2011 Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems

SMILER: Towards Practical Online Traffic Classification

Baohua Yang
Department of Automation,
Tsinghua University, China
Research Institute of
Information Technology,
Tsinghua University, China
ybh07@mails.tsinghua.edu.cn

Guangdong Hou
Department of Automation,
Tsinghua University, China
hgd05@mails.
tsinghua.edu.cn

Lingyun Ruan
Department of Automation,
Tsinghua University, China
rly07@mails.
tsinghua.edu.cn

Yibo Xue
Research Institute of
Information Technology,
Tsinghua University, China
Tsinghua National Lab for
Information Science and
Technology
yiboxue@tsinghua.edu.cn

Jun Li
Research Institute of
Information Technology,
Tsinghua University, China
Tsinghua National Lab for
Information Science and
Technology
junl@tsinghua.edu.cn

ABSTRACT

Network traffic classification is extremely important in numerous network functions today. However, most of the current approaches based on port number or payload detection are becoming increasingly impractical with the appearance of dynamic or encrypted applications. Even though some supervised learning based work were proposed, it is difficult to collect sufficient flow-labeled traces for training. On the other hand, online classification needs an early identification, which is still challenging for most well-known approaches. In this paper, we propose a semi-supervised learning based traffic classification approach named SMILER, which supports an early classification from the sizes of the first few packets (empirically 5 packets) of a flow. Experiments in real networks demonstrate that SMILER achieves 94% precision and 96% recall on average for all tested applications; even with disordered packets SMILER still works well. With a hybrid scheme, the performance is further improved. Meanwhile, SMILER performs fast in both classification and updating. All experimental results show that SMILER is practical for fast and accurate online traffic classification.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms

Algorithms, Measurement, Management

Keywords

Traffic Classification, Semi-Supervised Learning, Quality of Service

1. INTRODUCTION

Accurate and fast traffic classification according to application protocol types is one of the essential technologies employed in today's networking functions, as it is fundamental in the effective traffic control and network management tasks. For example, Internet Service Providers (ISP) rely on traffic classification to optimize network performance and provide quality-of-service (QoS) guarantees, and network operators also benefit from it in security issues, such as anomaly detection. On the other hand, application-aware network devices can achieve higher performance by reducing the sizes of rule-sets with knowledge of protocol types. In Snort's latest rule-set [10], only about 26.3% rules are for HTTP, thus with knowledge of application types, Snort could achieve better performance using a relatively smaller sub rule-set. Besides, researchers also need traffic statistics at application level to help explore the inherent characteristics of Internet and design future networks. All these demands make it a critical technology to classify network traffic accurately and fast in terms of application protocol type.

One type of traditional approach is based on port number. The idea was simple but successful because the traditional applications usually utilize the registered port numbers, *e.g.,* the port number 80 is assigned to HTTP protocol. However, this approach is becoming increasingly less effective since more and more applications do not use the standardized ports today[24, 13, 19]. In some cases, newly emerging applications may also employ well-known ports such as HTTP or FTP to hide themselves.

A more accurate type of approach is based on payload detection, which takes advantage of the Deep Packet Inspection (DPI) technology. In theory, traffic will be classified correctly with knowing the syntax of application protocols and the full examination of the payload. However, there are several reasons why this kind of approach is not always practical and efficient: a) Limitation of classification speed. Due to the high throughput of today's Internet (more than tens of Gbps), slow payload scanning becomes a frustrating solution in most cases; b) Privacy and legal issues. Internet users may not want their traffic content to be inspected; c) Signature. Accuracy of these approaches relies on precise signatures, which requires a prompt updating when a new application appears or the existing one changes. d) En-

978-0-7695-4521-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ANCS.2011.34

cryption. Payload-encrypted applications also decrease the efficiency of DPI-based approaches.

Recently, some novel works were proposed [22, 19, 17, 26, 12] using machine learning technology to classify traffic based on flow statistical information such as packet size, total bytes or flow duration. These machine learning based approaches need no payload detection. However, there are still some constraints to overcome. First, machine learning based classifiers vitally require a large amount of labeled flows for training, which are often difficult to acquire. Second, online traffic classification on lots of applications requires an early identification, which challenges the approaches that require statistics of complete flows, such as total bytes or flow duration. At last, traffic statistical results may vary in different networks, but most learning machine approaches are unable to tune their performance according to practical demands flexibly.

In light of the above analysis, we consider that a practical online traffic classification approach should satisfy the following requirements: (i) Accurate classification without accessing the packet content, especially for encrypted or NATed (Network Address Translation) traffic; (ii) An early identification from only the first several packets of a flow; (iii) Flexibility to be tuned according to users' demands and to be integrated with other solutions; (iv) High speed in both classification and updating.

In this paper, we propose a SeMi-supervIsed Learning based classifiER named SMILER, which provides fast online classification with only the sizes of the first few packets. Training with both labeled flows and unlabeled ones, SMILER guarantees encouraging classification accuracy for all tested applications while keeping high classification speed. Besides, our approach employs a confidence factor (c-factor) that indicates the credibility of the results. With a hybrid scheme (see Section 4.3), SMILER provides the flexibility to tune its performance to meet different requirements.

The highlights of our approach include:

- Early and accurate results are obtained from the sizes of the first few packets of each flow, which is essential for practical online traffic classification approaches. Especially, SMILER allows a classification on encrypted flows.
- Disordered packets can be handled. SMILER still works successfully even if some of the first few packets are out of order.
- Performance can be tuned flexibly. With a hybrid scheme, SMILER supports performance adjustment according to various demands in different situations.
- Flexibility of collaborating with other approaches is also provided. For example, it is easy to integrate SMILER with deep inspection solutions such as L7-filter [16].

In our prototype, SMILER is successfully integrated with NetMate [23], a flexible and extensible open-source network measurement tool. We utilize NetMate to collect packets and classify them into flows. The sizes of the first few packets of each flow are recorded in SMILER. The classifier of SMILER is lightweight, which is written in C code. Using the online prototype, we deployed SMILER system successfully within a campus network. The results demonstrate that SMILER provides a fast online classification (more than 6 Mfps[1]) with high accuracy over 90% for most applications. In addition, the best classification can be achieved with the sizes of the first 5 packets, which is similar as indicated in [1]. With the hybrid scheme, classification accuracy of SMILER can be further improved.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III introduces the basic assumptions and algorithms of semi-supervised machine learning. Section IV describes the details of designing SMILER system. Section V shows the evaluation and analysis of experimental results. The conclusions and suggestions for future work are given in the last section

2. RELATED WORK

In recent years, a number of researchers have been looking for practical methods for the traffic classification problem. Many of them try to study the Machine Learning (ML) technology [18], which usually utilizes several statistic features to describe the network traffic, including packet length, inter-arrival time distributions and the flow size [7, 17, 1], etc. Most ML-based traffic classification approaches focus on two types of learning: supervised learning and unsupervised learning. Though a lot of novel works have been proposed, we summarize the most relevant ones here due to space limitation.

