

Hierarchical Learning for Fine Grained Internet Traffic Classification

Original

Hierarchical Learning for Fine Grained Internet Traffic Classification / Grimaudo, Luigi; Mellia, Marco; Baralis, ELENA MARIA. - STAMPA. - (2012), pp. 463-468. (3rd International Workshop on TRaffic Analysis and Classification TRAC 2012 Limassol, Cyprus August 2012) [10.1109/IWCMC.2012.6314248].

Availability:

This version is available at: 11583/2502294 since:

Publisher:

IEEE

Published

DOI:10.1109/IWCMC.2012.6314248

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

Hierarchical Learning for Fine Grained Internet Traffic Classification

Luigi Grimaudo
Politecnico di Torino, Italy
luigi.grimaudo@polito.it

Marco Mellia
Politecnico di Torino, Italy
marco.mellia@polito.it

Elena Baralis
Politecnico di Torino, Italy
elena.baralis@polito.it

Abstract—Traffic classification is still today a challenging problem given the ever evolving nature of the Internet in which new protocols and applications arise at a constant pace. In the past, so called behavioral approaches have been successfully proposed as valid alternatives to traditional DPI based tools to properly classify traffic into few and coarse classes. In this paper we push forward the adoption of behavioral classifiers by engineering a Hierarchical classifier that allows proper classification of traffic into more than twenty fine grained classes.

Thorough engineering has been followed which considers both proper feature selection and testing seven different classification algorithms. Results obtained over actual and large data sets show that the proposed Hierarchical classifier outperforms off-the-shelf non hierarchical classification algorithms by exhibiting average accuracy higher than 90%, with precision and recall that are higher than 95% for most popular classes of traffic.

I. INTRODUCTION

The identification and characterization of network traffic is at the base of network management activities for an operator. Through the continuous monitoring of the traffic, security policies can be deployed and tuned, anomalies can be detected, changes in the users behavior can be identified so that QoS and traffic engineering policies can be continuously improved.

In the last years, several traffic classification techniques have been proposed to overcome the limit of original port-based classifiers. Most popular approaches are coarsely based on *deep packet inspection* (DPI) or *behavioral* techniques. In the first case, the traffic is classified looking for specific tokens inside the packet payload. Behavioral techniques try to overcome the limitations of DPI by exploiting some description of the application behavior by means of statistical characteristics, such as the length of the first packets of a flow. See [1] for a detailed discussion of related work.

Both DPI and behavioral classifiers are supervised techniques. However, in case of DPI, the training is often cumbersome and complex, since it involves in most cases the manual identification of the tokens and regular expressions that define a class. In case of behavioral classifiers instead, the adoption of classification algorithms allows to automatically define the rules to label flows, provided a good training set is available. Behavioral approaches bring other advantages with respect to DPI: i) They do not inspect the packet payload, thus preserving privacy, and can then be used for lighter monitoring such as the one offered by, e.g., netflow; ii) They can be easily

extended by going through a quicker retraining phase; iii) The decision process can be computationally lightweight since feature computation is typically much simpler than regular expression parsing.

However, behavioral classifiers suffer from some drawback too [2]: i) A proper training set must be available, including a training set for the “unknown” class, i.e., flows that do not belong to any of the targeted classes; ii) Training must be customized to the monitored network, i.e., training is not portable; iii) And they are known to provide good accuracy when considering *few and coarse* traffic classes, like HTTP vs Peer-to-Peer (P2P) vs email. The last issue is particularly critical given the current trend to have a convergence of most applications going over the same protocol, namely HTTP. Therefore one natural question arises: is it possible to push further behavioral classifiers to correctly identify a large and granular set of classes? For instance, could it be possible to identify application specific traffic that runs over HTTP, like distinguishing Facebook, YouTube, or Google Maps traffic? How to handle the unknown class? In this paper we address this latter problem by engineering and evaluating the performance of a novel Hierarchical behavioral classifier. The intuition is to split the classification process of flows into several stages. At the beginning, coarser classes are used, while in following stages finer grained classification is performed. Classifiers are organized in a tree-based structure, defined according to our domain knowledge. Each node is an independent classifier which operates on a subset of flow features specifically selected to maximize its accuracy, precision and recall. The root node simply separates flows into “unknown” or “known” protocols. The latter set is then classified into 7 classes, with P2P and HTTP appearing as generic classes to be further refined at the next step. For instance, 10 possible subclasses are possible for HTTP traffic.

