An Abacus for P2P-TV Traffic Classification

Silvio Valenti*, Dario Rossi*, Michela Meo†, Marco Mellia†, Paola Bermolen*

*TELECOM ParisTech, France
Email: first.last@enst.fr

†Politecnico di Torino, Italy
Email: first.last@polito.it

***Abstract*—This demo focuses on the online classification of traffic generated by P2P-TV applications, live video delivering services used by an ever increasing number of users worldwide. We designed a novel behavioural technique, which is able to reliably identify P2P-TV traffic simply based on raw counts of packets and bytes exchanged by the application during small-time windows. The demo software aims at showing the classification process and results, allowing users to interact on a simple active testbed with different live running P2P-TV applications.**

I. Introduction

We recently assisted to a rise of popularity of P2P-TV, i.e., applications which use the peer-to-peer paradigm to deliver live video content to the end-user. Since the increased popularity directly translates into an increase of P2P-TV traffic as well, this motivates the need for their reliable classification: yet, P2P-TV application identification is still a quite unexplored field. In [4] we designed, implemented and validated a classification engine, which is able to correctly identify end-points running these kind of applications by using signatures based on the count of the number of packets and bytes that such applications exchange with other peers during small-time windows. Our technique, which relies on Support Vector Machines [2] falls in the class of *behavioural classification* [1], a quite novel and light-weight approach which bases the classification process on the sole examination of the traffic patterns that applications generate.

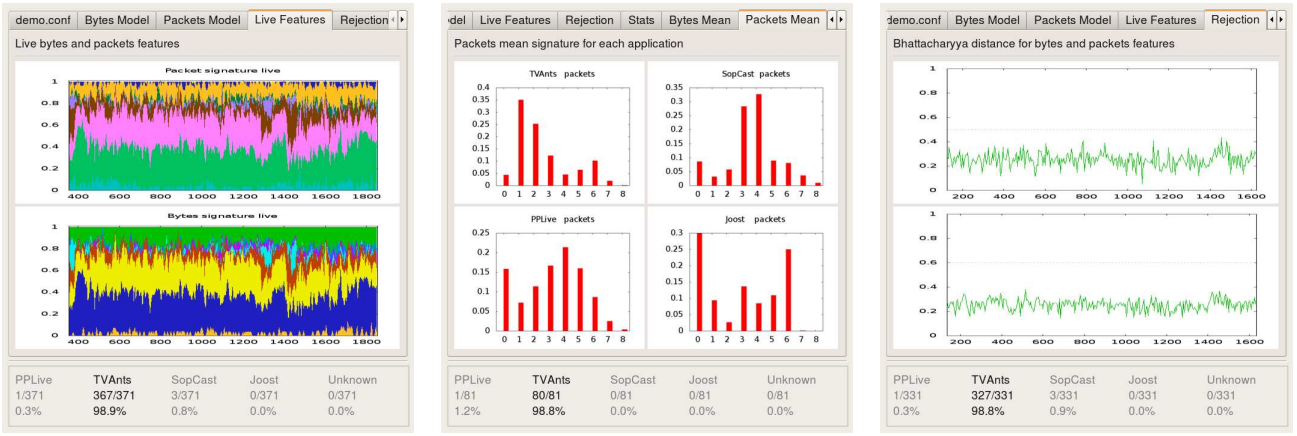
The goal of this demo is to illustrate the inner working of the classification framework [4], named *Abacus* from "Automated, behavioral application classification using signatures". In particular, we show the feasibility of *fine-grained* P2P-TV classification, i.e., the ability to distinguish between different P2P-TV applications, just by counting the number of exchanged packets and bytes. In practice, we setup a simple active testbed, where a probe PC runs different P2P-TV applications (namely PPLive, TVAnts, and SopCast) at the same time. The demo software captures this traffic, process it to extract the signatures needed by the classification engine and shows the classification results in real-time. Each step of this process is presented to the user in an interactive fashion, as users can control the target of the classification. Classification process is presented in an intuitive way, showing e.g., the temporal evolution of the Abacus signatures, of the classification results, and also of relevant statistics of the P2P-TV traffic.

II. Behavioural classification framework

In the following, we briefly describe our classification framework as well as the rationale behind it. For lack of space, we will omit details, which can be found in [4]. Basically, P2P-TV applications need to perform two different concurrent activities. The first is the download/upload of video content from/to other peers; the second is the maintenance of the P2P infrastructure, e.g., peers discovery and gossiping. Despite these tasks being common to all applications, each one performs them in its very own way. For example, concerning video transfers, some application prefers to download most of the video content from a few peers, establishing long-lived flows with them, whereas other applications prefer to download short fixed-sized "chunks" of video from many peers at the same time. Similarly, some application implements a very aggressive network probing and discovering policy, constantly sending small-size messages to many different peers, while others simply contact a few super-peers from which they receive information about the P2P overlay.

Based on the above remarks, we develop a simple framework able to pinpoint the design choices of different P2P-TV applications. In more detail, we focus on *UDP* traffic, which is the preferred transport layer protocol employed by P2P-TV applications. Moreover, since both signalling and video traffic are usually multiplexed on the same socket, we classify *end-points*, i.e. the pair IP address and transport layer port. We assume our classification engine to be sited at the *edge* of the network, where all the traffic exchanged by an end-point can be measured. Finally, we only consider the traffic *received* from an end-point, since live streaming applications need a steady downlink throughput to ensure a smooth playback.