2.1 Supervised Learning Based

Dunigan, T. et al. [7] first used the Principal Component Analysis (PCA) and density estimation to classify network traffic into different applications. The features used in [7] included packet size, inter-arrival times, and inter-packet correlations. The approach was evaluated only with a small dataset.

Roughan, M. et al. [22] introduced the Nearest Neighbor (NN) algorithm into traffic classification. This paper first categorized the classification features into five different classes: packet level (*e.g.,* packet length), flow level (*e.g.,* duration), connection level (*e.g.,* TCP window sizes), intra-flow and multi-flow, and outlined the classification framework based on statistical application signatures. By classifying applications into three different types: Bulk data, Interactive and Streaming, the lowest error rate varied from 2.5% to 3.4%. In [12], FPGA platforms were utilized to accelerate the identification based on k-NN. A high accuracy (above 99%) was achieved for classifying three multimedia applications while keeping sustaining 80 Gbps throughput.

Moore and Zue [19] used the Naive Bayes technique to classify Internet traffic. They took accuracy and trust as the evaluation metrics and achieved about 65% accuracy, while the accuracy could be improved to 95% overall with Naive

[1] fps means flow per second.

Bayes Kernel Estimation (NBKE) and Fast Correlation-Based Filter (FCBF) methods.

These approaches using supervised learning can get accuracy higher than 90% generally. However, they require large numbers of labeled flows of each application for training and are difficult to be deployed directly into online classification. Hence, other researchers utilize the unsupervised learning techniques.

2.2 Unsupervised Learning Based

McGregor, A. et al. published a work [17] that applied Expectation Maximization (EM) algorithm [6] to IP traffic classification. This work used a fixed set of classification features (packet size, inter-arrival statistics, byte counts, etc.) to cluster flows into different types of applications. The authors took a 6-hour Auckland VI trace from NLANR [9] for experiments and found that the traffic was divided into several groups of classes (bulk transfer, small transactions, multiple transactions, interactive traffic, etc.). However, the results did not show the influence of different features and how good the approach actually was.

Zander, S. et al. proposed a work [26] using AutoClass [5]. AutoClass is an unsupervised Bayesian classifier that also uses the EM algorithm to find the best cluster sets. The work was evaluated on three 24-hour traffic traces (Auckland-VI, NZIX-II and Leipzig-II) from NLANR [20]. As a further step from [17], this work also studied the evaluation and proposed an intra-class homogeneity metric.

Bernaille, L. et al. proposed a traffic classification approach using simple k-means clustering algorithm [1]. In contrast to previous work, this algorithm only detected the first few packets of a flow, which allowed an early classification. A similar clustering based work in [2] also utilized cluster algorithm to achieve an early classification with first few packets. However, both works cannot handle packet disorder and an additional identification is needed for the clustering results to get an exact application labeling.

2.3 Others

BLINC [14] proposed a new idea to associate host behaviors with application flows. Instead of studying individual flows, BLINC tried to classify hosts with their services accurately, such as web servers. However, BLINC cannot classify a single flow with its application type.

Erman et al. [8] first proposed a semi-supervised learning approach to train the classifier with only a few labeled and many unlabeled flows. They obtained the accuracy more than 90% for offline classification. However, when used in online classification, this approach got the accuracy lower than 80% even with the first 16384 packets of a flow, which also indicated a high storage cost and latency in online traffic classification.

Cao J. et al presented an online application identification approach [3] based on traffic statistical analysis such as flow duration, which investigated both host-level identification and flow-level identification with decision trees. However, [3] shows that those works based on whole flow statistics cannot support early detection, which prevents their usage in online scenarios.

In summary, supervised learning based approaches are strictly limited to large amount of labeled training flows, while unsupervised learning based methods do not require labeling. However, the results are often not that accurate for the traffic containing multiple types of applications, and labeling is still required to get the explicit type of the clustered traffic. Thus it is appealing to benefit from combining these two methods. Our approach is based on semi-supervised learning, and achieves fast and early classification with high accuracy, which implies it is a practical solution for online traffic classification in real networks.

3. SEMI-SUPERVISED LEARNING

3.1 Denotation and Assumption

Semi-supervised learning [4] that utilizes both labeled and unlabeled data for training has attracted more and more attention recently because it is usually difficult and expensive to collect a large amount of labeled training data.

In semi-supervised learning, the training data set is comprised of two parts: the labeled samples (each sample is denoted by a feature-label pair) $\boldsymbol{X^L} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{l}$ and the unlabeled samples $\boldsymbol{X^U} = \{\boldsymbol{x_j}\}_{j=l+1}^{l+u}$. For binary classification, $y_i \in \{-1, 1\}$; for multi-class classification, $y_i \in \{1, 2, \ldots, N\}$, where N is the number of tested classes. The final aim of semi-supervised learning is to learn a discriminant function $f : X \to Y$, which can calculate the correct label y for each sample x.

In real network traffic classification, substantial labeled samples are expensive to obtain. Thus, traditional supervised methods such as SVM [21] will be inappropritе due to the lack of labeled training set. On the contrary, unlabeled data is much easier to collect. For example, gathering network flows without labels is effortless through monitor tools, *e.g.,* tcpdump [11]. Hence, classifying network traffic based on semi-supervised learning method will be promising, which utilizes both labeled and unlabeled training data.

Before employing semi-supervised learning, certain assumptions have to be proposed first. [4] suggested gathering samples of the same class in one cluster with high probability. From this assumption, most samples should locate in regions that are far from the boundaries between the different clusters.

3.2 Transductive Support Vector Machine

TSVM [27] is an extension of the standard Support Vector Machine (SVM) to utilize both labeled and unlabeled data, which is described as follows: Assume that we have l labeled samples $\{x_i, y_i\}_{i=1}^{l}$ and u unlabeled samples $\{x_j^*\}_{j=1}^{u}$ with $x_i, x_j^* \in R^d$ and $y_i \in \{-1, +1\}$, we want to construct a linear classifier $sign(w^T x + b)$ that uses unlabeled data, especially when $l \ll u$.