We consider a benchmark in which 23 different classes are provided by an “oracle”. We use Tstat [3], our DPI-based tool as ground truth generator. Extensive and thorough experiments are run considering 22 different data set collected from a large ISP network and 3 additional data sets collected from our campus network. Results show that the proposed approach outperforms classical machine-learning based classification algorithms, which fail in handling flows of the “unknown” class, and when the number of samples in the training set is

TABLE I
SET OF PROTOCOLS IDENTIFIED BY TSTAT THAT HAVE MORE THAN 50
SAMPLES IN ONE OF THE DATA SETS USED FOR TRAINING SET.

| ID | Class | Byte | Flow | Application protocol |
|----|---------------|-------|------|---|
| 1 | Unknown/other | 1.2G | 355k | Unclassified or belonging to discarded classes |
| 2 | SMTP | 394M | 44k | Simple Mail Transfer Protocol - RFC 5321 |
| 3 | POP3 | 182M | 6k | Post Office Protocol - RFC 1939 |
| 4 | IMAP4 | 55M | 419 | Internet Message Access Protocol - RFC 3501 |
| 5 | SSL/TLS | 968M | 25k | Transport Layer Security protocol - RFC 5246 |
| 6 | MSN | 4M | 137 | Microsoft Messenger MSN Protocol |
| 7 | MSN HTTP | 12M | 162 | Microsoft Messenger MSN Protocol tunneled over HTTP |
| 8 | Flickr | 105M | 2k | Flickr Photo download over HTTP |
| 9 | ADV | 159M | 11k | Advertisement content download over HTTP |
| 10 | MegaUpload | 2.1G | 225 | Megaupload file download over HTTP |
| 11 | Gmaps | 218M | 2k | Google Maps images download over HTTP |
| 12 | Wiki | 28M | 661 | Wikipedia content download over HTTP |
| 13 | Facebook | 1.6G | 40k | Facebook web page content download over HTTP |
| 14 | OpenSocial | 6M | 241 | OpenSocial based social networks over HTTP |
| 15 | YouTube Video | 4.9G | 796 | YouTube flash video streams over HTTP |
| 16 | YouTube Site | 4G | 4k | YouTube web page static content download over HTTP |
| 17 | Flash Video | 848M | 560 | Generic flash video streams over HTTP |
| 18 | RTMP | 72M | 56 | Generic flash video streams over Real Time Messaging Protocol |
| 19 | Other Video | 200M | 60 | Generic video content over HTTP |
| 20 | ED2K Obf | 23.6G | 28k | Obfuscated Emule Protocol |
| 21 | ED2K | 59G | 17k | Plain Emule Protocol |
| 22 | BT | 3G | 14k | BitTorrent Peer Wire Protocol |
| 23 | BT MSE/PE | 3G | 16k | Encrypted BitTorrent Peer Wire Protocol |

heavily unbalanced, as typical in real scenarios. The hierarchical classifier instead achieves better results thanks to splitting the decision process into several stages, each involving fewer classes.

II. DATA SET AND CLASSES

For the experiments carried out in this paper we rely on the traffic monitoring and classification capabilities of Tstat [3], the passive sniffer developed at Politecnico di Torino since 2000 which is freely available from [4]. Tstat passively monitors network traffic carried on a link. It is capable to rebuild each TCP flow, computing a number of statistics. A complex DPI classifier is able to identify more than 50 different protocols. Its accuracy has proved to be very reliable in the past [5]. Among all possible traffic classes that Tstat is able to classify, we selected those for which at least 50 flows are present in each data set. Table I details the list of applications we target, showing also their predominance in one of the data sets used for training. Protocols and application are coarsely grouped to easy readability. As it is possible to see, we consider both simple and well known application protocols (SMTP/POP3/SSL/etc.), and finer grained classification. For example, we would like to distinguish among plain and obfuscated P2P protocols; for HTTP traffic, we would like to identify Facebook separately from social network platforms based on Google OpenSocial protocol. In total we consider 23 different classes.

