Semi-Supervised Network Traffic Classification

Jeffrey Erman¶, Anirban Mahanti§, Martin Arlitt‡¶, Ira Cohen‡, Carey Williamson¶
¶Department of Computer Science, University of Calgary
§Department of Computer Science and Engineering, Indian Institute of Technology (Delhi)
‡Enterprise Systems & Software Labs, HP Labs

Categories and Subject Descriptors: C.4 [Computer Systems Organization]Performance of Systems

General Terms: Algorithm, Measurement, Performance

Keywords: Traffic classification, Semi-supervised learning

1. INTRODUCTION

Identifying and categorizing network traffic by application type is challenging because of the continued evolution of applications, especially of those with a desire to be undetectable. The diminished effectiveness of port-based identification and the overheads of deep packet inspection approaches motivate us to classify traffic by exploiting distinctive flow characteristics of applications when they communicate on a network.

This paper proposes a traffic classification methodology that relies on using only flow statistics to classify traffic. We introduce a flexible mathematical framework that leverages both labeled and unlabeled flows. This *semi-supervised* [1] approach to learning a network traffic classifier is one key contribution of this work.

There are three main advantages to our proposed semi-supervised approach. First, fast and accurate classifiers can be obtained by training with a small number of labeled flows mixed with a large number of unlabeled flows. Second, our approach is robust and can handle both previously unseen applications and changed behavior of existing applications. Furthermore, our approach allows iterative development of the classifier by allowing network operators the flexibility of adding unlabeled flows to enhance the classifier's performance. Third, our approach can be integrated with solutions that collect flow statistics. For example, our framework can leverage recent work on flow estimation and can classify traffic at both the edge and the core of the network [5].

As a proof of concept, we implemented prototype offline and realtime classification systems. A distinguishing aspect of our work is the implementation of a realtime classifier in the Bro [6] Intrusion Detection System (IDS).

We also consider the longevity of classifiers [4]. Our experiments with long-term Internet packet traces suggests that classifiers are generally applicable over reasonably long periods of time (e.g., on the order of weeks) with retraining necessary when there are significant changes in the network usage patterns including introduction of new applications.

SIGMETRICS'07, June 12–16, 2007, San Diego, California, USA.
ACM 978-1-59593-639-4/07/0006.

2. CLASSIFICATION FRAMEWORK

Let $\mathbf{X} = \{X_1, \cdots, X_N\}$ be a set of flows. A flow instance X_i is characterized by a vector of attribute values, $\mathbf{X_i} = \{X_{ij} | 1 \leq j \leq m\}$, where m is the number of attributes, and X_{ij} is the value of the j^{th} attribute of the i^{th} flow. Also, let $\mathbf{Y} = \{Y_1, \cdots, Y_q\}$ be the set of traffic classes, where q is the number of classes of interest. The Y_i's can be classes such as "HTTP", "Streaming", and "Peer-to-Peer". Our goal is to learn a mapping from a m-dimensional variable X to Y. This mapping forms the basis for classification models.

In designing our classification method, there are two main challenges for classifying network flows. First, labeled examples are scarce and difficult to obtain. With few labeled examples, traditional supervised learning methods often produce classifiers that do not generalize well to previously unseen flows. Second, not all types of applications generating flows are known *a priori*, and new ones may appear over time. Traditional supervised methods force a mapping of each flow into one of q known classes, without the ability to detect new types of flows.

To address these challenges, we designed a method that combines unsupervised and supervised methods. Our classification method consists of two steps. We first employ a *clustering* [2] algorithm to partition a training data set that consists of scarce labeled flows combined with abundant unlabeled flows. We use the K-Means clustering algorithm in our results. Second, we use the available labeled flows to obtain a mapping from the clusters to the different known q classes (Y). This step also allows some clusters to remain unmapped, accounting for possible flows that have no known labels. The result of the learning is a set of partitions, some mapped to the different flow types. Our method is characterized in the literature as a *semi-supervised* learning method [1].

2.1 Mapping Clusters to Applications

The output of the K-Means clustering algorithm is a set of clusters, represented by their centroids, γ_k. Given a flow feature vector $\mathbf{x}$, we assign it to one of the clusters by finding the nearest centroid to $\mathbf{x}$, using:

$$C_k = \arg\min_k d(\mathbf{x}, \gamma_\mathbf{k}), \tag{1}$$

where $d(\cdot, \cdot)$ is the distance metric chosen in the clustering step. For K-Means with Euclidean distance, Eq. 1 amounts to the maximum likelihood cluster assignment solution.

However, knowing to which cluster a flow feature vector most likely belongs does not provide the actual classification to one of the application types. Therefore, we need a mech-

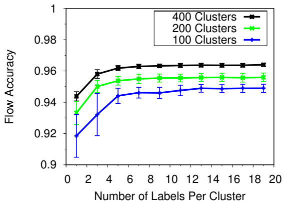


Figure 1: Selective Labeling of Flows

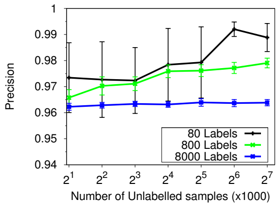


Figure 2: Training with (Un)labeled Flows

anism to map the clusters found by the clustering algorithm to the different application types.

