Journal of Telecommunications Systems manuscript No.
(will be inserted by the editor)

A streaming flow-based technique for traffic classification applied to 12+1 years of Internet traffic

Valentín Carela-Español · Pere Barlet-Ros · Albert Bifet · Kensuke Fukuda

Received: date / Accepted: date

Abstract The continuous evolution of Internet traffic and its applications makes the classification of network traffic a topic far from being completely solved. An essential problem in this field is that most of proposed techniques in the literature are based on a static view of the network traffic (i.e. they build a model or a set of patterns from a static, invariable dataset). However, very little work has addressed the practical limitations that arise when facing a more realistic scenario with an infinite, continuously evolving stream of network traffic flows. In this paper, we propose a streaming flow-based classification solution based on Hoeffding Adaptive Tree, a machine learning technique specifically designed for evolving data streams. The main novelty of our proposal is that it is able to automatically adapt to the continuous evolution of the network traffic without storing any traffic data. We apply our solution to a 12+1 year-long dataset from a transit link in Japan, and show that it can sustain a very high accuracy over the years, with significantly less cost and complexity than existing alternatives based on static learning algorithms, such as C4.5.

Valentín Carela-Español (✉)
UPC BarcelonaTech, Barcelona, Spain
Tel.: +34 934017182
Fax: +34 934017055
E-mail: vcarela@ac.upc.edu

Pere Barlet-Ros
UPC BarcelonaTech, Barcelona, Spain
E-mail: pbarlet@ac.upc.edu

Albert Bifet
HUAWEI Noah's Ark Lab, Hong Kong
E-mail: bifet.albert@huawei.com

Kensuke Fukuda
National Institute of Informatics (NII), Tokyo, Japan
E-mail: kensuke@nii.ac.jp

Keywords Traffic Classification · Machine Learning · Stream Classification · Hoeffding Adaptive Tree · Network Monitoring

1 Introduction

Napster, Edonkey, BitTorrent, Megaupload, Facebook or YouTube are just a few examples of popular applications that suddenly emerged or disappeared from the network, changing significantly the shape of Internet traffic. The Internet is a quickly and continuously evolving ecosystem, which makes the task of traffic classification more challenging year after year. As a consequence, the research community has thrown itself into the solution of this problem, but as pointed out in [1], this problem is still far from being completely solved.

State-of-the-art proposals for traffic classification are usually based on Deep Packet Inspection (DPI) or Machine Learning (ML) techniques [2–9]. These techniques extract in an offline phase a set of patterns, rules or models that capture a static view of a particular network and moment of time from a training dataset. This output is later used to classify the traffic of this particular network online. Although all these proposals theoretically achieve very good results in terms of accuracy, their application has not been as prolific as expected. This is arguably explained by the fact that these solutions do not address several practical issues that arise when they are deployed in real operational scenarios. One of this unadressed issues is that these techniques should be adapted not only to the particular scenario where they are deployed, but also to the continuous changes in the network traffic mix. This adaptation involves a complex and costly training process, which

must be performed periodically and usually requires human intervention.

On the contrary, this paper proposes a flow-based network traffic classification solution that can automatically adapt to the continuous changes in the network traffic. We introduce for the first time the use of Hoeffding Adaptive Tree (HAT) for traffic classification. In contrast to previous solutions that rely on static datasets, this technique addresses the classification problem from a more realistic point of view, by considering the network traffic as an evolving, infinite data stream. This technique has very appealing features for network traffic classification, including the following [10, 11]:

- It processes a flow at a time and inspects it only once (in a single pass), so it is not necessary to store any traffic data.
- It uses a limited amount of memory that can be configured to fit a pre-defined memory budget, independent of the length of the data stream, which is considered infinite.
- It works in a limited and small amount of time. The decision phase is lightweight enough to be used for online classification (see Section 6.3).
- It is ready to predict at any time, so the model is continuously updated and ready to classify with an accuracy comparable to batch machine learning techniques (see Section 6.2 and 6.5)

Our solution also has some interesting features that simplify its deployment in operational networks compared to other alternatives based on DPI or ML techniques [12]. The main problem with DPI-based techniques is that they rely on very powerful and expensive hardware to deal with nowadays traffic loads, which must be installed in every single link to obtain a full coverage of a network. Similarly, traditional ML-based techniques for traffic classification require access to individual packets, which involves the use of optical splitters or the configuration of span ports in switches. In contrast, our solution works at the flow level and is compatible with NetFlow v5, a widely extended protocol developed by Cisco to export IP flow information from network devices [13], which has already been deployed in most routers and switches. Although our solution uses NetFlow v5 as input, it can easily work with other similar exporting protocols (e.g., J-Flow, sFlow, IPFIX).

In order to present sound conclusions about the quality, simplicity and accuracy of our proposal we evaluate our traffic classification solution with the entire MAWI dataset [14], a unique publicly available dataset that covers a period of 13 years. The MAWI dataset consists of daily collected traces from a transit link in Japan since 2001. To the best of our knowledge, this is the first work that deals with such amount of real traffic data for traffic classification. Our results show that our solution for traffic classification is able to automatically adapt to the changes in the traffic over the years, while sustaining very high accuracies. We show that our technique is not only as accurate as other state-of-the-art techniques when dealing with evolving traffic, but it is also less complex and easy to maintain and deploy in operational networks.

The rest of this paper is organized as follows. The related work is briefly presented in Section 2. The proposed classification technique based on Hoeffding Adaptive Tree is described in Section 3. The methodology and the MAWI dataset used for the evaluation of our technique is presented in Section 4. Section 5 analyzes the impact of different configuration parameters of HAT when used for network traffic classification. Section 6 evaluates our solution based on HAT with the MAWI dataset and compares it with the decision tree C4.5 [15], a widely used supervised learning technique. Finally, Section 7 concludes the paper.

2 Related Work

Machine learning techniques for evolving data streams have been widely used in many fields during the last years [10, 16]. However, their application in the field of network traffic classification has been minimal despite of their appealing features. To the best of our knowledge just two works have applied similar techniques in this field. Tian et al. in [17, 18] presented an evaluation of a tailor-made technique oriented to evolving data streams. They compared it with different ML batch techniques from the literature (i.e., C4.5, BayesNet, Naive Bayes and Multilayer Perceptron). The results obtained are aligned with our results, however the dataset used was very limited for the evaluation of a stream data technique (i.e., 2 000 instances per application). Raahemi et al. introduced in [19] the use of Concept-adapting Very Fast Decision Tree [20] for network traffic classification. This technique, closely related to HAT, achieves high accuracy. However, the study focused only on the differentiation of P2P and non-P2P traffic. The dataset was labeled using a port-based technique with the problems of reliability it implies [9, 21, 22]. Unlike these previous works, our solution is based on a more reliable labeling technique [2, 3, 7, 8] and is evaluated with a comprehensive dataset with evolving data streams (i.e., 13 years of traffic, 4 billions of flows). We also perform a complete study of HAT in order to understand the impact of its different parameters on the classification of network traffic.

The problems that arise when a technique is deployed in an actual scenario have been scarcely studied in the literature. To the best of our knowledge, only our previous work [12] has addressed the problem of automatically updating the classification models without human intervention. In contrast, the features of this new proposal considerably reduce the requirements of [12]. Although it also needs a small sample of labeled traffic to keep the model updated, no traffic data is stored nor periodically retrainings are performed. These novel features make our proposal a solution very easy to maintain and deploy in operational networks.

