Abacus: Accurate Behavioral Classification of P2P-TV Traffic

Paola Bermolen[a], Marco Mellia[b], Michela Meo[b], Dario Rossi[a], Silvio Valenti[a,*]

[a] *TELECOM ParisTech, 46 rue Barrault, 75634 Paris, France*
[b] *Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy*

Abstract

Peer-to-Peer streaming (P2P-TV) applications offer the capability to watch real time video over the Internet at low cost. Some applications have started to become popular, raising the concern of Network Operators that fear the large amount of traffic they might generate. Unfortunately, most of P2P-TV applications are based on proprietary and unknown protocols, and this makes the detection of such traffic challenging per se. In this paper, we propose a novel methodology to accurately classify P2P-TV traffic and to identify the specific P2P-TV application which generated it. Our proposal relies only on the count of packets and bytes exchanged among peers during small time-windows: the rationale is that these two counts convey a wealth of useful information, concerning several aspects of the application and its inner workings, such as signaling activities, video chunk size, etc.

Our classification framework, which uses Support Vector Machines, accurately identifies P2P-TV traffic as well as traffic that is generated by other kinds of applications, so that the number of false classification events is negligible. By means of a large experimental campaign, which uses both testbed and real network traffic, we show that it is actually possible to reliably discriminate between different P2P-TV applications by simply counting packets.

Keywords: Traffic Classification, Support Vector Machine, P2P live-streaming

1. Introduction

The Internet has proved to have an awesome capability of adapting to new services, migrating from the initial pure datagram paradigm to a real multi-service infrastructure. One of the most recent step of this evolution is P2P-TV, i.e., large-scale real-time video-streaming services exploiting the peer-to-peer communication paradigm. There are several currently deployed P2P-TV systems [1, 2, 3, 4], which feature low-quality

* Corresponding author. Tel. +33.1.4581.7275, Fax +33.1.4581.7158
Email addresses: paola.bermolen@telecom-paristech.fr (Paola Bermolen), mellia@tlc.polito.it (Marco Mellia), meo@tlc.polito.it (Michela Meo), dario.rossi@telecom-paristech.fr (Dario Rossi), silvio.valenti@telecom-paristech.fr (Silvio Valenti)

and low-bitrate streaming, with high-quality systems just beyond the corner. In P2P-TV systems, hosts running the application, called *peers*, form an *overlay topology* by setting up virtual links over which information is transmitted and received. A source peer injects the video stream, by chopping it into data units of a few kilobytes, called *chunks*, which are then sent to a few other peers, called *neighbors*. Each peer contributes to the video diffusion process by retransmitting chunks to its neighbors following a swarming-like behavior, somehow inspired to file sharing P2P systems like BitTorrent.

P2P-TV is candidate for becoming the next Internet killer application as testified by the growing success of commercial systems (such as PPLive, SopCast, TVAnts and many others) which already attract millions of users every day [5]. **Also, Cisco estimates that, globally, P2P-TV traffic is now over 280 petabytes per month [6], and is projected to increase further over the next year. As a consequence, P2P-TV systems gathered the attention of the research community, interested in understanding their behavior and improve their performance, while the Internet Service Providers (ISP) community have raised some concerns about them.** Indeed, P2P-TV traffic may potentially grow without control, causing a degradation of quality of service perceived by Internet users or even the network collapse [7]. In fact while the downlink rate of peers is limited by the video stream rate, the uplink rate may grow unbounded as observed in [5]. Unfortunately, most successful P2P-TV systems follows a closed and proprietary design, so the algorithms and protocols they adopt are unknown.

As such, the identification of P2P-TV applications is a topic of growing interest. For instance, ISPs can be interested in blocking application A and at the same time explicitly supporting application B, because the ISP itself provides a service relying on B. In a similar way, an operator could be forced to block the traffic of an application for some infringements (e.g., copyright), while still protecting the traffic of the application used by other broadcasters. More possible uses of P2P-TV application classification can be found in the field of network *monitoring* (e.g., ranking applications according to their popularity), security (blocking a given application which is exploited for DDoS attacks or worm diffusion) and charging. We think that, with the growing diffusion of P2P-TV applications, ISP will soon be asking for tools enabling this kind of activities.

However, despite much valuable effort has already been devoted to the task of Internet traffic classification [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], P2P-TV traffic identification has only been partially addressed by payload-based mechanisms [11]. Among the different classification methodologies, a very promising one is the so called "behavioral" approach [18, 19]. Behavioral classification aims at identifying the traffic by the sole examination of transport-layer traffic patterns, and is very lightweight as it does not require neither packet-payload inspection nor per-packet operation. However, behavioral classification has so far achieved only coarse-grained classification of Internet applications, identifying broad application *classes* (e.g., interactive, P2P, Web, etc.) rather than discriminating different applications within the same class. Thus, the design of a fine-grained classification engine that only exploits behavioral characteristics remains, to date, an open problem.

In this work, we tackle precisely this issue by designing a novel behavioral classification framework, tailored for P2P-TV applications, which is able to achieve fine-grained classification (i.e., distinguish among applications). Our framework, which was described in a preliminary version in [20], uses simple application signatures gathered from the count of packets and bytes that peers exchange during small time windows. To validate the proposed classification engine, we carry out a thorough experimental campaign using both testbed traces and passive measurements collected from real networks. We consider four P2P-TV applications, chosen for their popularity among the large number of available ones. Our results show that the percentage of correctly classified traffic is above 95% of bytes. Moreover, the engine correctly labels as "unknown" the traffic generated by non P2P-TV applications, keeping the false positive rate (i.e., wrong classification of non P2P-TV traffic as such) below 0.1% in the worst case.

The rest of this paper is organized as follows. Sec. 2 defines the application signatures and describes the classification framework, providing justification of its founding rationale. **Sec. 3 thoroughly describes the workflow, methodology and datasets used to validate the classification engine. Sec. 4 then illustrates baseline classification results, providing an extensive study of the signature portability, to show that the proposed technique works in rather different network environments. A careful sensitivity and robustness analysis of the method to internal parameters is reported in Sec. 5. Afterwards, we show in Sec. 6 that an extended signature definition can further improve the classification accuracy. Finally, Sec. 7 overviews related works and Sec. 8 concludes the paper, summarizing the main messages and illustrating future research directions.**

2. Classification Framework

2.1. The Rationale

Our aim is to classify P2P-TV *end-points*, which can be identified by IP address and transport layer port pair $(IP, port)$. Typically, P2P-TV applications rely on UDP as the transport protocol. During installation, a single UDP port is selected at random, over which all the signaling and video traffic exchanged with other peers is multiplexed. Therefore, all the traffic going to/coming from a given *(IP,UDP-port)* end-point is actually destined to/sourced from the same P2P-TV application running on the host. This holds true for P2P-TV applications like PPLive[1], SopCast[2], TVAnts[3] and Joost[4][1], which we take as examples throughout this paper. Because of the continuous development of new applications, the choice of a representative set is definitely a difficult one. We decided to use the most popular applications at the time of experiments.

As mentioned before, we design a P2P-TV classification methodology that relies only on the evaluation of the *amount of information*, such as packets and bytes, exchanged by peers during small time-windows. The rationale is that a raw count of

[1]Joost became a Web-based application in October 2008, but at the time when the experiments were performed it offered VoD and live-streaming by P2P.

exchanged data conveys useful information concerning several aspects of P2P-TV applications.

A human analogy may help in clarifying the intuition. Suppose peers in the network are people in a party room: people generally have different behavior, e.g., they will be more or less talkative. As such, somebody may prefer lengthy talks with a few other people, whereas somebody else may prefer very brief exchanges with a lot of people. This is similar to what happens with P2P applications: some applications continuously perform peer discovery by sending few packets to a previously not-contacted peer; others tend to keep exchanging most of packets with the same peers.

Additionally, most P2P-TV applications have been designed around the concept of "chunks" of video, i.e., small units of information whose size is a typical parameter of each application[2]. Download of video content is thus performed using several chunks, and the size of flows carrying the video content is roughly a multiple of the chunk size. Moreover, P2P-TV video service has an almost constant downlink throughput, due to the nature of the video stream. By tracking the *breakdown* between the different contributors it is possible to highlight different policies that a particular application can adopt, namely, fetching chunks from many neighbors, or downloading from a restricted list of preferential peers. Yet, while any P2P-TV peer consumes equally, the amount of uploaded data can be significantly different from peer to peer, due to different configuration, such as upload capacity. For example, in [5], it is shown that uplink to downlink throughput ratio for PPLive varies in the $[0, 10]$ Mbps range, for the same downlink throughput of about 400 Kbps. In reason of the above observation, we assume that the classifier is located at the *edge* of the network (where all traffic exchanged by a given end-point transits), and consider only the *downlink* direction, i.e., traffic coming from the Internet and crossing the edge of the network into the end-point direction. Notice that once an endpoint has been identified by means of downlink traffic, the uplink traffic is classified as well. However, the additional use of the characteristics of the uplink traffic to support the classification represents an interesting direction for future work.

In the following, we restrict our attention to UDP traffic, although endpoint identification can be extended to applications relying on TCP at the transport layer as well. In case TCP is used, the client TCP port is ephemeral, i.e., randomly selected by the Operating System for each TCP connection. The TCP case thus requires more complex algorithms in case of traffic *generated* from a specific peer, since ephemeral ports differ among flows generated by the same peer. However, we point out that the ephemeral port problem vanishes if we focus on the downlink direction as we do in this work (i.e., since we need in this case to aggregate all traffic received by a TCP server port, that remains the same for all flows of any given peer).

2.2. Behavioral P2P-TV Signatures

Let us consider the traffic received by an arbitrary end-point $\mathcal{P} = (IP, port)$ during an interval of duration ΔT. We evaluate the amount of information received by $\mathcal{P}$ simply as the number of received packets. **In Sec. 6 we extend this concept to account**

[2] **Note that while *frames* are the typical unit of data generated by video encoders, the segmentation in *chunks* is instead imposed by the P2P application and is typically independent from the codec.**

also for the amount of bytes, which we will show to further improve classification performance.