For the linear separable case, the problem can be formulated in Equation (1).

$$\begin{aligned} \text{argmin} \quad & (y_1^*, \cdots, y_u^*, w, b): \\ & \frac{1}{2}||w||^2 \qquad (1) \\ \textit{subject to}: \quad & \forall_{i=1}^{n} : y_i[w \cdot x_i + b] \geq 1 \\ & \forall_{j=1}^{u} : y_j^*[w \cdot x_j^* + b] \geq 1 \end{aligned}$$

In Equation 1, y_j^* is the labeling on unlabeled data x_j^*. In this case, TSVM is to find a labeling of $y_1^*, \cdots, y_u^*$ for the tested data and a hyperplane $< w, b >$ which separates the originally labeled data from the newly labeled data with the margin of the maximum probability.

For non-separable data, we can also use the slack variables ξ_i as in the standard SVM, which is shown in Equation (2).

$$\begin{aligned} \text{argmin} \quad & (y_1^*, \cdots, y_u^*, w, b, \xi_1, \cdots, \xi_n, \xi_1^*, \cdots, \xi_u^*): \\ & \frac{1}{2}||w||^2 + T\sum_{i=0}^{n} \xi_i + T^* \sum_{j=0}^{u} \xi_j^* \qquad (2) \\ \textit{subject to}: \quad & \forall_{i=1}^{n} : y_i[w \cdot x_i + b] \geq 1 - \xi_i \\ & \forall_{j=1}^{u} : y_j^*[w \cdot x_j^* + b] \geq 1 - \xi_j^* \\ & \forall_{i=1}^{n} : \xi_i > 0 \\ & \forall_{j=1}^{u} : \xi_j^* > 0 \end{aligned}$$

Where T and T^* are parameters that allow a trade-off of the margin size against misclassifying training data or excluding test data, which can be tuned by users. Besides, the number of the tested samples to be assigned the label "+1", can also be specified manually. This parameter allows a adjustment for different performance requirements.

The labeling step of the unlabeled training data needs to be optimized in TSVM, which results in a combinatorial optimization problem. In theory, computing an exact TSVM is NP-hard, while we could utilize fast solutions to speed up the processing [15, 25].

There are two main advantages to utilize TSVM in traffic classification. First, its assumption of cluster agrees with the scenario of our problem, where plenty of unlabeled flows with few labeled ones are easy to collect. Furthermore, the classification phase needs low computation complexity to guarantee fast speed. Thus, we can achieve a fast online traffic classification with good accuracy.

4. SMILER CLASSIFICATION

In this section, we first introduce the framework of our SMILER system, then discuss the classification methods of binary-class case and multi-class case. After that, we detail the design of our hybrid scheme and the solution for traffic classification with packet disorder.

4.1 System Overview

There are three main parts in our whole system: *Metering*, *Training* and *Classification*. These three parts include six components overall: Meter, Judge, Other Slow Classifier (OSC), Trainer, Marker and Classifier. The framework is shown in Figure1 while functions of each component are described as follows.

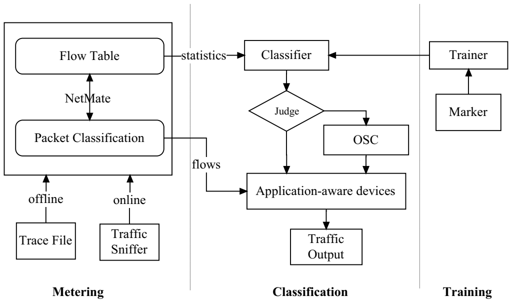

Figure 1: SMILER System Overview

4.1.1 Metering

There is only one component in this part, the Meter. We utilize Meter to collect flow information from off-line trace files or online traffic. There are large numbers of metering tools existed. We choose NetMate as our basic Meter component in our system because it is open-source, and quite flexible to insert new modules.

The original NetMate supports classifying the incoming packets into flows based on packet headers *e.g.,* (five tuples). Based on NetMate, we design a flow-management module to record the sizes of the first few packets of each flow for the classification. Noticeably, the packet size results will be collected sequentially with the packet id without waiting till the end of the flows, which facilitates the processing in the classification part.

4.1.2 Training

The training part includes the Trainer and the Marker. The Trainer component will generate the learning machine for the Classifier with the given training data. Usually, unlabeled flows can be collected by NetMate directly, while the labeled ones will be generated by other identification tools such as L7-filter after the collection. A flexible user interface is also provided as the Marker, which enables the manual checking on the labels of training flows. The accuracy of the labels will be improved with the additional checking step.

4.1.3 Classification

The Classification part consists of three components: Classifier, Judge and OSC. Main semi-supervised learning algorithms are implemented in the Classifier component. For basic classification, the Classifier component will calculate the classification results with a c-factor C (The definition will be given in Section 4.3).

If the hybrid scheme is enabled, there will be one more step. The Judge component will check the c-factor value with a given threshold τ, if $C \leq \tau$ (which means the classification results are not reliable enough), OSC will receive the picked low-confident flows and return the classification results on them. Thus different parts of the traffic may be classified by different components. SMILER provides the potential to support various types of OSC. In this paper, we take a DPI-based OSC, which typically classifies traffic accurately but slowly. However, we can tune the trade-off between the classification accuracy and the classfication speed by changing the value of τ. Generally, a smaller τ results in higher classification speed with lower accuracy ratio, and vice versa.

4.2 Classifier Design

In real networks, there are various types of network applications in the traffic simultaneously. We design SMILER to deal with the multi-class case for practical purposes. For the convenience and clarity of description, here we start with the binary-class case, which means the classification with $y_i \in \{+1, -1\}$, then we extend the method to the multi-class case.

4.2.1 Binary-Class Case

In the binary-class case, we utilize the captured flows to train a classifier that gives the result of true or false for testing flows. These training samples can be divided into two parts according to whether they own labels. We denoted the labeled set with $\{x_i, y_i\}_{i=1}^{L}$, where x_i is the i_{th} sample and y_i is the corresponding application type, and L is the size of the set. Meanwhile, unlabeled training set is denoted with $\{x_l^*\}$ and the size is U. We then design a classifier based on TSVM that predicts the labels of the testing flows with both labeled and unlabeled training data.