3 Classification of evolving network data streams

We propose a flow based traffic classification technique for evolving data streams based on Hoeffding Adaptive Tree. This technique has very interesting features for network traffic classification, and addresses the classification problem from a more realistic point of view, because it considers the network traffic as a stream of data instead of as a static dataset. This way, we better represent the actual streaming-nature of the network traffic and address some practical problems that arise when these techniques are deployed in operational networks. We describe our proposal to classify network traffic streams in this section. We first present the original Hoeffding Tree (HT) technique oriented to data streams and then we briefly describe the adaptation to deal with evolving data streams, called Hoeffding Adaptive Tree (HAT). Finally, we present the traffic attributes selected to perform the classification of the network traffic.

3.1 Hoeffding Tree

Hoeffding Tree (HT) is a decision tree-based technique oriented to data streams originally introduced by Hulten et al. in [20]. As already mentioned, stream-oriented techniques have many appealing features for network traffic classification: (*i*) they process an example at a time and inspect it only once (i.e., they process the input data in a single pass), (*ii*) they use a limited amount of memory independent of the length of the data stream, which is considered infinite, (*iii*) they work in a limited amount of time, and (*iv*) they are ready to predict at any time. However, these features considerably complicate the induction of the classification model. ML batch techniques (e.g., C4.5, Naive Bayes) are usually performed over static datasets, and therefore, they have access to the whole training data to build the model as many times as needed. On the contrary, models resulting from stream-oriented techniques should be inducted incrementally from the data they process just once and on-the-fly. Therefore, the technique cannot store any data related to the training, which makes the decision-making a critical task.

As already mentioned, HT induces a model in the form of a decision tree. The process of induction starts with a single node, named root, that is recursively split in more nodes creating different branches. The last nodes of the branches are named leaves and contain the class prediction. A key operation in the induction of a decision tree is to decide when to split a node in new branches. Batch techniques have access to all the data in order to perform this operation and decide the most discriminating attribute in each node. On the contrary, stream-oriented techniques do not have access to all the data because the input data is processed in a single pass. To address this problem, HT uses the Hoeffding bound [23] in order to incrementally induce the decision tree. Briefly, this bound guarantees that the difference of discriminating power between the best attribute and the second best attribute in a node can be well estimated if enough instances are processed. The more instances it processes the smaller is the error to decide whether a node should be split. The method to compute this discriminating power, which depends on the split criteria (e.g., Information Gain), as well as other HT parameters are later studied in Sec. 5. A detailed description of the Hoeffding Tree technique can be found in [20].

3.2 Hoeffding Adaptive Tree

Hoeffding Tree allows the induction of a classification model according to the requirements of a data stream scenario. However, an important characteristic of the Internet is that the stream of data continually changes over time (i.e., it evolves). Batch models should be periodically retrained in order to adapt the classification model to the variations of the network traffic, which is a complex and very costly task [12]. Hoeffding Adaptive Tree (HAT), proposed in [24], solves this problem by implementing the Adaptive Sliding Window (ADWIN). This sliding window technique is able to detect changes in the stream (i.e., concept drift) and provide estimators of some important parameters of the input distribution using data saved in a limited and fixed amount of memory, which is independent of the total size of the data stream. The main characteristic of the ADWIN technique is that the size of the sliding window is not fixed, but it is continuously recomputed online based

on the rate of change observed in the data. The interested reader is referred to [25] for more details on how ADWIN is implemented.

3.3 Inputs of our system

The implementation of our system can indistinctly receive two different types of instances: labeled and unlabeled flows. Depending on the type of instance, our solution will perform a classification (if the flow is not labeled) or a training operation (if it is labeled). The classification process labels a new unknown flow using the HAT model. The input of the classification process consists of a set of 16 flow features that can be directly obtained from NetFlow v5 data: source and destination port, protocol, ToS, # packets, # bytes, TCP flags, average packet size, flow time, flow rate and flow interarrival time. The choice of features is based on our previous work in [12]. Although it makes the classification more challenging, the use of standard Netflow v5 data considerably decreases the cost of deployment and computation requirements of the solution, given that the input is already provided directly by the routers.

The other type of instances our solution can receive are the retraining flows. These flows will be labeled by an external tool, as will be described later. In order to automatically update the model, our technique should receive training flows with the same set of 16 features used by the classification process together with the label associated to them. Unlike batch techniques, the retraining process is performed incrementally, which allows the model to be ready to classify at any time. Therefore, our solution can indistinctly deal with a mix of instances and operate with them according to their type (i.e., classification or retraining flows). The best ratio between classification and retraining instances depends on the scenario to be monitored. However, as shown in [12], a very small ratio of retraining instances (e.g., less than 1/400) is sufficient to keep a high accuracy along time. This labeling process can be performed with several techniques, including DPI, given that only a small sample of the traffic needs to be labeled, and therefore it is computationally lightweight. For instance, a common example would be the deployment of our solution in a network with several routers exporting NetFlow v5 data. The labeling of the training flows could be done with NBAR2 [26], using a small sample of the traffic from only one the routers. NBAR2 is a DPI-based technique implemented in the last versions of the CISCO IOS. Otherwise, activating NBAR2 in all the routers and with all the traffic is usually not possible, given the high computational cost and impact it would have on their performance. Another alternative is the use of the methodology presented in [12]. This consists of a small sample of data with full payload, which is labeled using an external DPI tool. This is the solution used in the evaluation presented in Sec. 6.

4 Methodology

This section describes the methodology used to evaluate the performance of our proposal. First, the tool used for the evaluation is presented and then, the dataset used as ground-truth for the evaluation is described.

4.1 MOA: Massive Analysis Online

Massive Online Analysis (MOA) [11] is a Java open source software for data stream mining. Unlike its well-known predecessor WEKA [27], MOA is oriented to the evaluation and implementation of machine learning techniques for data streams. It is specially designed to compare the performance of stream oriented techniques in streaming scenarios. MOA implements the HAT technique with a set of configuration parameters. In addition, it allows the use of batch techniques implemented in WEKA, which simplifies the comparison of traditional batch ML techniques like the decision tree C4.5.

MOA implements different benchmark settings to evaluate stream techniques. For our evaluation, we chose *Evaluate Interleaved Chunks* among the different options available in MOA. *Interleaved Chunks* uses all the instances dividing the stream in chunks (i.e., set of instances). Every chunk is used first for testing (i.e., evaluation of the classification accuracy) and then for training (i.e., the induction of the model).

We believe that this evaluation is the most representative because it uses the complete dataset (i.e., stream) for both testing and training. However, similar conclusions are drawn with other evaluations methods. In our evaluation we first use the default configuration of their parameters to simplify its comparison. We then study the impact of the chunk size (i.e., number of instances in each chunk) on its performance.

4.2 The MAWI Dataset

In order to obtain representative results for the evaluation of stream oriented techniques we need datasets that are long enough to capture the evolution of Internet traffic over time. We use the publicly available MAWI dataset [14] to perform the evaluation because

it has unique characteristics to study stream oriented techniques for network traffic classification. The MAWI dataset consists of 15-minutes traces daily collected in a transit link since 2001 (i.e., 13 years). Although it is a static dataset, its long duration and amount of data makes it the perfect candidate for the evaluation of our technique. Furthermore, its duration allows us to study the ability of HAT to automatically adapt to the evolution of the traffic.

To set the ground-truth of the MAWI dataset we used a DPI technique. The packets in the private version of this dataset are truncated after 96-bytes, which considerably limits the amount of information available for the DPI techniques. Because of this constraint we rely our ground-truth labeling on Libprotoident [2]. The most important feature of Libprotoident is that its patterns are found just in the first 4 bytes of payload of each direction of the traffic. Unexpectedly, that data is enough to achieve very high accuracy classification as shown in [2, 3]. However, the MAWI dataset is characterized to have asymmetric traffic that can reduce the effectiveness of the Libprotoident. We performed a sanitization process and focused on the TCP and UDP traffic from the MAWI dataset. Table 1 presents the top ten applications by flow along the thirteen years once the sanitization is applied. Also, we performed the evaluation of HAT with unidirectional flows, this way we are able to better classify the asymmetric traffic.