We partition the space $\mathbb{N}$ of the possible number of packets sent to $\mathcal{P}$ by another peer into $B_n + 1$ bins of exponential-size with base 2: $I_0 = (0, 1]$, $I_i = (2^{i-1}, 2^i]$ for $i = 1, \ldots, B_{n-1}$ and $I_{B_n} = (2^{B_n-1}, \infty]$. For each ΔT interval, we count the number N_i of peers that sent to $\mathcal{P}$ a number of packets $n \in I_i$; i.e., N_0 counts the number of peers that sent exactly 1 packet to $\mathcal{P}$ during ΔT; N_1 counts the number of peers that sent 2 packets; N_2 the number of peers that sent 3 or 4 packets and, finally, N_{B_n} is equal to the number of peers that sent at least $2^{B_n-1} + 1$ packets to $\mathcal{P}$. Let K denote the total number of peers that contacted $\mathcal{P}$ in the interval. The behavioral signature is then defined as $\underline{n} = (n_0, \ldots, n_{B_n}) \in \mathbb{R}^{B_n+1}$, where:

$$n_i = \frac{N_i}{\sum_{j=0}^{B_n} N_j} = \frac{N_i}{K} \tag{1}$$

The signature $\underline{n}$ is the observed probability mass function (pmf) of the number of peers that sent a given number of packets to $\mathcal{P}$ in a time interval of duration ΔT; this function is discretized according to the exponential bins described above. The choice of exponential width bins reduces the size of the signature, while keeping the most significant information that can be provided by the pmf. **In fact, since low order bins are much finer, short flows are likely to end up in different bins, even though the difference in their counts is small (e.g. flows composed by a single packet, two packets and three packets are counted respectively in the components n_0, n_1 and n_2). On the contrary, longer flows are coarsely grouped together in the higher bins. Intuitively, having a finer characterization of short flows can provide much information (e.g., distinguishing between single-packet probes versus short signaling exchanges spanning several packets), while there is no gain in having an extreme accuracy when considering long flows (e.g., distinguishing between 500 or 501 packet long flows). This intuition is discussed in Sec. 5, where we examine the impact of different binning strategies.**

Since $\underline{n}$ has been derived from the pure count of exchanged packets, we name it "Abacus", which is also a shorthand for "Automated Behavioral Application Classification Using Signatures". Before describing the whole classification process, let us show the expressiveness of the Abacus signatures, by presenting a few examples.

In Fig. 1-(a) we show an example of temporal evolution of the signature $\underline{n}$ for each of the four applications we consider in this paper (from left to right, Joost, SopCast, TVAnts and PPLive). Each plot is built by running a single application on a controlled peer for an hour and capturing the received packets. Then we process this traffic and compute a signature $\underline{n}$ for each interval $\Delta T = 5\,s$. Each graph has the time on the x-axis, while on the y-axis the value of each components n_i is reported. Each component is represented with a shaded area of a particular level of gray (with darker colors corresponding to low-order components, and lighter ones to high-order components). Moreover the components are staggered one above the others, so that n_0, the darkest component, extends from 0 to n_0, while n_1 extends from n_0 to $n_0 + n_1$ and so on. Clearly, as the overall signature is itself a pmf, the sum of all components is equal to 1.

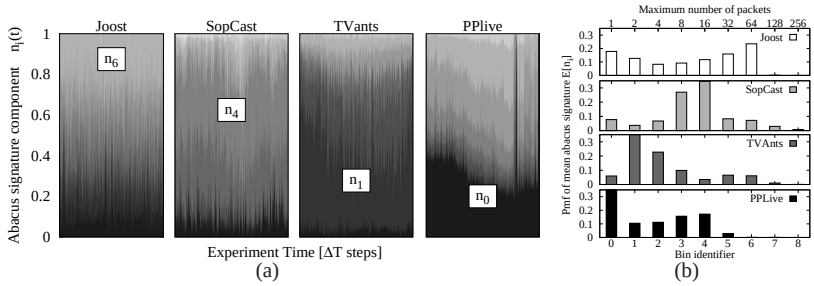

Figure 1: Temporal evolution of P2P-TV applications signature with a mark for the widest bin in (a), and mean value for each component of Abacus signatures in (b)

Each application has its own characteristic distribution, which is extremely different from the others. The most probable bin (i.e. the one which exhibits the highest values for most of the intervals ΔT) is highlighted in the figure, showing that it is different for each application. Interestingly, the most probable bin remains the same during most of the application lifetime, despite its actual width varies over time. Notice that the breakdown is not stationary over time for all applications: this is for instance the case of PPLive, as it emerges from the rightmost plot of Fig. 1-(a), which hints to transient or possibly "multi modal" behaviors. The dark vertical line toward the end of PPLive experiment corresponds to a sudden massive increase of n_0, due to a 10-seconds long blackout period (i.e. two ΔT intervals), where the end-point under observation was essentially receiving single-packet probes, and likely no video chunks.

To better highlight how Abacus signatures capture the differences between applications, we have computed the *average* of each single signature component n_i over all the intervals represented in Fig. 1-(a), and reported it in the histograms of Fig. 1-(b). Bin identifier i is reported on the x-axis, with top x-axis showing the maximum number of packets within the bin. Interesting behaviors stand out from the picture. For instance, Joost peers preferentially receive either a single or several $(32, 64]$ packets from any given peer. SopCast instead prefers middle-sized burst of $(5, 16]$ packets, while TVAnts prefers lower order bins $(2, 7]$ packets. Finally, PPLive highly prefers single packet exchanges. This confirms that different P2P-TV applications have remarkably different behaviors, just like humans at a party: in the next sections, we exploit this evidence for classification purpose. Obviously if two applications employ the same signaling and diffusion algorithms, they are characterized by a similar behavior and, thus, hardly recognizable. Under these assumptions, a fine-grained classification is no longer possible. Notice that this is a common problem with all classification methodologies: as far as the features are the same, the classifier is confused. For example, traditional Deep-Packet-Inspection (DPI) classifiers cannot distinguish two VoIP applications relying on the same protocol at the session level, e.g., RTP. Similarly, behavioral classifier that use packet size and inter-packet-gap as features cannot distinguish VoIP applications that use the same Codec.

3. Methodology and Dataset

Classification of P2P-TV traffic can be performed by exploiting the Abacus signatures so far described through any supervised learning machine. In this work, we resort to Support Vector Machines (SVM), which are well known for their discriminative power in both the learning machine community [21], and have more recently peered in the Internet traffic classification literature as well [22]. Specifically, we make use of `LibSVM` implementation [23] of SVM. Since providing a through overview of SVM is beyond the scope of this paper, we invite the reader to go through [21, 22] for further details concerning SVM. In this section, we describe the workflow we follow in the evaluation, as well as describe dataset used for our experimental campaign.

3.1. Workflow overview

From a high-level perspective, the P2P-TV classification workflow consists of three phases: the *training* phase, whose output is a model that can be used for the *classification* phase. A third phase is needed to *validate* the classification results against a reference ground truth. In what follows, we detail and explain the workflow with the help of Fig. 2.

In the context of SVM, entities to be classified are represented by means of some distinctive features, which in our case are the Abacus signatures computed on real traffic. Since SVM is a supervised algorithm, the machine must be *trained* first; this process is illustrated in the top part of Fig. 2. During the training phase, the machine is fed some sample Abacus signatures, with the associated labels that specify the generating P2P-TV applications. Notice that preliminary to this process, an *oracle* (e.g. DPI) is used to associate the protocol label to the traffic signatures. Oracle labels are considered accurate, thus representing the *ground truth* of the classification. In our case, using the testbed traces described later in this section, we generate labeled signatures of known P2P-TV applications: specifically, for each trace, we build an Abacus signature $\underline{n}$ every interval of ΔT seconds. Each signature is then possibly chosen, at random, to be included in the training set: impact of training set size and selection policy on the classification performance is discussed in details in Sec. 5.

Irrespectively of the chosen oracle, the output of this phase is a *trained model*, which is basically a careful selection of samples from the original training set. In more detail, given a geometric representation of computed features in a multidimensional space, the idea of SVM is to *partition* the space into regions, each based on a few representative samples of each class called *Support Vectors*. Defining the surface that partitions the space into areas is however tricky, since training signatures can be spread out, so that complex surfaces should be described. The key idea of SVM is to map, by means of *kernel* functions, the training features into a higher dimensional space so that samples belonging to different classes can be separated by the simplest surface, i.e., an hyper-plane. The training phase is precisely the procedure to build the *separation surface*. In our case, Abacus signatures $\underline{n}$ represent points in a (B_n+1)-dimensional space, which is then transformed into an infinite-dimensional space by means of a Gaussian kernel, where the partition

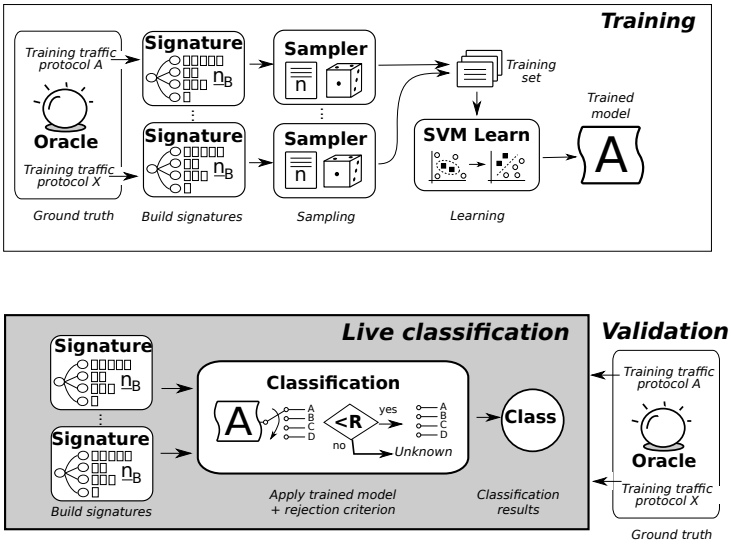

Figure 2: Classification framework: Model training (top) and validation (bottom). Live classification is performed as in validation, except that no ground truth is available as in the validation case.

in subregions takes place. We examine the impact of other kernels later on in Sec. 5.

The SVM generated during the training phase is used to associate a label to a previously unseen signature: this process is called *classification* and is represented in the bottom part of Fig. 2 (shaded gray area). To classify a new signature, SVM first remaps it in the higher dimensional space, and assigns it to the class whose region of space it falls into. Finally, SVM classification is accepted provided it passes a *rejection criterion*, to correctly discard non P2P-TV traffic. In fact, as SVM partitions the space into regions, it will *always* label any new sample as belonging to one region specified during the training phase. We thus devise a rejection criterion of which here we only convey the main idea, deferring mathematical details to Appendix A. The criterion is based on a measure of distance between signatures in the probability distribution space. For each new sample classified by SVM as label c, we evaluate the distance of the new sample from the center of training vectors having label c: whenever the distance exceeds a rejection threshold R, the SVM classification is rejected and the sample is labeled as "Unknown".

In case the classification performance is under analysis, classification results are finally compared with the reference ground truth label. This process, known as *validation* phase, allows to assess the expected performance of the classification engine in real operational networks: baseline results are reported in Sec. 4, with special attention to the signatures portability in Sec. 4.2 and sensitivity to parameters setting in Sec. 5.

Finally, it is worth remarking that, apart from the training phase, the overall

framework is well suited for live classification, yielding a classification result every ΔT s, (which is the parameter that defines the *reactivity* of our classifier). Notice as well that all operations performed on observed traffic are extremely lightweight so that our classifier can cope also with high rates of traffic (a brief analysis of the scalability of our Python implementation is reported at the end of Sec. 5). As a side note, we have also released an open-source demo of our classifier, available online [24], that allows the user to run Abacus in real-time, on live traffic captured on a real network interface. At the same time, we point out that in all our experiments we used Abacus as an *offline* classifier on pre-recorded traces with associated ground-truth, in order to perform multiple experiments on the same dataset, which is is needed to have a trustworthy validation of the classification performance. However, from the above discussion it follows that results are representative of live classification as well.