Standard TSVM can be formulated in Equation (3).

$$\begin{aligned} \min_{w,\{y^*\}_{j=1}^{U}} \quad & \frac{\lambda}{2}||w||^2 + \frac{1}{2L}\sum_{i=1}^{L} l(y_i w^T x_i) \\ & + \frac{\lambda^*}{2U}\sum_{j=1}^{U} l(y_j^* w^T x_j^*) \qquad (3) \\ \mathit{subject\ to:} \quad & \frac{1}{U}\sum_{j=1}^{U}\max[0, sign(w^T x_j^*)] = r \end{aligned}$$

Where λ^* is given by the user to control the influence of unlabeled data, and will be determined by cross-validation. r is usually set as the ratio of the samples with label "+1" in the labeled set while l is the loss function, which is set as $l(x) = l_2(x) = max(0, 1 - |x|)^2$.

Algorithm 1 Algorithm of Single-Class Case

```
Input:
    Labeled training data $(x_1, y_1), \cdots, (x_n, y_n)$
    Unlabeled training data $x_1^*, \cdots, x_n^*$
    Parameters $\lambda, \lambda^*$ in Equation (3)
Output:
    The hyperplane of classification $< w, b >$
Initialization:
    $(w, b)$ :=solve_svm_pro( $(x_1, y_1), \cdots, (x_n, y_n), \lambda$ )
$\lambda^{'} := 10^{-5}$  //some small value
while $\lambda^{'} < \lambda^*$ do
    do
    $(w, b, \vec{y}^*)$ :=solve_svm_pro( $(x_1, y_1), \cdots, (x_L, y_L)$,
    $(x_1^*, y_1^*), \cdots, (x_U^*, y_U^*), \lambda, \lambda^{'}$ ) //Solve this problem using the modified finite Newton linear  $l_2$-SVM
    Labels switching procedure
    while number of the samples switched labels $> 0$
    $\lambda^{'} = 1.5 \times \lambda^{'}$
end while
return $(w, b)$
```

Algorithm 1 shows how to train a TSVM classifier $< w, b >$ for binary-class. The classifier will calculate $R = wx_{unseen} + b$ for an unseen sample and give the label, as shown in Equation (4).

$$y = \begin{cases} +1 & R > 0 \\ -1 & R < 0 \end{cases} \qquad (4)$$

In Algorithm 1, the function of solve_svm_pro means to solve a problem in Equation (5).

$$\min_{w} \frac{\lambda}{2}||w||^2 + \frac{1}{2L}\sum_{i=1}^{L} l_2(y_i w^T x_i) + \frac{\lambda^{'}}{2U}\sum_{j=1}^{U} l_2(y_j^* w^T x_j^*) \quad (5)$$

Equation (5) is a supervised SVM problem, where $\lambda^{'}$ and $y_1^*, \cdots, y_U^*$ are both known parameters.

4.2.2 Multi-Class Case

Real traffic classification requires classifying multiple types of protocols. Hence the binary-class method of TSVM will be incapable of handling this situation. To extend to the multi-class case, we take advantage of a set of binary-class classifiers to construct the classifier for multiple classes. Then the task will be how to design these binary-class classifiers simultaneously and generate the multi-class classifier based on them. Generally, there are two types of combination techniques: *One verse One* and *One verse All*.

- One verse One
  For each pair in the two categories, a binary-class classifier will be trained. If there are N classes in total, $\frac{N(N-1)}{2}$ classifiers will be returned using this method. After the binary-class classifier generation, a vote is carried out among all classifiers to gain the final prediction label.

- One verse All
  For each category, the whole data set is divided into two parts: "target" sub one and "non-target" sub one. With this division, a two-class classifier can be trained. Thus for N classes, N classifiers will be generated. If prediction conflict occurs between different classifiers for the same testing sample, the one with the largest $\frac{R}{||w||}$ value will be taken.

We have tested both methods, and found that the *One verse One* method performs better than the latter one in solving traffic classification problems. Each time a one-verse-all classifier is trained, flows of "non-target" applications are considered as in one class, which can usually unbalance the training data. Besides, although samples of the "non-target" category may be gathered into several clusters according to their protocol types, we use the hyperplane classifiers as the binary-case classifier in the original feature space. However, it will be difficult for such a classifier to classify several clusters into two categories correctly. Moreover, the performance of the final combined classifier is strongly influenced by these hyperplane classifiers, while each *One verse One* classifier only needs to deal with one pair from each category, which makes it much easier to obtain the correct results.

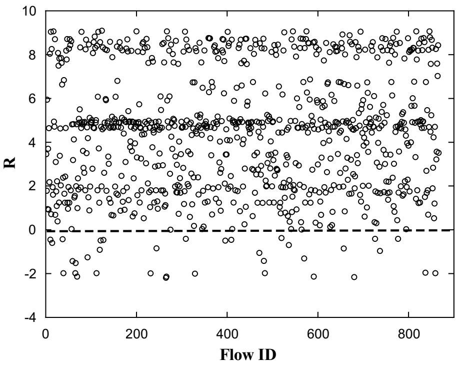

Figure 2: Classification Plane using HTTP Flows

An alternative explanation can be obtained intuitively. Samples with d features can be viewed as points in a d-dimensional hyperspace. The *One verse One* classifier has $\frac{N(N-1)}{2}$ hyperplanes as the final classifiers, which divide the classification space into more subspaces than the *One verse All* based method (only with N hyperplanes). Thus the *One verse One* based method promises more precise results in the classification.

Based on the above analysis and experimental results, we choose the *One verse One* method in SMILER. Both the training and classification procedures are described in Algorithm 2.

Algorithm 2 Algorithm of Multi-Class Case

```
Input:
    Labeled training data $(x_1, y_1), \cdots, (x_L, y_L)(y_i \in \{1, \cdots, N\})$
    Unlabeled training data $x_1^*, \cdots, x_U^*$
    Parameters $\lambda, \lambda^*$ in Equation (3) Tested sample $x_{unseen}$
Output:
    Predict label $y_{unseen}$ for one test sample $x_{unseen}$
    Training Part:
    for each pair of categories $i, j \in \{1, \cdots, N\}, i < j$ do
        train a one-verse-one classifier $< w_{ij}, b_{ij} >$ using Algorithm 1.
    end for
    Classification Part:
    set $y = [0, \ldots, 0] \in R^N$
    for each category $i, j \in \{1, \cdots, N\}, i < j$ do
        if $w_{ij}x_{unseen} + b_{ij} > 0$ then
            $y(i) = y(i) + 1$
        else
            $y(j) = y(j) + 1$
        end if
    end for
    return $y_{unseen} = arg \max_i y(i)$
```

4.3 Hybrid Scheme

The key idea of our hybrid scheme is to apply a confidence factor (c-factor) to help SMILER determine intelligently which part of traffic should be transferred to OSC. The definition of c-factor is put forward based on the observation of SMILER's classification results.