After the labeling and the sanitization process, the MAWI dataset consists of almost 4 billions of unidirectional labeled flows. To the best of our knowledge this is the first paper in the network traffic classification field that deals with this large amount of data, which is necessary to extract sound conclusions from our evaluation.

5 Hoeffding Adaptive Tree Parametrization

In this section, we study the parametrization of Hoeffding Adaptive Tree for network traffic classification. As described in Section 4 we use MOA and the MAWI dataset to perform the evaluation. Since this is the first work to use Hoeffding Adaptive Tree for network traffic classification the configuration of the different parameters of HAT and their impact on network traffic classification remain unknown. Because of this, we next present a complete study of the impact of the different parameters of HAT when applied to network traffic. We have studied a total of ten parameters implemented in MOA for HAT: *numeric estimator*, *grace period*, *tie threshold*, *split criteria*, *leaf prediction*, *stop memory management*, *binary splits*, *remove poor attributes*, *no preprune*, and *split confidence*. To obtain a final configuration, we begin the parametrization using the default values in MOA. Then, after the evaluation of each parameter, the best option is selected and used in the rest of the evaluation. In this section we chose 40 million of instances to perform the evaluation. We split them in four different dates to ensure the representativeness of the results, more exactly we have selected the first 10 million of instances from October 2001, January 2004, July 2008 and March 2011. We perform a specific experiment for each date and then compute the average of them to present the results. After the parametrization Section 6 presents an evaluation with the complete MAWI dataset. We briefly describe each parameter, however, we refer the interested reader to [24] for a detailed explanation.

Two main metrics are used in this evaluation in order to show that HAT can be as accurate as batch techniques but using less resources. *Accuracy* is the first metric used and it measures the classification quality of the models. This metric is computed by dividing the total amount of correctly classified instances by the total amount of instances classified. *Cost* is the second metric used and it evaluates the amount of memory and computation time used by the models. The cost metric is computed by multiplying the amount of RAM memory (Gb) and the amount of CPU time (hours) used by the models.

5.1 Numeric Estimator

An important issue of ML techniques oriented to data streams is how they deal with numeric attributes. Unlike most batch ML techniques (e.g., C4.5, Naive Bayes), the techniques for data streams can only pass one time over the data. Because of that, the discretization of the features (i.e., numeric attributes are transformed into discrete attributes) is a more difficult task. MOA implements 4 different numeric estimators for classification using HAT: Exhaustive Binary Tree, Very Fast Machine Learning (VFML), Gauss Approximation (i.e., default one) and Quantile Summaries (i.e., Greenwald-Khanna). Figure 1 (top) presents the performance results of this criteria. We tested different values for each numeric estimator, however, we studied more values of the VFML numeric estimator given its better results. These values correspond to the number of bins used for discretization of the numeric attributes. Gauss Approximation as much as Greenwald-Khanna obtain very poor results. The best numeric estimators in our scenario are VFML and the Exhaustive Binary Tree (BT). More specifically, VFML 1 000 and the Exhaustive Binary Tree are the most accurate. The good performance

Table 1: Top 10 Applications by Flow in the MAWI Dataset

Year	Top 1	Top 2	Top 3	Top 4	Top 5
2001	HTTP (49.44%)	DNS (42.11%)	DEMONWARE (3.27%)	SMTP (2.37%)	FTP (0.52%)
2002	HTTP (41.30%)	DNS (37.75%)	OPASERV (11.81%)	DEMONWARE (4.16%)	SMTP (1.79%)
2003	HTTP (30.22%)	DNS (22.55%)	OPASERV (22.46%)	SQL EXPLOIT (19.47%)	SMTP (1.87%)
2004	HTTP (38.77%)	DNS (26.45%)	SQL EXPLOIT (12.11%)	OPASERV (10.46%)	SMTP (3.40%)
2005	HTTP (31.02%)	DNS (30.80%)	SQL EXPLOIT (13.85%)	SKYPE (8.09%)	MSN (3.91%)
2006	DNS (33.34%)	HTTP (31.51%)	SQL EXPLOIT (11.43%)	SKYPE (6.28%)	BITTORRENT (4.39%)
2007	DNS (50.42%)	HTTP (31.61%)	BITTORRENT (3.82%)	SKYPE (3.37%)	SMTP (2.81%)
2008	DNS (50.82%)	HTTP (26.52%)	BITTORRENT (5.27%)	SKYPE (4.13%)	SQL EXPLOIT (3.86%)
2009	DNS (44.31%)	HTTP (22.04%)	BITTORRENT (20.50%)	SKYPE (4.27%)	GNUTELLA (2.74%)
2010	DNS (48.67%)	HTTP (26.75%)	BITTORRENT (9.82%)	TEREDO (4.29%)	SKYPE (3.76%)
2011	DNS (39.91%)	HTTP (29.55%)	BITTORRENT (13.48%)	SKYPE (5.48%)	TEREDO (4.30%)
2012	DNS (44.93%)	HTTP (31.30%)	BITTORRENT (11.11%)	TEREDO (4.17%)	SKYPE (2.12%)
2013	DNS (54.87%)	HTTP (26.78%)	BITTORRENT (6.33%)	NTP (5.16%)	SIP (1.27%)

Year	Top 6	Top 7	Top 8	Top 9	Top 10
2001	NETBIOS (0.43%)	GNUTELLA (0.37%)	CALL OF DUTY (0.28%)	HALF LIFE (0.22%)	IRC (0.19%)
2002	EMULE (0.62%)	FTP (0.48%)	GNUTELLA (0.43%)	MSN (0.23%)	IRC (0.21%)
2003	EMULE (1.22%)	FTP (0.27%)	NORTON (0.23%)	GNUTELLA (0.2%)	MSN (0.18%)
2004	MSN (2.74%)	SKYPE (1.76%)	NETBIOS (1.07%)	GNUTELLA (0.51%)	FTP (0.30%)
2005	OPASERV (3.11%)	SMTP (2.41%)	BITTORRENT (2.10%)	TDS (1.21%)	SMB (0.42%)
2006	SMTP (2.66%)	OPASERV (1.73%)	MSN (1.66%)	PPLIVE (1.60%)	SMB (0.58%)
2007	SQL EXPLOIT (2.74%)	SSH (1.67%)	MSN (0.84%)	FTP (0.37%)	EMULE (0.34%)
2008	SMTP (3.39%)	SSH (2.04%)	MSN (1.61%)	QQ (0.26%)	ORBIT (0.24%)
2009	SQL EXPLOIT (1.40%)	SMTP (1.7%)	SSH (0.83%)	EMULE (0.76%)	PPSTREAM (0.32%)
2010	SSH (1.89%)	SMTP (1.17%)	SQL EXPLOIT (0.68%)	SIP (0.48%)	NTP (0.41%)
2011	NTP (2.30%)	SSH (1.01%)	SMTP (0.59)	EMULE (0.58%)	SIP (0.43%)
2012	SSH (1.50%)	NTP (1.31%)	SIP (0.56%)	SMTP (0.44%)	CANON BJNP (0.36%)
2013	SKYPE (1.18%)	SSH (1.11%)	PANDO (0.93%)	SMTP (0.47%)	CANON BJNP (0.33%)

obtained by VFML can be related to the properties of the features used for the classification (i.e, NetFlow v5 features).

Apart from the accuracy, another important feature to take into account is the overhead every option implies. Note that this technique should work online and deal with a huge amount of data in a limited amount of time. Because of this, it is important to keep the solution as lightweight as possible while keeping a high accuracy. Figure 1 (bottom) presents the model cost of each numeric estimator in our evaluation. Greenwald-Khanna, Gauss Approximation, and VFML 10 and 100 are hidden behind VFML 1 000. The huge difference of load between the three most accurate techniques makes the VFML 1 000 the best numeric estimator for our scenario.