As a matter of fact, the only offline operation is constituted by the training phase. Concerning this point, we anticipate that the results of our portability analysis show that a well chosen training set can be used to successfully classify traffic in a variety of network conditions, so that offline retraining is rarely required. Moreover, the collection of the training set is a very simple process, which can be easily automated. This is not the case for many other classifiers, usually presenting a much more complicated training phase, which can even involve manual inspection of the packet traces (e.g. DPI requires analysis of the traffic to extract keywords or regular expressions that identify the target protocol).

3.2. Dataset and Oracle

Assessing classification performance is known to be a non-trivial task due to the difficulty to know the "ground truth", i.e., what is the actual application that generated the traffic [9]. **Testing the classification engine by means of artificial traffic (e.g., by generating traffic in a testbed) solves the problem of knowing the ground truth (i.e., you are the oracle), but synthetic traces are hardly representative of real world traffic.** Assessing the performance against traffic traces collected from operative networks is therefore mandatory. Moreover, even when considering real traffic traces, performance of the classifiers can be affected by the scenario (e.g, corporate and residential networks have very different traffic mixes). **Still, the major problem when dealing with real traffic traces is finding the ground truth: for instance, no reliable DPI mechanism exists nowadays for P2P-TV.**

In reason of the above trade-off, we adopt a mixed approach: we use both (i) traces actively gathered in a large scale testbed, and (ii) passive traces collected from different real operational networks. **Testbed traces contain P2P-TV traffic only and allow us to evaluate the engine capability to correctly discriminate P2P-TV applications and correctly label all P2P-TV traffic. Conversely, real network traces do not contain any P2P-TV traffic and allow us to verify that the engine correctly handles unknown applications as well (i.e., do not label other traffic as P2P-TV).**

3.2.1. Testbed Traces

To overcome the testbed representativeness limitation, we setup a large testbed involving multiple measurements points. The testbed was setup **in May 2008** in the

Table 1: Summary of the hosts, sites, countries (CC) and access types **(NAT=Network Address Translation, FW=Firewall)** of the peers involved in the testbed.

Host	Site	CC	Access	NAT	FW
1-4	BME	HU	high-bw	-	-
5			DSL 6/0.512	-	-
1-9	PoliTO	IT	high-bw	-	-
10			DSL 4/0.384	-	-
11-12			DSL 8/0.384	Y	-
1-4	MT	HU	high-bw	-	-
1-3	FT	FR	high-bw	-	-
1-4	ENST	FR	high-bw	-	Y
5			DSL 22/1.8	Y	-
1-5	UniTN	IT	high-bw	-	-
6-7			high-bw	Y	-
8			DSL 2.5/0.384	Y	Y
1-8	WUT	PL	high-bw	-	-
9			CATV 6/0.512	-	-

Table 2: Details about the Testbed Traces

Application	Hr	UDP Signatures	UDP Packets	UDP Bytes	UDP%
SopCast	36	26k	17.2M	7.5G	92.5
TVAnts	36	26k	14.2M	7.1G	33.0
PPLive	26	19k	11.7M	5.1G	70.7
Joost	30	22k	6.1M	6.4G	99.5
Total	128	93k	48.2M	26.1G	73.7

context of NAPA-WINE, a 7th Framework Programme project funded by the EU [7]. The testbed involved more than 30 controlled peers, hosted at 7 different Institutions, scattered over 4 European countries and connected to 9 different Autonomous Systems. **Details concerning the experiments are reported in Tab. 1, and a thorough analysis of P2P-TV application behavior is available in [25]. During the experiments, each PC ran the same application for one hour, during which all involved peers were forced to watch the same channel at the same time. SopCast, TVAnts, PPLive and Joost were run: in all (but Joost) cases, the nominal stream rate was 384 kbps ($\sim$550 kbps) and Windows Media 9 Encoder was used. PCs were synchronized via NTP and MS Windows scheduler was used to automatically start the application and tune to the selected channel. Each PC captured the packet-level traces for the whole duration of the experiment (no packet sampling or loss occurred, and broadcast/multicast packets were filtered out).**[3].

[3]Traces differ because during the experiment some application could not successfully run, e.g., due to peer failure, or bad network condition.

Since our classifier operates only on downlink traffic, that is the traffic directed to the target endpoint, we first extracted the relevant UDP packets from the traces. The overall duration, number of signatures (i.e. snapshot of $\Delta T = 5\ s$) as well as number of UDP packet and UDP bytes finally included in our dataset per application are reported in Tab. 2. Notice that the table also reports the percentage of UDP traffic relatively to the overall amount of IP traffic in the captured testbed traces. With the exception of TVAnts, we gather confirmation that UDP traffic is prevalent, accounting to about 3/4 of the total traffic volume, and more than 90% in case of SopCast and Joost. Notice that this changes with respect to previous work [26] in which TCP was found to be responsible for the bulk of the exchanges. As the versions of the software that we used in our experiments is more recent than that used in [26], the data confirms that P2P-TV software is continuously evolving[27], and that in this evolution UDP is preferred over TCP. In the TVAnts case, instead, the software version did not evolved from 2006, in which case UDP/TCP ratio is in agreement with [26].

Overall, the testbed is representative of about 130 hours worth of video streaming, 93k signatures samples, 48M packets and 26 GBytes of data. **Moreover, as different network setups (e.g., access technologies, security policies, private/public addresses, etc.), different content (popular vs unpopular channels) and peers configurations (hardware, OS) were part of the testbed, we are confident that the heterogeneity of the dataset is representative of a wide range of scenarios.**

3.2.2. Real Traces

Real traffic traces are collected from two different networks in Italy.

- CAMPUS (C) is a 5-days long trace, collected during one working week at the edge router of Politecnico di Torino LAN, which is representative of a typical data connection to the Internet [28]. The LAN contains about 7000 hosts, whose users can be administrative, faculty members and students. Most of the traffic is due to TCP data flows carrying web, email and bulk traffic, since a firewall blocks all P2P file sharing applications.

- ISP (I) is a 1-day long trace collected from one of the main broadband ISP in Italy which offers triple-play services over an all-IP architecture to more than 5 millions of users. The ISP network is representative of a very heterogeneous and uncontrolled scenario, in which customers are free to use the network without restriction. Traffic is sniffed at a PoP level, to which about 500 users are connected, using more than 2000 different IP addresses considering VoIP phones, set-top-boxes, PCs.

In both cases, traces were collected during May 2006, when P2P-TV applications were not popular in such networks. As such, they are instrumental to assess the amount of false positive classification (i.e., non-P2P-TV traffic classified as P2P-TV).

As detailed in Appendix A, the rejection criterion is very effective in avoiding false alarms: in 72% of the CAMPUS and 85% of the ISP traffic volume respectively, no false positive classification is possible *a priori*. However, our aim is to

Table 3: Details about Real Traces. To perform a worst-case analysis, only end-points that can lead to false positive classification are considered (28% of the CAMPUS and 15% of the ISP overall traffic volume).

Network	Traffic	Signatures	Packets	Bytes
CAMPUS	UDP	1.9M	73.6M	10.6G
	Skype	0.5M	11.9M	2.2G
	DNS	0.2M	5.0M	0.7G
ISP	UDP	0.7M	28.5M	24.9G
	eDonkey	0.3M	9.8M	1.4G
	DNS	24.4k	0.6M	37.8M

gather conservative performance bounds: therefore, we build a *worst-case scenario* for the comparison, so that the actual performance in operational networks can be expected to be much more robust. To devise the worst-case scenario, we do not consider unknown traffic that Abacus would reject by design (e.g., such as single-flow client-server traffic, see Appendix A for further details), but instead take into account only the subset of traffic that Abacus could actually misclassify (i.e., accepting unknown traffic as P2P-TV and generating thus a false alarm), which constitutes merely the remaining 28% of the CAMPUS and 15% of the ISP traffic traces.

As reported in Tab. 3, we consider both the aggregated UDP traffic volume (that Abacus can misclassify) produced by all applications in the CAMPUS and ISP traces, as well as relevant UDP traffic subsets, representative of both P2P and client-server applications. The rationale of this choice is that we want to test whether false-positive classification is more likely to arise when considering P2P applications or traditional client-server services. Specifically, we consider Skype and eDonkey traffic as examples of voice and file-sharing P2P applications, and DNS as an example of traditional client-server service. To reliably identify eDonkey, we develop and implement a DPI classifier, based on [29, 30], while we classify Skype with out previous work [15], and rely on Tshark DPI protocol inspection capabilities to isolate DNS traffic.

4. Experimental Results

This section reports the results of our experimental campaign. **We start by considering signatures that are defined on the number of packets exchanged, providing first some base-line results in a general enough scenario. Then we investigate the *signature portability*, across different space, time and network conditions: this is done to assess if a classifier trained with signatures gathered under a given set of conditions, is able to correctly identify traffic generated in completely different settings (e.g., different ISPs, access technologies, networks conditions, different TV channels, different times, etc.).** Classification performance is expressed in terms of

- the amount of True Positive (TP) classification, i.e., number of tests for which the classifier identifies the correct P2P-TV application given that the sample was

Table 4: P2P-TV Classification Performance: Confusion Matrix of Testbed and Real Traces (Signatures)

	PPLive	TVAnts	SopCast	Joost	Unk
PPLive	**81.66**	0.58	9.55	2.32	5.90
TVAnts	0.49	**98.51**	0.18	0.77	0.04
SopCast	3.76	0.11	**89.62**	0.32	6.19
Joost	2.84	0.55	0.28	**89.47**	6.86

	PPLive	TVAnts	SopCast	Joost	TNR
CAMPUS	2.42	2.23	0.01	0.02	**95.3**
ISP	0.66	0.13	0.43	0.10	**98.7**

actually of that P2P-TV class. TPs are counted considering the testbed traces.

- the amount of False Positive (FP) classification, i.e., number of tests for which the classifier identifies the sample as any of the P2P-TV application, despite the sample was not actually of any P2P-TV class. FPs are counted considering the real traces.

Similarly, True Negatives (TN) are tests in which the classifier correctly rejects (i.e., does not classify it as P2P-TV) a sample which was indeed not generated by a P2P-TV application. Finally, False Negatives (FN) are instead tests for which a sample of a P2P-TV application was misclassified (i.e., rejected or classified as another P2P-TV application).

TP (and FP) results are usually normalized with respect to the total positive (and negative) samples. The TP-Rate (TPR) and the FP-Rate (FPR) are defined as follows:

$$TPR = \frac{TP}{TP+FN}, \quad FPR = \frac{FP}{FP+TN},$$
$$FNR = \frac{FN}{TP+FN}, \quad TNR = \frac{TN}{FP+TN}.$$

4.1. Baseline results

In this first set of experiments, we report results considering the following parameters: for each application, the training set includes samples extracted considering 2 peers at random from each group of $N = 7$ networks taking part to the experiment. From all signatures they generate, 4000 signatures are randomly extracted to define the training set, which corresponds to about 17% of all signatures. Experiments are then repeated 10 times, randomly changing the training set and so the validation set at each run. Finally, average classification results are computed. We consider signatures generated using $\Delta T = 5$s intervals. Classification are performed using SVM with a Gaussian kernel and exponential bins $B_n = 8$, with a rejection threshold $R = 0.5$. Parameters sensitivity and optimization is discussed later in the remaining part of this section.