To run performance evaluation on actual traffic, packet traces have been collected from two real networks: a nationwide ISP in Italy that offers us three different vantage points, and our Campus network. ISP vantage points expose traffic of three different Points-of-Presence (POP) in different cities in Italy; each PoP aggregates traffic from more than 10,000

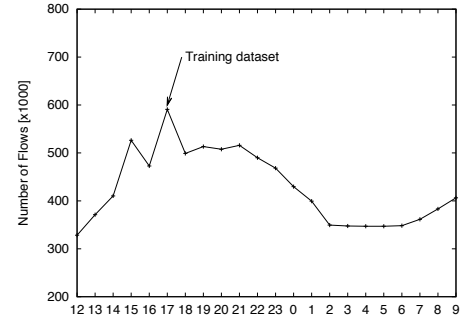


Fig. 1. Number of flows in each ISP data set of 1 hour.

ISP customers, which range from home users to Small Office Home Office (SOHO) accessing the Internet via ADSL or Fiber-To-The-Home technology. It represents therefore a very heterogeneous and challenging scenario. We define a data set as the set of all flows observed from a vantage point during a one-hour long time interval. In this paper we focus our attention to one of the three ISP vantage points from which we have collected 22 different data sets, i.e., 22h long trace. Fig. 1 shows the number of flows that are present in each ISP data set. As expected, the number of flows grows during the day when web traffic is predominant. During the night, fewer flows are present, most of them due to P2P traffic. At 17:00, traffic reaches the peak. We consider this particular data set as “training data set” in the following. Other data sets are used for testing and validation purposes. Table I details the breakdown of the h.17 data set among different classes for Bytes and flows. Notice that some classes have several thousands of flows, while others count no more than few tens of flows.

Finally, 3 completely different data sets have been collected from our campus network and will complete our analysis. This represent a different scenario, in which traffic generated by more than 20,000 students, professors and staff members is present. In this scenario, there is little P2P traffic, since a firewall blocks plain P2P protocols.

A. Performance metrics

Performance of a classifier are typically assessed considering the *overall accuracy, recall, precision* and *F-measure* [6].

Accuracy, is the ratio of the sum of all True Positives (prediction and ground truth are in agreement) to the sum of all tests, for all classes. Accuracy however is biased toward the most predominant class in a data set.

Precision, for a given class, is the ratio of True Positives and the sum of True Positives and False Positive (a sample of another class that has been labeled as of this class). It determines the fraction of samples that actually turns out to be positive in the group the classifier has declared as a positive class. The higher the precision is, the lower the number of false positive errors committed by the classifier.

Recall, for a given class, is the ratio of the True Positives and the sum of True Positives and False Negatives (a sample of the class is labeled as not). It measures the fraction of positive

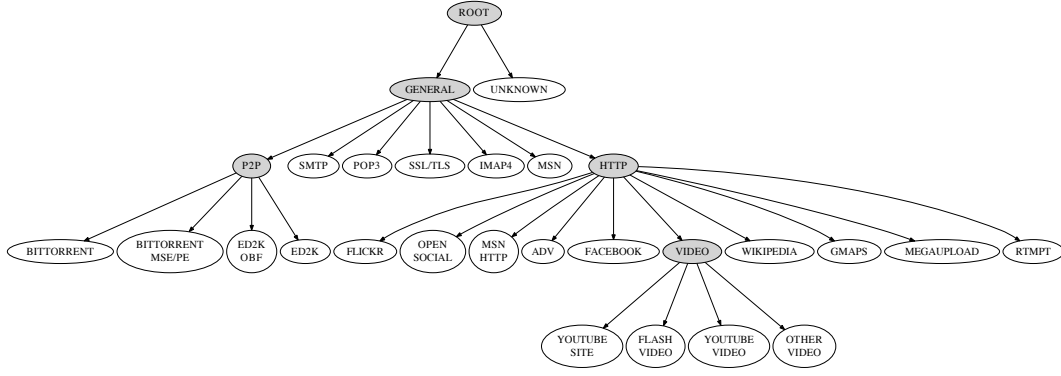


Fig. 2. Tree structure for the Hierarchical classifier.

samples correctly predicted by the classifier. Classifier with large recall have very few positive example misclassified as the negative class.