5.2 Grace Period

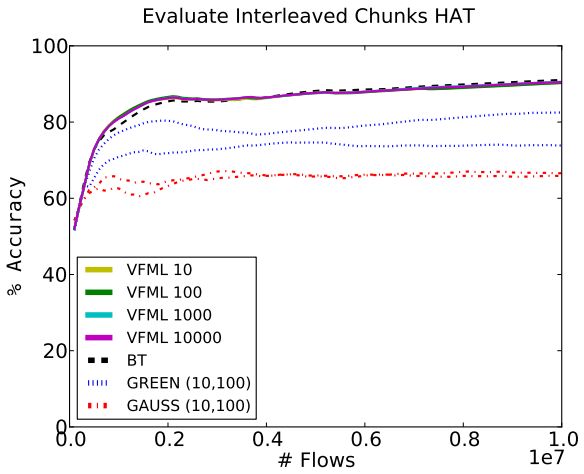
The next parameter studied is the *grace period*. This parameter configures how often (i.e., how many instances between computations) the values in the leafs of HAT are computed. This computation is performed in order to decide if a further split is necessary. This computation is considerably costly and the impact of each instance in the result of this computation is small. Therefore, it is reasonable to perform this computation periodically instead of repeating it for each instance. High values would reduce the cost of the technique but slow down the growth of the tree, thus decreasing its accuracy in theory. Figure 2 (top) presents the impact of different grace values on the accuracy of the technique. At first glance there are no huge differences between the different values. As expected the lowest value is initially getting the best results since it is extracting the knowledge by quickly splitting the leaves. However, we are dealing with a data stream and making a decision with few instances can sometimes produce inaccuracies in the future. In Fig. 2 (top) the most accurate grace periods are 1 000 and 200 (i.e., default one).

The negligible impact of this parameter on the accuracy can be related to the nature of the traffic mix in our dataset. However, the importance of this parameter is its ability to decrease the overhead of the technique without decreasing significantly its accuracy. Figure 2 (bottom) presents how the different values of the grace period affects to the cost of the technique. We decided to use 1 000 as grace period giving it is the best trade-off between accuracy and load.

5.3 Tie Threshold

A well-known parameter from decision tree techniques is the *tie threshold*. Sometimes two or more attributes in a leaf cannot be separated because they have identical values. If those attributes are the best option for splitting the node the decision would be postponed until they differ and this can decrease the accuracy. Figure 3 (top) presents the accuracy obtained with different values of the *tie threshold* parameter. The most accurate value is 1, closely followed by 0.5 and 0.25.

In order to decide between the most accurate tie thresholds we rely on the cost of the model they produce. Figure 3 (bottom) shows that 0.25, and 1 are the



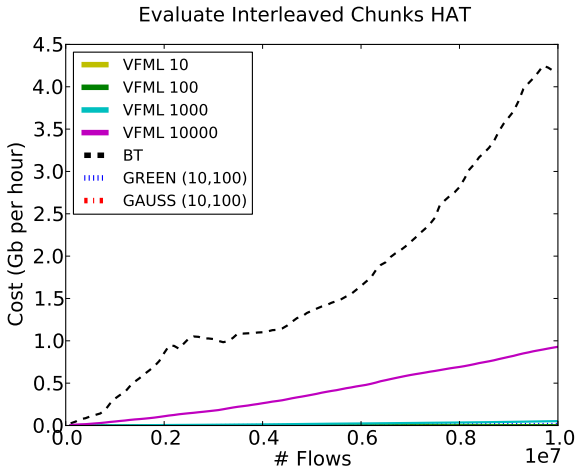
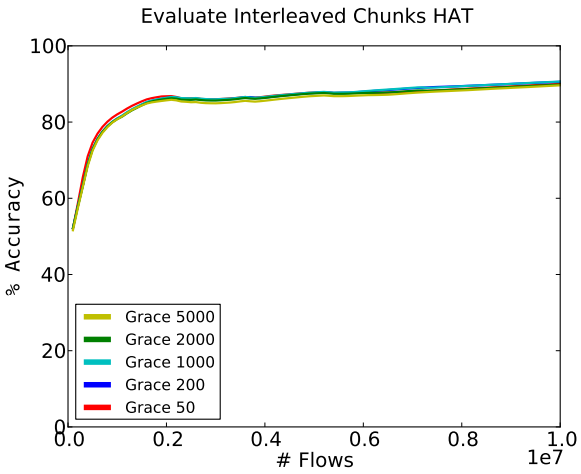


Fig. 1: Impact of the *Numeric Estimator* parameter



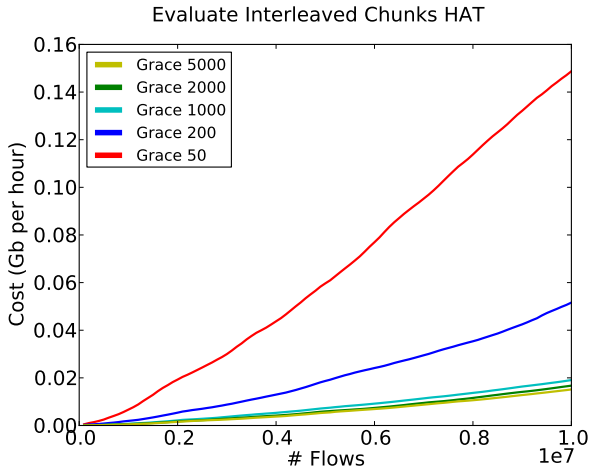


Fig. 2: Impact of the *Grace Period* parameter

best options among the three more accurate values. We decided to use 1 as tie threshold because, unexpectedly, although initially it achieves similar performance to 0.25, the final cost follows a lower inclination.

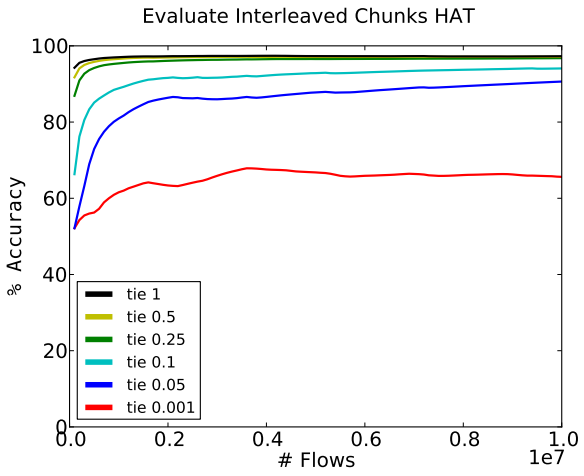
5.4 Split Criteria

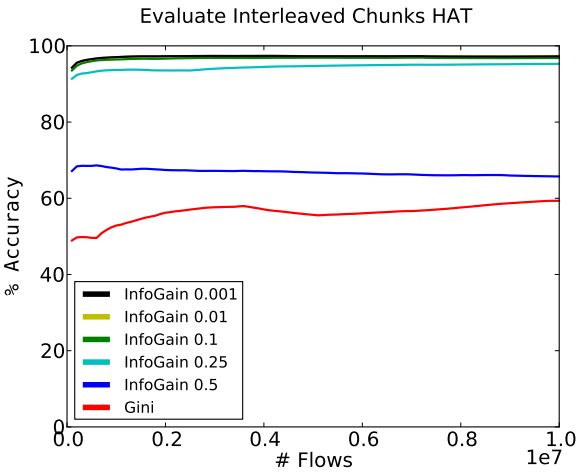
As mentioned before, the *grace period* indicates when to compute the necessary values to decide if a node should be split. This computation refers to the *split criteria*. This parameter decides when an attribute is enough discriminative to split a node. There are two approaches implemented in MOA: Information Gain and Gini. Figure 4 (top) presents the accuracy obtained with the Gini split criteria and different values of the Information Gain. These values correspond to the minimum fraction of weight required to down at least two branches. The performance of the Gini option is considerably poor in our scenario. Regarding the different values of the Information Gain, the values 0.001, 0.01 and 0.1 achieve the highest accuracies.