The top part of Tab. 4 reports the classification performance of Abacus **on the testbed traces** adopting a "confusion matrix" representation. Each row considers traffic of a given application, and each column reports classification results. Values on

the diagonal correspond to True Positives (highlighted in bold), whereas elements outside the diagonal correspond to False Negatives, which accounts for both misclassified samples, and rejected samples **(which fall in the last column Unk, "Unknown")**. Performance are expressed for the time being in terms of signatures (i.e., groups of packets received during a ΔT interval) but we will also consider classification performance in terms of packets, bytes and peers later on. It can be seen that, in the worst case, about 81% of individual signatures are correctly classified. The most difficult application to identify appears to be PPLive, which is confused with SopCast (9.55%) or Joost (2.32%). Other applications show higher TPR, with TVAnts showing almost perfect match. On average, about 4.5% of P2P-TV signatures are rejected, thus being labeled as Unknown.

Bottom part of Tab. 4 reports results considering the real traces dataset. Since no P2P-TV traffic is present in this dataset, True Negative Ratio (TNR) is the main index to be considered (boldface, rightmost table column). Results show that the rejection criterion adopted is very robust, so that less than 5% of signatures samples are misclassified in the *worst case*. Recall that this 5% misclassification percentage refers to the about 15% of the traffic that Abacus could misclassify **(see Appendix A for further details) so that the overall TNR is actually much higher: namely, considering all the UDP traffic traces of the real networks, more than 99% of the signatures do not raise any false alarm**. Left part of the Table details the breakdown of False Positives: PPLive and TVAnts are the cause of most misclassification, while Joost practically causes no False Positives.

4.2. Signatures Portability

We now evaluate *network portability* of Abacus signatures. The objective is to answer the question: how generic is a training performed considering traces collected in a network? Our testbed dataset is different enough to see what happens when, for example, the classifier is trained considering a trace collected in a University Campus network, and then used in a totally different network, like a ADSL scenario. Moreover, both the access types and the channel popularity could impact the accuracy of the training set, which we deal with in the following. Besides, we are interested in testing how often the signatures have to be redefined, considering P2P-TV traces gathered in different years. Finally, we also test how robust the classifier is in presence of high packet loss or limited bandwidth.

For the sake of simplicity, we consider only the testbed traces and we no longer apply the rejection criterion. Results are summarized in Tab. 5: the first column reports the experiment label, the second column (Train) states which training set was used, while the third column (Test) reports the dataset using for the classification process. TPR for each application are reported in the subsequent columns. To ease the comparison, the first row (labeled *Ref*) reports the baseline results: notice that TPR is slightly higher with respect to Tab. 4, since we do not apply the rejection criterion.

4.2.1. Portability across Network Sites (NS)

In the first scenario, we consider traffic captured from PCs running at different institutions, i.e., in different Countries, networks, etc. (see Tab. 1). We start by considering peers that are all placed in corporate or campus networks, with high-bandwidth

connections to the Internet. There are 7 of such sites. For each application, we select 4 sites, and used traffic collected there for the training. Then, traces collected in the remaining 3 networks are classified to evaluate TPR. To gather robust results, we consider every possible combination $\binom{7}{4} = 35$ of training and validation subsets. For each combination, 3 tests are performed with different random training samples.

Results are reported in the raw labeled as *NS* in Tab. 5, which shows that signatures are network-portable under homogeneous settings: indeed, the largest performance drop is 4%, which corresponds to the PPLive case.

4.2.2. Portability across Access Technologies (AT)

We now test to what extent signatures are portable across different access technologies, e.g., ADSL versus High Bandwidth (HB) access. As noted in [5], nodes with high-bandwidth access can act as "amplifiers", providing content to possibly several peers; conversely, ADSL peers may only act as "forwarder" due to the limited uplink capacity. Though we consider only the downlink traffic, such different behaviors can impact the Abacus signatures, e.g., due to a different fraction of signaling packets a peer receives. For example, an amplifier peer can receive many small sized acknowledgments, while a low-capacity peer mainly receives large packets containing video data. We therefore split the testbed dataset into two parts: the first contains traces collected from all High Bandwidth PCs, while the second contains ADSL PCs. Three tests are performed: (i) classifying ADSL traces using ADSL training set, (ii) classifying ADSL traces using the HB training set and (iii) classifying HB traces using ADSL training set. Each test has been repeated 10 times, and average results are reported.

Results are reported in rows labeled *AT* in Tab. 5. Overall, Abacus signatures confirm their portability even across different access networks: for TVAnts, SopCast and Joost, results are modestly impacted by train/test combination (being 8% of reduced TPR the worst case). In case of PPLive, the TPR drops to 58% when HB training is used to classify ADSL traffic. This is likely due to the fact that PPLive is very aggressive in exploiting HB peers upload capacity, so that the number of peers sending acknowledgments shifts the signature toward low bins, i.e., few acknowledgment packets are received from a given peer. ADSL peers, on the contrary, contribute with little upload bandwidth, so that the incoming traffic is mainly due to video chunks received as trains of packets, i.e., groups of large data packets that are received from contributing peers.

4.2.3. Portability across Channel Popularity (CP)

We now consider what is the impact of channels with different popularity. Channel popularity indeed may significantly influence the P2P-TV application behavior: for example, while in popular channels a large number of peers are available, for unpopular channels only a few peers can be used to exchange the video content. We performed a second experiment considering a very popular (POP) channel using PPLive. We selected PPLive since it is the P2P-TV application for which Abacus showed the worst performance so far. The total number of peers observed during this experiment was larger than 200000, while in the previous dataset less than 56000 peers were observed. We refer to this dataset as a unpopular channel (UNP). As before, we evaluate the portability over all combination of train/test sets, repeating the experiments 10 times.

Table 5: Signature portability: TPR evaluation

	Train	Test	PP	TV	SO	JO
Ref	ALL	ALL	**84.84**	**98.51**	**92.63**	**91.50**
NS	4/7	3/7	78.90	97.61	90.30	88.61
AT	ADSL	ADSL	83.48	97.86	95.61	91.36
	ADSL	HB	79.63	93.73	87.30	90.61
	HB	ADSL	58.28	98.15	93.70	81.55
CP	POP	POP	95.88	-	-	-
	POP	UNP	48.59	-	-	-
	UNP	POP	94.79	-	-	-
TI	2008	2006	18.81	98.44	51.06	-
EI	HB	Bw	91.14	76.80	75.76	-
	HB	Delay	88.19	84.62	77.80	-
	HB	Loss	75.22	91.77	84.31	-

Results are reported in the rows labeled *CP* in Tab. 5. First, PPLive classification performance improves when it comes to the classification of popular channels (i.e., TPR in POP/POP and UNP/POP cases is about 95% versus the about 85% of the UNP/UNP case used as reference). Nonetheless, we observe that the classification of UNP dataset when training has been done considering the POP dataset leads to poor performance (TPR drops to less than 50%). This partly limits the portability across channels. A simple solution might consist in building a training set containing a mixture of signatures from both traces, which raises the TPR again to about 85%. This result suggests that channel popularity should be explicitly taken into account when building the training set, by including samples that are representative of different channel popularity.

4.2.4. Portability over Time (TI)

We now focus on the signatures portability over different periods of time. From a practical point of view, this allows to know how often classifier should be retrained. We resort to the traffic traces used in [26], that authors kindly made available to the scientific community. Traces of [26] were collected in July 2006 during the Fifa World Cup: the study focused on the same applications we examined in this article, with the exception of Joost which was not available at that time. Overall, the time-portability measurements account for 14 hours of video, $14M$ packets and 2.3 Gbytes of data. We classify this old dataset using the Abacus classifier trained with the dataset collected in 2008. (same training set of Sec. 4.1). Notice that the network environment was also different, so that we are *jointly* evaluating time and network portability. Results are reported in the row labeled *TI* of Tab. 5, which shows that TVAnts is correctly classified, SopCast has a TPR of 51%, while PPLive is almost completely misclassified. This suggests that some applications changed drastically their behavior from July 2006 to March 2008. Notice that TVAnts was at version 1.0.58 in [26] and it is now at 1.0.59, which likely means that little changes have been implemented. On the other hand, SopCast moved from version 0.9 to version 3.0.3 PPLive from 1.1.0 to 1.9.15, hinting

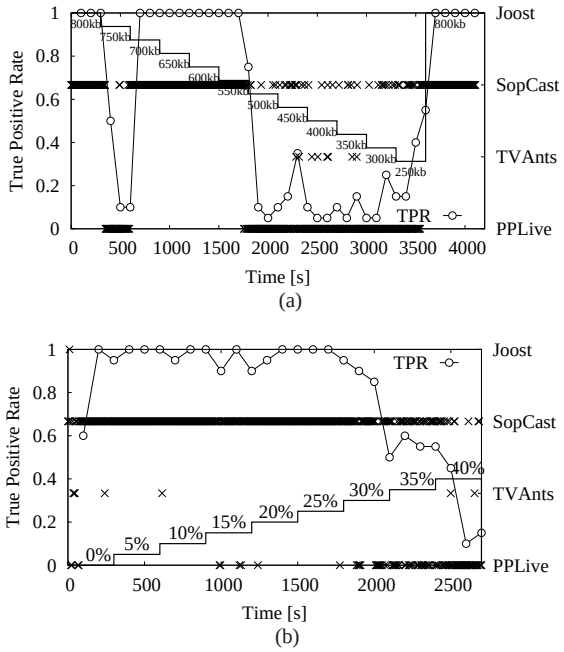

Figure 3: Portability over Emulated Impairment: example of temporal evolution of SopCast classification for decreasing bottleneck bandwidth (a) and increasing packet loss rate (b)

to clearly more drastic changes **as explained in [27, 31]**. Thus, in case applications do not change their internal algorithms, the Abacus signatures are extremely portable across time –even across years– as we see in the case of TVAnts. On the other hand, if an application implements new algorithms which result in new behavior, then Abacus should undergo a new training phase. However, similar considerations are valid for any kind of classifier, from port-based ones, to DPI or behavioral classifiers. In fact, whenever the features change, all classifiers must be trained again (e.g. by changing the port number, updating the DPI signature or re-training the behavioral/statistical features).

4.2.5. Portability over Emulated Impairments (EI)

As a final case, we consider whether Abacus signatures are portable across different network conditions. We consider the traces gathered in an active testbed [32], where changing network conditions were artificially enforced. In particular, in these experiments, a Linux router was used to emulate some network conditions: (i) bandwidth, (ii) delay and (iii) packet losses were imposed on the downlink path to the PC running the P2P-TV application. We refer the reader to [32] for a complete description of the testbed; here, we only point out that impairments range from mild to very tough conditions (e.g., 200 Kbps of available downlink bandwidth, delay up to 2 s and packet losses up to 40%). Traces gathered in this testbed are classified considering the HB

training set, and results are reported in the last lines of Tab. 5 labeled *EI*. Even in these extreme conditions, Abacus still exhibits very high TPR, which can still exceed 90% for some applications, with a worst case of about 75%. Reported results are averaged over all the time varying conditions, including very distorted scenarios. Classification results are differently impaired by different network conditions. For example, PPLive is mostly affected by loss increase, while TVAnts classification results are more sensitive to bandwidth change. SopCast results are mostly affected by bandwidth and delay changes.