F-Measure, a widely used metric in classification, weights both precision and recall in a single metric by taking the harmonic mean: $2 \times \text{Recall} \times \text{Precision} / (\text{Recall} + \text{Precision})$.

In this paper we report Recall and F-Measure metrics to assess per-class performance, while Accuracy will be provided when comparing overall results. The complete set of results can be found in [1]. All experiments have been carried out using *RapidMiner* [7] on a 8-cores Intel Xeon E5450 based server PC equipped with 32GB of ram. Computational costs will be reported considering this setup.

III. HIERARCHICAL CLASSIFICATION

All classification algorithms share the same idea: given a description of the object to classify in terms of “features”, find the most likely class according to a model that has been derived from a set of objects properly labeled, i.e., the “training set”. Which algorithm and which features to use are key points to address in the design of the classifier. Our proposal has been designed by performing a thorough selection among different alternatives. The key and novel idea we leverage is to build a classification scheme which is based on a *hierarchy of classifiers*. This allows each classifier to work on a limited subset of classes and on a specialized subset of features, i.e., the features that are most suitable to distinguish among the considered classes. In the following we describe the overall process.

A. Hierarchy Definition

All classification algorithms are known to suffer when the number of classes they have to choose among increases. For example, it can be easy to split P2P traffic from HTTP traffic. How to however correctly classify the single application running over HTTP may be trickier. Moreover, for example the features that allow to separate P2P traffic from HTTP traffic may be useless when trying to separate YouTube from Facebook flows.

The key idea we leverage in this paper is to design a classification scheme based on a *hierarchy of classifiers*. At

first, the flow will be classified into few coarse classes. At the following stages, finer and finer grained classification is achieved. To define the hierarchy, we rely on our domain knowledge. Fig. 2 shows the Hierarchical classifier we propose in this paper. Gray nodes are sub-classifiers and white nodes represent the final classes. We use five classifiers. At the root, flows are split among the “known” and “unknown” classes. Then, a general classifier decides among protocols that we know it is easy to distinguish: P2P, HTTP, SMTP, etc., are well defined classes that have been already shown to be easily identified using behavioral algorithms [2]. At the next step, some classes can be further split into subclasses. For example, P2P traffic is split into BitTorrent versus eMule, while HTTP traffic is split into finer grained applications. Finally, video streams over HTTP will be further classified among YouTube streams, YouTube web site objects, generic Flash Video, or other Video streams.

In the following, for comparison purposes, we consider a classical classifier based on a single stage, in which the classification decision has to be taken at the root node directly. We refer to this case as “Flat” classifier.

B. Feature Selection

For each classifier, the proper set of features must be selected. In the context of traffic classification, most of the proposals so far relies on a set of features that have been chosen based on authors’ domain knowledge. For example, [8] uses a list of features that the authors think to be good to distinguish P2P traffic from client-server traffic. Similarly, [9] uses the size of the first packets as features given the focus on the “early traffic classification”. While the choice of the features can be intuitive when dealing with few classes of traffic, it becomes suddenly difficult to properly select the most prominent features that allow to distinguish between a large list of applications. For example, how to distinguish YouTube video streams from other flash video streams?