Let us consider a single endpoint $\mathcal{P}_x = (IP_x, port_x)$, which in $\Delta T = 5s$ long time-window is contacted by a finite number of peers, $\mathcal{P}_j$, $j = \{1, \ldots, K(x)\}$. We want to asses which is the portion of these peers that sends to $\mathcal{P}_x$ a given number of packets or bytes. For this, suppose that each peer $\mathcal{P}_j$ sent p_j packets and b_j bytes to $\mathcal{P}_x$. We define a partition of $\mathbb{N}$, $\{I_0, \ldots, I_i, \ldots, I_B\}$ such that $I_0 = [0, 1]$, $I_i = [2^{i-1} + 1, 2^i]$ and $I_B = [2^B, \infty)$, using a different value of B for packets (Bp) and bytes (B_b). Without loss of generality, let us focus on the count of packets. We then calculate the number $N_i^x(p)$ of peers that sent a number of



(a) (b) (c)

Fig. 1. Screen-shot of the demo: (a) temporal evolution of signatures, (b) mean value of signatures for different applications, (c) rejection criterion

packets that falls in the bin I_i, i.e. $N_i^x(p) = \sum_{j=1}^{K(x)} \mathbf{I}_{\{p_j \in I_i\}}$. With this information, we can construct $p_x = (p_x^1, \ldots, p_x^{B_p})$ where $p_x^i = N_i^x(p) / \sum_{i=0}^{Bp} N_i^x(p)$, and we obtain a normalized vector which can be viewed as an empirical probability density function (pdf). We proceed similarly with the byte counters to construct the byte-wise signature b_x. Such simple signatures allow to capture crucial P2P-TV design choices. For instance, if an application implements an aggressive probing strategy, it will have a large number of peers falling in the low-order bins (i.e., the first components of its p_x signature will be large). Similarly, if an application uses video chunks that are, say, 16-packets long, its signature will exhibit a large fraction of peers falling in the 4-th bin.

By concatenating p_x and b_x in a single vector we obtain the *Abacus* signature, which is then fed to Support Vector Machines, a well-known class of algorithms for supervised multi-class classification. The SVM is trained off-line, with a data-set gathered from a large pan-European testbed [3], that contains signatures from all applications we want to classify. In the demo, live classification is performed every ΔT seconds, where we derive a new signature for the end-point under analysis and fed it to the trained SVM.

A natural question is how to handle traffic generated by applications that are *unknown* to the trained SVM, i.e., which are not included in the training set. To discard this traffic we designed a rejection criterion, based on the Bhattacharyya distance between empirical pdfs. Basically, we classify as "unknown" a signature that is too distant from the mean signature of the class to which it was previously assigned by SVM. Details of this procedure can be found in [4].

III. Abacus Demo

The demo software is a complete implementation of the *Abacus* classification framework, featuring a simple graphical interface which allows us to control and see the system in action. The demo runs either with pre-recorded traces, or live by directly capturing the traffic on a network interface.

The user first has to select an end-point from a list automatically generated based on the observed traffic. For instance, since the probe PC runs three applications (i.e., PPLive, SopCast and TVAnts) at the same time, the list will contain three active end-points. Once a socket is selected, the classification process (i.e. *Abacus* signature computation, SVM classification and application of rejection criterion) starts. Fig. 1 reports some screenshots taken at different times of a live session of the demo with TVAnts running on the probe PC. In Fig. 1-(a) the temporal evolution of the bytes and packets signature is shown, where components of each signature are vertically staggered and represented with different colours; as each signature is a pdf, the sum of all its components is equal to 1. A simple visual inspection of the signatures shows that some components are "dominant" over the others, which are furthermore different across applications. In Fig. 1-(b), we represent the *mean* of the packets signatures of the SVM training set: differences are so evident that it could be possible to tell which application is running just by visually comparing the live-signature with the training ones. Finally Fig. 1-(c) shows the temporal evolution of the distance used by the rejection criteria: the threshold can be *interactively* set, in order to evaluate the impact of this parameter on the system performance. Besides the plots of Fig. 1, the demo software present other statistics (e.g., the downlink throughput, number of contacted peers, number of received packets, etc.), which are both interesting per se, and insightful for what concerns the different activities of P2P-TV applications.

References

[1] T. Karagiannis e al. "BLINC: multilevel traffic classification in the dark," *ACM Communication Review,* Vol. 35, No. 4, 2005

[2] N. Cristianini, J. Shawe-Taylor, "An introduction to support Vector Machines and other kernel-based learning methods," *Cambridge University Press*, New York, NY, 1999.

[3] NAPA-WINE , http://www.napa-wine.eu.

[4] S. Valenti, D. Rossi, M. Meo, M. Mellia, P. Bermolen, "Accurate, fine-grained classification of P2P-TV applications by simply counting packets", in *International In Traffic Measurement and Analysis (TMA) Workshop at IFIP Networking'09* Aachen, Germany, May 2009