Peer-to-peer traffic classification: exploiting human communication dynamics

Alessandro Finamore*, Marco Mellia*, Michela Meo*, Dario Rossi†, Silvio Valenti†
*Politecnico di Torino, Italy (first.last@polito.it)
†TELECOM ParisTech, France (first.last@enst.fr)

***Abstract*—This demo focuses on the online classification of traffic generated by P2P applications, considering two approaches with radically different designs. One approach is *payload* based: it inspects the packet payload to automatically gather a stochastic description of the content, thus inferring the *syntax* of the application protocol rather than payload *semantic*. The other approach we consider is *behavioral*: it analyzes the transport level exchanges of P2P applications, discriminating between different *protocol dynamics*. Both approaches achieve very reliable classification but, in reason of their different design, have their pros and cons. For instance payload-based classification fails when data is fully encrypted (e.g., IPsec, or encrypted TCP exchanges), while the behavioral classifier is unable to classify a single flow (i.e., as protocol dynamics need the observation of multiple flows). The demo software allows the user to interact with the classification engines, e.g. by injecting traffic to classify or by tuning and inspecting the classification process.**

I. Introduction

During the last few years, Internet traffic classification has gained considerable importance. Moreover, due to its widespread, P2P traffic has been one of the main targets of new classification methodologies. In this demo, we consider two of such approaches, having an orthogonal design. On the one hand, we have *Kiss* [1]–[3], which is a payload based classifier, built on the stochastic inspection of packet content – i.e., the "language" spoken by P2P applications. On the other hand, we have *Abacus* [4]–[6] which is a behavioral classifier, based on the raw count of packets and bytes exchanged by peers – i.e., the "loquacity" of P2P applications.

We build over the demonstration [5], that only illustrated the process of behavioral classification (Abacus), on which we integrate the payload-based classification process (Kiss). This allows to compare the classification process and accuracy of the two classifiers which, as shown in [7], are both very reliable. The definition of very expressive signatures and the use of state of art Support Vector Machines (SVM) in the classification process, which are known for their discriminative power, contribute to the high accuracy. During the demo, the user is given the chance of interacting with the classification process, by either actively injecting traffic in the network to feed the classifier engines, or by inspecting, tweaking and comparing the classification processes and results.

Moreover the demo reports information about the memory occupation and CPU consumption of the classifiers, hence allowing the user to have an idea of the cost of the classification. Our previous investigation of this issue in [7] shows that such an evaluation is not trivial, as the resource required by a classifier is usually heavily dependent from traffic mixture and condition. As the demo reports real-time measurements about computational resource consumption of both classifiers, the user is able to see how different traffic conditions directly impact the performance of the classifiers in this respect.

In the reminder of this document, we briefly describe the signatures used by Kiss and Abacus. Because of the limited space we mainly try to provide the reader with the intuition behind the classifiers' design, with the help of both an analogy with human communications and a few pictorial examples. Finally we spend some words on the demo itself, also presenting some screenshots from a running session.

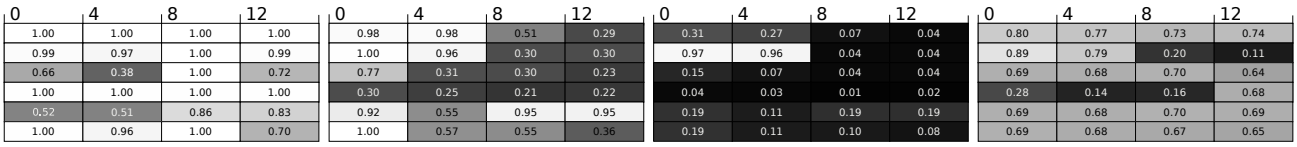
II. Kiss Signatures

The first approach we consider is based on the analysis of packet payload, trying to detect the syntax of the application protocol, rather that its semantic. The process is better understood by contrasting it with Deep Packet Inspection (DPI), which typically searches keywords to identify a specific protocol. With a human analogy, this corresponds to trying to recognize the foreign language of an overheard conversation by searching for known words from a small dictionary (e.g., "Thanks" for English language, "Merci" for French, "Grazie" for Italian and so on).

The intuition behind Kiss is that application-layer protocols can however be identified by statistically characterizing the stream of bytes observed in a flow of packets. Kiss automatically builds protocol signatures by measuring entropy (or Chi-Square test) of the packet payload. Considering the previous analogy, this process is like recognizing the foreign language by considering only the cacophony of the conversation, letting the protocol syntax emerge, while discarding its actual semantic.