In machine learning field, well-known algorithms have been proposed to solve the problem of feature selection, i.e., techniques for selecting a subset of relevant features for building robust learning models [10]. Among the different algorithms, the “minimum-Redundancy-Maximum-Relevance” (mRMR)

TABLE II
SELECTED FEATURE ON THE SERVER TO CLIENT TRAFFIC.

| Classifier | Features | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------|----------|----------|---------------|--------------|-----------|------------|------------|------------|------------------|------------------|-----|------------------|------------------|----------|----------|----------|----------|--------------|-----------|---------|---------|-------------------|------------|------------|------------|------------|------------|-------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|-----------|------------|----|----|
| | TCP port | RST sent | PURE ACK sent | unique bytes | data pkts | data bytes | RFC1323 ws | RFC1323 ts | WINDSCALE factor | Server SACK req. | MSS | max segment size | min segment size | RWND max | RWND min | CWND max | CWND min | initial CWND | stdev RTT | min TTL | max TTL | last segment time | msg 1 size | msg 2 size | msg 5 size | msg 7 size | msg 8 size | msg 10 size | # segments | seg 1 size | seg 2 size | seg 3 size | seg 4 size | seg 5 size | seg 6 size | seg 7 size | seg 8 size | seg 9 size | seg 10 size | seg 7 IPG | # features | | |
| Flat | x | x | | | | | x | x | | | | x | x | | | | x | x | | x | | x | x | x | x | x | x | x | | x | x | x | x | x | x | x | x | x | x | x | | 26 | |
| ROOT | | | | | | | x | | x | | | | | | | | | | | x | x | | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | 10 | |
| General | x | | | | | | x | | | | | x | x | | x | | x | x | | | | | | | | | | | | x | x | x | | | | | | | | | | | 10 |
| P2P | | | | | | | | | | x | | | | | | | | | | | | | | x | | | | | | | | | | | | | | | | | | | 8 |
| HTTP | x | x | x | x | x | x | | | x | | x | x | | x | x | x | | x | x | x | | | x | x | | | | x | x | x | | | | | | | | | | | | x | 22 |
| Video | | | | | | | | x | | x | x | | | x | x | | x | | | | | | x | x | | | | | | x | | | | | | | | | | | x | 11 | |

algorithm is considered as the state-of-the-art [11]. mRMR is an approximation of the theoretically optimal maximum-dependency feature selection that maximizes the mutual information between the joint distribution of the selected features and the classification variable. The input of the feature selection algorithm is a “training” data set, in which *all possible* flow features are provided and flows are correctly labeled. The algorithm selects then the subset of most relevant features to properly assign the correct class. As initial set of features, we use all behavioral layer-4 features that are provided by Tstat. The overall list includes more than 200 features, most of which have been proposed in the past literature. For the sake of brevity we do not report the complete list.

Feature selection can be run independently for each classifier. This allows us to actually select a *different set of features* for each sub-classifier, a key and desirable property. The results of the feature selection are reported in Table II which report the subset of features selected for each classifier considering server to client flow features ([1] details the whole sets). Three considerations hold: First, the list of selected features includes some intuitive choices, but also some unexpected selections. For example, the server RWND scale factor have been found to be useful by the ROOT and HTTP classifier only. Second, different classifiers use different features. Third, the Flat classifier has to consider 45 (26+19) features entailing a larger complexity; at most 35 (22+13) features have been selected for any hierarchical stage.

C. Classification Algorithm Selection

The proper classification algorithm has to be selected among the large number of approaches discussed in the literature: Naive Bayes, Bayesian Kernel Estimation, Rule Based, Decision Trees, Neural Networks, Support Vector Machine (SVM), K-Nearest Neighbor (K-NN) are popular techniques, each leveraging some different idea [10]. Most of these have also been used in the context of traffic classification [2], [12] with good results when dealing with *few* classes.

We run a preliminary set of experiments to see which is the classifier that would guarantee the best performance. For each algorithm, we consider the training data set. We apply the ten-fold cross-validation methodology to estimate the accuracy of each classifier.

Figure 3 reports the average among classes of the F-Measure and the Recall, on top and bottom plot, respectively. Performance of the Flat classifier (black) and the Hierarchical classifier (gray) are reported for each classification algorithm. First, notice that we were not able to complete the test of the SVM and the K-NN Flat classifiers, that were not able to complete the experiment after three days. As well known, dealing with a large number of classes and features poses computational issues for some algorithms. The Hierarchical solution scales better, since each classifier has to deal with a smaller number of classes and features. More details are provided in Sec. IV-C.

