Traffic Classification On The Fly

Laurent Bernaille[†] Renata Teixeira[†] Ismael Akodjenou[†] Augustin Soule[‡] Kave Salamatian[†]
[†] Université Pierre et Marie Curie - Paris VI
[‡] Thomson Paris Lab

ABSTRACT

The early detection of applications associated with TCP flows is an essential step for network security and traffic engineering. The classic way to identify flows, i.e. looking at port numbers, is not effective anymore. On the other hand, state-of-the-art techniques cannot determine the application before the end of the TCP flow. In this editorial, we propose a technique that relies on the observation of the first five packets of a TCP connection to identify the application. This result opens a range of new possibilities for online traffic classification.

Categories and Subject Descriptors

C.2.3 [**Network Monitoring**]: Network Management; I.2.6 [**Learning**]: Unsupervised Learning

General Terms

Measurement, Algorithms, Management

Keywords

Traffic classification, Applications, Machine Learning

1. INTRODUCTION

Enterprise or campus networks usually impose a set of rules for users to access the network in order to protect network resources and enforce institutional policies (for instance, no sharing of music files or gaming). This leaves network administrators with the daunting task of (1) identifying the application associated with a traffic flow on-the fly and (2) control user's traffic when needed. Therefore, accurate classification of traffic flows is an essential step for administrators to detect intrusion or malicious attacks, forbidden applications, or simply new applications (which may impact the future provisioning of network resources).

Previous work has proposed a number of methods to identify the application associated with a traffic flow. The simplest approach consists in examining TCP port numbers. Port-based methods are simple because many well-known applications have specific port numbers (for instance, HTTP traffic uses port 80 and FTP port 21). However, the research community now recognizes that port-based classification is inadequate [1, 2, 3, 4], mainly because many applications use dynamic port-negotiation mechanisms to hide from firewalls and network security tools. An alternative approach is to inspect the payload of every packet. This technique can be extremely accurate when the payload is not encrypted, but it is unrealistic alternative. First, there are privacy concerns with examining user data. Second, there is a high storage and computational cost to study every packet that traverses a link (in particular at very high-speed links).

Detecting the application associated associated with a flow is of limited interest once the flow is finished (except for provisioning and maybe for pricing). To address these challenges, we propose a novel technique that only uses the *size of the first few data packets* of each TCP flow to identify the application associated with a TCP flow. Our method runs at the *edge* of the network, i.e., where the network connects to the Internet, and hence is able to capture all packets associated with a TCP flow in both directions (from sender to receiver and vice-versa). The accurate classification of flows based on information contained only in the first few packets opens new possibilities for online classification (i.e. security, SLAs, etc.). Our method is in sharp contrast with the current state of the art.

2. STATE OF THE ART

After port-based classification has been debunked and given the limitations of searching payloads for signatures, there has been a new trend to classify traffic based on summarized flow information such as duration, number of packets and mean inter-arrival time [2], [5], [6], [3]. BLINC [4] introduced a new approach for traffic classification. It associates Internet hosts with applications. Instead of studying TCP (or UDP) flows individually, it looks at all the flows generated by specific hosts. BLINC is able to accurately associate hosts with the services they provide or use (application server, web client, etc.). However, it cannot classify a single TCP flow.

All classification techniques that use flow statistics can only identify the nature of a flow when the flow is finsihed. BLINC has to gather information from several flows for each host before it can decide on the role of a host. These requirements prevent the use of these methods online. In contrast, our method relies only on the first few packets of a TCP flow. This early classification is essential to allow automatic blocking, filtering, or recording of specific applications. It also limits the amount of memory required to store information associated with each flow.

3. KEY POINTS

This section discusses our assumptions and the two main insights of our method: the analysis of only the first few packets of the flow, and the use of unsupervised clustering to detect a set of flows that share a common behavior.

3.1 Assumptions

Our goal is to build a classifier that runs online and accurately identifies the application associated with a TCP flow as early as possible. Our work levarages the following observations:

- **Access to both directions of a TCP connection**. Edge networks usually connect to the Internet at few locations. The vast majority in one or two locations. We assume that the network administrator monitors both directions of all the edge links.
- **Offline access to flows of a number of applications**. Our method only needs packet size information. However, for training purpose, we need to know the application associated with a sample of flows. There are two alternative techiniques for collecting this training trace: (i) directly from one of the monitored links using high-end packet monitoring cards, we can analyze the payloads to identify the application corresponding to a given flow; or (ii) generated in a controlled fashion by explicitly launching instances of each of the desired applications and collecting all packets exchanged.
- **Online access to header of all packets**. We use only the information on the packet header to do our online classification, which is much smaller and of fixed size.