Interestingly, results improve when considering PPLive classification in the case of bandwidth limitations. While this seems counter intuitive, it can be explained considering that most False Negatives obtained from other applications are actually misclassified as PPLive. This suggests that PPLive signatures are more variable and spread out, avoiding FN classification for PPLive but possibly causing more FP classification for other applications.

As an example, Fig. 3 reports the time evolution of two different experiments of SopCast classification, considering a scenario in which the available bandwidth is decreasing (top plot), or the packet loss rate is increasing (bottom plot). Every 5 minutes network conditions are artificially worsened by either reducing the available bandwidth by 50 Kbps, or by increasing the packet loss rate by 5%. The resulting impairment profile is reported in the picture.

Fig. 3 plots individual classification decisions, taken each $\Delta T = 5$s: these are represented with crosses, referring to the right y-axis, and allow to see when and how the application has been eventually misclassified. The picture also reports the True Positive Rate, evaluated over 20 consecutive signatures (i.e., 100 seconds), represented as a continuous dotted line referring to the left y-axis. Considering top plot, which refers to bandwidth limited scenario, it can be seen that as soon as the bottleneck bandwidth kicks in, SopCast is misclassified as PPLive during a brief period, possibly hinting to a sudden reaction of the application to the anomalous conditions. Then, SopCast is correctly classified until the available bandwidth drops too low: afterward, SopCast TPR drops quickly, being most of the time misclassified as PPLive and seldom with TVAnts. At the end of the experiment, when the bottleneck bandwidth is removed, SopCast is again correctly classified. Similar considerations hold for the loss scenario depicted in the bottom plot of Fig. 3, in which samples are misclassified only when loss rate exceeds 30%.

5. Sensitivity Analysis

After evaluating the effect of external conditions on the classifier performance, in this section we rather focus on its internals. In fact we present the results of the experiment carried on to investigate the *sensitivity* of the classification to parameter changes, so to select the settings which guarantee the best performance.

5.1. Impact of the Rejection Threshold R

Irrespectively of the precise distance metric used in the rejection criterion (whose mathematical details are reported in Appendix A), the selection of the

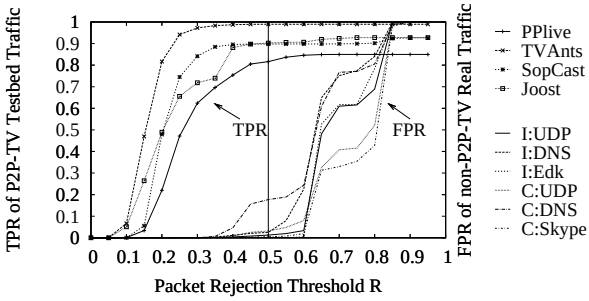

Figure 4: TPR and FPR as a function of the rejection threshold R evaluated on packet feature.

rejection threshold R is guided by the following tradeoff: R should be large to maximize the TPR (i.e., avoid classifying P2P-TV as Unknown), while R should be small to minimize the FPR (i.e., avoid classifying unknown traffic as P2P-TV).

We evaluate the TPR and FPR as a function of R in Fig. 4, where a solid vertical line at $R = 0.5$ represents the threshold used so far. It can be seen that TPR of P2P-TV applications quickly saturates to an asymptotic value for $R \geq 0.5$. Conversely, the FPR of non-P2P-TV traffic increases only for large values of R, and for low values of $R \leq 0.5$ almost no false alarm is raised. Among the various traffic, only DNS traffic is sometimes misclassified as P2P-TV traffic. This confirms $R = 0.5$ to be a good choice of the threshold.

5.2. Impact of Time Interval ΔT

The choice of the value of the ΔT parameter is driven by the following trade off. On the one hand, timely detection of P2P-TV traffic needs ΔT to be small. On the other hand, sufficiently large time intervals must be considered to estimate the signature. Moreover, to limit computational complexity and the generated amount of information, network monitoring entities (such as NetFlow [33] probes) typically operate on larger timescales.

Results are reported in Fig. 5, where the best results for each application is labeled with a star. As expected, medium-duration window (e.g., $\Delta T = 5\,\mathrm{s}$) yields higher TPR for most applications, providing a more timely classification as well. Smaller values of ΔT limit the estimation of the bin distribution, impairing classification accuracy. Interestingly, for large windows (e.g., $\Delta T = 60\,\mathrm{s}$) the discriminative power of the Abacus signatures only mildly degrades for three out of four applications. Only for PPLive we observe a decrease of 20% for the TPR, which is due manly to the rejection criterion being too aggressive and discarding correct classifications, suggesting that for longer ΔT the rejection criterion should be more carefully tuned.

5.3. Impact of Training Set Size

We now assess the classification sensitivity to variations on the training set *size*, i.e., the impact of the number of samples that form the training set. Indeed, the training set should be large enough to be representative of the application behavior under a

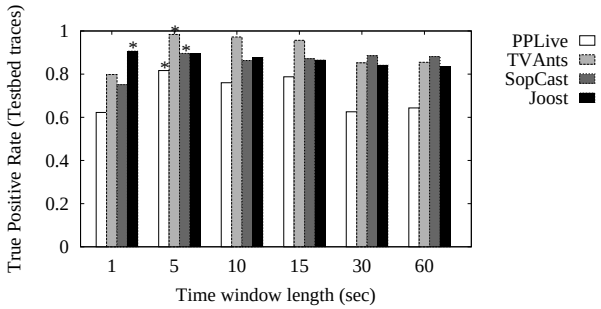

Figure 5: P2P-TV TPR for different values of the time interval ΔT. Best-case for each application is labeled with a star $*$ sign.

large range of conditions. On the other hand, the SVM training and classification computational costs benefit of a smaller set. Moreover, a too large training set could result in the well-known phenomenon of *over-fitting*, resulting in poor classification performance. Fig. 6-(a) reports the TPR for each application, as a function of the number of signatures used in the training phase per each application. For each value of the training set size, we run 10 independent experiments over which results are averaged. The bottom x-axis reports the number of signatures used for training on a logarithmic scale, while the upper x-axis reports the percentage of training samples versus the total testbed dataset. Training set size extends up to 4000 signatures per application, which corresponds to the 17% early used early used in Sec. 4.

Results show that no over-fitting phenomenon is experienced, since the TPR increases with the increase of the training set size. Best results are obtained considering 4000 signatures per application, which validates the choice made in previous section. Notice that even by drastically decreasing the training set size to about 300 signatures per application, the corresponding decrease in TPR is only modest, e.g., 8% in the worst case of PPLive, while TVAnts shows excellent results even with an extremely reduced training set. This interesting performance is the result of both the discriminative power of SVM, and the descriptive expressiveness of Abacus signatures. Clearly, a better characterization of each application behavior is achieved including more signatures, as reflected by the improved performance.

5.4. Impact of Training Set Diversity

We now fix the training set size and focus on the training set *diversity*, i.e., the number of different peers from which signatures are selected. Our aim is to roughly assess whether it is sufficient to observe a single peer in a given network to gather an adequate description of the application behavior in that network, or whether the observation of several peers is necessary. To answer this question, we fix the overall training set size to 4000 signatures per application and vary the number of peers selected as reference in each network (see Tab. 1 for details on the number of peers). Each experiment is repeated 10 times to collect average results. Fig. 6-(b) shows the TPR obtained considering a reference set of *one*, *two* or *all* peers for each network in the testbed.

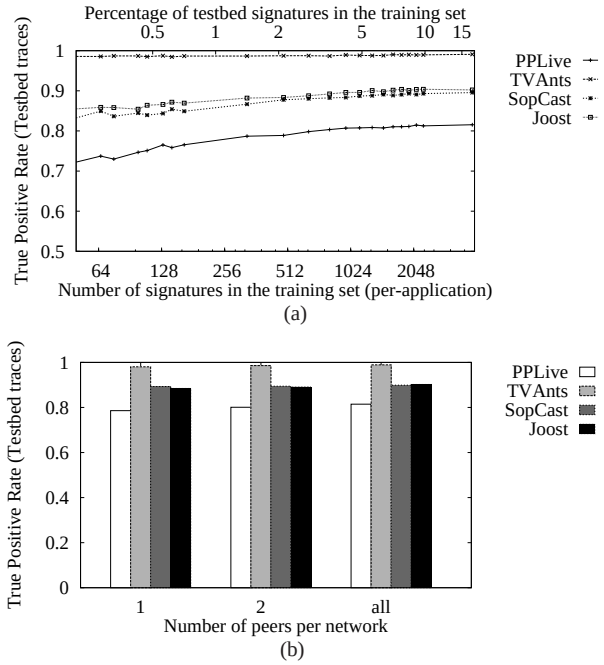

Figure 6: Impact of size (a) and diversity (b) of the training set

Results show that the increase of the number of peers only provides a very limited gain on the classification performance. From a practical perspective, this is a very desirable property: even a single trace is sufficient to build expressive signatures.

5.5. Impact of SVM Kernel and Binning Strategy

Since the core of our classification framework exploits SVM, all parameters that are susceptible of affecting its performance need to be investigated as well. Therefore, we focus on two main choices concerning SVM: (i) the *kernel* function and (ii) *binning* strategy.

The kernel function is used to map the training points to an hyper-space where they can be separated by hyper-planes. The SVM literature is very rich of kernel functions, which are more or less indicated for different kinds of data. In our study we evaluate three well-known kernels: the general-purpose *gaussian* kernel (K_G), the *linear* kernel (K_L) and the Bhattacharyya kernel (K_B). The linear kernel (3) is simply the dot product of two feature vectors, while the Bhattacharyya kernel (4) can be obtained by substituting each features with its square root [34]. **As we will explain in Appendix A, we use the Bhattacharyya distance [35] between probability mass functions as a core tool for the rejection criterion, as it is an extremely valuable metric to quantify the separability of two classes: a natural question is therefore whether the kernel function (4) can be helpful to better separate the different applications also from the SVM standpoint.**

Table 6: Classification performance and cost for different binning strategies, SVM kernels. In bold the best results.