Second, the Hierarchical classifier outperforms the Flat classifier considering any classification algorithm. Average F-Measure and Recall are both smaller than 80% for the Flat classifier. On the contrary, the Hierarchical classifier achieves performance higher than 95% for both metrics when a Decision tree is used. This suggests that the problem in designing a Flat classifier is not in the choice of the classification algorithm; rather, any algorithm performs poorly with a large number of traffic classes. Therefore some ingenuity has to be used to improve performance, justifying the need for a hierarchical solution.

For the Hierarchical classifier, the Decision Tree is the only classifier that achieves excellent results for all sub-classifiers in the hierarchy. Other algorithms exhibit more variable results. For example, the SVM performs very well for P2P classification, but it performs poorly for Video classification. Note that the Hierarchical classifier allows also the selection of different classification algorithms for each internal sub-classifier. In the following we restrict our attention to the Decision Tree classifier only.

IV. APPLICATION TO REAL TRACES

We now provide a more extensive and thorough performance evaluation. We start by considering the performance of the Hierarchical versus Flat classifier considering each subclass. We consider as training data set the ISP trace collected at h.17, and the h.18 trace for testing. Figure 4 details the results. Top plots compare the absolute F-Measure for each class; while plots on the bottom quantify the improvement guaranteed by the Hierarchical classifier for F-measure and Recall, respectively. Classes appear in the same order as in

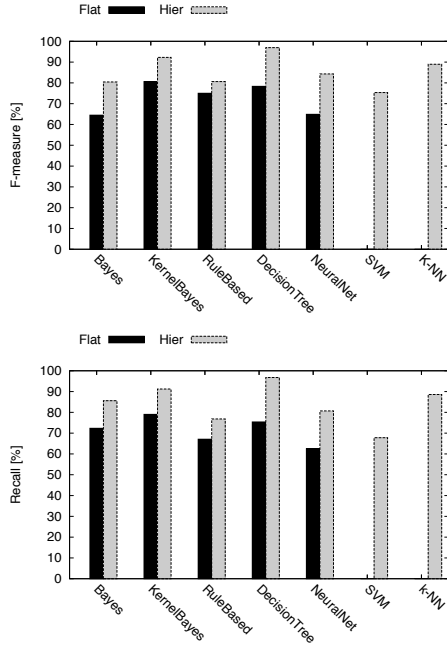


Fig. 3. Comparison of different classification algorithms. Average F-measure and Recall considering ten-fold cross-validation test on a 1h long trace from ISP.

Table I. Results allow to appreciate the benefit of the Hierarchical approach. F-Measure improves for all classes by 28% on average. Notably, some classes are basically ignored by the Flat classifier, e.g., MSN. On the contrary, the Hierarchical classifier deals with MSN flows at the Generic sub-level, where only 7 classes have to be identified. The F-Measure for MSN class then tops to 98%.

Recall improves by about 10% overall, since for some classes the Flat classifier is already achieving good results. In some cases, the Recall decreases by some percentage points. Notice that these are border cases in which the Flat classifier reaches good Recall, but bad F-Measure, i.e., bad Precision. For instance, consider the You-Tube Video class. In this case, the number of False Negatives is small, but the number of False Positives is very high. The Hierarchical classifier improves the F-Measure (thus lowering the False Positive) and overall it performs much better also in this case (F-measure grows by 80%).

Notice that also popular classes are misclassified by the Flat classifier. For example, the Unknown class has very poor performance. Since the Recall is only 15%, the number of False Negative is very large. This is clearly critical, making it impractical to use the Flat classifier given that most of the unknown flows will be classified as one of the known classes. The Hierarchical classifier on the contrary is able to achieve excellent performance, with Recall and F-Measure higher than 95%.

