Accurate, Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets

Silvio Valenti[1], Dario Rossi[1], Michela Meo[2], Marco Mellia[2], and Paola Bermolen[1]

[1] TELECOM ParisTech, France
first.last@telecom-paristech.fr
[2] Politecnico di Torino, Italy
first.last@polito.it

Abstract. We present a novel methodology to accurately classify the traffic generated by P2P-TV applications, relying only on the count of packets they exchange with other peers during small time-windows. The rationale is that even a raw count of exchanged packets conveys a wealth of useful information concerning several implementation aspects of a P2P-TV application – such as network discovery and signaling activities, video content distribution and chunk size, etc. By validating our framework, which makes use of Support Vector Machines, on a large set of P2P-TV testbed traces, we show that it is actually possible to reliably discriminate among different applications by simply counting packets.

1 Introduction

The Internet proved to have an amazing capability of adapting to new services, migrating from the initial pure datagram paradigm to a real multi-service infrastructure. One of the most recent steps of this evolution is constituted by P2P-TV, i.e., large-scale real-time video-streaming services which exploit the peer-to-peer communication paradigm, and already count millions of users worldwide.

As such, the identification of P2P-TV applications is a topic of undoubted interest, which has not been addressed yet, despite the valuable effort already devoted to the task of traffic classification [1,2,3,4,5,6,7,8,9]. In this field *behavioral classification* [1,2] is a novel approach which aims at identifying the traffic generated by network hosts or end-points by the sole examination of their traffic patterns (e.g. number of hosts contacted, transport layer protocol employed, number of different ports used, etc.). This approach is very light-weight, as it requires neither the inspection of packet payload as in [3,4], nor operations on a per-packet basis as in [7,8]. However, so far, behavioral classification has been able only to discriminate broad application *classes* (e.g., interactive, P2P, Web, etc.) rather than different applications *within* the same class.

This work is the first to propose a *fine-grained* classification engine which only exploits behavioral characteristics – namely, the count of packets exchanged by peers during small time-windows. Our framework, which is tailored for P2P-TV applications such as PPLive, SopCast, TVAnts and Joost[1], makes use of the Support Vector Machines. We validate the engine by means of a large and diverse set of traces collected

[1] Since October 2008 Joost is no more using P2P to deliver video content.

over a pan-European testbed: experimental results show that it is possible to discriminate among different P2P-TV applications by simply counting packets – as true positive classification accounts to more than 81% of packets, bytes and peers in the *worst case*.

2 Classification Framework

The Rationale

Our aim is to classify P2P-TV end-points, identified by a network address and transport layer port pair $(IP, port)$. Typically, a P2P-TV application running on a given IP host multiplexes signaling and video traffic exchanged with other peers on a single port. We assume our engine to be situated at the *edge* of the network, where all the traffic exchanged by a given end-point transits. Furthermore, we restrict our attention to UDP traffic only, as it is the transport layer protocol preferred by P2P-TV applications.

Since UDP is a connectionless transport protocol, we cannot exploit any kind of flow semantic to perform the classification. As such, we rely solely on the count of packets a P2P-TV application exchanges with other peers during small time-windows. Indeed, we advocate that application *signatures* based on the raw packet count convey a wealth of useful information, tied to several design aspects of an application (i.e., overlay discovery and signaling activities, video diffusion policy, etc.).

A human analogy may help in clarifying this intuition. Let us compare peers in the network to people in a party room: human beings have rather different attitudes and behaviors, just as peers do. For instance, somebody prefers lengthy talks with a few friends: similarly, some application tends to keep exchanging data with the same peers as long as possible. Somebody else, on the contrary, may prefer to briefly chat with a lot of people, just like applications with an intense network discovery activity and a dynamic diffusion of the video content would do.

Furthermore P2P-TV applications exchange the video stream in *chunks*, i.e., minimum units of data with a fixed length, that are thus transferred with the same number of packets: since each application independently selects its own chunk size, differences in this choice will be reflected by the raw packet count.

Finally, in the following we consider only the downlink traffic direction. Indeed, we point out that P2P-TV applications need a rather steady downlink throughput to ensure a smooth playback: in fact, it has been observed that while peers *consume* equally, they do not *contribute* equally [11] to the video diffusion. Therefore, we expect the downlink traffic direction alone to convey all the needed information for a correct classification.