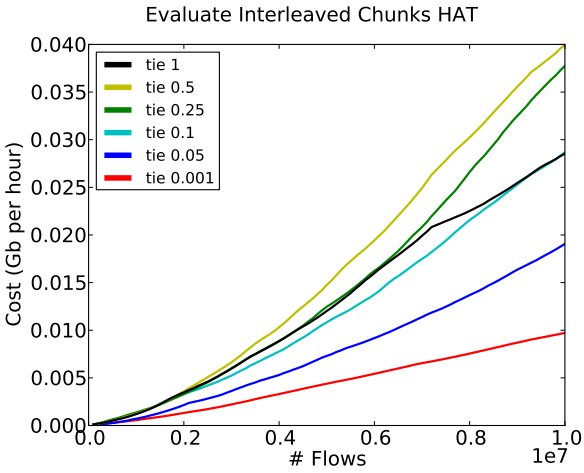
Figure 4 (bottom) shows how the cost of technique is impacted by the different split criteria. We decided to use the Information Gain value 0.001 because it is the lightest among the most accurate.

5.5 Leaf Prediction

An important feature of HAT is that, since the model is continuously being updated, it is always ready to classify. The next parameter is related to this classification and describes how HAT performs the classification decision at leaf nodes. MOA implements three different approaches: Majority Class, Naive Bayes and Naive Bayes Adaptive. The Majority Class approach consists of assigning the most frequent label in that leaf. Apart from the most frequent label in a leaf, we have much information related to the instance (i.e., at-







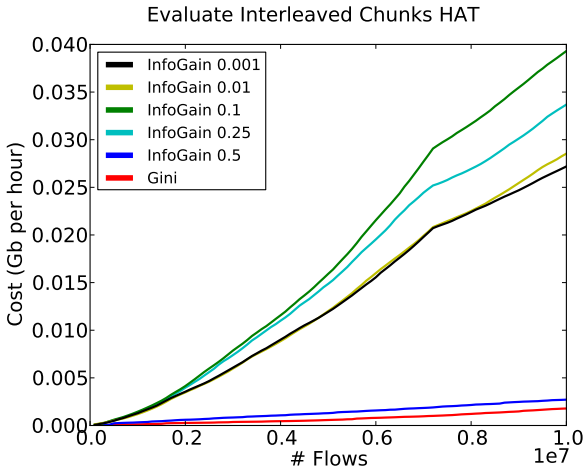


Fig. 3: Impact of the *Tie Threshold* parameter

Fig. 4: Impact of the *Split Criteria* parameter

tributes). The Naive Bayes approach tries to use this extra information to make a more accurate prediction. This approach computes the probability an instance belongs to the different possible labels from a leaf based on its attributes. The most probable label is the one assigned. However, this technique can reduce the accuracy depending on the scenario. The Naive Bayes Adaptive approach tries to take advantage of both approaches by combining them. It computes the error rate of the Majority Class and Naive Bayes in every leaf, and use for future predictions the approach that has been more accurate so far. Figure 5 (top) presents the accuracy obtained with the different approaches. Unexpectedly, the Naive Bayes approach obtains very poor results. As described in [28], the experimental implementation in MOA does not change the memory management strategy when Naive Bayes is enabled and this can impact on its performance. On the other hand, the Majority Class and the Naive Bayes Adaptive approaches obtain similar high accuracies.

Figure 5 (bottom) shows how the different approaches impact on the solution in terms of model cost. Taking into account these results we decided to use Majority Class as the leaf prediction technique. Apart from having a lower cost, while achieving similar high accuracy, the Majority Class approach is not affected by other parameters. Approaches based on Naive Bayes can decrease its accuracy if parameters like *removing poor attributes* or *stopping memory management* are activated. The negligible impact of the Naive Bayes approach on the accuracy can be result of the traffic mix and the features used for the classification (i.e, NetFlow v5 features).

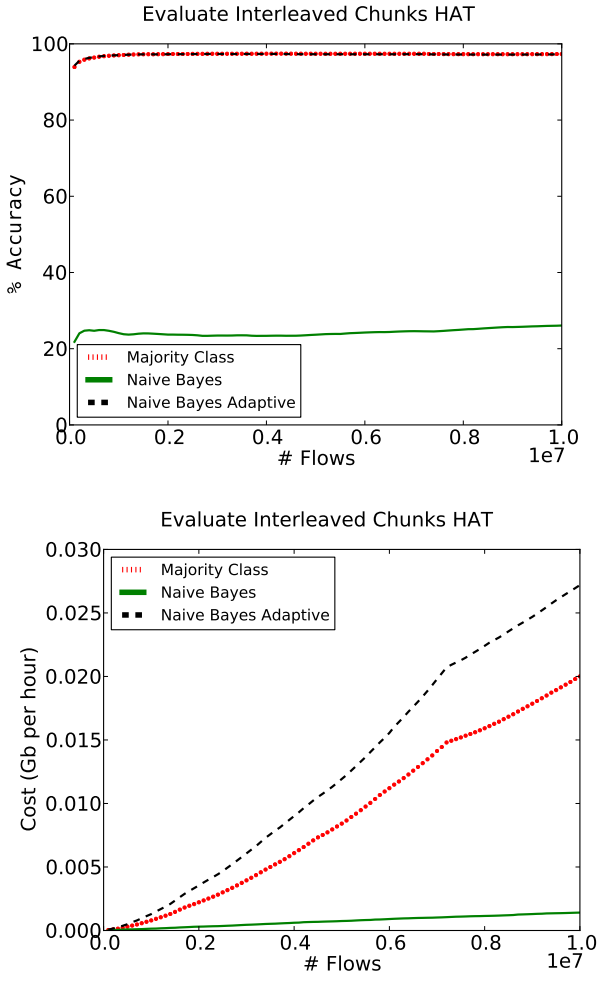


Fig. 5: Impact of the *Leaf Prediction* parameter

5.6 Other Parameters

So far, the parameters studied have substantially impacted the accuracy or cost of HAT. However, we have also evaluated some parameters with marginal impact. This is the case of the *Stop Memory Management* parameter. When this parameter is activated HAT stops growing as soon as the memory limit is reached. However, it seems that the default value of the memory limit in MOA is never reached or this parameter is not implemented for the HAT technique. The *Binary Split* parameter, describing if the splits of a node have to be binaries or not, has also a marginal impact. We truly believe that this result is directly related to our scenario characteristics. All our attributes are numerically and hence all the splits performed are almost always binary splits. The last parameter studied with marginal impact is the *Remove Poor Attributes* parameter. This feature removes attributes in the leafs whose initial values indicate their uselessness for the splitting decision. In our scenario, these parameters have not impacted on the accuracy of HAT. However, a marginal improvement has been observed in terms of cost. Thus, we also activated them in the final configuration.

Table 2: HAT parametrization

Parameter	Value
Numeric Estimator	VFML with 1 000 bins
Grace Period	1 000 instances (i.e., flows)
Tie Threshold	1
Split Criteria	Information Gain with 0.001 as minimum fraction of weight
Leaf Prediction	Majority Class
Stop Memory Management	Activated
Binary Splits	Activated
Remove Poor Attributes	Activated

We have also studied the parameters *No PrePrune* and *Split Confidence* and no differences have been observed. As a result, none of them are activated in our final configuration.

