Controlling False Alarm/Discovery Rates in Online Internet Traffic Flow Classification

Daniel Nechay, Yvan Pointurier, and Mark Coates
Department of Electrical and Computer Engineering
McGill University, Montréal, Québec, Canada
Email: daniel.nechay@mail.mcgill.ca, yvan@ieee.org, mark.coates@mcgill.ca

***Abstract*—Existing Internet traffic classification techniques achieve impressively low misclassification rates, but do not provide performance guarantees for particular classes of interest. In this paper, we propose two novel online traffic classifiers — one based on Neyman-Pearson classification and one based on the Learning Satisfiability (LSAT) framework — that can provide class-specific performance guarantees on the false alarm and false discovery rates, respectively. We also present a preprocessor for our classifiers that predicts, after the reception of only a small number of packets, whether a flow will be 'large' (as defined by a network operator). Only these resource-intensive flows are passed to the classifier, greatly reducing the computation burden imposed. We validate our methodology by testing our approaches using traffic data provided by an ISP.**

I. Introduction

Online Internet traffic classification involves the automatic association of a user-defined class to a traffic flow after the arrival of only a small number of its packets. Accurate traffic classification enables the provision of application-specific Quality-of-Service (QoS) guarantees [1] and simplifies the enforcement of service level agreements. ISPs can use the flow labels to prioritize time-sensitive traffic (e.g., VoIP and video-conferencing) and block (or throttle) undesirable traffic (e.g., peer-to-peer). The classification results also provide useful statistics for network provisioning and security assessment.

Unfortunately, classifying Internet traffic is not a trivial task. Port numbers assigned by the Internet Assigned Numbers Authority (IANA) are not a good indicator of the application, because application developers frequently ignore the assignments, and use arbitrary (or even random) port numbers. Recently, machine learning techniques, including classification and clustering methods, have been proposed for traffic classification, with the underlying features derived from flow statistics [1]–[10]. One important attribute missing from the classification methods proposed thus far is the provision of application/class-specific performance guarantees, particularly bounds on *false alarm rates* and *false discovery rates*. In this paper, we address this deficiency, proposing online traffic classifiers that are designed to satisfy constraints on these performance metrics.

The false alarm rate for class i (FAR_i) refers to the expected fraction of the flows that do not belong to traffic class i that are incorrectly classified as belonging to i. This concept can be extended to a pairwise false alarm rate FAR_{ij} for classes i and j, which specifies the expected proportion of the flows belonging to class j that are incorrectly labeled as class i by the classifier. The false alarm rate differs from the false discovery rate, FDR_i, which is the expected fraction of incorrectly classified flows among all traffic flows classified as class i. This can also be extended to a pairwise false discovery rate, FDR_{ij}, which is the expected fraction of all flows classified as class i which do in fact belong to class j.

A network operator can select which of these constraints is appropriate according to the specific goal of the classification procedure. To illustrate the need for controlling both of these performance metrics, consider a scenario where 1100 flows are simultaneously present in a network, with the following actual breakdown: 900 HTTP flows, 100 peer-to-peer (P2P) flows and 100 VoIP flows. Here the HTTP class dominates and it is possible to obtain a relatively high overall classification accuracy (low misclassification rate) by simply classifying the majority of flows as HTTP. However, such a strategy leads to unacceptably low detection rates for the VoIP and P2P classes. Any procedure, such as prioritization or rate-limiting, applied to flows that such a classifier labels as VoIP or P2P, will be ineffective. It is clear that optimizing the global misclassification rate is insufficient.

Our approach allows the operator to set class-specific performance guarantees. For example, suppose that the operator wants to give higher priority to VoIP traffic and wants to bound the number of flows that are incorrectly prioritized. By specifying that the pairwise FAR of each pair of classes must not exceed 5%, the resultant classifier ensures that at most 50 non-VoIP flows would be classified as VoIP, and no more than 10 of the true VoIP flows are incorrectly identified as HTTP or P2P. Alternatively, suppose that the operator wishes to block P2P traffic, but ensure that of the blocked flows only a small percentage are incorrectly blocked. By training a classifier with the P2P FDR constraint set to 1%, the operator can guarantee that of every 100 blocked flows, at most one is incorrectly blocked.

A. Contribution

Although numerous traffic classification methodologies have been proposed, none provide strict guarantees on performance, particularly false alarm/discovery rates. Such guarantees are essential if a network operator wishes to employ traffic flow control procedures based on the classification labels. The primary contribution of this work is the proposal of two novel online classifiers that provide class-specific performance guarantees. The first classifier controls false alarm rates, and is based on a multi-class generalization of Neyman-Pearson (NP) classification [11]–[15]. The second controls false discovery rates, and is based on a multi-class generalization of the Learning Satisfiability framework (LSAT) [16]. The FAR/FDR constraints are the parameters that must be specified by the operator. Both classifiers are implemented as generalizations of the 2ν-Support Vector Machine (2ν-SVM) [17], which implies that classification can be performed online with a small number of elementary arithmetic operations. Our classifiers operate solely on statistics derived from packet headers, which are relatively easy to obtain.

We also describe a pre-processing mechanism to speed up the classification process, which employs a predictor to identify "large" flows (as defined by the network operator). The fast pre-processor ensures that the more computationally intensive multi-class classification procedure does not have to be applied to small flows, which terminate before any traffic control method can have a meaningful impact.

B. Paper Structure

The rest of the paper is organized as follows. Section II discusses the related work, Section III provides formal statements of the problems we address, and Section IV provides the background on 2ν-SVMs, Neyman-Pearson classification, and the learning satisfiability framework. Section V presents the novel algorithms used in our classifiers. Section VI explains how we obtained and processed the data for our validation experiments, and Section VII provides the results from our experimentation with real traffic. Section VIII summarizes the work and indicates future avenues of research.

II. Related Work

There are two broad categories of network traffic classification methods: Deep Packet Inspection (DPI) [18], [19] and Shallow Packet Inspection (SPI) [1], [3], [10], [20]–[22]. DPI involves payload analysis; it is relatively slow, raises privacy and legal concerns, and is rendered ineffective when flows are encrypted. SPI determines flow features by collecting statistics derived from analyzing the headers of the packets in a flow and uses this information to classify the flow.

Of the SPI methods, there have been proposals for both supervised and semi-supervised classification. The semi-supervised approaches [8]–[10] do not require that all flows are labeled during training; clustering techniques (k-Means, Gaussian mixture models, and spectral clustering) are applied to identify suitable classes. Subsequent labeling of a few flows in each cluster leads to reasonably good classification performance [10]. This has the potential benefit of allowing the discovery of new application classes.