Behavioral P2P-TV Signature

More formally, let us consider the traffic received by an end-point $\mathcal{P}_x$=$(IP_x, port_x)$ during an interval ΔT, which, for the remainder of this work, we fix to $\Delta T = 5$ seconds. During this interval, peer $\mathcal{P}_x$ will be contacted by $K(x)$ other peers, namely $\mathcal{P}_1 \ldots \mathcal{P}_{K(x)}$, receiving a different number of packets from each of them, say $p_1 \ldots p_{K(x)}$. Then, we derive the number N_I^x of peers that sent a number of packets in an interval $I = [a, b]$ to peer $\mathcal{P}_x$ i.e. denoting with $\mathbf{1}\{\cdot\}$ is the indicator function:

$$N_I^x = \sum_{j=1}^{K(x)} \mathbf{1}\{p_j \in I\} \qquad (1)$$

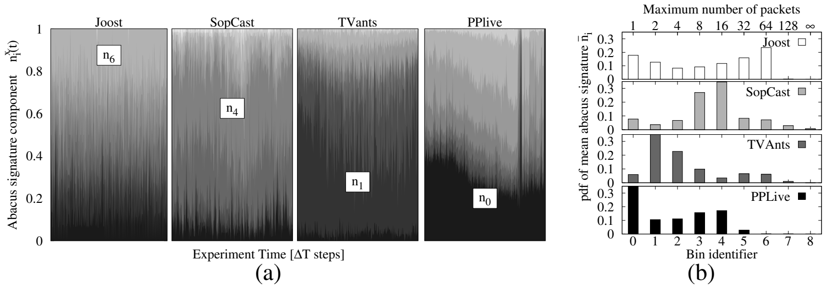


Fig. 1. Abacus signatures of P2P-TV application: (a) temporal evolution and (b) mean value

In particular we use $B + 1$ intervals of exponential width $\{I_0, \ldots, I_i, \ldots, I_B\}$ such that $I_0 = (0, 1]$, $I_i = (2^{i-1}, 2^i]$, and $I_B = (2^B, \infty]$. In other words, $N_i^x = N_{I_i}^x$ will count the number of peers sending to $\mathcal{P}_x$ a number of packets in the interval $(2^{i-1}, 2^i]$, while $N_B^x = N_{I_B}^x$ will count all peers sending at least 2^B packets to $\mathcal{P}_x$. As previously explained, we expect that if the application performs network discovery by means of single packet probes and uses $C = 16$ packet long chunks, there will be a large number of peers falling into the N_0^x and N_4^x bins. For each time interval ΔT, we then build a behavioral signature $\underline{n}^x = (n_0^x, \ldots, n_B^x) \in \mathbb{R}^{B+1}$, by normalizing N_i^x over the total number $K(x)$ of peers that contacted $\mathcal{P}_x$ during that interval:

$$n_i^x = \frac{N_i^x}{\sum_{b=0}^{B} N_b^x} = \frac{N_i^x}{K(x)}, \quad \text{and} \quad |\underline{n}^x| = \sum_{i=0}^{B} n_i^x = 1 \tag{2}$$

Since signature $\underline{n}^x$ has been derived from a pure count of the number of exchanged packets, we name it *abacus* (shorthand for “automated behavioral application classification using signatures”). An example of the temporal evolution of abacus signatures $\underline{n}^x(t)$ is given in Fig. 1-(a). considering the behavior of an arbitrary peer $\mathcal{P}_x$ during 1-hour long experiment for the four different applications. Time of the experiment runs on the x-axis in multiples of ΔT, whereas y-axis reports the *cumulative* abacus signature, using different fading colors for different bins. Bins are ordered from bottom to top, so that bin number 0 (which is the darkest one), starts at the bottom of the y-axis scale and extends until n_0^x. Subsequent bins are then *incrementally* staggered (with progressively lighter colors), so that the k-th bin starts at $\sum_{i=0}^{k-1} n_i^x$ and the last bin extends until $|\underline{n}^x| = 1$.

Already at a first glance, we notice that for any given application one bin (which is labeled in the picture) is remarkably wider than the others. Moreover, while the widest bin differs across applications, it keeps roughly the same for any given application, during most of the experiment duration, despite its actual width changes over time. This can be more easily gathered by comparing the *mean* per-application signature, averaged over all time intervals, reported in Fig. 1-(b). for instance, during a 5-seconds interval, Joost peers tend to exchange either a single or several (33–64) packets to any given peer, whereas SopCast performs less probing sending also fewer (9–16) packets.