		Recall (TPR)				Support Vectors (SV)				
Bins	Kernel	PPLive	TVAnts	SopCast	Joost	PPLive	TVAnts	SopCast	Joost	Total
Exponential	Gaussian	**81.66**	98.51	**89.62**	**89.47**	**1015**	**106**	**845**	**415**	**2381**
	Bhattacharyya	77.73	98.52	88.58	87.99	1759	110	1185	798	3852
	Linear	73.44	**98.54**	88.55	87.42	2062	219	1348	956	4585
Constant	Gaussian	**67.12**	**97.86**	**89.76**	**69.66**	**853**	**81**	**654**	**635**	**2223**
	Bhattacharyya	65.27	97.14	89.58	68.27	1215	113	902	755	2985
	Linear	64.90	97.70	89.45	68.88	1382	316	911	1091	3700

$$K_G(\underline{x}_i, \underline{x}_j) = e^{-\gamma||\underline{x}_i - \underline{x}_y||^2} \quad (2)$$

$$K_L(\underline{x}_i, \underline{x}_j) = \underline{x}_i \cdot \underline{x}_j \quad (3)$$

$$K_B(\underline{x}_i, \underline{x}_j) = \sqrt{\underline{x}_i \cdot \underline{x}_j} \quad (4)$$

As far as *bin distribution* is concerned, we use either $B_{exp} = 9$ exponential-width bins (base 2), or $B_{fix} = 255$ constant-width bins (1-packet steps), both spanning over the $[0, 255]$ packets range. Recall that the number of bins impacts both memory requirement and computational complexity, so that exponential binning should be preferred in case of comparable classification performance.

Results are shown in Tab. 6, which reports classification results in terms of the TPR of P2P-TV applications and the number of Support Vectors (SV) of the trained model. The latter is a measure of the classification computational cost, since the number of operations that has to be performed to classify each signature grows linearly with the number of SVs. The cost of the training phase is not considered, since it is an offline operation rarely performed. Notice also that, due to the type of operations in (2), (3) and (4), the selected kernel has impact on the computational cost – with the Linear kernel being light-weighted, the Gaussian kernel the most expensive and the Bhattacharyya kernel in between the other two.

Tab. 6 collects results highlighting the best choices using bold font. Irrespectively of the binning strategy, the Gaussian kernel yields consistently better results for both TPR and number of SVs. An important decrease in the performance is observed when considering constant binning, where the TPR for PPLive and Joost falls below the 70%. This is mostly due to the rejection criterion, which wrongly identifies as unknown a conspicuous number of signatures: indeed, the Bhattacharyya distance is less effective with this longer signature containing many zero values, which result in bigger distances from the class center. Results obtained with the Bhattacharyya kernel are almost equal to the linear kernel, with the advantage that the number of SV is smaller. Finally it must be noted that TVAnts requires a very small number of SV to obtain very good performance, irrespectively of the binning and kernel choice. In contrast PPLive, the most difficult application to classify, requires a number of SV that is ten times the number of TVAnts for its best choice of binning and kernel.

With respect to the bin distribution choice, the use of exponential binning reduces the memory consumption and the number of operations to be performed by B_{fix}/B_{exp},

Table 7: Extended Abacus Signatures: Confusion Matrix of P2P-TV Application

	Signatures: Confusion Matrix				
	PPLive	TVAnts	SopCast	Joost	Unk
PPLive	**95.42**	0.22	1.86	0.36	2.14
TVAnts	0.06	**99.84**	0.10	0.00	0.00
SopCast	0.98	0.15	**97.55**	0.03	1.29
Joost	0.21	0.01	0.01	**94.97**	4.80

Table 8: Extended Abacus Signatures: Classification Results per Signature, Packets, Bytes and End-Point

	Signatures			Packets			Bytes			Peer	
	TP	Mis	Unk	TP	Mis	Unk	TP	Mis	Unk	TP	Unk (n)
PPLive	95.42	2.44	2.14	98.11	1.60	0.29	98.32	1.54	0.14	100.0	0.0 (0)
TVAnts	99.84	0.16	0.00	99.77	0.23	0.00	99.82	0.17	0.01	100.0	0.0 (0)
SopCast	97.55	1.17	1.29	99.18	0.78	0.04	98.96	0.98	0.06	97.06	2.94 (1)
Joost	94.97	0.23	4.80	99.50	0.25	0.25	99.62	0.23	0.15	93.33	6.67 (2)

i.e., almost a factor of 30. For example, assuming 1 GBytes of RAM, $B_{exp} = 9$ exponential bins would allow to compute about $15M$ end-points considering 64bit floating point notation. With the same amount of memory, using B_{fix} linearly spaced bins allows to track roughly $0.5M$ end-points. Considering CPU time, a server equipped with an Intel Xeon E5345 clocked at 2.33GHz reaches 3000 classifications per second using exponentially distributed bins. Given that a signature is produced every $\Delta T = 5$ s, about 15000 end-points could be classified in real-time even by our non optimized Python code. Considering linearly distributed bins, only 126 classifications per second are computed, allowing to classify no more than 630 end-points.

Results from this analysis reinforces the selection of an exponential binning strategy in combination with the Gaussian kernel.

6. Improving the Accuracy: Extending the Signature

In this section we augment the Abacus signature to include not only the number of packets received by each peer, but also the number of received bytes. Following the same procedure, we consider a ΔT time interval in which the endpoint $\mathcal{P}$ receives $b_1, \ldots, b_K$ bytes from K peers. $B_b + 1$ exponential-width classes are identified, according to the number of bytes received from $\mathcal{P}$, and counting the occurrences of each class in B_i. The byte-wise signature $\underline{b}$ is then obtained by normalizing the count B_i over the total number of received bytes. The tuple $\underline{b}$ is a pmf, whose component b_i can be interpreted as the probability that an arbitrary peer sends between $(2^{i-1}, 2^i]$ bytes to $\mathcal{P}$. For byte-wise signatures, we set the number of bins to $B_b = 14$.

We define the application signature by concatenating the packet-wise $\underline{n}$ and byte-wise $\underline{b}$ signatures in a single vector $\underline{a} = (\underline{n}, \underline{b})$. Since the extended signature $\underline{a} = (\underline{n}, \underline{b})$ is composed of two parts, we can define two rejection thresholds, considering $\underline{n}$ or $\underline{b}$ only. We therefore report in Fig. 7 the TPR and FPR as a function of the rejection threshold R applied to byte signatures. **Contrasting Fig. 7 with Fig. 4, we observe that the bytes signature exhibit a better behavior also with respect to the rejection**

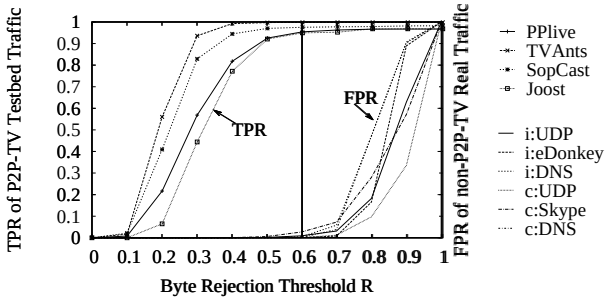

Figure 7: TPR and FPR as a function of the rejection threshold R evaluated on bytes feature.

criterion. In fact, on the one hand, TPR curves saturate much faster, which means that points of the same application are better clustered; on the other hand, the FPR curves start showing up for larger values of the threshold, which is even better because we can safely adopt a larger value for R, obtaining at the same time lower FPR and higher TPR. Given these considerations, we decided to apply the rejection criterion only to the byte-wise signatures $\underline{b}$ with a threshold $R = 0.6$.

We perform the classification based on the extended signature $\underline{a}$ with a byte-wise rejection criterion and a rejection threshold $R = 0.6$. Results reported in Tab. 7 are gathered for $\Delta T = 5$s, with a training set of 4000 signatures extracted at random. Compared to previous results of Tab. 4, the extended signature leads to significant performance improvement, so that TPR is now about 95% in the worst case, and misclassification probability is reduced to few percentage points.

To better appreciate results, Tab. 8 reports performance considering correctly classified *packets*, *bytes* and *peers*. Packet-wise and byte-wise performance can be directly gathered by taking into account the number of packets and bytes carried by each signature; the peer classification is instead evaluated considering a *majority* criterion, so that a peer is classified as running application X if the majority of time such peer samples have been classified as X. Tab. 8 reports the percentage of correct classification (TP), of misclassification (Mis, corresponding to the sum by rows of non diagonal values in the confusion matrix) and rejection (Unk) for all the above metrics. Notice that FN=Mis+Unk. Interestingly, performance improves when the number of correctly classified *packets* and *bytes* is considered, suggesting that misclassification occurs when signatures carry few data, e.g., when the application is possibly malfunctioning. In case of *peer* classification, reliability of end-points identification increases as well. Only 3 hosts are classified as not running any P2P-TV application, and notably there is no misclassification. Investigating further, we found that rejected cases correspond to peers that received a small amount of traffic, and, thus, possibly were not playing any video.

We now assess the benefits of the extended signatures on the effectiveness of the rejection criterion. We again consider real traffic collected from operational networks, considering only the worst-case traffic portion for which possible false positives may be triggered. Tab. 9 reports results referring to the extended signatures, showing the

Table 9: Non P2P-TV Traffic in Campus and ISP traces: False Positive Ratio FPR and FP Confusion Matrix. To perform a worst-case analysis, only end-points that can lead to false positive classification are considered (28% of the CAMPUS and 15% of the ISP overall traffic volume)

			FP Confusion Matrix			
Traffic		FPR	PP	TV	SO	JO
C	UDP	**2.70**	0.57	1.00	1.13	-
	Skype	0.04	-	0.03	0.01	-
	DNS	0.17	0.02	0.10	0.05	-
I	UDP	**0.90**	0.61	0.14	0.15	-
	eDonkey	0.09	0.02	0.04	0.03	-
	DNS	0.44	0.03	0.33	0.08	-

false positive rate (FPR) and the breakdown of false alarms between the different P2P-TV applications. First, notice that the number of false alarms is very limited, being only 2.7% in the worst-case traffic subset: however, if all UDP traffic is considered, FPR drops to less than 0.1%. This negligible number of false alarms confirms the reliability of the classification engine. Moreover, false positive rate is low for individual applications too: indeed, it is very rare that eDonkey or Skype traffic is confused with any P2P-TV application (0.09% and 0.04% of false positives).

7. Related Work

Despite Internet traffic classification is a relatively recent research field, there is already a large number of works on the topic [8, 9, 10, 12, 13, 14, 16, 17, 40, 18, 41, 19, 51]. In this section we will provide only a brief overview of the classifiers of this vast body of literature in order to place our work among the others. For more details and a complete organic description we refer to the survey of Nguyen et al. [36] and to the quantitative comparison of Kim et al. [37].

In the first days of the Internet, identifying the generating application of some network traffic was not a issue: protocols were assigned well-known transport-layer ports by IANA, so a simple look up of such value in the packet header was enough to achieve a good classification. Unfortunately *port-based* classification has become unreliable [9] as protocols use non-standard ports or, worse, hide themselves behind other protocols' ports. Hence, researchers had to come up with novel ways to solve this problem.