3.2 Intuition behind the method

Our goal is to identify the application associated with a TCP flow as early as possible. We design a classifier that uses information available in the header of the first P packets of the flow to identify the application associated to a flow. Specifically, we use only the size of the data packets, not using TCP control packets (SYN, ACKs, etc). The size of the first few packets is a good predictor of the application associated with a flow because it captures the application's negotiation phase, which is usually a pre-defined sequence of messages and distinct among applications.

We illustrate our approach with SMTP and E-Donkey flows captured at the edge of a large university network to shed light on the behaviors of applications. Figure 1 presents the size of the first packet versus the size of the second (negative values represents packets sent from the TCP server to the client and positive from client to server) for every TCP flow of these two applications. E-Donkey clients initialize request for files, hence the positive sign for the size of the first packet, whereas the SMTP sever initiates negotiation. This simple example illustrates the expressiveness of this two-dimensional representation of TCP flows to distinguish the application associated with each flow. In fact, the traces we study consist of ten major applications and we found that 5 packets were enough to distinguish their behaviors.

3.3 Unsupervised clustering

To extract groups of flows that share a common communication behavior, we borrow techniques from machine learning. We use *unsupervised clustering* as it relies on unlabeled data samples (in our case, the size of the first few packets of a TCP flow) to find natural groups (or *clusters*) in a dataset, whereas supervised clustering uses a pre-labeled set of samples to construct a model for each cluster.

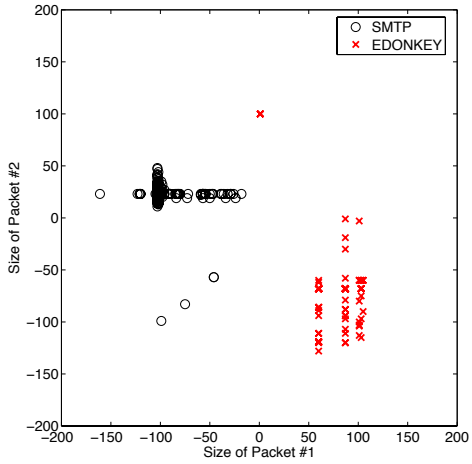

Figure 1: SMTP versus Edonkey flows.

Although the traffic classification mechanism presented in [3] uses Naive Bayes Classifiers, an example of supervised clustering, unsupervised learning is more appropriate for traffic classification because it does not rely on pre-defined classes. A single application can have multiple behaviors that should each have a model. For example, Figure 2 presents the size of the first packet versus the size of the second of a set of FTP flows. This figure shows that FTP has three very distinct behaviors: (i) command flows correspond to a control flow, in which a data connection is negotiated and in which clients ask for data; (ii) download flows corresponds to files transfer from client to server; and (iii) upload flows are file transfers from server to client.

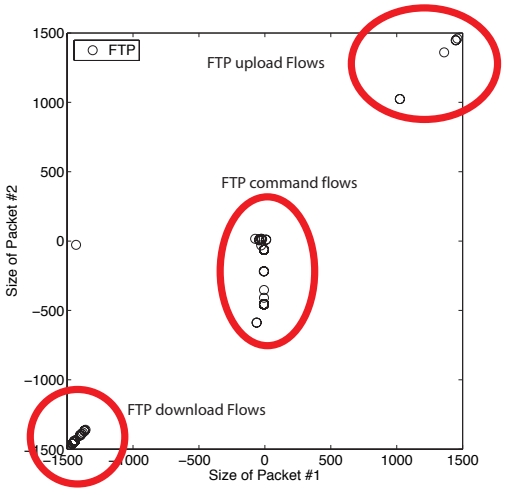

Figure 2: Example of a multi-modal application: FTP.

FTP is one example of a multi-modal application, in our traces we observe many other such applications. Unsupervised learning finds groups of flows that share the same behavior, which are easier to model.

4. METHODOLOGY

Based on the observations from the previous section, we propose a traffic classification mechanism that works in two phases: an offline learning phase and an online traffic classification phase. First, the learning phase uses a set of training data to cluster TCP flows that have common behavior. Then, the traffic classification phase uses these classes to determine the application associated with each TCP flow. To verify that the behaviors of applications do not vary with time, we use one data set for the learning phase and another for the evaluation of the derived classification method. These two traces were collected several month apart on the same link.