Finally, similarly to other ML-based techniques, HAT can be used in ensembles techniques. MOA implements several ensembles methods (e.g., bagging, boosting) that basically combine several models to improve the final accuracy. However, this improvement comes with a higher computational cost. Given that we already achieve a very high accuracy with the current configuration we dismissed the use of ensembles techniques in our scenario.

Table 2 presents the final configuration of the parameters obtained in this section. We use this configuration for the evaluation of the HAT technique for network traffic classification.

6 Hoeffding Adaptive Tree Evaluation

Once the best configuration is selected we compare the HAT technique with a well-known technique from the literature. The goal of this comparison is to show that our solution can be as accurate as batch-oriented techniques, but with the appealing features of those oriented to streams. As mentioned in Sec. 1, batch techniques are usually built from a static dataset and do not address the ever-changing nature of the Internet traffic [29] or rely on complex custom-made solutions [12]. However, our solution can automatically adapt to changing traffic conditions without storing any data and being always ready to classify. For this comparison, we chose the J48 technique as a representative example of batch-oriented techniques, which is an open source version of the C4.5 decision tree implemented in WEKA. We selected this technique because it has been widely used for network traffic classification [5,6,12,29], achieving very good results when compared with other techniques [4,30].

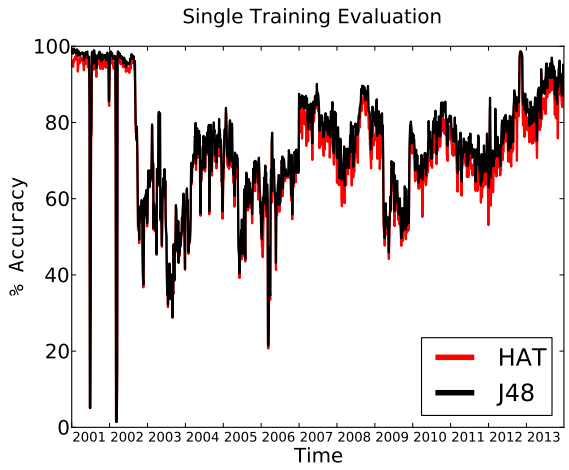


Fig. 6: Single training configuration

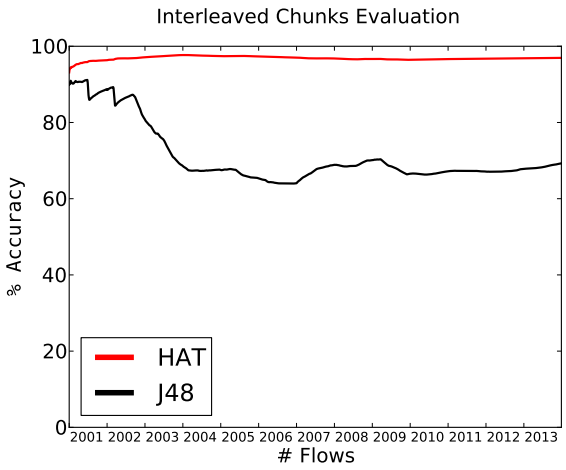


Fig. 7: *Interleaved Chunk* evaluation with default configuration

6.1 Single Training Evaluation

Usually ML-based network traffic classification solutions presented in the literature are evaluated from a static point of view using limited datasets. The first evaluation performed pretends to show the temporal obsolescence of the models produced with static datasets [12, 29]. To achieve this goal we performed an evaluation applying just an initial training with 3 million of flows in 2001 for the complete classification of the 13 years of traffic of the MAWI dataset. The accuracy of both techniques is substantially degraded in this evaluation showing that the models should be periodically updated to adapt to the changes in the traffic. The deep drops in the accuracy are related to new applications that are not present in the initial training dataset. The increment of accuracy during the last years of the evaluation is due to the change of the traffic mix in the MAWI dataset. As showed in Table 1, there is an increment of traditional applications (i.e., DNS, HTTP and NTP) and a decrease of novel applications (i.e., BitTorrent and Skype) during those years. Giving that this evaluation is performed from a static point of view, HAT is not able to make use of its interesting features for streams.

6.2 Interleaved Chunk Evaluation

The second experiment consists of an *Interleaved Chunk* evaluation with the default evaluation method of MOA. That is, a stream-based evaluation where the 4 000 million of flows from the 13 years of the MAWI dataset are segmented in chunks of 1 000 instances that are first used to classify and later to train. Figure 7 presents the results regarding this evaluation. Our solution achieves considerably better results than the J48 batch technique. This can be easily explained by the fact that this evaluation methodology is oriented to evaluate incrementally inducted techniques. The J48 batch technique creates a new decision tree from scratch with every chunk of 1 000 instances forgetting all the previous knowledge extracted. In contrast, our solution updates the classification model with the new information but considering also all the information extracted so far, which results in a more robust classification model.

6.3 Chunk Size Evaluation

As shown in the previous experiment, J48 is significantly less accurate than HAT with the default stream-based evaluation. However, that difference seems mainly because of the small chunk size that produces very poor J48 trees. In order to address this problem, we next study the impact of the chunk size on both techniques. We evaluate six different chunk sizes (i.e., 1, 100, 1 000, 10 000, 100 000, 1 000 000 flows) in the *Interleaved Chunk* evaluation. Given the large number of executions involved, we decided to use a sample of more than 4 million of flows of the MAWI dataset in this experiment. Figure 8 shows the accuracy of both techniques for each chunk size. Given that HAT builds its tree incrementally, it is barely affected by the chunk size, achieving always a very high accuracy. Unlike HAT, J48 is substantially impacted by the chunk size. As expected, the small values of the chunk size (i.e., 1, 100, 1 000) produce inaccurate J48 trees. Only the highest chunk sizes (i.e., 100 000 and 1 000 000) are able to achieve similar accuracies to the HAT technique.

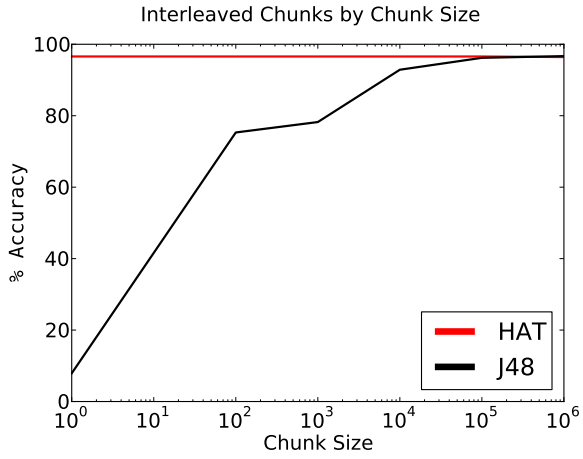


Fig. 8: Accuracy by chunk size

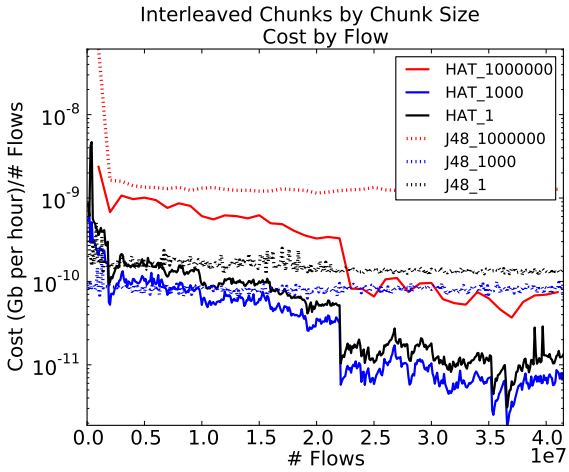


Fig. 9: Cost by chunk size

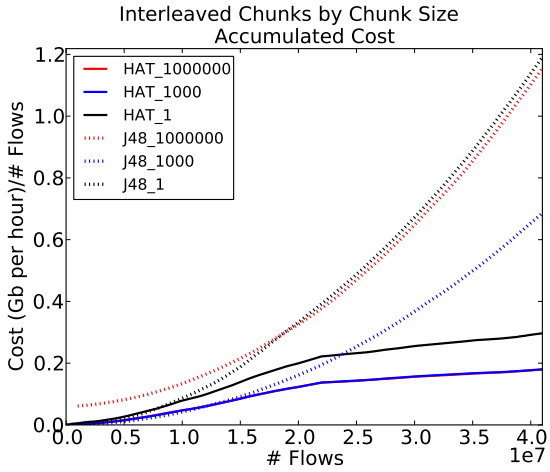


Fig. 10: Accumulated cost by chunk size

Moreover, large chunk sizes imply the storage of large amounts of traffic as we will discuss next.