For testing data x, the classifier gained in SMILER is a set of hyperplanes $< w_{ij}, b_{ij} >$. Each hyperplane works out a discrimination value $R = wx + b$ for x. R can be viewed as an assigned distance from the hyperplane. If $R \geq 0$, x will be regarded as a certain class. Specially, when $R = 0$, the training data locates just at the hyperplane. We infer that the smaller the R is, the less confident the classification result will be. Moreover, our experimental results turn out to support this inference. Figure2 shows a hyperplane to classify HTTP flows. In this figure, most HTTP flows locate above the hyperplane (with positive R values), while only few flows locate below it. Notice that even for flows with negative R values, they still stay close to the hyperplane.

In Figure2, the testing set consists of 866 HTTP flows, which are shown as the rings, and the hyperplane is shown as a dotted line. If the data is positive (above the hyperplane), it can be classified correctly. Figure2 also shows that some samples are misclassified as negative (below the hyperplane), most of which locate near the hyperplane.

It is commonly believed that if the distance value to the hyperplane is larger, the prediction label will be more reliable. We define a confidence factor C based on this distance value. This factor should range in $[0.5, 1.0]$ and be proportional with $\frac{|R|}{||w||}$. When $\frac{|R|}{||w||}$ approaches ∞, C should approach 1.0. When $\frac{|R|}{||w||} = 0$, the testing sample locates at the hyperplane, which indicates an equal probability to be either class, so C should be 0.5 at this time.

Based on the analysis above, we give the definition of c-factor in Equation (6).

$$C = \frac{1}{1 + e^{-\frac{|R|}{||w||}}} \tag{6}$$

In our experiments, the final classifier is not a single hyperplane but a combination of different hyperplanes. Hence for each testing sample, every hyperplane gives the value of $\frac{|R|}{||w||}$, which will result in different C values. The minimum, maximum or mean value of C can be taken to present the confidence of the global classifier. In theory, when using the global classifier to classify a sample, only a few hyperplanes including the correct one will get close to the sample, while other hyperplanes will keep far from the sample location. Thus C_{max} and C_{avg} are usually quite large, which can not present the confidence precisely. This makes us believe that C_{min} is the most effective parameter to measure the confidence, which will be taken when $\frac{|R|}{||w||} = \min\{\frac{|R_{ij}|}{||w_{ij}||} | i, j \in \{1, 2, \ldots N\}, i < j\}$

4.4 Packet Disorder

In real networks, some packets may be out of order after multi-path transferring. Currently, most solutions are based on the packets reassembly technique. This technique may cause some problems for fast online traffic classification. For example, if the classification approach needs information of the first k packets of a flow, the classification cannot be started until all the k packets arrive and are reassembled. Even worse, some packet may be lost and need to be re-transferred, thus such type of classification will cause large processing latency and storage complexity.

Table 1: Global Statistics of Traces

Trace	Date	Flow	Packet	Byte
$trace_{08}$	08/12	377503	37296499	32768012521
$trace_{10}$	10/07	3394	2252118	2203749653

Table 2: Application Statistics of Traces

Apps	$trace_{08}$		$trace_{10}$	
	Flow %	**Byte %**	**Flow %**	**Byte %**
HTTP	24.32	7.64	19.17	41.32
P2P	30.11	29.90	44.04	29.34
DNS	7.60	0.34	N/A	N/A
HTTPS	1.64	0.21	7.0	1.95
FTP	1.13	0.14	6.23	11.80
SSH	0.98	0.13	11.90	0.33
IM	0.81	0.10	0.73	0.30
SMTP	0.74	0.06	1.00	4.78

We solve this problem by designing the classifier compatible with missing features. All valid features should be no less than 0, and we set the values of the missing features to 0. Since $R = wx + b$, the procedure is the same as setting the corresponding w to 0. That means when some feature is missed, we use an approximate hyperplane to do the classification instead of the exact one. The new hyperplane will be parallel to the dimension of the missing feature in the feature space. Commonly, this modification might bring some accuracy loss.

5. EVALUATION AND ANALYSIS

5.1 Traffic Traces and Testbed Setup

As most current publicly available trace sets (*e.g.,* NLANR [9]) do not contain any application information for security and privacy reasons, our evaluations are based on real traces collected in a large institute containing hundreds of various servers in a campus network. The trace set consists of two parts: $trace_{08}$ (collected in 2008) and $trace_{10}$ (collected in 2010). However, it would be of great help for researchers if a reliable and accurate marked trace bench could be proposed.

For time consideration, these traces are not quite huge but both are collected in different periods for the purpose of credibility. To validate SMILER's classification results, we first label the testing flows in the traces by examining the content[2].

The basic statistic information and the application distribution are shown in Table 1 and Table 2 respectively. In Table 2, we find that HTTP and P2P based applications take a large percentage in both flow and byte. Comparing the two trace files, some interesting characteristics are observed. For example, the byte percentage of HTTP increases by more than 400% after two years while the flow percentage decreases, which is mostly due to the video applications over HTTP have increased a lot in recent years. Besides, the percentage of HTTPS also increases a lot, which implies that people are paying more and more attention to their security and privacy. From the traces, we focus on 6 representative applications for our evaluations, including Web (non-video HTTP), Video (over HTTP), FTP, SMTP, BT and SSH. Although only 6 famous applications are considered in our current evaluations, we will continue to collect more results on other applications.

Our test bed is integrated with NetMate, which is used as the metering platform. The test bed is running on Ubuntu 10.04 OS (Linux kernel v2.6.32-24-generic), installed on a PC with Intel CPU T7500 (2.2 GHz, 4 M Cache) and 2 GB DDR-II memory. All programs are written in C codes.

5.2 Performance Parameters

In order to evaluate the efficiency of SMILER, we utilize a few performance parameters. To describe these parameters, several definitions are declared first. For an application category A:

True Positive (TP) The flows of application A are classified as A correctly, which is a correct result for the classification;

True Negative (TN) The flows not in A are classified as not in A, which is also a correct result;

False Positive (FP) The flows not in A are misclassified as A. For example, a non-P2P flow is misclassified as a P2P flow. FP will produce false warnings for the classification system;

False Negative (FN) The flows in A are misclassified as some other category. For example, a true P2P flow is not identified as P2P. FN will result in identification accuracy loss.