A substantial body of work is emerging on supervised classification [1]–[7]. The cited papers are only a representative sample of traffic classification research; for a much more complete survey, see [23]. Labeling can be done by manual inspection or using automated tools such as Bro [24]. Many classification methods have been evaluated: Naïve Bayes, C4.5 Decision trees, Bayesian network and Naïve Bayes tree algorithms, k-Nearest Neighbours, Linear Discriminant Analysis, Quadratic Linear Analysis, and Support Vector Machines. None of these algorithms appears to significantly outperform the others, and issues such as feature selection appear to play a more important role than the underlying classification methodology. In [2], [3], the authors collected a large number of flow statistics and applied discriminator selection methodologies to identify the most informative and relevant features. We follow a similar approach to reduce the size of the feature space.

Online classification necessitates that decisions are made after the observation of a small fraction of the packets in a flow. Bernaille et al. demonstrate that it is possible to accurately classify traffic based on flow statistics gathered from only the first five packets [8]. Erman et al. develop a layered classification architecture, where decisions are made, and then refined, based on features collected at packet milestones (8 packets, 16, 32, etc.) [10]. We base our classification on statistics derived from the first k packets; in our simulations we set $k = 6$ as this is the earliest that we can start classifying traffic with our data set.

There has been limited work on developing techniques that allow an operator to control traffic classification performance [23]. González-Castaño et al. focus on identifying whether a flow is a P2P application (binary classification), and describe techniques for tuning SVM parameters to control class-specific misclassification rates [5]. Karagiannis et al. present a heuristic approach in [25] which provides coarse-tunability of performance. None of these techniques is designed to provide performance guarantees.

With regard to the statistical methodology, we adopt a Neyman-Pearson classifier to control false alarm rates [11]–[15] and employ the Learning Satisfiability (LSAT) framework to control false discovery rates [16]. The algorithms we present are novel extensions of these learning methodologies to multi-class problems.

III. Problem statements

In this section, we formalize three traffic classification problems with direct relevance to network operation management. The problem of online traffic classification consists of associating a "class" to a network flow, given information or "features" that can be derived from the first several packets of the flow. A flow is a sequence of TCP packets with the same source IP address, destination IP address, source port, and destination port. We call X the d-dimensional random variable corresponding to the flow features; here the feature space is $\mathbb{R}^d$. Denote by $x \in \mathbb{R}^d$ the features of a sample flow. To each flow is associated an output Y. Call $y \in \mathbb{R}$ the output for a sample flow. In the case of large flow prediction (problem 3 below), $Y \in \mathbb{R}$ is the number of bytes in the flow at termination. In the case of the classification problems, Y is discrete and finite, and we call $Z = Y \in \{1 \ldots, c+1\}$ the class of the flow and z the class of a sample flow. In this case, each flow can be associated to one of $c + 1$ classes, where the $(c+1)$th class is a default class for flows that cannot be associated to one of the first c classes. Notice that Y and Z are random variables: for any given feature vector, the associated output or class is a probabilistic quantity. We denote by Q an unknown probability measure on $\mathbb{R}^d \times \mathbb{R}$ and we assume that pairs (X, Y) are independently identically distributed according to Q.

A. Problem 1: Application classification with FAR constraints

The goal of Problem 1 is to design a multi-class classifier $f : \mathbb{R}^d \rightarrow \{1, \ldots, c+1\}$ which, given the features x of the flow, associates a class $\hat{z} = f(x)$ to the flow. The classifier must meet *false alarm rate* constraints specified by the operator. These constraints may be either single-class or pairwise constraints. The single class constraints take the form of a maximum false alarm rate α_i for each class $i \in \{1, \ldots, c\}$. The single-class false alarm rate $\mathrm{P}(\hat{Z} = i | Z \neq i)$ for a class i is the probability that a flow is detected as class i while, in fact, it is not of class i. The pairwise false alarm rate $\mathrm{P}(\hat{Z} = i | Z = j)$ for a pair of classes (i, j) is the probability that flow h is detected as class i while, in fact, it is of class j. The pairwise constraints take the form of maximum allowable pairwise false-alarm rates $\alpha_{i,j}$ for any two classes $i \neq j$. In this problem setting, we allow the operator to specify multiple single-class constraints or multiple pairwise constraints, but not a mixture of the two types of constraints, because this can lead to conflicting requirements.

The classifier must then minimize the total misclassification rate while meeting the false alarm rate constraints set by the operator. Formally, this involves finding the classifier

$$f^* = \arg\min_f \mathrm{P}(\hat{Z} \neq Z) \text{ where } \hat{Z} = f(X) \tag{1}$$

such that, for single class constraints:

$$\forall i \in \{1, \ldots, c\} : \mathrm{P}(\hat{Z} = i | Z \neq i) \leq \alpha_i, \tag{2}$$

or, for pairwise constraints:

$$\forall i, j \in \{1, \ldots, c\}, j \neq i : \mathrm{P}(\hat{Z} = i | Z = j) \leq \alpha_{i,j}. \tag{3}$$

B. Problem 2: Application classification with FDR constraints

Problem 2 also addresses the design of a multi-class classifier, but the constraints are now on false discovery rates. The single class false discovery rate for class i is defined as $\mathrm{P}(Z \neq i | \hat{Z} = i)$. By bounding the FDR, the operator is limiting the proportion, among all flows classified as class i, of flows that are actually not from class i. The pairwise false discovery rate for a pair of classes (i, j) is defined as $\mathrm{P}(Z = j | \hat{Z} = i)$, that is the probability that, among the flows classified as class i, a flow is actually of class j. In Problem 2, the operator may specify a set of single class FDR thresholds (β_i) for each class $i \in \{1, \ldots, c\}$ or a set of pairwise FDR thresholds $(\beta_{i,j})$ for two classes $i \neq j$.