As important as the accuracy is the cost of the techniques. The J48 decision tree, as a batch technique, needs to store first the data of each chunk to continuously build the model from scratch, which results in huge memory requirements. Figure 9 presents the cost (i.e., Gb per hour) by flow in log scale directly obtained from MOA. For clarity, only the extremes values (i.e., 1, 1 000 000) and the default value (1 000) are plot. The rest of values follow a similar behavior as the 1 000 chunk size. Initially, all the sizes have a high cost per flow, especially the smallest and the highest chunk sizes (i.e., 1 and 1 000 000). The cost quickly decreases after the initial peak. However, it decreases differently for both techniques. After the initial peak, the cost of J48 remains more or less constant along time. The cost for J48 among the different chunk sizes is similar but the highest chunk size (i.e., 1 000 000), being more than five times higher. In contrast, the cost of HAT rapidly decreases to very low values. Even with the highest chunk size it is able to decrease the cost similarly to the lowest values of the J48 technique. The constant cost of J48 is related to the cost of the training of each model for each chunk. Unlike J48, the model of HAT is incrementally built. Once it is consistent (i.e., around 2 million in our evaluation) only small modifications are applied in the model for every chunk.

To better show the differences in the cost of both techniques, Figure 10 presents the accumulated cost of both techniques by chunk size. The growth of the cost by the HAT technique is almost plain after 2 million of flows. On the other hand, J48 has a continuous growth along time. It is important to note that this evaluation is done with a static dataset of 4 million. However, the difference of cost between both techniques would considerably increase in an infinite stream-based scenario (e.g., network traffic classification).

In summary, in a stream-based scenario the HAT technique is usually more accurate than J48. Only when high chunk sizes are used J48 is able to be as accurate as the HAT technique. Furthermore, HAT consumes less resources than the J48 decision tree, especially when those high chunk sizes (i.e., 100 000 and 1 000 000) are used to increase the accuracy of J48.

6.4 Periodic Training Evaluation

In order to compare our results with other retraining proposals from the literature, we modified the original idea of the *Interleaved Chunk* evaluation by following the configuration proposed in [12]. The new evaluation consists of the use of chunks of 500 000 instances for

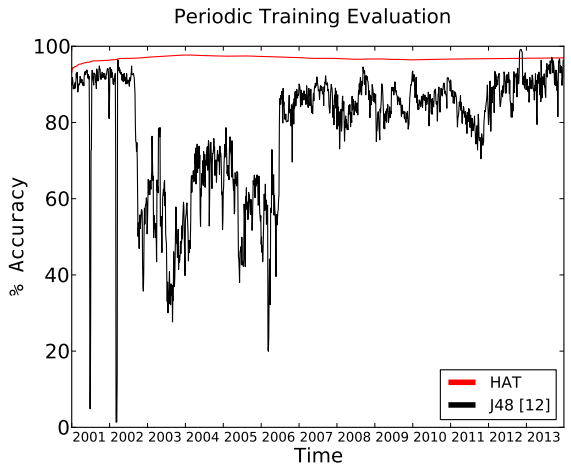


Fig. 11: *Interleaved Chunk* comparison with [12] configuration

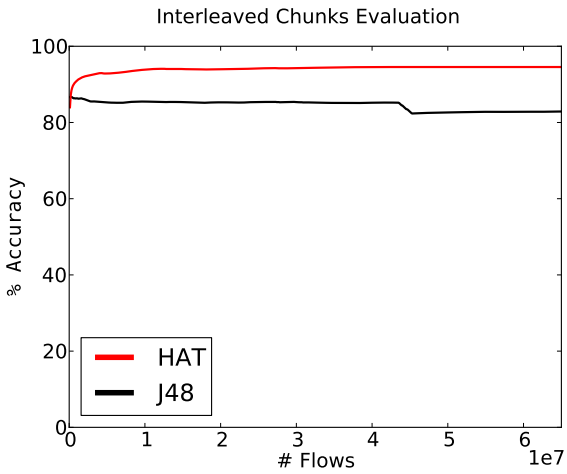


Fig. 12: *Interleaved Chunk* evaluation with CESCA dataset

training and 500 000 000 for testing, using the last seen chunk to train the next group. This evaluation represents the scenario presented in Section 3.3, where a sample of the traffic is labeled by a DPI-based technique to retrain the model, while it is used to classify all the traffic. Therefore, with the exception of the first chunk, the complete MAWI dataset is classified. We selected 500 000 as chunk size derived from the results obtained in [12]. However, in [12] the retrained decision is based on a threshold accuracy while, in our evaluation, due to software constraints, it is based on the amount of instances processed (i.e., 500 000 000). Although the evaluation has been changed, the operation to compute the accuracy is maintained to make the comparison possible. Figure 11 presents the results of this evaluation. The accuracy of the J48 technique has been improved significantly. However, the stable accuracy seen in the previous evaluation has changed to a more volatile one. This is because the initial configuration is continuously retrained and quickly adapting itself to the changes in the traffic. The results suggest that in this particular dataset, the retraining should be performed more often in order to adapt faster to the changes in the traffic with the related cost it would produce. Note however that the choice of the chunk size for HAT is quite irrelevant as shown in Section 6.3

6.5 External evaluation

So far, we presented the parametrization and evaluation of the HAT technique with the MAWI traffic. The results show that the HAT technique is, at least, as accurate as a state-of-the-art technique, such as C4.5 (i.e., J48 in MOA) but with considerably less costs. In order to show these results are not only related to the MAWI dataset, we next evaluate the performance of the HAT technique with a different dataset. We used the CESCA dataset used in [12] to compare the performance of HAT and J48 and make easier the comparison between both works. The CESCA dataset is a fourteen-days packet trace collected on February 2011 in the 10-Gigabit access link of the Anella Científica, which connects the Catalan Research and Education Network with the Spanish Research and Education Network. A 1/400 flow sampling rate was applied accounting for a total of 65 million of labeled flows. We use the default configuration of the *Interleaved Chunk* evaluation and the parametrization obtained in Section 5 for the HAT configuration. Although possible tuning could be applied to this specific scenario, we show that the configuration obtained in Section 5 seems suitable for other scenarios. Figure 12 shows that, similar to Fig. 7, the HAT technique is more accurate than J48 in a stream-based scenario. The smaller differences in terms of accuracy with the CESCA dataset can be related to a less heterogeneous traffic mix and a shorter dataset (i.e., 14 days vs 13 years).

7 Conclusions

In this paper we propose a new stream-based classification solution based on Hoeffding Adaptive Tree. This technique has very appealing features for network traffic classification: (*i*) processes an instance at a time and inspects it only once, (*ii*) uses a predefined amount of memory, (*iii*) works in a bounded amount of time and (*iv*) is ready to predict at any time. Furthermore, our technique is able to automatically adapt to the changes

of the traffic with just a small sample of labeled data, making our solution very easy to maintain. As a result, we are able to accurately classify the traffic using only Netflow v5 data, which is already provided by most routers at no cost, making our solution very easy to deploy.