Fig. 1 reports examples of mean Kiss signatures for popular P2P-TV applications like PPLive, SopCast, TVAnts and Joost[1]. The picture represents the application layer header, where each group of 4 bits is individually considered: for each group, the amount of entropy is quantified by means of a Chi-Square test χ^2 with respect to the uniform distribution. The syntax of the header is easy to interpret: low χ^2 scores hint to high randomness of the corresponding group of bit, due to

[1]Joost is no longer P2P but Web based



(a) Joost (b) SopCast (c) TVAnts (d) PPLive

Fig. 1. Mean kiss signatures, 24 chunks of 4 bits each (higher value and lighter color correspond to higher determinism)

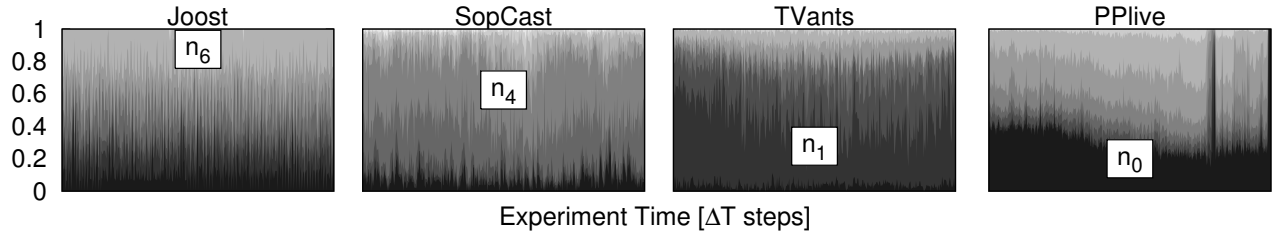


Fig. 2. Temporal evolution of Abacus signatures. Darker color correspond to low order bins, carrying less traffic. Bins are exponential so that $X_i \propto 2^i$, and a mark denotes the most likely bin.

obfuscation or encryption; high χ^2 scores instead are characteristic of deterministic fields, such as addresses or identifiers; intermediate values correspond to changing fields, such as counters and flags. As protocol languages are different, Kiss signatures allow to easily distinguish between applications [1], [3]

III. Abacus Signatures

The Abacus classifier leverages the observation that P2P applications perform different concurrent activities at the same time.

One activity, namely *signaling*, needed for the maintenance of the P2P infrastructure, is common to all applications. Still applications differ in the way they actually perform the signaling task, as this is affected by the overlay topology and design (e.g., DHT lookup versus an unstructured flooding search) and by implementation details (e.g., packet size, timers, number of concurrent threads.)

The *data-exchange* activity in instead related to the type of offered service (e.g., file sharing, content, VoIP, VoD, live streaming, etc.). Again, applications are remarkably different, both considering implementation details (e.g., codec, transport layer, neighborhood size, etc.) or the offered service (e.g., low and relatively stable throughput for P2P-VoIP, higher but still relatively stable aggregated incoming throughput for P2P-VoD and TV, largely variable throughput for file-sharing, etc).

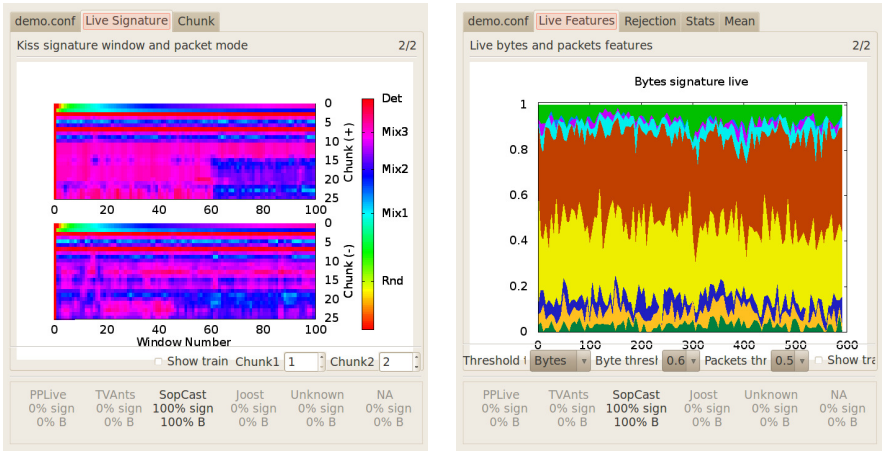
Such difference are so striking, that it is actually possible to finely differentiate between different P2P applications offering the same service: in what follows, we make an explanatory example considering the case of P2P-TV applications. We consider P2P-TV applications and contrast the possible ways in which they implement the live TV service. Concerning video transfers, for example, some application may prefer to download most of the video content from a few peers, establishing long-lived flows with them, whereas other applications may prefer to download short fixed-sized "chunks" of video from many peers at the same time. Similarly, some application may implement a very aggressive network probing and discovering policy, constantly sending small-size messages to many different peers, while others may simply contact a few super-peers from which they receive information about the P2P overlay. As such, some peers will be "shy" and contact a few peers, possibly downloading most of the data from them, while others will be "easy-going" and contact many peers, possibly downloading a few data from each.