Again, we are seeking to design the classifier f^* that minimizes the total misclassification rate while meeting false discovery rate constraints set by the operator:

$$f^* = \arg\min_f \mathrm{P}(\hat{Z} \neq Z) \text{ where } \hat{Z} = f(X) \tag{4}$$

such that, for single class constraints:

$$\forall i \in \{1, \ldots, c\} : \mathrm{P}(Z \neq i | \hat{Z} = i) \leq \beta_i, \tag{5}$$

or, for pairwise constraints:

$$\forall i, j \in \{1, \ldots, c\}, j \neq i : \mathrm{P}(Z = j | \hat{Z} = i) \leq \beta_{i,j}. \tag{6}$$

C. Problem 3: large flow detection

Problem 3 involves the design of a predictor, which based on a set of features derived from the first few packets of a flow, specifies whether the flow will contain (in its entirety) a "small" or "large" number of bytes. Denote by L the maximum number of bytes for which a flow is considered "small" and by U the minimum number of bytes for which a flow is considered "large". Our goal is to determine, after observing only a small number of packets, whether it is worthwhile employing a more computationally expensive procedure to classify the flow. We strongly desire to classify every flow that has a large expected size. At the same time, we want to ensure that of the flows we do pass on to the classifier, only a tiny fraction are small (because processing these wastes time).

We formulate the predictor design as a constrained level set estimation problem. We denote by Q' the marginalization of Q on the feature space $\mathbb{R}^d$ and by Y the total number of bytes of a given flow at flow termination time. The goal is to identify the largest subset G of the feature space, that is, the subset of the feature space with maximum measure Q', which satisfies the following two constraints. The first constraint (C_0) enforces that any flow with feature vector $x \in G$ has an expected total size (at flow termination time) of at least U bytes. The second constraint, (C_1), enforces a false discovery rate constraint, and ensures that over the entire set G, the probability that a candidate flow is actually "small" (contains less than L bytes) is smaller than ω. The constants L, U, ω are set by the operator. Formally, Problem 3 is equivalent to determining the largest set

$$G^* = \arg\max_{G \subset \mathbb{R}^d} Q'(G) \tag{7}$$

such that:

$$C_0 : \quad E[Y | X = x] \geq U \quad \forall x \in G, \tag{8}$$

$$C_1 : \quad \mathrm{P}(Y < L | X \in G) < \omega. \tag{9}$$

IV. Background Material

A. Learning Satisfiability (LSAT)

The Learning Satisfiability framework, introduced in [16], is a formalization of a class of learning problems that involve multiple constraints. The goal is to learn a set in the input (feature) space that simultaneously satisifes multiple output constraints. The constraints are expressed in terms of expectations and/or event probabilities. The distinguishing features of the LSAT framework are that (i) multiple performance criteria must be satisfied; and (ii) output behaviour is assessed only on the solution set.

Formally, the one-class LSAT problem can be expressed as follows. Let $\mathcal{P}$ denote a collection of probability measures on $\mathbb{R}^d \times \mathbb{R}$ (the vector product of the d-dimensional real-valued feature space and the single dimension real-valued output space). Let $\mathcal{G}$ denote a collection of candidate sets in the input space and let $C : \mathcal{G} \times \mathcal{P} \to \mathbb{R}^{m+1}$ be a constraint function mapping each set and unknown probability measure to a $(m+1)$-dimensional vector of real numbers. For a given probability measure Q, we are interested in the largest set $G \in \mathcal{G}$ that satisfies the constraint set $C(G,Q) \geq 0$, where the inequality is applied element-by-element. Let Q' denote the marginalization of Q onto the feature space. Then the LSAT optimization problem is:

$$\max_{G\in\mathcal{G}} Q'(G) \quad \text{subject to} \quad C(G,Q) \geq 0$$

A solution may not exist, depending on the nature of the constraints and Q (in such cases, we consider the empty set to be a default solution). Since the maximization is over Q', the goal is to build a set that includes as many input elements as possible but satisfies all of the specified constraints.

The LSAT framework permits two types of set-based output constraints:

1. Point-wise Constraint: $C(G,Q) = C(x,G,Q)$ is a function of the input variable x, and the constraint takes the form $C(x,G,Q) \geq 0, \ \forall \ x \in G$.
2. Set-average Constraint: $C(G,Q)$ is only a function of the set G, and the constraint $C(G,Q) \geq 0$ is only satisfied "on-average" over the set G.

In this paper, the application of LSAT in Problem 2 involves a number of set-average constraints (false discovery rates). The application of LSAT in Problem 3 involves both a pointwise constraint and a set-average constraint. The pointwise constraint $C_0(G,Q)$ is of the form $E[Y|X=x] - U \geq 0, \quad \forall x \in G$, where Y is the number of bytes in the flow, and U is a threshold defining a "large" flow. The set-average constraint is $P(Y \geq L | X \in G) - (1-\omega) \geq 0$, where L is a threshold specifying the maximum size of a "small" flow and $1-\omega$ is a probability threshold. This latter constraint specifies that the probability, averaged over the identified set, that a given flow is actually "small", is less than ω.

B. The 2ν-Support Vector Machine

Support vector machines are a very effective classification approach [26]. These algorithms consist of two steps: (i) transforming the input features x_i via a mapping $\Phi : \mathbb{R}^d \to \mathcal{H}$ where $\mathcal{H}$ is a high-dimensional Hilbert space; and (ii) constructing a hyperplane (the decision boundary) in $\mathcal{H}$ according to the max-margin principle. The intuitive idea is that the vectors become easier to separate in the high-dimensional space. The mapping to the high dimensional space is achieved via a kernel function $k(x,x') = \langle x, x' \rangle_{\mathcal{H}} \triangleq \langle \Phi(x), \Phi(x') \rangle$.

If the feature vectors are completely separable, then the max-margin principle will lead to the hyperplane being placed such that the distance to the nearest point is maximized. If the two classes cannot be separated by a hyperplane, then the constraint is relaxed by introducing slack variables ξ_i. When x_i is on the wrong side of the boundary, $\xi_i > 0$ and x_i is a *margin error*. The hyperplane is specified in terms of the normal vector $w \in \mathcal{H}$ and the affine shift $b \in \mathbb{R}$; the support vector classifier is then

$$f_{w,b}(x) = \operatorname{sgn}(\langle w, x \rangle_{\mathcal{H}} + b). \tag{10}$$

Note that $w = \sum_{i=1}^n \alpha_i Z_i \Phi(x_i)$, and the *support vectors* are those x_i with $\alpha_i \neq 0$.