A. Robustness versus time

One interesting question to answer is how the performance of a classifier change over time. Assume to train the classifier

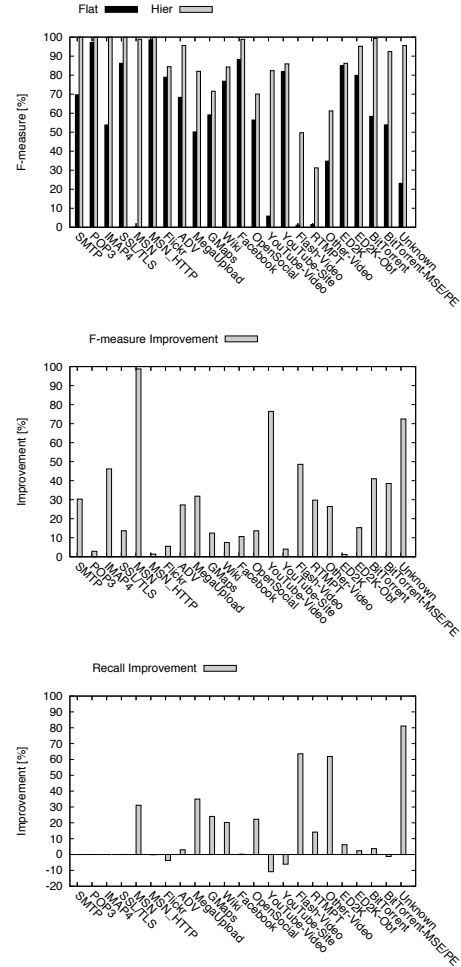


Fig. 4. F-Measure and Recall for each class for the Hierarchical and Flat classifiers. Training on h.17 data set and testing on h.18 data set. ISP trace.

with a given data set collected at a given time. What happens if the classifier is used later? To answer this question we consider the whole ISP data set, which is 22h long. Training of the classifier is done considering the usual h.17 data set. Then performance is evaluated on the other 21 different data sets. To validate the statistical significance of the performance improvements, we used the paired t-test [13] at 95% of significance level for each data set. The overall Accuracy is reported in Figure 5. It shows that the Hierarchical classifier significantly outperforms the Flat classifier. The former guarantees an overall accuracy always higher than 88%, while the latter achieves reasonable performance only during night time when the traffic is dominated by P2P traffic and thus few classes are “active”. During the day it barely reaches 70% of overall Accuracy.

B. Experiment considering other data sets

We have repeated the experiment considering other data sets. For the sake of brevity, we report only one experiment considering two 1-hour long traces collected from our campus LAN at h.15 and h.19 on a normal working day. As previously,

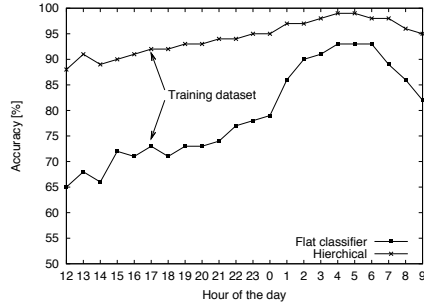


Fig. 5. Accuracy of the Hierarchical classifier when used in real time. One day long data set from ISP.

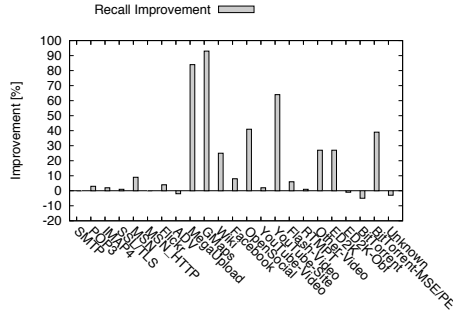


Fig. 6. Improvement for each class for the Hierarchical and Flat classifiers. Testing on Campus data set.

training has been done considering the h.15 trace and testing is done on the h.19 trace. The Recall improvement is reported in Figure 6. Also in this case the Flat classifier provides good results for some classes, while it completely misses others, while the hierarchical classifier improves results especially for less popular classes.

C. Computational Complexity

To gauge the overall computational costs of the classifiers, we were able to completely classify a 1h long data set in less than 1 second and using a very limited amount of memory; i.e., classification cost are very light. The Flat classifier can classify 89,750 flows per second, while the hierarchical classifiers tops to more than 368,400 decision per second. This results are mainly due to the adoption of a Decision Tree classifier at each node. Memory cost is also negligible. Notice that the Hierarchical classifier can be naturally implemented using parallel processes organized in a pipeline. These results show that it is possible to actually use the classifier in on-line system.