These differences are shown in Fig. 2, which depicts the temporal evolution of (a simplified version of) the signature used for traffic classification. To capture the above differences, we asses the shyness of a peer P by gauging the proportion of peers that send to P a given amount of traffic in the range $X_i = [X_i^-, X_i^+]$. We then evaluate an empirical probability mass function p_i (pmf) by normalizing the count n_i of peers sending $x \in X_i$ traffic (e.g., packets or bytes), and by ordering the bins such that $X_{i-i}^+ \leq X_i^-$, i.e. low order bins contain less traffic (for reason detailed in [4] we use exponential binning, so that $X_i \propto 2^i$).

In Fig. 2, darker colors correspond to lower bins, and bins are staggered so that they extend to 1 (due to pmf). In the picture, the most likely (i.e., $\text{argmax}_i n_i$)bin is indicated with a textbox: it can be seen that each application has a behavior that, although not stationary over time, is however remarkably different from all the others.

IV. Demo Overview

A full-fledged implementation of both classifiers is included in the demo software. Basically the demo runs the two classification engines concurrently on the same traffic, which is either read from a recorded trace or directly captured live on network interface. The data used internally by the classification algorithms is exported at different point of their execution, before being used for the final classification. Then a user-friendly GUI (i) displays this data in a number of graphs



(a) Kiss window (b) Abacus window

Fig. 3. Example screenshots from the demo software.

updated in real time, besides (ii) allowing the user to interact with the classification process.

At the beginning the user is presented with two windows, one for each classification algorithm which is run independently from the other one. The user has first to select the socket (i.e. IP address and UDP port) which will be target of the classification: traffic exchanged by this socket is then processed and classified. As it can be seen in the screenshots in Fig. 3, the result of the classification is constantly displayed and updated in the bottom part of the window, as a percentage of correctly classified signatures and bytes.

Fig. 3 shows two examples of the kind of data displayed by the demo software. The Kiss picture on the left shows the evolution over time of the Chi-Square metric calculated for the first 24 groups of 4 bits in the packet payload. The top part plot is related to traffic received by the target endpoint, while the bottom part is related to traffic sent by the endpoint. Color is used to denote the level of randomness of the chunk: red chunks correspond to constant fields, whereas purple, blue, sky blue and yellow represent increasing levels of randomness. The Abacus picture on the right shows, instead, the time change of the distribution of peers according to the number of bytes sent to the target socket. Each bin of the distribution is represented with a different color and bins are stacked one over the other, building a well-defined and rather stable pattern.

Notice also that both windows contain some controls, just above the classification results. These allow the user either to tweak the parameters of the classifiers (e.g. the rejection criterion threshold for Abacus), or to select the information showed by the demo (e.g., to toggle the display of an additional window, which provides a view on the training set used by the classification engines.

V. Demo Requirements

The demo has very few requirements in term of equipment: in fact, the classification engines run smoothly even on a low-profile laptop. Although the demo can run on pre-recorded traces (e.g., by replaying them with TCPReply), we we believe that participants would greatly appreciate to watch the classifier challanged by live traffic, e.g. by running the P2P-TV applications on one machine and by capturing the traffic and running the demo on another one.

However, for this setup to be feasible (i.e., in order the P2P-TV applications to run properly and generate the traffic to be classified), *a wired Internet access* is definitely required. In fact a wireless access, furthermore shared with all participants to the conference, does not provide enough bandwidth to support P2P live-streaming.

Acknowledgement

This work has been funded by Celtic Project TRANS

References

[1] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection," in *Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537*, May 2009.

[2] G. L. Mantia, D. Rossi, A. Finamore, M. Mellia, and M. Meo, "Stochastic Packet Inspection for TCP Traffic," in *IEEE International Conference on Communications (ICC'10)*, (Cape Town, South Africa), May 2010.

[3] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection classifier for udp traffic," *IEEE Transactions on Networking, to appear*, 2010.

[4] S. Valenti, D. Rossi, M. Meo, M.Mellia, and P. Bermolen, "Accurate and fine-grained classification of p2p-tv applications by si mply counting packets," in *Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537*, pp. 84–92, May 2009.

[5] S. Valenti, D. Rossi, M. Meo, M. Mellia, and P. Bermolen, "An abacus for p2p-tv traffic classification," in *IEEE INFOCOM, Demo Session*, (Rio de Janeiro, Brazil), April 2009.

[6] D. Rossi and S. Valenti, "Fine-grained traffic classification with Netflow data," in *ACM IWCMC 2010, 1st International Workshop on TRaffic Analysis and Classification (TRAC)*, (Caen France,), June 28 - July 2 2010.

[7] A. Finamore, M. Mellia, M. Meo, D. Rossi, and S. Valenti, "Kiss to Abacus: a comparison of P2P-TV traffic classifiers," in *Traffic Measurement and Analysis (TMA'10), LCNS*, (Zurich, Switzerland), April 2010.