An extension to the SVM framework that supports cost-sensitive classification is the 2ν-support vector machine of [17]. The 2ν-SVM solves the optimization problem in (11). Here $\nu_+ \in [0,1]$ and $\nu_- \in [0,1]$ provide a method for globally adjusting the weight associated with incorrect assignment to the class I^+ or I^-, and n_+ and n_- are the number of points belonging to the $+$-class and $-$-class, respectively.

$$\min_{w,b,\xi,\rho} \frac{(\nu_+ + \nu_-)||w||^2}{2} - 2\nu_+\nu_-\rho + \frac{\nu_-}{n_+}\sum_{i\in I^+}\xi_i + \frac{\nu_+}{n_-}\sum_{i\in I^-}\xi_i \tag{11}$$

$$\begin{array}{rrcll} \text{s.t.} & Z_i(k(w,x_i)+b) & \geq & \rho - \xi_i & \text{for } i=1,\ldots,n \\ & \xi_i & \geq & 0 & \text{for } i=1,\ldots,n \\ & \rho & \geq & 0. & \end{array}$$

The 2ν-SVM provides a method for controlling the false-alarm rate of the classifier by appropriately adjusting the values of ν_+ and ν_-. The algorithm, described in [15], for Neyman-Pearson classification, involves conducting a grid search over the SVM parameters ν_+ and ν_-, and solving the 2ν-SVM optimization problem for each setting, with the goal of determining the classifier that achieves the lowest misclassification rate P_M while meeting a specified constraint on the false-alarm rate P_F. Since these rates cannot be evaluated directly, they must be estimated; we use k-fold cross validation on the training data to do so.

Learning-satisfiability problems can also be addressed using an extended version of the 2ν-SVM. The extension is expressed in (12) and involves the introduction of weights γ_i that are specific to feature vectors. We generate an artifical label $D_{i,j}$ and a cost $\gamma_{i,j}$ for each data point x_i and constraint j. In order to apply cost-sensitive classification, we must collapse the $D_{i,j}$ to a single label D_i. To each constraint we assign a weight λ_j that provides a mechanism for adjusting the relative importance of the constraints. If $D_{i,j} = 1$ for all j, then we set $D_i = 1$ and $\gamma_i = \sum_{j=0}^k \lambda_j \gamma_{i,j}$. A similar procedure applies if $D_{i,j} = 0$ for all j. The situation is more complicated if $D_{i,j}$ differs for various constraints. In this case, we set $D_i = 1$ and $\gamma_i = \sum_{D_{i,j}=1} \lambda_j \gamma_{i,j}$. However, we also construct an auxiliary data point $(X_{\tilde{i}}, D_{\tilde{i}}, \gamma_{\tilde{i}})$, with $D_{\tilde{i}} = 0$ and $\gamma_{\tilde{i}} = \sum_{D_{i,j}=0} \lambda_j \gamma_{i,j}$.

$$\min_{w,b,\xi,\rho} \frac{(\nu_+ + \nu_-)||w||^2}{2} - 2\nu_+\nu_-\rho + \frac{\nu_-}{n_+}\sum_{i\in I^+}\xi_i\gamma_i + \frac{\nu_+}{n_-}\sum_{i\in I^-}\xi_i\gamma_i \tag{12}$$

$$\begin{array}{rrcll} \text{s.t.} & D_i(k(w,x_i)+b) & \geq & \rho - \xi_i & \text{for } i=1,\ldots,n \\ & \xi_i & \geq & 0 & \text{for } i=1,\ldots,n \\ & \rho & \geq & 0. & \end{array}$$

To solve the learning satisfiability problem using this extended 2ν-SVM, we must search over the possible values of λ_j, ν_+ and ν_- to find the largest set that satisfies all of the constraints [16]. In each case, we test whether the set satisfies constraints by performing cross-validation on the input data. This algorithm is limited to constraints where one can identify an appropriate mapping to labels and costs. This can, however, be achieved for a wide range of important constraints, including those involving bounds on pointwise or set-average expectation or tail probabilities.

V. Traffic classification algorithms

We now present the SVM-based algorithms used to solve the problems formalized in Section III. We have adopted SVM approaches because there are existing frameworks for controlling FARs and FDRs using cost-sensitive SVMs (our novel methodological contribution in this paper is to extend them from binary to multi-class problems). We do not anticipate that the SVM will intrinsically outperform alternative classification approaches, but it has been shown to achieve similar performance [23].

A. Application classification with FAR constraints

Standard multi-class SVM techniques do not scale well with the number of classes for training and prediction, and implementations involving several cascaded binary SVMs generally achieve comparable performance [26]. Our approach is to design a chain of c binary classifiers $f = (f_{s(1)}, ..., f_{s(c)})$. Each binary classifier $f_{s(i)}$ decides whether a flow belongs to class $s(i)$ ($f_{s(i)}(x) = 1$) or not ($f_{s(i)}(x) = 0$), where $s(\cdot)$ is a permutation of the original c classes (excluding the default "catch-all" class $c+1$). Let $\hat{z} = \arg \min_{s(1) \leq i \leq s(c)} \{s(i) : f_{s(i)}(x) = 1\}$ be equal to the first class for which one of the binary classifiers $f_{s(i)}$ detects a match for a flow with training vector x. If none of the classifiers $f_{s(i)}$ has detected a match, then set $\hat{z} = c+1$.

We now propose a heuristic training methodology. Although it is unlikely to identify the optimal classifier (even within the restriction to the class of SVM classifiers), it does perform a reasonably comprehensive search of the space of possible binary classifier chains. We describe the training methodology for single-class FAR constraints (the extension to pairwise FAR constraints is straightforward).

Each classifier $f_{s(i)}$ is implemented as a binary 2ν-SVM, taking as parameters ν^+, ν^-, and the kernel parameter σ. We do not address the problem of kernel selection in this work, and simply adopt a Gaussian kernel $K(x, x') = \exp\left(-\sigma \|x' - x\|^2\right)$. We discretize the SVM parameter space, and for each tuple (ν^+, ν^-, σ), for each class $s(i)$, we determine an SVM using all training features and the binary training labels $Z'_i = \delta_{Z=s(i)}$. We now have a set of candidate binary SVMs for each class. Our goal is to choose the ordering s and the best parameter tuple for each classifier $s(i)$, in order to identify the chain of classifiers that minimizes the misclassification rate $\mathrm{P}(\hat{Z} \neq Z)$ subject to meeting FAR constraints.