***Payload-based* classifiers [8, 9, 10, 11] inspect the content of packets well beyond the transport layer headers, looking for distinctive protocol signatures (either keywords or matching regular expressions). This family of classifiers has long provided extremely accurate results [9] and has been implemented in several commercial software products. Nevertheless, although research have designed automatic ways to extract signature from packets [8, 10, 11], payload based classifier are extremely greed in terms of computational resource and fail by design with encrypted traffic.**

***Statistical-based* classification [12, 13, 14, 15, 17, 16] is based on the rationale that, being the nature of the services extremely diverse (e.g., Web vs VoIP), so will**

be the corresponding traffic generated (e.g., short packets bursts of full-data packets vs long, steady throughput flows composed of small-packets). Such classifier exploits several flow-level measurements to characterize the traffic of the different applications [12, 13, 14], on which they usually apply data mining techniques. Unsupervised clustering of traffic flows [12] does not require training and allows to group flows with similar features together, possibly identifying novel unexpected behaviors. Supervised machine learning techniques, instead, need to be trained with know traffic, but are able to provide a precise labeling of the traffic. If all of these first works are focused on *post-mortem* traffic classification (i.e., after the flow ends), recently similar techniques have been applied to packet-level properties of network flows (e.g. size and direction of the very first packets), and are usually referred as *early* traffic classification [16, 17].

Finally, *behavioral-based* classification [18, 19, 51] aims at identifying the traffic generated a host by the sole examination of the generated traffic patterns (e.g., how many hosts are contacted, with which transport layer protocol, on how many different ports, etc.): the idea is that different applications generate different patterns. For instance, a P2P host will contact many different peers typically using a single port for each host, whereas a Web server will be contacted by different clients with multiple parallel connections. The combination of metrics able to capture such differences with machine learning techniques has yielded extremely promising results for the coarse-grained classification of hosts. This approach is very light-weight, as it does not require neither to inspect portions of the packet payload as in [8, 10], nor to perform operations on a per-packet basis as in [16, 17]. Moreover, given the current tendency toward flow-level monitors such as NetFlow [33], the possibility to operate on the sole basis of behavioral characteristics is a very desirable property for classification

Our work on Abacus, of which a preliminary version appeared in [20] fits in this last category. Compared to [20], this paper has some fundamental additions: first the signatures have been *extended* to include the counts of bytes besides the counts of packets, which greatly improved the overall accuracy; second, in this work we perform a detailed *sensitivity analysis* of classification performance as well as a thorough *portability assessment* of the signatures across different network conditions. Besides, we point out that recent results [38] suggest that the technique may be applicable to a larger class of P2P traffic in general.

We also point out that in [11] we already proposed a classifier that can successfully target P2P-TV traffic, named KISS. However, KISS uses a totally different approach to traffic classification and rather belongs to the family of payload-based techniques (which also means that it fails with completely encrypted traffic). As a matter of fact, KISS stochastically inspects packet payload to infer a statistical description of the protocol employed by an application by means of a measure of the randomness of group of bits of the payload itself (which constitute the KISS signatures). In [39] we compared Abacus with KISS, showing how KISS is able to classify a wider range of protocols (i.e., not only P2P-TV), though at the cost of an higher computational complexity.

We conclude this overview with an overall consideration on the applicability of the classifiers. With few exceptions such as [40], the wide majority of the classifi-

cation algorithms proposed in literature cannot be directly applied in the network core. Limitations can be either intrinsic to the *methodology* (e.g., behavioral classification typically focuses on endpoint [18] or end-hosts [41] activity), or tied to the *computational complexity* (e.g., DPI [8, 9, 10, 11] cannot cope with the tremendous amount of traffic in the network core), or to *state scalability* (e.g., flow-based classification [14, 12] requires to keep a prohibitive amount of per-flow state in the core), or to *path changes* (path instabilities or load balancing techniques can make early classifications techniques such as [16, 17] fail in the core). Although P2P classification in the core is possible with traffic dispersion graphs (TDGs)[42], we point out that TDGs only achieve coarse-grained classification as opposite to Abacus. At the same time, we point out that classifying traffic at the network ingress point is a reasonable choice for ISPs: indeed, traffic can be classified and tagged at the access (e.g., DiffServ IP TOS field, MPLS, etc.), on which basis a differential treatment can then be applied by a simple, stateless and scalable core (e.g., according to the class of application.).

Finally, for the sake of completeness, we mention a few references on other relevant works on P2P live-streaming system. As this body of literature is extremely vast, we focus on works that study existing and popular systems (i.e., that are interesting for ISPs to classify), and leave work that focus on the design of new systems out of scope. Indeed, after the first pioneering works [43, 44] presenting this innovative way of streaming content across thousands of hosts using swarm-like unstructured system (somehow inspired by Bittorrent [45]), the main reason of the research community interest on P2P-TV is clearly the success of commercial software as [1, 2]. With this regard, measurement of P2P-TV applications are the focus of [46, 47, 26, 48, 25, 32]. Specifically, [46] focuses on PPLive, [47] on UUSee, [48] on Zattoo, while [26, 25, 32] perform a comparison of several popular applications (the first considers PPlive, SopCast and TVAnts, the second adds Joost and the latter also adds TVUplayer).

8. Conclusions

This work proposed Abacus, a novel behavioral approach for fine-grained classification of P2P-TV applications. Our methodology relies only on the simple count of packets and bytes exchanged amongst peers during small time-windows. Our classification engine, which makes use of Support Vector Machines during the decision process, correctly classifies about 95% of packets, bytes and peers in the worst case. Moreover, the classification engine raises very few false alarms, well below 0.1% in the worst case. Such astonishing performance is the result, on the one hand, of the discriminative power of SVM, and, on the other hand, of the descriptive expressiveness of Abacus signatures.

A large set of experiments has been carried over to assess Abacus performance, both considering parameter sensitivity, and signature portability: results prove that the proposed approach is very robust to both. Training the Abacus classifier is simple, as signatures can be generated automatically using a very small number of traces. In terms of both memory requirements and computational complexity, Abacus is also very lightweight. Moreover, the fact that behavioral data used by Abacus are directly

available from commonly deployed NetFlow monitors makes it apt to be deployed in real network environments.

While Abacus opens the way for fine-grained classification engines working solely on behavioral data, it also raises a number of interesting points, which we plan to address in future work. First of all, while we proved Abacus to be extremely reliable on P2P-TV traffic, it would be interesting to investigate whether Abacus can classify applications of other P2P classes as well: encouraging results [38] suggest this to be possible, but further analysis on wider datasets is needed to confirm the finding.

Second, the behavioral statistics on which Abacus decisions are taken are extremely reliable when the classification is applied to *all* the traffic observed by an endpoint. Another open point concerns the performance of Abacus when the classification engine is moved from the access (e.g., DSLAM) deeper into the aggregation network (e.g., at the first or second IP router), where *not* all the traffic can be observed. In this case, although the observed traffic is only a *subset* of the whole endpoint traffic, nevertheless the breakdown of peers in low and high contributors bins, as measured by the Abacus signatures, may not change significantly in practice across subsets. Indeed, first consider that since Abacus signatures are normalized, changes in the raw traffic volume do not affect the signature definition, other than for quantization effect. Second, considering a partitioning in subsets that is due to, e.g., IP routing or flow-level load balancing, the peers in the subset would be independently sampled: thus, provided that enough peers are sampled by a subset, the resulting subset would contain both short flows (e.g., peer discovery and overlay maintenance) as well as long flows (e.g., data transmission in multiples of the chunk size) without any particular bias affecting their breakdown. As part of our future research, we aim at digging this issue further, by observing the same traffic from multiple vantage points in the network, and applying the classification to different subsets to verify to what extent the above considerations hold in practice.

Finally, the information required by Abacus relies on simple operations performed on the raw count of exchanged packets and bytes. While the Abacus signatures are extremely expressive, nevertheless they constitute a single choice within the extremely large spectrum of behavioral signatures (e.g., any combination of operations on flow-level data). To gather a more complete picture of fine-grained behavioral classification, we finally aim at systematically exploring the space of behavioral signatures, by building a large set of potentially expressive features and exhaustively investigating their classification performance.

Appendix A. Rejection Criterion

SVM is a powerful classification algorithm, but for our purpose of network traffic classification it presents one simple shortcoming. Recall that a SVM trained model is composed of two parts: first a mapping from the original features space to a multidimensional space; second a set of hyperplanes individuated by Support Vectors, which defines a *partition* of the target space into regions, each corresponding to a possible classification outcome. The problem is that, in this partitioned

space, a new point is always deemed to fall into a region, hence it will always be associated to one of the label represented in the training set. Unfortunately in traffic classification, we also need to deal with "other" traffic, generated by different application. To overcome this issue, we define a rejection criterion, whose aim is basically to recognize to traffic belonging to *none* of the target training classes.

Given that Abacus signatures are probability mass functions, we use a measurement index suitable to quantify distribution similarity. Given two pmfs, there exist several indexes to evaluate their degree of similarity. The Bhattacharyya distance (BD) [35] is a measure of *divergence* of two probability density (or mass) functions. Given two pmfs p and q over n discrete values, the Bhattacharyya distance $BD(p,q)$ is defined by:

$$BD(p,q) = \sqrt{1-B} \quad \text{where} \quad B = \sum_{i=1}^{n} \sqrt{p_i q_i} \tag{A.1}$$

Bhattacharyya distance, which is a particular case of the Chernoff distance, has several properties. First, it verifies the triangular inequality. Values of BD close to zero indicates strong similarity (if $p_i = q_i \ \forall i$, $B = 1$ and $BD = 0$) whereas values close to one indicates weak similarity. The Bhattacharyya coefficient $B \in [0,1]$ is the scalar product between the two vectors $p' = (\sqrt{p_1}, \ldots, \sqrt{p_n})$ and $q' = (\sqrt{q_1}, \ldots, \sqrt{q_n})$, which leads to a geometric interpretation of the coefficient B. In fact it can be seen as the cosine of the angle between p' and q'. The Bhattacharyya distance has been successfully applied in different contexts such as signal selection [49], or classification [50].

In our context, we use BD to measure the *separability* of two traffic classes. In particular, we *reject* the SVM label C of a signature $\underline{n}$ whenever the distance $BD(\underline{n}, E[\underline{n}(C)])$ exceeds a given threshold R, i.e., the sample will be labeled as *unknown*. $E[\underline{n}(C)]$ is the average signature computed on the training samples of application C. Notice that the average signature $E[\underline{n}(C)]$ identifies the center of the cluster formed by all training set signatures of application C. In other words, we accept SVM decision conditionally to the fact that the observed traffic signature $\underline{n}$ lies within a radius R from the center of the SVM training set for that class. The selection of the threshold value is simple but delicate, as it heavily influences the performance of the classification in terms of both True Positive Rate and False Positive Rate. In the paper we perform this process twice, for the simple as well as the extended signatures.