The performance parameters utilized in our evaluations, including *precision*, *recall* and *accuracy*, are also widely employed in traffic classification and many machine learning technologies. Following are the definitions:

Precision The percentage of samples classified as A that are really in class A, which can be calculated with $TP/(TP+FP)$. Precision can be given for a specified class or for multiple classes on average;

Recall The percentage of samples in class A that are correctly classified as A, which can be calculated with $TP/(TP+FN)$. Recall can be given for a specified class or multiple classes on average, too;

Accuracy The percentage of samples that are correctly classified, which can be calculated with $(TP+TN)/(TP+TN+FP+FN)$. Notice this overall accuracy is calculated totally with all tested classes;

Besides the above parameters, we also adopt a *confusion matrix* to help describe the details of the classification results.

Confusion Matrix An $N \times N$ matrix, whose element (i, j) is the percentage of flows which belong to class i but are misclassified as class j. This matrix shows the details of the incorrect classification results, which can be used for

[2] Note only the popular applications are labeled to test.

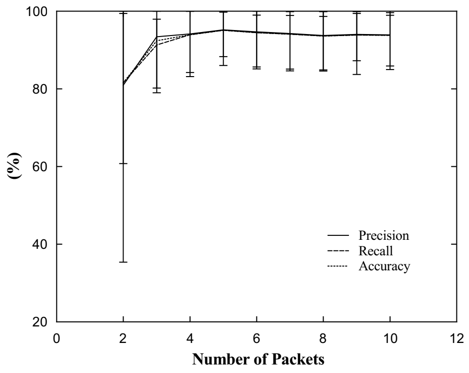

Figure 3: Precision, Recall and Accuracy Results with Different Number of Packets

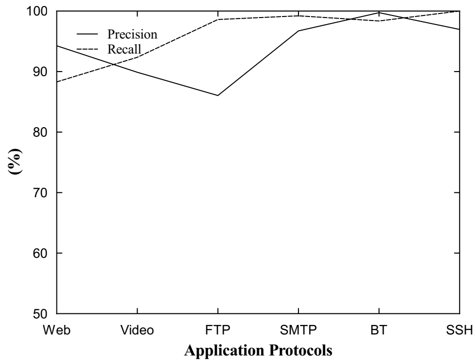

Figure 4: Classification Results for Applications with 5 Packets

exploring inherent characteristics of each application.

5.3 Classification Results

5.3.1 Results with Different Packet Number

Our approach tries to classify a flow with only the sizes of the first k packets (without TCP control packets). In theory, large k would improve the accuracy of the classification results, but increase the latency. Thus it is appealing to choose a suitable k to achieve an accurate result in time.

To find out a reasonable k, we carry out the classification experiments with different k values. To train SMILER, we take 100 labeled flows and 250 unlabeled flows randomly from the training set. Each experiment is repeated 5 times on different training samples with k from 2∼10, and we take the average one as the final result. Experimental results of precision, recall and accuracy are shown in Figure3.

Precision In Figure3, the average precision curve increases from about 80% to more than 94% when packet number increases from 2 to 4. For packet number over 4, the average precision keeps over 93%, where the best one is 95.2% with 5 packets. In addition, the variance of precision among different applications is reduced from around 40% to less than 20% when packets number increase from 2 to 4, while for more than 4 packets, the variance only changes little. This result implies that the precision increases notably with the increasing of packet number when $k \leq 4$, and keeps stable when $k > 4$.

We also check the category of applications with the worst and the best precision with the smallest packet number, which are Video (over HTTP) and BT respectively. This interesting result depicts that it is very challenging to classify popular video applications (around 60% precision) with only few packet sizes, while BT can be tackled quite well with few packets (nearly 100%).

Recall As shown in Figure3, the average recall curve is quite similar with the one of average precision, which is increased from about 80% to 94% with packet number from 2 to 4. With packet number increasing from 4 to 10, the average recall fluctuates between 93.61% (with 8 packets) and 95.09% (with 5 packets), which also implies that more than 4 packet sizes guarantee a good result. The variance of recall is over 60% with only 2 packet sizes, and decreases sharply with more than 4 packets. For example, with 5 packet sizes, the variance is only 11.7%. We further check the detailed results with 2 packets. The worst application is Video (over HTTP) as well, which suggests that the identification of Video (over HTTP) is an appealing question for today's traffic classification research.

Table 3: Confusion Matrix

ratio %	Web	Video	FTP	SMTP	BT	SSH
Web	88.29	6.34	3.77	0.97	-	0.63
Video	6.31	92.36	0.98	-	0.18	0.18
FTP	0.80	0.60	98.60	-	-	-
SMTP	-	-	-	99.21	0.79	-
BT	0.85	0.13	0.13	-	98.39	0.49
SSH	-	-	-	-	-	100.00

Accuracy Figure3 also illustrates the accuracy of all applications classification, which shows a growth from 81.6% to 93.9% when packet number rises from 2 to 4, and for more than 4 packets, the accuracy keeps over 93.6%. Surprisingly, the best accuracy is also obtained at 5 packets, which is 95.1%. This result is quite similar to that indicated in [2], which achieves above 90% accuracy with 4 packets by enterprise trace.

Best Number Selection From results in Figure3, we summarize a basic conclusion as follows. When the packets number k is small (such as 2), the results are not good for all the metrics and the variances are large. This is partly because only two features have not enough information to represent the characteristics of the flows' distribution. With the increasing of k, the performance improves rapidly at first and then keeps at a high level, and the best performance occurs at 5 packets.

With the first 5 packets, Figure4 shows the precision and recall results of all tested applications, including Web, Video, FTP, SMTP, BT and SSH. From Figure4, the precision ranges from 86.04% to 99.73%, while the recall ranges from 88.29% and 100%. All these results show that SMILER

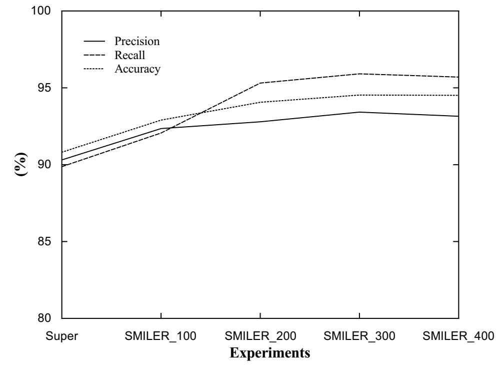

Figure 5: Accuracy Comparison between SMILER and Supervised Learning

achieves good precision and recall at the same time for all applications.