In order to compare candidate classifier chains, we adopt a cost function to assess performance:

$$R(f) = \sum_{s(i)} \frac{1}{\alpha_{s(i)}} \max(P_F(s(i)) - \alpha_{s(i)}, 0) + P_M(s(i)) \quad (13)$$

Here $P_F(s(i))$ is the FAR for class $s(i)$, and $P_M(s(i))$ is the misclassification rate. Since we cannot directly evaluate these quantities, we must estimate them based on the training data, and the empirical cost function becomes:

$$\begin{aligned} \widehat{R}(f) &= \sum_{s(i)} \widehat{R}(f_{s(i)}) \qquad (14) \\ &= \sum_{s(i)} \frac{1}{\alpha_{s(i)}} \max(\widehat{P}_F(s(i)) - \alpha_{s(i)}, 0) + \widehat{P}_M(s(i)) \end{aligned}$$

This cost function, first proposed for binary classification in [15], has a number of desirable properties: it is minimized by the classifier f^* described in Section III-A in the binary case; it can be estimated using the training data (we use a five-fold cross-validation procedure to estimate the FARs and misclassification rates); and as α_i tends towards 0, classifiers violating the FAR constraints are more heavily penalized. By adopting this cost function, we place equal emphasis on the misclassification rate for each class. This may not be desirable if it is more important to achieve a low misclassification rate for one specific class; a possible extension of the approach we propose is to associate different weights with the class-specific misclassification rates. Observe that the cost function is dependent on the ordering s, which essentially determines the tie-breaking procedure for the binary SVMs. The FARs and misclassification rate for class $s(i)$ are determined not only by the individual SVM for that class, but also by the preceding SVMs in the chain.

Suppose we have chosen an ordering s. As expressed in (14), we can decompose the global cost function into a sum of class-specific costs. We choose the SVM parameters for this ordering using a branch-and-bound search over the possible parameter tuples. The search procedure operates as follows. First, we evaluate the partial cost $\widehat{R}(f_{s(1)})$ for each possible set of parameter tuples for class $s(1)$. We choose the parameter tuple that achieves minimum class-specific cost, and then evaluate $\widehat{R}(f_{s(2)})$ for all parameter tuples for class $s(2)$. Note that these costs are dependent on the parameters we chose for the SVM employed for class $s(1)$. We choose the classifier $f_{s(2)}$ which minimizes the partial cost $\widehat{R}(f_{s(1)}) + \widehat{R}(f_{s(2)})$, and then repeat the process for classes $s(3), \ldots, s(c)$. At this point, we can evaluate the total cost for the chosen set of parameter tuples, and this forms a bound on the minimum cost. The search procedure now explores other branches (for example, the second-best parameter tuple for class $s(1)$), eliminating any branch as soon as its partial cost exceeds the established bound. This search is reasonably efficient, because evaluation of the partial costs is a very simple calculation, and the nature of the cost function means that many candidate classifier chains do not need to be evaluated because their partial costs grow very rapidly compared to the bound on the minimum cost. For example, any chain that violates one or more FAR constraints automatically has a much higher cost than a chain that can meet all constraints.

In order to choose the best ordering, we can either repeat the process described above for every possible ordering, which is computationally feasible for a small number of classes (less than 6). When the number of classes is larger, we adopt two heuristics to reduce the number of orderings we evaluate: (i) initially place classes with stringent FAR constraints near the start of the chain; (ii) after evaluation of an ordering, promote any class that has a high partial cost, either because FAR constraints are violated or the class-specific misclassification rate is high. The reason for the promotion is that classifiers near the start of the chain have greater flexibility. If the class membership probabilities are significantly different, it is also important to place classes with small representation early in the chain.

At the end of the training stage, we have identified an ordering s and the SVM parameters ($\nu^+_{s(i)}$, $\nu^-_{s(i)}$, $\sigma_{s(i)}$) for each classifier $f_{s(i)}$, which is equivalent to knowing the weights $w_{s(i)}$ that define each SVM $f_{s(i)}$. The training procedure involves significant computation, but can be executed in the order of minutes, which is reasonable considering that we expect the distribution of flows and associated applications to remain stable for hours or days (although it may be necessary to train different classifiers to address diurnal variations).

B. Application classification with LSAT

We now turn to Problem 2, where the network operator wants to set target FDRs rather than FARs as constraints. We again use a chain of binary classifiers with class order specified by the permutation s, but, in this case, each building block is an LSAT binary classifier. Training is the same as the FAR case, with the following exceptions. First, the grid search on the SVM parameter space now includes the supplemental parameter $\lambda_{s(i)}$, which corresponds to the weight associated with the FDR constraint for class $s(i)$. Second, we use a different cost function, as suggested in [16], to jointly achieve FDR constraints and minimize the misclassification rate.

Problem 2 maps to the LSAT framework described in Section IV as follows. Consider the binary problem of detecting whether a flow with feature vector X is of class $s(i)$ with class FDR constraints $\beta_{s(i)}$. Call $G_{s(i)} \subset \mathbb{R}^d$ the set of feature vectors for which class $s(i)$ is detected, that is $f_{s(i)}(x) = 1$ if and only if $x \in G_{s(i)}$. Seeking the classifier $f^*_{s(i)}$ which minimizes the misclassification rate is thus equivalent to seeking the largest subset $G^*_{s(i)}$ of the feature space:

$$G^*_{s(i)} = \arg \max_{G_{s(i)} \subset \mathbb{R}^d} Q'(G_{s(i)}) \tag{15}$$

such that the FDR constraint for class $s(i)$ is met:

$$C_0 : \qquad P(Z = 0|\hat{Z} = 1) < \beta_{s(i)}. \tag{16}$$

C_0 is equivalent to $P(Z = 1|X \in G) - (1 - \beta_{s(i)}) > 0$, which is in the form of a set-average constraint as described in Section IV-A.

This is implemented using the same greedy algorithm as in Section V-A, where each block is now a 2ν-SVM adapted to the LSAT problem as described in IV-B. As suggested in [16], we use the misclassification rate $P_M(s(i))$ as the cost function to order candidate classifiers for class $s(i)$; to enforce the FDR constraints, we discard any set of parameters ($\nu^+_{s(i)}$, $\nu^-_{s(i)}$, $\sigma_{s(i)}$, $\lambda_{s(i)}$) that result in a SVM that does not meet the FDR constraint on the training data. We search for the best class ordering s using the same strategy as in the FAR-constrained case, i.e., an exhaustive search when the number of classes is small and the application of strategic search heuristics when the number of classes is large.

C. Early detection of large flows

Problem 3, the early detection of large flows, differs from the first two problems, because the output associated with each feature vector is no longer a class label, but is an integer Y representing the total number of bytes in the flow. The network operator provides two thresholds L and U, which respectively define the upper-limit for a "small" flow, and the lower-limit for a "large" flow. Our goal is to identify the largest region G in the feature space such that the expected value of Y at every point in the region is greater than U and the probability $P(Y < L|X \in G) < \omega$.