TVants prefers instead lower order bins (2–4 packets), and PPLive makes a significant use of single packet exchanges, possibly to discover the network, while the rest of its activity is more spread out over the other bins.

Support Vector Machines

Our classification framework makes use of Support Vector Machines (SVMs) [10], which are well known among the *supervised* learning methods for their discriminative power. In SVM, entities to be classified are represented by means of some distinctive "features", i.e., the abacus signatures in our case. SVM classification is a two-phase process. First, SVM needs to be trained with supervised input (i.e., abacus signatures of known traffic and the corresponding application label). The output of this phase is a model, which can then be applied in a second phase to classify previously unseen signatures.

Given a geometric representation of features in a multi-dimensional space, the training phase partitions the feature space into a set of classes, using a few representative samples of each class. Then, during the classification phase, any new point is assigned to the most likely class, depending on the zone the point falls into. Defining the delimiting surfaces is complex, since training points can be spread out on the feature space: the key idea of SVM is to remap the original space into a higher dimensional one, so that different classes can be separated by the simplest surfaces, i.e., hyper-planes. To assess the classification results, signatures are computed over known validation traffic (different from the one used in the training phase), and are then fed to SVM model: finally, classification results are compared with the real label.

Rejection Criterion

An important point is that, since SVM induces a *partition* on the abacus feature space, any new point is necessarily labeled as one of the applications offered to SVM during the training phase. Since we trained our machine only with P2P-TV traffic, any unknown application would be mistakenly classified as P2P-TV. Therefore, in order to have an effective classification engine, we need to define a *rejection criterion*.

Given two probability density functions, there exist several indexes to evaluate their similarity. The Bhattacharyya distance BD [12] is a measure of the divergence of two pdfs, which verifies the triangular inequality. In the case of two discrete probability p and q in $\mathbb{R}^n$, it is defined by:

$$BD(p,q) = \sqrt{1-B} \quad \text{where} \quad B = \sum_{k=1}^{n} \sqrt{(p(k) * q(k))} \tag{3}$$

B is known as Bhattacharyya coefficient and $0 \leq B \leq 1$. Values of BD near to zero indicates strong similarity (if $p(k) = q(k) \quad \forall k$, $B = 1$ and $BD = 0$) whereas values near to one indicates weak similarity.

In our context we *reject* the SVM classification label C of a sample signature n whenever the distance $BD(n, \overline{n}(C))$ exceeds a given threshold R, where $\overline{n}(C)$ is the average signature computed over all training set signatures of application C. In other words, we accept the SVM decision only if the signature n lies within a radius R from the center of the SVM training set for that class. Otherwise we label the signature sample as “unknown”. For the time being we set $R = 0.5$ and discuss the impact of this choice, as well as its motivation, later on.

3 Experimental Results

Testbed setup
Assessing traffic classification performance is known not to be a trivial task due to the difficulty to devise a reliable “oracle” to known the “ground truth”, i.e., what was the actual application that generated the traffic [4]. 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 care must be taken in order to ensure testbed traces to be representative of real world traffic.

Therefore, to overcome this issue, we setup a large testbed in the context of NAPA-WINE, a 7th Framework Programme project funded by the EU [13], whose main features are summarized in Tab. 1. Partners took part in the experiments by running P2P-TV clients on PCs connected either to the institution LAN, or to home networks having cable/DSL access. In more detail, the setup involved a total of 44 peers, including 37 PCs from 7 different industrial/academic sites, and 7 home PCs. Probes are distributed over four countries, and connected to 6 different Autonomous Systems, while home PCs are connected to 7 other ASs and ISPs. Moreover, different experiments and peers configurations (hardware, OS version, channel popularity, etc.) further ensure that the testbed is representative of a wide range of scenarios. We considered four different applications, namely PPLive, SopCast, TVAnts and Joost and we performed several 1-hour long experiments during April 2008, where partners watched the same channel at the same time and collected packet-level traces. In all cases, the nominal stream rate was 384kbps. Overall, the testbed dataset amounts to about 5.5 days worth of video streaming, $100 \cdot 10^3$ signatures samples, $48 \cdot 10^6$ packets, 26 GBytes of data.