However, there exist some cases where no false alarm can be raised (i.e., non P2P-TV traffic will be always classify as unknown), which makes Abacus robust by design. Let us consider the case when traffic is received from one peer only. Then, the Abacus signature $\underline{n}$ is a vector containing a single 1 at the bin i^*. **In this case, the distance from the center $E[\underline{n}(C)]$ (C for short) of the cluster of an arbitrary application will be $BD(\underline{n}, C) = \sqrt{1-\sqrt{C_{i^*}}}$. Suppose we choose a threshold R, then we reject the classification if:**

$$BD(\underline{n}, C) = \sqrt{1-\sqrt{C_{i^*}}} < R$$

from which we can derive an acceptance condition on the value of C_{i^*}:

$$C_{i^*} < (1-R^2)^2$$

In case we set $R = 0.5$ as in Sec. 4, we have that a signature is rejected whenever its most likely bin C_{i^*} exceeds $(1 - 0.5^2)^2 = 0.5625$: notice from Fig. 1 this is never the case for any P2P-TV applications, whose most likely bins remain below 0.3. In other words, the criterion is robust with respect to P2P-TV applications (whose signatures are not rejected) and with client-server applications as well (since any signature containing a single bin has forcibly $C_{i^*} = 1$ and is thus rejected). Similarly, consider the case when traffic is received by only two peers: any such signature is a linear combination of two unit vectors and simple mathematical considerations can prove that it will be rejected too. Therefore, also in this case, the signature is always rejected (classified as "unknown") and no false alarm is raised. To summarize, the criterion rejects any peer contacting two or less other peers during a given time interval ΔT – which basically means that client-server traffic will never raise any false alarm, but will rather be correctly classified by the engine as "Unknown".

The robustness of the rejection criterion has allowed us to focus, throughout the paper, on a worst case analysis of the false alarm rate. Indeed, we recall that considering the Real Trace dataset, used to evaluate the amount of False Positive classification, the number of signature composed by less or equal to two peers is verified for a large fraction of the samples: 62% and 82% in CAMPUS and ISP respectively, to which a 72% and 85% of UDP volume corresponds. All this large amount of traffic *cannot be misclassified by design*, adding to the robustness of the classification framework. Therefore, our evaluation focused on the remaining signatures (i.e., with more than two peers), that could be classified by Abacus, so to gather conservative results.

Appendix B. Acknowledgments

This work was funded by EU under the FP7 Collaborative Project "Network-Aware P2P-TV Applications over Wise-Networks" (NAPAWINE). Authors wish to thank colleagues O. Fourmaux and T. Silverston for making their traces [26] available to the scientific community. Many thanks to the anonymous reviewers, whose invaluable help assisted us in greatly improving the quality of the paper.

[1] PPLive, http://www.pplive.com.

[2] SOPCast, http://www.sopcast.com.

[3] TVAnts, http://www.tvants.com.

[4] Joost, http://www.joost.com.

[5] X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross "A Measurement Study of a Large-Scale P2P IPTV System", *IEEE Transactions on Multimedia*, Dec. 2007

[6] "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014", `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html`, June 2010

[7] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini and D. Rossi, "Building a cooperative P2P-TV application over a Wise Network: the approach of the European FP-7 STREP NAPA-WINE",*IEEE Communication Magazine,* Vol. 64, No. 6, April 2008.

[8] S. Sen, O. Spatscheck, D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures", *13th International Conference on World Wide Web (WWW'04),* pp. 512-521, New York, NY, May 2004.

[9] AẆ. Moore, K. Papagiannaki, "Toward the Accurate Identification of Network Applications", *In Passive and Active Measurement (PAM'05),* Boston, MA, USA, March/April 2005

[10] J. Ma, K. Levchenko, C. Kreibich, S. Savage, G. M. Voelker, "Unexpected Means of Protocol Inference", *6th ACM SIGCOMM Internet Measurement Conference (IMC'06),* pp. 313-326, Rio de Janeiro, BR, October 2006.

[11] A.Finamore, M.Mellia, M.Meo, D.Rossi, "KISS: Stochastic Packet Inspection Classifier for UDP Traffic", *IEEE Transactions on Networking, to appear, 2010,*

[12] A. McGregor, M. Hall, P. Lorier, J. Brunskill, "Flow Clustering Using Machine Learning Techniques", *PAM'04*, Antibes Juan-les-Pins, Fr., pp. 205-214, April 2004.

[13] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, "Class-of-Service Mapping for QoS: a Statistical Signature-based Approach to IP Traffic Classification", *4th ACM SIGCOMM Internet Measurement Conference (IMC'04),* Taormina, IT, pp. 135-148, October 2004.

[14] A. W. Moore, D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques", *ACM SIGMETRICS '05*, Banff, Alberta, Canada, 2005

[15] D.Bonfiglio, M.Mellia, M.Meo, D.Rossi, P.Tofanelli, "Revealing Skype Traffic: when Randomness Plays with You", *ACM SIGCOMM'07,* Kyoto, Japan, August 2007.

[16] L. Bernaille, R. Teixeira, K. Salamatian, "Early Application Identification," *Conference on Future Networking Technologies (CoNEXT'06),* Lisboa, PT, December 2006.

[17] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, "Traffic Classification Through Simple Statistical Fingerprinting", *ACM Computer Communication Review,* Vol. 37, No. 1, pp.5-16, Jan 2007.

[18] T. Karagiannis, K. Papagiannaki, M. Faloutsos "BLINC: Multilevel Traffic Classification in the Dark", *ACM Communication Review,* Vol. 35, No. 4, pp. 229 - 240, 2005.

[19] K. Xu, Z. Zhang, S. Bhattacharyya, "Profiling Internet Backbone Traffic: Behavior Models and Applications", *ACM SIGCOMM'05,* Philadelphia, PA, pp. 169-180, August 2005.

[20] S. Valenti, D. Rossi, M. Meo, M.Mellia and P. Bermolen, “Accurate and Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets”, *In Traffic Measurement and Analysis (TMA) Workshop at IFIP Networking’09*, Aachen, Germany, May 2009.

[21] N. Cristianini, J. Shawe-Taylor, “An introduction to Support Vector Machines and Other Kernel-based Learning Methods”, *Cambridge University Press*, New York, NY, 1999.

[22] N. Williams, S. Zander, G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification”, *ACM Computer Communication Review*, Vol. 36, No. 5, pp.5-16, 2006

[23] C.C. Chang, C.J. Lin, “LIBSVM: A Library for Support Vector Machines”, available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[24] S. Valenti, D. Rossi, M. Meo, M. Mellia, P. Bermolen, “An Abacus for P2P-TV traffic classification”, In *IEEE INFOCOM 2009, Demo Session*, April 2009

[25] D. Ciullo, M.A. Garcia, A. Horvath, E. Leonardi, M. Mellia, D,Rossi, M Telek, P. Veglia, “Network Awareness of P2P Live Streaming Applications: A Measurement Study”, *IEEE Transaction on Multimedia*, Vol. 12, No. 1, pp. 54–63, Jan 2010.

[26] T. Silverston, O. Fourmaux “Measuring P2P IPTV Systems”, *In Proceedings of ACM NOSSDAV*, June 2007

[27] Y. Huang, T.Z.J. Fu, and D. Chiu, J.C.S Lui, and C. Huang, “Challenges, design and analysis of a large-scale P2P-VoD system”, *ACM Computer Communication Review*, Vol. 38, No. 4, pp.375-388, 2008

[28] M.Mellia, R.Lo Cigno, F.Neri, “Measuring IP and TCP behavior on edge nodes with Tstat”, *Computer Networks*, Vol.47, No.1, 2005 pp. 1-21, January 2005.

[29] “IPP2P home page”, `http://www.ipp2p.org/`.

[30] Y. Kulbak, D. Bickson, “The eMule protocol specification”, *Technical Report Leibniz Center TR-2005-03*, School of Computer Science and Engineering, The Hebrew University, 2005.

[31] G. Huang, “Experiences with PPLive,” *Keynote speech at ACM SIGCOMM’07 Workshop on P2P-TV*, Kyoto, Japan, August 2007.

[32] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, M. Meo, “P2P-TV systems under adverse network conditions: a measurement study”, *IEEE Infocom*, Rio de Janeiro, April 2009

[33] B. Claise, Ed. “Cisco Systems NetFlow Services Export Version 9” *IETF RFC 3954*, October 2004

[34] T. Jebara, R. Kondor, “Bhattacharyya and Expected Likelihood Kernels”, *In Proc. of Conference on Learning Theory (COLT’03)*, Washington D.C., USA, August 2003.

[35] A. Bhattacharyya, “On a Measure of Divergence Between Two Statistical Populations Defined by Probability Distributions”, *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[36] T. Nguyen, 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.

[37] H. Kim, KC Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K.Y. Lee, “Internet traffic classification demystified: Myths, caveats, and the best practices”, In Proc. of ACM CoNEXT ’08, 2008

[38] D. Rossi, S. Valenti, “Fine-grained traffic classification with Netflow data”, *In TRaffic Analysis and Classification (TRAC) Workshop at IWCMC 2010*, Caen, France, June 2010.

[39] A.Finamore, M. Meo, D. Rossi, S. Valenti, “Kiss to Abacus: a comparison of P2P-TV traffic classifiers”, In *Traffic Measurement and Analysis (TMA) Workshop at PAM’10*, Zurich, Switzerland, April 2010.

[40] J. Erman, A. Mahanti, M. F. Arlitt, C. L. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core”, *WWW 2007*, Banff, Canada, May 2007.

[41] T. Karagiannis, K. Papagiannaki, N. Taft, M. Faloutsos, “Profiling the End Host”, *In Passive and Active Measurement (PAM’07)*, Louvain-la-neuve, Belgium, Apr 2007.

[42] M. Iliofotou, H. Kim and P. Pappu and M. Faloutsos, M. Mitzenmacher, G. Varghese, “Graph-based P2P Traffic Classification at the Internet Backbone”, in *12th IEEE Global Internet Symposium (GI2009)*, Rio de Janeiro, Brazil, April 2009

[43] X. Zhang, J. Liu, B. Li, TSP Yum, “CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming”, In *Proc. of IEEE INFOCOM ’05*, Mars 2005

[44] X. Liao, H. Jin, Y. Liu, LM. Ni, D. Deng, “Anysee: Scalable live streaming service based on inter-overlay optimization”, In *Proc. of IEEE INFOCOM ’06*, Mars 2006

[45] B. Cohen, “Incentives build robustness in Bittorrent”, In *Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[46] X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross, “A Measurement Study of a Large-Scale P2P IPTV System,” *IEEE Transactions on Multimedia*, Vol.9, No.8, pp.1672-1687, Dec. 2007.

[47] C. Wu, B. Li and S. Zhao, "Exploring large-scale peer-to-peer live streaming topologies," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, Vol. 4, No. 3, 2008, pp. 1–23.

[48] H.Chang, S.Jamin and W.Wang, "Live streaming performance of the Zattoo network," *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC'09)*, Chicago, Illinois, Nov. 2009.

[49] T. Kailath, "The Divergence and Bhattacharyya Distance Measures in Signal Selection", *IEEE Transaction on Communication*, Vol. 15, pp. 52–60, 1967.

[50] K. Matusita, "A Distance and Related Statistics in Multivariate Analysis", *In Proc. of International Symposium on Multivariate Analysis, Academic Press P.R. Krishnaiah (ed.)*, pp. 187-200,

[51] R. Wang, Y. Liu, Y. Yang, X. Zhou, "Solving P2P Traffic Identification Problems Via Optimized Support Vector Machines", *IEEE/ACS International Conference on Computer Systems and Applications,(AICCSA '07)*, pag. 165-171, May 2007.