As discussed in Section IV-A, this can be formulated as a binary LSAT problem with one pointwise constraint and one set-average constraint. We previously described a 2ν-SVM algorithm for solving an equivalent LSAT problem in [16], and we employ the same approach here.

VI. Data and processing

We obtained a traffic trace from OmniGlobe Networks, a Canadian ISP that specializes in providing satellite-link Internet service, corresponding to 24 consecutive hours of traffic on April 15 and 16, 2008. In this work, we only consider the TCP flows in the trace. We define a flow as a 4-tuple (source IP address, destination IP address, source TCP port, destination TCP port). For each TCP flow in the trace, we extracted a large number of features and derived a notion of "ground truth" for the class and number of bytes (Z for Problems 1 and 2 and Y for Problem 3). That is, the application corresponding to each flow and the total number of bytes in the flow. The features and ground truth were used to train our classifiers, and to assess their performance in a validation step.

We used tcpdump to capture up to the first 100 bytes for each packet (including the headers – from MAC layer to the TCP layer). We sliced our full trace into 24 single-hour traces, and for each hour of traffic, we identified the flows. We ignored flows which did not start or did not end within the considered hour. We verified that these flows crossing hour boundaries represented a negligible amount of traffic. Each flow was processed individually to extract the features and ground truth of the class and number of bytes.

To extract features from each flow, we use the tcptrace tool [27]. This tool can provide 142 different statistics for a

TABLE I
APPLICATION BREAKDOWN FOR FLOWS > 6 PACKETS

	Flows		Size	
Application	**Number**	**Percentage**	**GB**	**Percentage**
HTTP	315375	78.3%	4.1	74.6%
HTTPS	20736	5.2%	0.29	5.4%
MSN	3364	0.8%	0.04	0.7%
POP3	1311	0.3%	0.01	0.2%
OTHER	61870	15.4%	1.05	19.1%

flow, including the IP addresses and TCP ports for source and destination, given only the captured headers. The tool is also able to give the same set of statistics for truncated flows, that is, flows for which we only process the first k packets. This is important in our context, because we are striving to classify traffic in real-time. We exclude the 4 statistics that define a flow from the default 142 statistics returned by tcptrace and we use a subset of the remaining 138 statistics to train and evaluate our classifiers.

To associate an application to each flow, we use Bro, an intrusion detection tool capable of deep packet inspection [24]. We show the complete application breakdown in terms of flows and bytes in Table I for flows of more than 6 packets; in Section VII we evaluate our classifiers after the 6th packet for each flow. For this network, HTTP traffic is dominating the traffic mix. We also notice that the users of this network do not engage in P2P file exchange (P2P traffic is blocked by the ISP). For the "Other" class, over 80% of these flows are smaller than 10 packets so they disappear from the network rather quickly. Typical applications that we found in the "Other" class included port scans and broken TCP connection flows.

VII. EXPERIMENTAL RESULTS

The main goal of our experiments is to validate that we are able to set performance guarantees on classes of interest while still achieving a high accuracy. We first reduce the input feature space so only relevant features are processed by our classifiers. In [2], Williams et al. compare different feature selection methods. In our work we employ the so-called "correlation-based with best first search (forward)" method, which performs well according to [2]. The resulting features for our data set are: (1) total number of bytes sent from the client to the server; (2) number of packets with the FIN field set between the client and the server; (3) the window scaling factor used between the client to the server; (4) total number of bytes truncated in the packet capture between the client and server; and (5) total number of packets truncated in the packet capture between the server and the client. In our experiments, the training set is comprised of 1000 flows randomly selected from the first hour of traffic. We perform classification after six packets of a flow are detected (parameter k). We focus on classifying each flow as one of the four dominant applications in our data set (HTTP, HTTPS, MSN messenger and POP3), or identifying it as belonging to the catch-all default class "other". To emulate a real-time environment, we do not process flows smaller than the milestone of k packets.

A. Classification with FAR constraints

Our first experiment explores the performance of our algorithm for traffic classification under FAR constraints. In this experiment, the flows comprising the training set were selected randomly from the first hour (1000 in total). The remaining 23 hours were used as test data. We compare the performance of three classifiers: (i) a baseline multiclass SVM classifier, as described in [4]; (ii) our proposed FAR-constrained classifier with a single-class FAR constraint on the HTTP class ($\alpha\{HTTP\} = 0.4\%$) and (iii) our proposed FAR-constrained classifier with a pairwise FAR constraint on the HTTP and HTTPS classes ($\alpha\{HTTPS, HTTP\} = 0.05\%$). Since there were only 4 classes, our algorithms searched over all possible orderings (24) of the classifiers in the binary chain.

We initially evaluate performance based on the remaining flows from the first hour, for which the classifier is best matched. The baseline classifier achieves the highest overall accuracy (lowest misclassification rate) at 98.5%, but the accuracies of the FAR-constrained classifiers are only marginally lower at 97.7% and 97.6%, respectively. In terms of single-class FAR for the HTTP class, the baseline classifier has an FAR of 3.7%, whereas the single-class FAR-constrained classifier reduces this below the threshold to 0.3%. The pairwise FAR constrained classifier achieves a pairwise FAR of 0.02%, which is at the threshold of 0.02%; the baseline classifier is at 0.07%.

Figure 1 displays the results of the experiment for hours 2-24. As indicated in the top panel, the accuracy of the classifiers varies over the different periods, ranging from 89.3% to 99.1%, but no classifier achieves a consistently superior accuracy. The middle panel shows that our proposed single-class FAR constrained classifier consistently achieves a FAR below 1.0%, whereas the FAR for the baseline multi-class SVM is much larger and reaches values as large as 30%. Our proposed classifier does not always achieve a FAR below the specified constraint of 0.4%, which may be partially due to discrepancies between the first hour of traffic and the remaining 23 (the training data may not sufficiently represent the later flows). The bottom panel examines the pairwise HTTPS-HTTP FAR, and indicates that the pairwise constrained classifier achieves a pairwise FAR consistently lower than that of the baseline multi-class SVM classifier; again the pairwise FAR constraint is not always met.