In order to asses the ability of our system to correctly label as unknown the traffic generated by non P2P-TV applications, we also collected packet level traces from our campus network. Particularly we isolated the traffic generated by two widely adopted P2P applications, i.e. Skype and eDonkey as examples of respectively P2P voice and file-sharing applications. To identify eDonkey we employed a DPI classifier based on [14], while for Skype we resorted to [9]. The final dataset amounts to about 2.2GBytes and 1,4GBytes of data for Skype and eDonkey respectively, which correspond to $500 \cdot 10^3$ and $300 \cdot 10^3$ signatures.

Discriminating P2P-TV applications
We use the signatures extracted from the testbed traffic to assess the ability of the engine to reveal P2P-TV traffic and to distinguish the different applications. Numerical results reported in the following are obtained by training the SVM with 20% of the testbed signatures selected at random, and using the remaining 80% for validation. Experiments are then repeated 10 times, randomizing the training set each time, so to gather robust results. Performance are expressed in terms of the amount of True Positive (TP, i.e. classifying label X correctly as X), and False Negative (FN, i.e. labelling a X sample as Y) classifications, and by measuring the TP-Rate (TPR) or *recall*, defined as TPR=TP/(TP+FN).

Table 1. Summary of the hosts, sites, countries (CC), autonomous systems (AS) and access types of the peers involved in the experiments

Host	Site	CC	AS	Access	Nat	FW	Host	Site	CC	AS	Access	Nat	FW
1-4	BME	HU	AS1	high-bw	-	-	1-4	ENST	FR	AS4	high-bw	-	Y
5			ASx	DSL 6/0.512	-	-	5			ASx	DSL 22/1.8	Y	-
1-9	PoliTO	IT	AS2	high-bw	-	-	1-5	UniTN	IT	AS2	high-bw	-	-
10			ASx	DSL 4/0.384	-	-	6-7				high-bw	Y	-
11-12			ASx	DSL 8/0.384	Y	-	8			ASx	DSL 2.5/0.384	Y	Y
1-4	MT	HU	AS3	high-bw	-	-	1-8	WUT	PL	AS6	high-bw	-	-
1-3	FFT	FR	AS5	high-bw	-	-	9			ASx	CATV 6/0.512	-	-

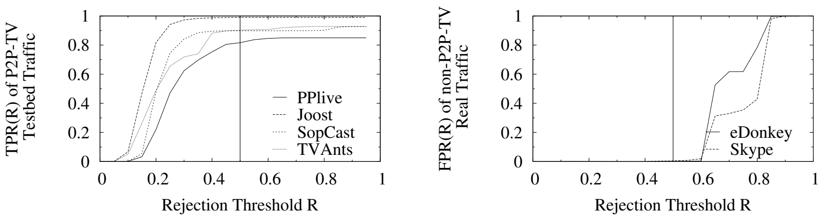
Table 2. Confusion matrix of P2P-TV application (left table) and per signature, packets, bytes and end-point classification results (right table)

	Signatures: Confusion Matrix					Signatures			Packets			Bytes			Peer	
	PPLive	TVants	SopCast	Joost	Unk	TP	Mis	Unk	TP	Mis	Unk	TP	Mis	Unk	TP	Unk (n)
PPLive	81.66	0.58	9.55	2.32	5.90	81.7	12.4	5.9	91.3	8.7	0.0	91.6	8.4	0.0	96.2	3.8 (1)
TVants	0.41	98.84	0.15	0.57	0.04	98.8	1.2	0.0	99.6	0.3	0.1	99.6	0.3	0.1	100	0 (0)
SopCast	3.76	0.11	89.62	0.32	6.19	89.6	4.2	6.2	94.7	1.7	3.6	94.0	1.8	4.2	94.4	5.6 (2)
Joost	2.84	0.55	0.28	89.47	6.86	89.5	3.7	6.8	92.1	2.3	5.6	92.2	2.4	5.4	93.3	6.6 (2)

Let us start by observing the left part of Tab. 2, which reports the classification performance relative to individual end-point signatures samples, corresponding to $\Delta T = 5$ s. worth of traffic, adopting a "confusion matrix" representation. For each row, testbed traffic signatures are classified using SVM and the classification result is reported in different columns. Diagonals of the matrix correspond to correct classification TPR, whereas elements outside the diagonal correspond to FN misclassification. Particularly the last column reports the traffic which is classified as "unknown" by the rejection criterion. It can be seen that, in the worst case, individual signatures are correctly classified nearly the 82% of the times. The application most difficult to identify appears to be PPLive, which generates 9.6% of SopCast False Positives, while for the all others the TP percentage exceeds 89%. All applications but TVAnts generate about 6% of "unknown" false negative (i.e. rejected due to a large BD distance.)