We evaluate our technique using the publicly available MAWI dataset, 4 000 millions of flows from 15-minutes traces daily collected in a transit link in Japan since 2001 (13 years). We first evaluate the impact of the different parameters on the HAT technique when used for traffic classification and then compare it with one of the state-of-the-art techniques most commonly used in the literature (i.e., C4.5).

The results show that our technique is an excellent solution for network traffic classification. It is not only more accurate than traditional batch-based techniques, but it also sustains this very high accuracy over the years with less cost. Furthermore, our technique does not require complex, ad-hoc retraining systems to keep the system updated, which facilitates its deployment and maintenance in operational networks.

Acknowledgements This research was funded by the NII International Internship Program, by the Spanish Ministry of Economy and Competitiveness under contract TEC2011-27474 (NOMADS project) and by AGAUR (ref. 2014-SGR-1427).

References

1. Dainotti, A., Pescapè, A., Claffy, K.C.: Issues and future directions in traffic classification. IEEE Network **26**(1), 35–40 (2012). URL http://dx.doi.org/10.1109/MNET.2012.6135854
2. Alcock, S. and Nelson, R.: Libprotoident: Traffic Classification Using Lightweight Packet Inspection. Tech. rep., University of Waikato (2012). [Online]. Available: http://www.wand.net.nz/publications/lpireport, as of June 22, 2015
3. Carela-Español, V., Bujlow, T., Barlet-Ros, P.: Is our ground-truth for traffic classification reliable? In: Proceedings of the 15th International Conference on Passive and Active Network Measurement, PAM'14, pp. 98–108. Springer (2014). URL http://dx.doi.org/10.1007/978-3-319-04918-2_10
4. Lim, Y.s., Kim, H.c., Jeong, J., Kim, C.k., Kwon, T.T., Choi, Y.: Internet traffic classification demystified: On the sources of the discriminative power. In: Proceedings of the 6th International COnference, Co-NEXT '10, pp. 9:1–9:12. ACM, New York, NY, USA (2010). URL http://doi.acm.org/10.1145/1921168.1921180
5. Nguyen, T.T., Armitage, G.: A survey of techniques for internet traffic classification using machine learning. Commun. Surveys Tuts. **10**(4), 56–76 (2008). URL http://dx.doi.org/10.1109/SURV.2008.080406
6. Carela-Español, V., Barlet-Ros, P., Cabellos-Aparicio, A., Solé-Pareta, J.: Analysis of the impact of sampling on netflow traffic classification. Computer Networks **55**(5), 1083–1099 (2011). URL http://dx.doi.org/10.1016/j.comnet.2010.11.002
7. Alcock, Shane and Nelson, Richard: Measuring the Accuracy of Open-Source Payload-Based Traffic Classifiers Using Popular Internet Applications. In: IEEE 38th Conference on Local Computer Networks Workshops (LCN Workshop on Network Measurements), pp. 956–963 (2013). URL http://dx.doi.org/10.1109/LCNW.2013.6758538
8. Bujlow, T., Carela-Espaol, V., Barlet-Ros, P.: Independent comparison of popular dpi tools for traffic classification. Computer Networks (76), 75–89 (2015). URL http://dx.doi.org/10.1016/j.comnet.2014.11.001
9. de Donato, W., Pescape, A., Dainotti, A.: Traffic identification engine: an open platform for traffic classification. Network, IEEE **28**(2), 56–64 (2014). URL http://dx.doi.org/10.1109/MNET.2014.6786614
10. Gama, J.a., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 329–338. ACM, New York, NY, USA (2009). URL http://doi.acm.org/10.1145/1557019.1557060
11. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research **11**, 1601–1604 (2010). URL http://www.jmlr.org/proceedings/papers/v11/bifet10a.html
12. Carela-Español, V., Barlet-Ros, P., Mula-Valls, O., Sole-Pareta, J.: An automatic traffic classification system for network operation and management. Journal of Network and Systems Management (2013). URL http://link.springer.com/article/10.1007/s10922-013-9293-1
13. Cisco IOS NetFlow: [Online]. Available: http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html, as of June 22, 2015
14. MAWI Working Group Traffic Archive: [Online]. Available: http://mawi.wide.ad.jp/mawi/, as of June 22, 2015
15. Quinlan, J.: C4. 5: Programs for Machine Learning. Morgan Kaufmann (1993)
16. Gama, J.: A survey on learning from data streams: current and future trends. Progress in Artificial Intelligence **1**(1), 45–55 (2012). URL http://link.springer.com/article/10.1007/s13748-011-0002-6
17. Tian, X., Sun, Q., Huang, X., Ma, Y.: Dynamic online traffic classification using data stream mining. In: Proceedings of the 2008 International Conference on Multi-Media and Information Technology, MMIT '08, pp. 104–107. IEEE Computer Society, Washington, DC, USA (2008). URL http://dx.doi.org/10.1109/MMIT.2008.185
18. Tian, X., Sun, Q., Huang, X., Ma, Y.: A dynamic online traffic classification methodology based on data stream mining. In: Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering - Volume 01, CSIE '09, pp. 298–302. IEEE Computer Society, Washington, DC, USA (2009). URL http://dx.doi.org/10.1109/CSIE.2009.904
19. Raahemi, B., Zhong, W., Liu, J.: Peer-to-peer traffic identification by mining ip layer data streams using concept-adapting very fast decision tree. In: Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence - Volume 01, ICTAI '08, pp. 525–532. IEEE Computer Society, Washington, DC, USA (2008). URL http://dx.doi.org/10.1109/ICTAI.2008.12
20. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pp. 97–106. ACM,

New York, NY, USA (2001). URL http://doi.acm.org/10.1145/502512.502529
21. Moore, A.W., Papagiannaki, K.: Toward the accurate identification of network applications. In: Proceedings of the 6th International Conference on Passive and Active Network Measurement, PAM'05, pp. 41–54. Springer-Verlag, Berlin, Heidelberg (2005). URL http://dx.doi.org/10.1007/978-3-540-31966-5_4
22. Dainotti, A., Gargiulo, F., Kuncheva, L.I., Pescape, A., Sansone, C.: Identification of traffic flows hiding behind tcp port 80. In: Communications (ICC), IEEE International Conference on, pp. 1–6 (2010). URL http://dx.doi.org/10.1109/ICC.2010.5502266
23. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American statistical association **58**(301), 13–30 (1963). URL http://dx.doi.org/10.2307/2282952
24. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII, IDA '09, pp. 249–260. Springer-Verlag, Berlin, Heidelberg (2009). URL http://dx.doi.org/10.1007/978-3-642-03915-7_22
25. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Siam International Data Mining Conference, pp. 443–448 (2007). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144.2279
26. NBAR2 or Next Generation NBAR - Cisco: [Online]. Available: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6616/qa_c67-697963.html, as of June 22, 2015
27. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explorations **11**(1), 10–18 (2009). URL http://dx.doi.org/10.1145/1656274.1656278
28. Bifet, A., Kirkby, R.: Data stream mining a practical approach. Citeseer (2009). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.192.1957
29. Li, W., Canini, M., Moore, A.W., Bolla, R.: Efficient application identification and the temporal and spatial stability of classification schema. Computer Networks **53**(6), 790–809 (2009). URL http://dx.doi.org/10.1016/j.comnet.2008.11.016
30. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. SIGCOMM Comput. Commun. Rev. **36**(5), 5–16 (2006). URL http://doi.acm.org/10.1145/1163593.1163596