B. Classification with FDR constraints

Our second experiment examines the performance of our proposed algorithms for traffic classification under FDR constraints. We compare the performance of three classifiers: (i) the baseline multiclass SVM classifier; (ii) an unconstrained binary-chain classifier (in which we set $\beta = 100\%$ for all classes) and (iii) our proposed FDR-constrained classifier with a single class FDR constraint on the HTTPS class ($\beta\{HTTPS\} = 5\%$). Training is again performed using 1000 flows randomly selected from the first hour. We conduct performance evaluation using the remaining flows in the first

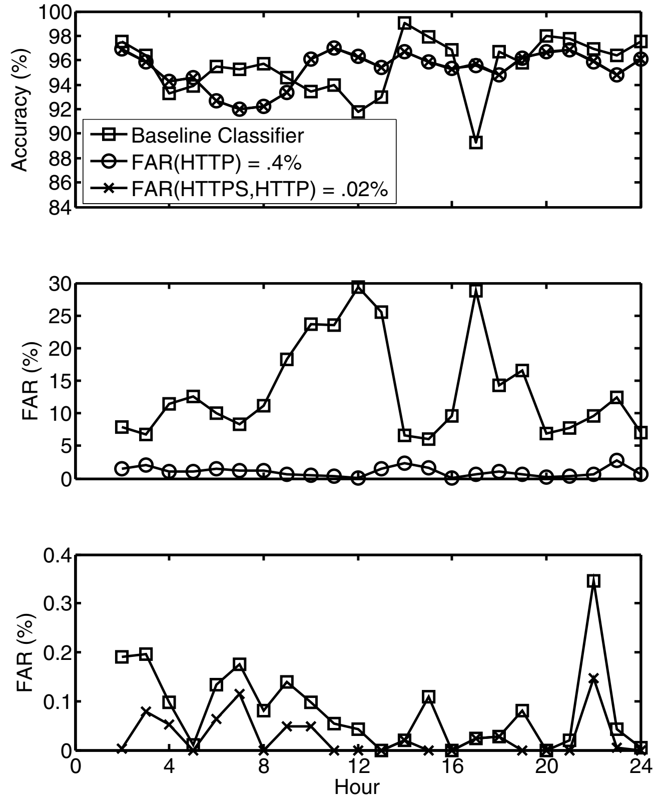

Fig. 1. Top Plot: The overall accuracy for hours 2-24 for the three classifiers discussed. Middle Plot: The False Alarm Rate (FAR) of HTTP for the baseline classifier and for the NP classifier when the FAR for HTTP is set to .4%. Bottom Plot: The pairwise FAR for HTTP flows being misclassified as HTTPS for the baseline classifier and the NP classifier where the FAR{HTTPS,HTTP} is set to .02%.

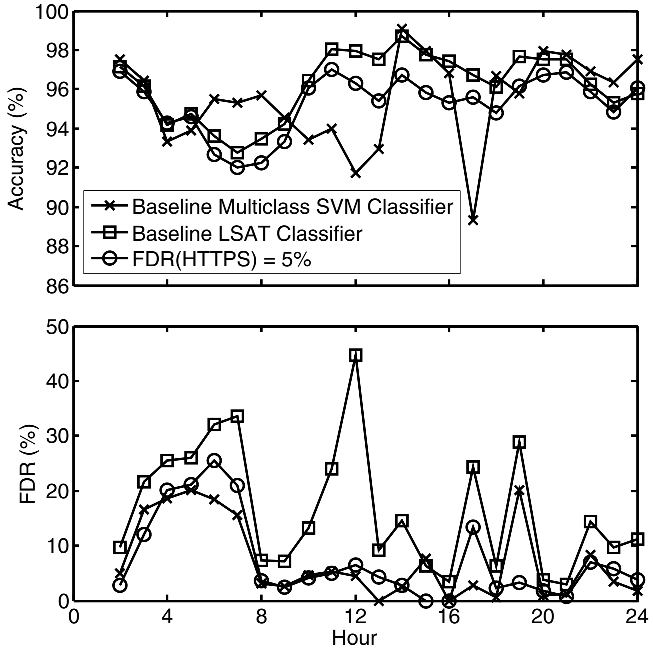

Fig. 2. Top Plot: The overall accuracy for hours 2-24 for the three classifiers discussed. Bottom Plot: The False Discovery Rate (FDR) of HTTPS for the baseline multiclass SVM classifier, unconstrained LSAT binary-chain classifier and for the FDR-constrained classifier where the FDR for HTTPS is set to 5%

hour, and then examine the stability of the performance over the remaining 23 hours.

For the first hour, the baseline multiclass SVM classifier achieves the highest overall accuracy at 98.5%, but the accuracies of the other two classifiers are only marginally lower, with the unconstrained binary-chain classifier achieving 98.0% and the FDR-constrained classifier achieving 97.9%. The FDR for the remaining flows of the first hour for the unconstrained binary chain classifier is 7.0% while with our proposed classifier we were able to reduce the FDR to 4.2%.

Figure 2 shows the results of the experiment for hours 2-24. In the top panel, the overall accuracies for all three classifiers are comparable, although the accuracy of the multiclass SVM drops significantly for hours 10-13 and hour 17. In the bottom panel, the FDR achieved by our proposed constrained classifier is comparable to the baseline multiclass SVM classifier and significantly lower than that of the unconstrained binary chain classifier (which exceeds 20% for several hours and reaches 44.6% for hour 12). The FDR-constrained classifier meets the specified constrained (or marginally exceeds it) for hours 1-2 and 8-24, but the FDR for hours 3-7 all exceed 10%. This indicates that there may be a need to train several classifiers for different periods of the day. The results do indicate that a binary-chain classifier can achieve performance comparable to that of a multiclass SVM, with the advantage that it is significantly more scalable as the number of classes increases.

C. Online classification complexity

One other area that needs to be addressed is whether our classifiers can predict incoming flows fast enough to function in an online environment. Running a single SVM for a sample flow requires extracting the features x, and computing the quantity $f_{w,b}(x)$ as in (10). Note that the dot-product $\langle w, x\rangle_{\mathcal{H}}$ is actually computed through a kernel evaluation, i.e. $\langle w, x\rangle_{\mathcal{H}} = k(w, x)$. Evaluating (10) thus requires summation of the results of N_v (where N_v is the number of support vectors of the considered SVM) kernel evaluations. The full classifier may require to go through up to c SVMs, leading to a computational cost of $N_v c$ kernel evaluations. In our simulation work, N_v was typically in the order of a few hundreds and $c = 4$, leading to the computation of a few thousands of kernel functions — chosen here to be the exponential of the norm of a d-dimensional vector ($d = 5$). We observe that adding classes to the classifier only increases the computational complexity linearly. In our experiments, classifying a flow took around 1 ms on a modern general purpose computer. This indicates that the flow classification problem is well-suited to implementation in high-speed routers with dedicated hardware.