Confusion Matrix In Table 3 we give the confusion matrix with the first 5 packets. From this matrix, we can see the detailed classification results of all tested applications. Take the first column for example, 6.31% Web flows are misclassified as Video, with 0.80% as FTP and 0.85% as BT. Most error ratio values are small, while the largest one occurs between Web and Video, which implies the distributions with sizes of the first 5 packets between Web and Video flows are similar. The possible reason is that they both utilize HTTP protocol, thus similar initializations might be utilized.

In summary, the best results can be achieved when k equals to 5. Consequently, we take the sizes of the first 5 packets as our classification features in the following experiments.

5.3.2 SMILER vs Supervised Learning

To validate the effectiveness of SMILER, we compare the performance of SMILER with the representative supervised learning classifier, SVM. In each experiment, 25 flows are labeled for training the super learning approach and different numbers of unlabeled flows are also provided randomly for the training.

Figure5 shows the comparison results in terms of precision, recall and accuracy. In experiment "Super", we classify traffic using super learning approach which is trained by 25 labeled flows, and for other experiments, we utilize SMILER trained with different unlabeled flows (100, 200, 300 and 400) besides the labeled ones. From Figure5, we can see that on all three performance parameters, SMILER obtains better results than the supervised one. Specially, experiment "SMILER_300" achieves an average accuracy of 94.53%, while for the super learning classifier it is only 90.81%. All the results show that after the unlabeled data is added for training, the precision, recall and accuracy results are all improved.

We also notice that with unlabeled training data growing from 300 to 400, the performance doesn't increase but drop a little. This may be because of the random data selection and the over-fitting caused by excessive training data. On the basis of our experiment results, we suggest the ratio between the labeled data and the unlabeled one be chosen around 10.

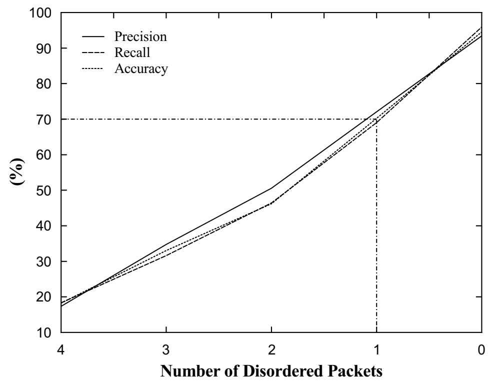

Figure 6: Performance Results with Different Number of Disordered Packets

5.3.3 Packet Disorder

Out of order packets are difficult to handle for most machine learning based traffic classification approaches because disordered packets cause the missing of features for online classification. To evaluate the performance of SMILER with packet disorder, we carry out experiments with different numbers of out of order packets. Since the definitions of packet disorder vary from each other, to show a fair comparison, we consider packets with wrong order as loss in our experiments, because we believe that a practical online traffic classification should achieve an early identification even with incomplete information. We randomly drop several packets from the first five packets and compare the average results in terms of precision, recall and accuracy.

Figure6 shows the results, from which we can see that packet disorder brings down all three performance metrics, however our approach still works successfully for all cases. It is clear that better performance is achieved with less out of order packets. Specially, our approach achieves about 70% average accuracy with a single out of order packet. On the other hand, the performance of SMILER increases almost linearly by reducing disordered packets. All the results demonstrate the good stability of SMILER for packet disorder.

5.3.4 Hybrid Scheme

Flexibility of supporting hybrid scheme is another advantage of SMILER. To validate the accuracy improvement by hybrid scheme, we carry out experiments with different c-factor values using a DPI based OSC. In Figure7, we show the accuracy results with a c-factor relax ratio, which means the upper-bound of accepting flows over c-factor. For example, 25% relax ratio means we choose the splitting margin as $\min(C) + 25\% \times (\max(C) - \min(C))$. From Figure7, we can see that the accuracy is improved by transferring flows to the OSC. With 40% relax margin, nearly 100% accuracy is achieved for all three definitions of C.

Figure7 also demonstrates that the performance using C_{min} is better than that with C_{max} and C_{avg}, which is inferred by our analysis in Section 4.3. All results illustrate that our hy-

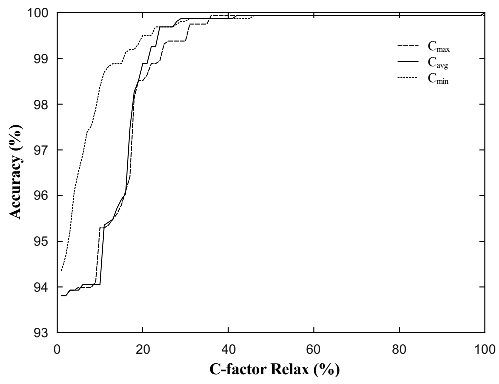

Figure 7: Accuracy Comparison of Hybrid Scheme with Different C-factors

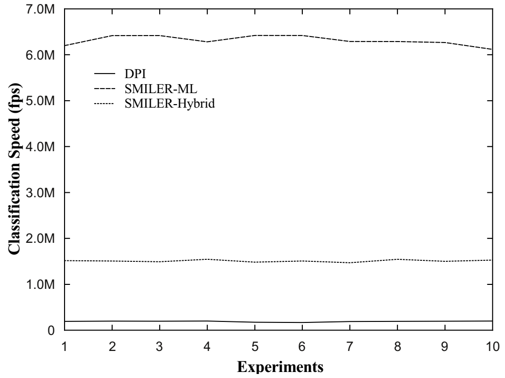

Figure 8: Classification Speed Comparison

brid scheme improves the classification accuracy effectively.

5.3.5 Classification Speed and Updating Time

We compare the classification speed of SMILER-ML (pure machine learning based) and SMILER-Hybrid (10% of SMILER's traffic is processed by OSC) with the traditional DPI (packet content detection) based classifier. To estimate the typical classification speed of the DPI classifier, we assume that only 16 bytes data per packet in the first five ones of each flow are examined, and the data structures of DPI only require 2 M bytes storage, which is quite favorable for simulating current DPI classifiers. 10 experiments are carried out, among which each one runs a classification of 30 seconds. Besides, we use the million flow per second (Mfps) as the classification speed unit.