4.1 Learning phase

The learning phase is performed offline and consists in detecting common behaviors in a set of flows. It takes as input a short packet trace with TCP flows from a mix of applications. Our study uses a one hour trace collected at the edge of a university network. This trace also includes packet payloads. We apply a packet trace analyzer to this data set to extract TCP flows. First, to group flows into clusters, we need to evaluate the similarity between them. We associate each flow with a spatial representation based on the sizes of its first P packets. The representation we use is quite simple: flows are represented by points in a P-dimensional space where each packet is associated to a dimension. The coordinate on the dimension p is the size of packet p in the flow. In order to differentiate packets sent by the TCP-server and the TCP-client, packets sent by the server have a negative coordinate. The similarity between flows is then evaluated with a simple metric: the Euclidean distance between their associated spatial representation.

In order to extract common behaviors in such a space we rely on the well-known K-Means algorithm [7]. Once we have found natural clusters in the training set, we model and analyze each cluster. The modeling step consists in defining a heuristic to associate a new flow to a cluster. We use a simple heuristic: we compute the euclidean distance between the new flow and the center of each predefined cluster, and then choose the cluster for which this distance is minimum. The analysis step consists in examining the composition of each cluster. We process the training flows with a payload analysis tool [8] that is a able to accurately determine the application associated with each flow. This step allow us to decide how to label a flow once it has be assigned to a cluster.

We studied different numbers of packets and found that the best separation of applications among clusters was observed with 5 packets. We also tried different numbers of clusters for the K-Means algorithm and found that, for 5 packets, 50 clusters was the best trade-off between behavior separation and complexity.

The learning phase outputs two sets: one with the description of each cluster (or the center of the cluster) and the other with their composition (or the set of applications represented in the cluster). We use both these sets to classify flows online.

4.2 Online classification

Figure 3 presents the structure of the online classifier. This classifier can run either at a management host that has online access to packet headers or in a network processor at the router. We assume that the network administrators deploy a monitoring card that is able to process the header of all packets traversing the link [9]. This process is very light since it only involves retrieving information from the IP and TCP headers (the size of the packet and the flow id).

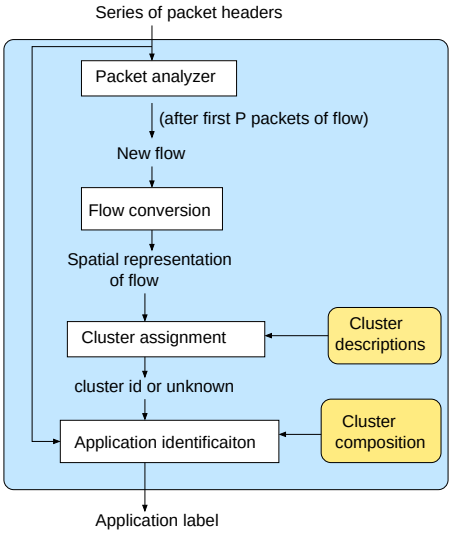

Figure 3: Design of online classifier.

Our classifier takes as input the series of packet headers for both directions of an edge link. A **packet analyzer** extracts the 5-tuple (protocol, source IP, destination IP, source port, destination port) and the packet size. The analyzer filters out control traffic (the three packets of the TCP handshake) and stores the size of every packet in both directions of the connection. When it has the size for the first P packets of the connection, it sends this information to the **flow conversion** module, which maps the new flow to a spatial representation. Then, the **cluster assignment** module searches all the cluster descriptions to find the best fit for the new flow and the **application identification** module selects which application is most likely associated with the flow given the set of applications that compose the cluster. We chose a simple heuristic to label associate unknown flows with applications: it is labeled with the application that is the most common in the cluster.

5. EARLY EVALUATION RESULTS

This section gives a proof of concept of our flow classification method. Further work is necessary to conclude in the applicability of the method. We build a prototype classifier using matlab. First, we use a training data set for the learning phase. Based on the cluster descriptions and cluster compositions found in the learning phase, we emulate the online classification using a trace collected six month afterwards on the same link. For validation purposes, we label this trace using the payload analysis tool used in the learning phase [8].