We use a probabilistic assignment to find the mapping from clusters to labels: $P(Y = y_j|C_k)$, where $j = 1, ..., q$ (q being number of application types) and $k = 1, ..., K$ (K being the number of clusters). To estimate these probabilities, we use the set of flows in our training data that are labeled to different applications $(\mathbf{x_i}, \mathbf{y_i})$, $i = 1, ..., L$, where L is the total number of different labeled applications. $P(Y = y_j|C_k)$ is then estimated by the maximum likelihood estimate, $\frac{n_{jk}}{n_k}$, where n_{jk} is the number of flows that were assigned to cluster k with label j, and n_k is the total number of (labeled) flows that were assigned to cluster k. To complete the mapping, clusters that do not have any labeled examples assigned to them are defined as "Unknown" application types, thus allowing the representation of previously unidentified application types.

Finally, the decision function for classifying a flow feature vector $\mathbf{x}$ is the maximum *a posterior* decision function:

$$y = \arg \max_{y_1, \ldots, y_q} (P(y_j|C_k)), \quad (2)$$

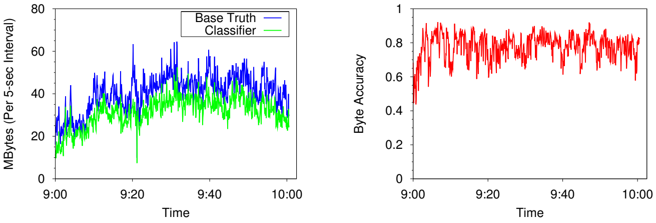
where C_k is the nearest cluster to $\mathbf{x}$, as obtained from Eq. 1.

3. SEMI-SUPERVISED RESULTS

Labeling of training feature vectors is likely to be one of the most time-consuming steps of the classification process, especially because many Internet application purposefully attempt to avoid detection. Our semi-supervised approach to training the classifier leverages the fact that clustering attempts to form disjoint groups, wherein each group consists of objects that bear a strong similarity to each other [3]. Thus, we test the hypothesis that if a few flows are labeled in each cluster, we have a reasonable basis for creating the cluster to application type mapping.

The first experiment considers the possibility of the entire training data set being unlabeled. In this case, we can selectively label a few flows from each cluster and use these labeled flows as the basis for mapping clusters to applications. Fig. 1 presents results from this experiment. We assume that we are provided with 64,000 unlabeled flows. Once, these flows are clustered we randomly label a fixed number of flows in each cluster. Interestingly, the results show that with as few as two labeled flows per cluster and $K = 400$, we can attain 94% flow accuracy. The increase in classification accuracy is marginal once five or more flows are labeled per cluster.

For the second set of experiments, results of which are shown in Fig. 2, we utilized 80, 800, and 8,000 labeled flows, and mixed these labeled flows with varying numbers of unlabeled flows to generate the training data set. Both labeled and unlabeled flows were randomly chosen from the test data



(a) Bytes Classified

(b) Byte Accuracy

Figure 3: Performance of Realtime Classifier

set [4]. These training flows were used to learn the flow to application mapping, with $K = 400$ in the clustering step, and we tested the resulting classifier on the same trace.

Fig. 2 reports the precision [4] of the classifier. We observe that for a fixed number of labeled training flows, increasing the number of unlabeled training flows increases our precision. This is an important empirical result because unlabeled flows are relatively inexpensive to obtain and the penalty for incorrect labeling of a flow might be high (e.g., assigning lower priority to business critical traffic). Thus, by simply using a large sample of unlabeled flows, the precision rate can be substantially increased.

Detailed results of our offline classifier can be found in [4].

4. REALTIME CLASSIFICATION

A fundamental challenge in the design of the realtime classification system is to classify a flow as soon as possible. Unlike offline classification where all discriminating flow statistics are available *a priori*, in the realtime context we only have partial information on the flow statistics.

We address this challenge by designing a layered classification system. Our layers are based upon the idea of *packet milestones.* A packet milestone is reached when the count of the total number of packets a flow has sent or received reaches a specific value. We include the SYN/SYNACK packets in the count. Each layer is an independent model that classifies ongoing flows into one of the many class types using the flow statistics available at the chosen milestone. Each milestone's classification model is trained using flows that have reached each specific packet milestone.

This layered approach allows us to revise and potentially improve the classification of flows. The memory overhead of our approach is linear with respect to the number of flows because we use the same feature set at all layers.

Fig. 3 presents example results. We see that the classifier performs well with byte accuracies typically in the 70 to 90% range. Further results can be found in [4].

5. REFERENCES

[1] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning.* MIT Press, Cambridge, MA, 2006.

[2] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* Wiley, second edition, 2001.

[3] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification using Clustering Algorithms. In *SIGCOMM'06 MineNet Workshop*, Pisa, Italy, September 2006.

[4] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/Online Traffic Classification Using Semi-Supervised Learning. Technical report, University of Calgary, 2007.

[5] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson. Identifying and Discriminating Between Web and Peer-to-Peer traffic in the Network Core. In *WWW'07*, Banff, Canada, May 2007.

[6] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Comput. Networks*, 31(23-24):2435–2463, 1999.