 4. The average TP and FP ratios as functions of the number of bytes used from the first packet for different training set sizes in case of the Random Forest classifier.

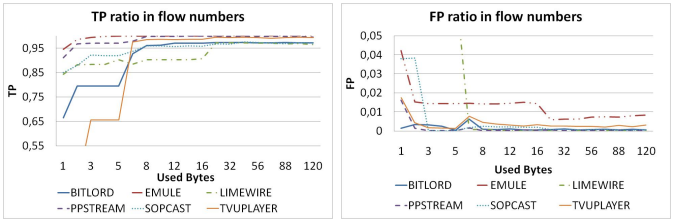

Figure 5. The TP and FP ratios for the different protocol classes (for flows) as a function of the number of used bytes in case of Random Forest.

as training data, respectively, and the remaining flows were used as test data. Each experiment was repeated ten times, using a random selection of the training data. The results of the tests with ten-fold cross-validation are also reported.

One can see that using only a few hundred training flows yields TP rates above 90%. Also, for all training set sizes the performance improves in a similar manner as observed in Figure 3, increasing between 1 and 8 bytes and then smoothing out. We can also notice that the byte-based measures sharply improve around 3 bytes. The reason of this observation may be that only a small portion of protocol messages is responsible for real data transfer (i.e., it is easier to identify flows carrying large amount of data), or it may just simply be an artifact of our data set (where certain patterns randomly occur just for a few elephant flows).

Figure 5 presents the TP and FP ratios (in flow number) for the different protocol classes in the ten-fold cross-validation measurement. We can observe that the necessary amount of information depends on the protocol class, for example, the Gnutella-client LimeWire requires more data than other protocols (this is in agreement with earlier findings of [9], although it seems that using around 30 bytes of the payload is sufficient in contrast to the 300-400 bytes observed in [9], but this may also be the result of not having enough LimeWire flows in our data sets).

C. Robustness

In a real network environment it is expected that the classifier has to deal with a lot of traffic that need not be

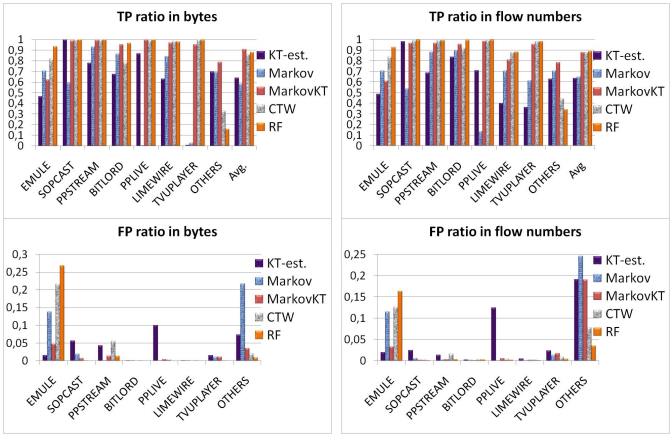

Figure 6. Comparison of the different machine learning methods on the aggregated data set extended with the *others* class.

classified, or several protocols and applications unknown to the classifier. Treating such situations inappropriately leads to a large increase in the FP rate. To reduce this effect, we introduce a new class, called *others*, composed of traces from diverse applications (Skype, encrypted BitTorrent, HTTP, etc.) that represents unknown traffic types. Then the problem is extended to classify a flow to one of the earlier mentioned P2P protocols, or to *others*.

The effect of this modification was measured (using ten-fold cross-validation) on the combined data set composed of WIRELESS and LAN, including the *others* traffic class, using the first 16 bytes of the first packet of each flow. A comparison of the results, presented in Figure 6, to the case without the *others* class (cf. Figure 1) shows that the TP ratio (for the better algorithms) is only slightly decreased for most of the P2P protocols, with the exception of Emule that is often confused with traffic from the *others* class. This is also reflected in the FP ratio, where high values can mostly be observed for Emule (and for PPLive in case of the MARKOV classifier, as before). The classification results are the worst for the *others* class in both TP and FP ratios (worsening substantially the average TP ratio), but, in fact, here we are most sensitive to the errors that classify *other* traffic to a known P2P traffic (which is reflected in the FP rate of the P2P traffic classes). The latter type of error seems to be substantial for the RF algorithm, which has the best performance in the tests without the *others* class. The reason for this performance degradation may come from the fact that the *others* class is quite diverse, and it may happen that our classification algorithms see a certain traffic type only in the test set, in which case such traffic may easily get misclassified.

Another important issue in traffic classification, which should be taken into account, is the asymmetric nature of routing in, e.g., backbone networks. This phenomenon often yields that we are only able to observe traffic from just one direction of a flow. To handle this situation, we can apply our method to the first few payload bytes of the first backward packet. Ten-fold cross-validation performance of the resulting 16-byte RF classifier on the combined WIRELESS-

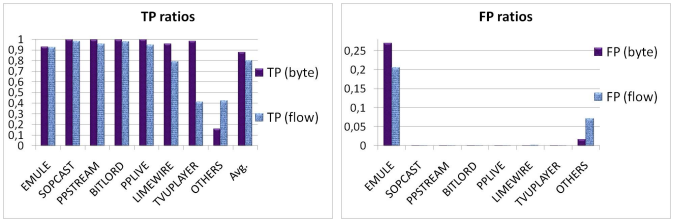

Figure 7. The performance of the Random Forest classifier using the first 16 bytes of the first backward packet of each flow.