Table 1 presents the accuracy of our classifier when it considers the first five packets of each flow. We measure accuracy by comparing the application labels given by our classifier to the ones obtained through payload analysis. We were able to correctly identify more than 80% of flows of almost of the applications. The only exception is POP3. Our

classifier labels 86.8% of POP3 flows as NNTP and 12.6% as SMTP, because POP3 flows always belong to clusters where POP3 is not the dominant application. This error can be easily fixed if we consider extra information. In this example, if we had used the destination port to label POP3 flows, we would achieve over 90% accuracy. We are currently working on more sophisticated labeling techniques.

Application	Accuracy
edonkey	84.2%
ftp	87%
http	99%
kazaa	95.24%
nntp	99.6%
pop3	0%
smtp	84.4%
ssh	96.92%
https	81.8%
pop3s	89.8%

Table 1: Accuracy of flow classification.

6. LIMITATIONS AND CHALLENGES

The initial results observed with our method on a small trace are encouraging. The method is promising as it allows early classification of applications and is quite simple. However, the method has some limitations that we discuss below. Most of these limitations are easy to overcome, while others are more fundamental and affect most classification methods to date.

Multi-homed networks. Large networks often have multiple connections to the Internet. In this case, we can extend our approach to monitor all access links and aggregate information on a machine where the classification will take place.

Packet order. In IP networks, packets may be out of order or may appear more than once. Out-of-order packets will change the spatial representation of the flow, which will impact the quality of our classification. Fortunately, we only need the first five packets to arrive in order, which is more likely. On the network we studied, less than 4% of TCP flows had an out-of-order packet within the first five data packets of the flow.

Sampling. On high speed links, monitors cannot collect all packets. The accuracy of our method will degrade fairly quickly under packet sampling. If instead the network adopts flow sampling [10], then our method will work unaltered.

Applications with similar behaviors. If two distinct applications start by exchanging five packets of approximately the same size, then we classify both with the same label. To solve this problem, we are working on more sophisticated labeling heuristics that take into account other information available in the first few packets (such as port numbers and inter-arrival times).

Applications with unknown behaviors. The learning phase gives a model for the traffic present in the training data. Using this model on other networks could be inefficient. Although a specific behavior for an application precisely described on a trace should not change when observed on another network or at a different time, a new network can (and most likely will) have different application behaviors. We can identify new behaviors using limited cluster radiuses. Using this limit, the we return an accurate application label to previously-defined behaviors and classify as "unknown" all flows that are not represented in the training data.

Short flows. It is well known that the traffic is made of a large majority of short flows and a small number of very large flows. We need to test our method on these very short flows. In this scenario, if it is successful, it will not work better than most other techniques. However, previous classification techniques have not specifically studied these short flows and they may exhibit some limitations that have not been observed yet.

False matches. Most classifiers to date are based on heuristics and may misclassify a flow. This possibility of error, even if small, prevents the use of traffic classification techniques to automatically filter or block flows. Network administrators can use our method to identify suspicious flows early and store all packets in them. They can later audit the stored data

Evasion. The main challenge to traffic classification techniques is evasion. For instance, an "attacker" could easily evade our method by padding packet payloads in order to modify sizes. Payload analysis tools cannot classify encrypted packets, port-based methods are deceived by a simple change of port, and approaches relying on summarized flow information are sensitive to simple alterations of packet sizes and inter-arrival times.

To summarize, we now need to analyze our method on a much broader panel of traces. This is a difficult task as (i) we need the full trace with full payload for validation purposes (such traces are rare in the community); and (ii) we more sophisticated techniques to label a flow after it has been assigned to a cluster. We are also working on circumventing the limitation and the weaknesses of our method.

7. REFERENCES

[1] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding?," in *Globecom*, 2004.

[2] 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 *IMC 2004*, 2004.

[3] A. Moore and D. Zuev, "Internet traffic classification using bayesian analysis," in *Sigmetrics 2005*, 2005.

[4] T. Karagiannis, D. Papagiannaki, and M. Faloutsos, "Blinc: Multilevel traffic classification in the dark," in *SIGCOMM 2005*, 2005.

[5] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *PAM 2004*, 2004.

[6] D. Zuev and A. Moore, "Traffic classification using a statistical approach," in *PAM 2005*, 2005.

[7] J. McQueen, "Some methods for classification and analysis of multivariate tions," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–2297, 1967.

[8] Qosmos, "Traffic designer (www.qosmos.com)."

[9] Endace, "Endace (www.endace.com)."

[10] N. Hohn and D. Veitch, "Inverting sampled traffic," in *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, (New York, NY, USA), pp. 222–233, ACM Press, 2003.