VIII. Conclusions

Although existing Internet traffic classifiers are able to achieve good overall accuracy, none can provide class-specific performance guarantees, particularly on the critical metrics of False Alarm Rate (FAR) and False Discovery Rate (FDR). In this paper, we have proposed two novel on-line classification algorithms that are designed to provide such guarantees. The algorithms are multi-class extensions of the Neyman-Pearson classification and Learning Satisfiability frameworks. We have validated the performance of the proposed methods using traffic traces obtained from Omniglobe Networks, a Canadian ISP that specializes in satellite-based Internet connectivity for remote areas. The experiments indicate that the techniques significantly reduce the FARs and FDRs compared to a state-of-the-art baseline approach, and almost achieve the specified constraints, at the expense of a very slight reduction in overall classification accuracy.

Although our experiments provide some validation of our proposed methodologies, a data set with a more diverse set of applications is required to provide a more comprehensive assessment. In particular, a dataset with significant volumes of time-sensitive traffic (e.g., VoIP and streaming video) would provide the opportunity to examine how well the classifier performs as the first stage of a Quality-of-Service engine that assigns different priorities to flows according to the identified class labels.

There are a number of methodological extensions that are of interest. Our proposed algorithms strive to minimize the global misclassification rate, but in many cases it might be preferable to prioritize the misclassification rate of a few critical classes. There are also scenarios where it is desirable to impose a mixture of FAR and FDR constraints, and it should be possible to combine the two algorithms we have proposed into a single entity.

Acknowledgments

The authors would like to thank OmniGlobe Networks for providing us with Internet traffic traces for our research.

References

[1] 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 *Proc. ACM SIGCOMM Internet Measurement Conf.*, Sicily, Italy, Oct. 2004.

[2] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification,” *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, 2006.

[3] A. W. Moore and D. Zuev, “Internet traffic classification using Bayesian analysis techniques,” in *Proc. ACM SIGMETRICS*, Banff, AB, Canada, June 2005.

[4] Y. Yang, Y. Liu, S. Li, and X. Zhou, “Solving P2P traffic identification problems using Support Vector Machines,” in *IEEE/ACS Int. Conf. Comp. Syst. and Applications*, Amman, Jordan, May 2007.

[5] F. J. González-Castaño, P. S. Rodríguez-Hernández, R. P. Martínez-Álvarez, and A. Gómez-Tato, “Support vector machine detection of peer-to-peer traffic in high-performance routers with packet sampling,” in *Int. Conf. Adaptive and Natural Computing Algorithms*, Warsaw, Poland, Apr. 2007.

[6] Z. Li, R. Yuan, and X. Guan, “Accurate classification of the Internet traffic based on the SVM method,” in *Proc. IEEE Int. Conf. Comm.*, Glasgow, Scotland, June 2007.

[7] Y. Liu, H. Liu, H. Zhang, and X. Luan, “The Internet traffic classification an online SVM approach,” in *Proc. IEEE Int. Conf. Information Networking*, Busan, South Korea, Jan. 2008.

[8] L. Bernaille, R. Teixeira, and K. Salamatian, “Early application identification,” in *Proc. CoNEXT Int. Conf. Emerging Networking Experiments and Technologies*, Lisboa, Portugal, Dec. 2006.

[9] L. Bernaille and R. Teixeira, “Early recognition of encrypted applications,” in *Proc. Passive and Active Measurement Conf.*, Louvain-la-neuve, Belgium, Apr. 2007.

[10] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/Realtime network traffic classification using semi-supervised learning,” *Performance Evaluation*, vol. 64, pp. 1194–1213, 2007.

[11] C. D. Scott and R. D. Nowak, “A Neyman-Pearson approach to statistical learning,” *IEEE Trans. Inform. Theory*, vol. 51, no. 11, pp. 3806–3819, 2005.

[12] D. Casasent and X. W. Chen, “Radial basis function neural networks for nonlinear Fisher discrimination and Neyman-Pearson classification,” *Neural Networks*, vol. 16, no. 5, pp. 529–535, 2003.

[13] A. Cannon, J. Howse, D. Hush, and C. Scovel, “Learning with the Neyman-Pearson and min-max criteria,” Los Alamos National Laboratory, Tech. Rep., June 2002, lA-UR 02-2951.

[14] T. Landgrebe and R. Duin, “On Neyman-Pearson optimisation for multiclass classifiers,” in *Proc. Pattern Recognition Assoc. of South Africa*, Langebaan, South Africa, Nov. 2005.

[15] M. A. Davenport, R. G. Baraniuk, and C. D. Scott, “Controlling false alarms with support vector machines,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Proc. (ICASSP)*, Toulouse, France, May 2006.

[16] F. Thouin, M. J. Coates, B. Erikkson, R. Nowak, and C. Scott, “Learning to Satisfy,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Proc. (ICASSP)*, Las Vegas, NV, USA, Apr. 2008.

[17] H. Chew, R. Bogner, and C. Lim, “Dual-ν support vector machine with error rate and training size biasing,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Proc. (ICASSP)*, Salt Lake City, UT, Jun. 2001.

[18] “QOSMOS - Deep Packet Inspection - Information Extraction.” [Online]. Available: http://www.qosmos.com

[19] D. Antoniades, M. Polychronakis, S. Antonatos, E. P. Markatos, S. Ubik, and A. Oslebo, “Appmon: An application for accurate per application traffic characterization,” in *Proc. of IST Broadband Europe*, Geneva, Switzerland, Dec. 2006.

[20] H. Jiang, A. W. Moore, Z. Ge, S. Jin, and J. Wang, “Lightweight application classification for network management,” in *Proc. SIGCOMM Workshop on Internet Network Management: The Five-Nines Workshop*, Kyoto, Japan, Aug. 2007.

[21] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” *ACM SIGCOMM Computer Communications Review*, vol. 36, no. 2, pp. 23–26, 2006.

[22] J. Erman, A. Mahanti, and M. Arlitt, “Internet traffic identification using machine learning,” in *Proc. IEEE GLOBECOM*, San Francisco, CA, USA, Nov. 2006.

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

[24] V. Paxson, “Bro: a system for detecting network intruders in real-time,” *Comput. Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[25] T. Karagiannis, D. Papagiannaki, and M. Faloutsos, “BLINC: Multilevel traffic classification in the dark,” in *Proc. ACM SIGCOMM*, Philadelphia, PA, USA, Aug. 2005.

[26] B. Schölkopf and A. J. Smola, *Learning with kernels*. Cambridge, MA, USA: MIT Press, 2002.

[27] “tcptrace - Official Homepage.” [Online]. Available: http://www.tcptrace.org/