LAN dataset is shown in Figure 7. The results indicate that the method is quite applicable for this scenario, resulting only in a $1-2\%$ performance drop relative to the results of classification from the forward packets.

Traffic classification algorithms often suffer from the problem that they behave differently under different network conditions and use features specific to the underlying networks (e.g., connection type). We expect that this is not the case with our algorithms, as the first 16 bytes of each flow are unlikely to carry network specific information. To see how our methods work in changing network environments, we trained an RF classifier on the WIRELESS data set, and tested it on LAN. The two data sets were recorded in different time with different computers and connection types, so our classifier can only utilize protocol specific similarities. The tests show very similar performance to the one obtained for the combined data set (details omitted due to space limitations), with an approximately 10% drop in the TP rate for LimeWire (corresponding to the changed proportion of UDP traffic), while the results for the other classes remained essentially the same. In case of LimeWire, using some more bytes may significantly improve the result, as indicated in Figure 5.

V. CONCLUSION AND FUTURE WORK

In this paper we showed that effective classification of P2P traffic can be performed based on the first few bytes of the first packet of each flow. The proposed classifiers are based on some machine learning algorithms (such as random forests or context trees) and can reach a remarkable accuracy over 95% using as limited data as the first 16 bytes of the first (or the first backward) packet of each flow. This result is competitive to other state-of-the-art algorithms. The advantage of our method is that, unlike traditional DPIs, no human expertise is needed to design the appropriate signatures, only a collection of sample flows from each class is necessary.

On the other hand, our method is able to classify only a few traffic types, leaving open the question if it can scale up to handle many more protocol types, too. Preliminary results to handle unknown traffic were also provided, but further steps are necessary in this direction. Moreover, the performed tests were made on actively collected data: in this case the ground truth is known without doubt, but the traffic patterns are less realistic than in real traces. However, we believe that the fact that our algorithms use only the first 16 bytes of the flows makes our method less sensitive to such errors (apart from the fact that the TP and FP rates are highly dependent on the composition of the whole data set). Nevertheless, measurements on real traffic traces (with high quality labeling) should be performed in the future.

ACKNOWLEDGMENTS

This research was supported in part by the Hungarian Scientific Research Fund and the Hungarian National Office for Research and Technology (OTKA-NKTH CNK 77782 and NAP 2005/KCKHA005), and by the PASCAL2 Network of Excellence (EC grant no. 216886). S. Laki was supported by the project TAMOP 4.2.1/B-09/1/KMR-2010-0003 of the National Development Plan. We would like to thank H. Kim for providing the code for the Coral classifier [4], [9] and G. Szabó and A. Finamore for helpful discussions.

REFERENCES

[1] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection," in *TMA '09: Proceedings of the First International Workshop on Traffic Monitoring and Analysis*, (Berlin, Heidelberg), pp. 117–125, Springer-Verlag, 2009.

[2] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *IMC '06: Proc. of the 6th ACM SIGCOMM conf. on Internet measurement*, (New York, NY, USA), pp. 313–326, ACM, 2006.

[3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[4] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, (New York, NY, USA), pp. 1–12, ACM, 2008.

[5] M. Pietrzyk, J.-L. Costeux, G. Urvoy-Keller, and T. En-Najjary, "Challenging statistical classification for operational usage: the adsl case," in *IMC '09: Proc. of the 9th ACM SIGCOMM conf. on Internet measurement*, (New York, NY, USA), pp. 122–135, ACM, 2009.

[6] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *SIGCOMM '05: Proc. of the 2005 conf. on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 229–240, ACM, 2005.

[7] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos, "Link homophily in the application layer and its usage in traffic classification," in *INFOCOM'10: Proceedings of the 29th Conference on Information Communications*, (Piscataway, NJ, USA), pp. 221–225, 2010.

[8] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, (New York, NY, USA), pp. 1–12, ACM, 2006.

[9] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, October 2004.

[10] D. Rossi and S. Valenti, "Fine-grained traffic classification with netflow data," in *IWCMC '10: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, (New York, NY, USA), pp. 479–483, ACM, 2010.

[11] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the validation of traffic classification algorithms," in *Passive and Active Network Measurement* (M. Claypool and S. Uhlig, eds.), vol. 4979 of *Lecture Notes in Computer Science*, pp. 72–81, Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-79232-1_8.

[12] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inform. Theory*, pp. 199–207, Mar. 1981.

[13] F. M. J. Willems, Y. N. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: Basic properties," *IEEE Trans. Inform. Theory*, vol. IT-41, pp. 653–664, May 1995.

[14] L. Breiman, "Random forests," *Machine Learning*, no. 1, pp. 5–32, 2001.

[15] "Opendpi webpage." http://www.opendpi.org/.

[16] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, and D. Rossi, "Live traffic monitoring with tstat: Capabilities and experiences," in *Proc. 8th International Conference on Wired/Wireless Internet Communications (WWIC 2010)*, (Luleå, Sweden), pp. 290–301, 2010.