The comparison results are shown in Figure8. Figure8 illustrates that both SMILER-ML (over 6 Mfps) and SMILER-Hybrid (over 1.5 Mfps) achieve much higher classification speed than DPI based one (only about 0.2 Mfps). This is because SMILER imposes no detection of packet content, which is usually quite expensive in both time and storage. All results in Figure8 propose that SMILER is also quite fast and lightweight in speed.

Meanwhile, the preprocessing phase of SMILER is also very fast in our experiments. For example, training with a dataset of 1000 flows (100 labeled and 900 unlabeled ones) only needs less than 0.1 second, which is much smaller than the preprocessing of DPI based methods (usually more than tens of seconds on large rule-set). This quick preprocessing promises SMILER as a practical approach in updating.

5.4 Discussion

Experimental results show that SMILER achieves superior performance in terms of precision, recall and accuracy. By utilizing the sizes of the first 5 packets of flows, SMILER provides an average precision of about 94%, recall over 96% and accuracy over 95% for all tested applications. These results imply an inherent relationship between packet sizes and application types. Compared with supervised learning approaches, SMILER also provides an evident performance improvement by employing unlabeled flows.

Although experimental results prove SMILER a practical online classification approach, there are still several interesting observations to be discussed. To compare SMILER with supervised learning approaches, we repeat each experiment 5 times with randomly chosen training data and a performance jitter is obtained among them. Take the supervised learning one for example, the fourth accuracy result is about 10% higher than the second one. This indicates that training data selection can affect the classification result, which is universal in machine learning based approaches actually.

An interesting question is why the sizes of the first few packets can determine the application types so accurately in SMILER. Since various network applications are designed for different specialized aims, we guess the main reason might be the similar design requirements for applications of the similar goal. For example, we observe that most packets in the first five ones of Video over HTTP and FTP flows are 1518-byte sized, but this number is quite small in IM protocols (*e.g.,* Gtalk). This similarity may be due to the reference to related protocols when new applications are designed.

Another question is how to design a novel hybrid scheme. The hybrid schemes aims to take advantage of other approaches to improve the accuracy. Take the DPI based OSC for example, accuracy can be improved by transferring selected flows to OSC, however, an unsuitable selection may even result in worse performance. If most easily-classified flows are transferred to OSC, the classification speed will be reduced seriously without any accuracy improvement. That is the main reason why we utilized a distance-based c-factor to optimize this selection. Experimental results show that our current c-factor works well for all tested applications. However, it is still an interesting topic to design effective c-factors in future.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a practical online traffic classification approach called SMILER, which utilizes semi-supervised learning technologies to classify traffic based on the sizes of the first few packets of each flow. SMILER allows an early identification without any inspection of packet content. With disordered packets, SMILER still works well. Furthermore, a hybrid scheme is proposed to help SMILER satisfy various performance requirements.

In order to evaluate SMILER, we deploy our approach in a real campus network. Experimental results show that SMILER achieves about 94% precision, over 96% recall and over 95% accuracy on average with only the sizes of the first 5 packets of each flow. Even with disordered packet, SMILER still shows a good robustness in performance. Our approach also performs well in classification speed and updating time. The average classification speed is over 6 Mfps, which is tens of times higher than that of packet content inspection based approaches. With large training sets (*e.g.,* 1000 flows), SMILER only requires less than 0.1s to generate the new classifier. The hybrid scheme further improves the classification performance. All these results prove that SMILER is quite practical for fast and accurate online traffic classification.

Through the experimental results, there are still several open issues. For example, with very short flows (less than three packets) or lots of disordered packets, SMILER's accuracy might be reduced, which implies that approaches that are more robust should be designed in future. Our future work also includes improving the classification performance of SMILER and evaluating on various network environments.

7. ACKNOWLEDGMENTS

We would like to thank the reviewers for their insightful comments, and people of the organizations that support our work in evaluations.

8. REFERENCES

[1] L. Bernaille, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006.

[2] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proceedings of the 2006 ACM CoNEXT Conference*, pages 1–12. ACM, 2006.

[3] J. Cao, A. Chen, I. Widjaja, and N. Zhou. Online identification of applications using statistical behavior analysis. In *IEEE GLOBECOM*, pages 1–6. IEEE, 2008.

[4] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[5] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. *Advances in Knowledge Discovery and Data Mining*, 180, 1996.

[6] A. Dempster, N. Laird, D. Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B: Methodological*, 39(1):1–38, 1977.

[7] T. Dunigan, G. Ostrouchov, and L. UT-Battelle. Flow characterization for intrusion detection. *ORNL/TM-2001/115, November*, 2000.

[8] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation*, 64(9-12):1194–1213, 2007.

[9] W. N. R. Group. Auckland-vi trace. http://www.wand.net.nz/wits/auck/6/, 2001.

[10] S. Inc. Snort. http://www.snort.org.

[11] V. Jacobson, C. Leres, and S. McCanne. Tcpdump. http://www.tcpdump.org/.

[12] W. Jiang and M. Gokhale. Real-time classification of multimedia traffic using fpga. In *2010 International Conference on Field Programmable Logic and Applications*, pages 56–63. IEEE, 2010.

[13] T. Karagiannis, A. Broido, and M. Faloutsos. Transport layer identification of p2p traffic. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 121–134. ACM New York, NY, USA, 2004.

[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *Proceedings of the ACM SIGCOMM*, page 240, Philadelphia, Pennsylvania, USA, 2005. ACM.

[15] S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6(1):341, 2006.

[16] J. Levandoski, E. Sommer, and M. Strait. L7-filter. http://l7-filter.sourceforge.net/.

[17] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. *Lecture Notes in Computer Science*, 3015:205–214, 2004.

[18] T. Mitchell. Machine learning and data mining. *Communications of the ACM*, 42(11):36, 1999.

[19] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the 2005 ACM International Conference on Measurement and Modeling of Computer Systems*, pages 50–60. ACM New York, NY, USA, 2005.

[20] NLANR. Passive measurement and analysis (pma) project. www.nlanr.net.

[21] V. N.Vapnik. *The nature of statistical learning theory.* Springer, USA, 1995.

[22] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 135–148, Taormina, Sicily, Italy, 2004.

[23] C. Schmoll and S. Zander. Netmate. www.ip-measurement.org/tools/netmate.

[24] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th International Conference on World Wide Web*, pages 512–521. ACM New York, NY, USA, 2004.

[25] V. Sindhwani and S. Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 484, Washington, USA, 2006.

[26] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Conference on Local Computer Networks*, volume 30, page 250. IEEE, 2005.

[27] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2006.