Considering training cost, Table III reports the overall time need to perform a training on a 1h long trace. The campus network data set is considered, in which a total of 1.6M flows is present. Both total CPU execution time and total memory usage are reported considering the training phase. As it can be seen, the adoption of a hierarchy of classifiers allows to greatly reduce the computational cost and the maximum memory required at any given time. Each sub-classifier indeed benefits from the reduced number of classes and features. Moreover,

TABLE III
COMPUTATIONAL AND MEMORY COST FOR DIFFERENT CLASSIFIERS TO EXECUTE A TRAINING PHASE ON A 1H LONG CAMPUS DATA SET.

| | Flat | Root | General | HTTP | P2P | Video | Total |
|--------------|------|------|---------|------|-----|-------|-------|
| CPU time [s] | 7849 | 1207 | 389 | 589 | 48 | 74 | 2307 |
| Memory [GB] | 29 | 17 | 11 | 13 | 3.4 | 2.5 | 46.9 |

fewer flows have to be considered to build the model and only those flows that belong to the subset of considered classes have to be taken into account. Note that the training phase cost is relatively important since it has to be seldomly performed off-line.

V. CONCLUSION

In this paper we presented a novel Hierarchical classifier, based on classification algorithms, that achieves excellent results even when considering fine grained classification of Internet TCP flows. We implemented a proper feature selection, selected the best approach among 7 different classification algorithms, tested the approach by ten-fold cross validation and t-test to check the significance of the experiments.

Considering real traffic traces captured from operative networks, we have shown the benefit of using a hierarchical approach: i) classification performance is boosted, ii) computational complexity is reduced, iii) robustness to the training set is achieved. Results demonstrate that behavioral classifiers can be finally considered a reliable means for fine grained traffic classification in the real world.

ACKNOWLEDGMENT

This work was supported by Narus, Inc. Authors are deeply thankful to Dr. Ram Keralapura and Dr. Antonio Nucci for their support and fruitful discussions.

REFERENCES

- [1] L. Grimaudo, et al., "Hierarchical Learning for Fine Grained Internet Traffic Classification", available from <http://www.tlc-networks.polito.it/mellia/papers/TR31012012.pdf>
- [2] H. Kim, et al., "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices", *ACM CoNEXT*, Madrid, SP, December 2008.
- [3] A. Finamore, et al., "Experiences of Internet traffic monitoring with Tstat", *Network*, IEEE, Vol.25, N.3, pp.8-14, May 2011.
- [4] <http://tstat.polito.it>, Tstat home page.
- [5] M. Pietrzyk, et al., "Challenging Statistical Classification for Operational Usage : the ADSL Case", *ACM IMC*, Chicago, IL, November 2009.
- [6] P.N. Tan, et al., "Introduction to data mining," *Pearson Addison Wesley Boston*, 2006
- [7] I. Mierswa, et al., "YALE: Rapid Prototyping for Complex Data Mining Tasks," *12th ACM SIGKDD - KDD-06*, Philadelphia, PA, August 2006.
- [8] T. Karagiannis, et al., "Blinc: Multilevel traffic classification in the dark", *ACM SIGCOMM*, Philadelphia, PA, August 2005.
- [9] L. Bernaille, et al., "Early application identification", *ACM CoNEXT*, Lisboa, PT, December 2006.
- [10] D. J. Hand, et al., "Principles of Data Mining", MIT press, 2001.
- [11] H.C. Peng, et al., "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 8, pp. 1226-1238, 2005.
- [12] T. Nguyen, et al., "A Survey of Techniques for Internet Traffic Classification using Machine Learning" *IEEE Communications Surveys and Tutorials*, vol. 10 no. 4, 2008.
- [13] T.G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms", *Neural Computation*, V.10, N.7, pp.1895-1923, MIT press, 1998.