We next quantify the classification performance also in terms of the number of correctly classified *packets*, *bytes* and *peers*. In more detail, to each signature a precise number of packets and bytes directly corresponds, so that the per-packets and per-byte metrics can be directly evaluated. In the case of per-peer classification, we instead combine several classification decisions, and evaluate whether the *majority* of signature samples for a given end-point has been correctly classified over its whole 1-hour long experiment. We point out that, while the classification engine is able to take a decision "early" (more precisely, after a delay of ΔT seconds), in the latter case of end-point classification we actually need *all* observations of a given experiment, falling therefore in the context of "late" classification. Right portion of Tab. 2 reports the percentage of correct classification (TPR), of misclassification (Mis, corresponding to the sum of row values that fall outside of the diagonal in the confusion matrix) and rejection (Unk) in terms of signature, packets, bytes and peer metrics; notice that FN=Mis+Unk.

Interestingly, we see that performance improves for all applications, and especially for PPLive, when considering *packets* and *bytes* metrics with respect to *signature*



(a) (b)

Fig. 2. TPR of P2P-TV (a) and FPR of non-P2P-TV (b) as a function of the rejection threshold R

samples: this means that misclassification happens when fewer packets/bytes are received (i.e., when the application is possibly mal-functioning). In case of end-point classification, reliability slightly increases, as the recall for all applications is greater than 93%. While results are more than satisfactory, yet we observe that identification of some peer fails even in the case of late classification, with a total of 5 tests classified as unknown, as highlighted in the last column of the table. Digging further we actually found that mainly 3 hosts are responsible for the misclassification, and moreover all of them actually showed abnormal functioning during the experiments.

Classifying the Unknown

If the rejection criterion generates about 5% of additional false negatives for the classification of P2P-TV applications, it reveals to be very effective in correctly handling unknown applications. In fact for both Skype and eDonkey traces our engine raises only 0.1% of false alarms: in other words, only 0.1% of the signature samples are not label as “unknown” as they should, but are rather labeled as one of the P2P-TV applications.

Results early shown highly depend on the rejection threshold R, whose choice depends on the following tradeoff. Intuitively, R should be as large as possible, to avoid classifying P2P-TV as Unknown (i.e., maximize the TPR) but, at the same time, R should be as small as possible to avoid classifying irrelevant traffic as P2P-TV (i.e., minimize the false positive rate, FPR). To validate the choice of $R = 0.5$ we proceeded as follows. Using testbed traces, we empirically evaluate the TPR as a function of the rejection threshold R, which is depicted in Fig. 2-(a). It can be seen that TPR quickly saturates, meaning that no P2P-TV signature is rejected when $R \geq 0.5$. We then use the non-P2P-TV traffic from our campus network to instead evaluate the FPR as a function of R, shown in Fig. 2-(b). In this case, due to the partitioning approach of SVM, eDonkey and Skype signatures are forcibly labeled by SVM as one of the P2P-TV applications: however, the BD distance of the labeled signature from the center of the cluster is likely higher than that of a true P2P-TV application. This clearly emerges from Fig. 2-(b), which show that for low values of $R \leq 0.5$, practically no false alarm is raised.

We specify that these are preliminary results, and that we plan to test the effectiveness of the rejection criterion on a wider range of non P2P-TV protocols as a future work. Yet, we showed that our rejection mechanism can correctly handle two widely used applications, representative of two different families of P2P protocols, by successfully identify them as unknown.

4 Conclusions

This work proposed a novel technique for the classification of P2P-TV applications, which relies on the count of packets exchanged amongst peers during small time-windows, and makes use of Support Vector Machines.

Through measurement collected in a large testbed, we show that our classification engine, is able to correctly classify more than 81% of signatures in the worst case. If performance is evaluated considering packets, bytes or peers metrics, correct classifications amount to 91% in the worst case. Moreover the rejection criterion we designed is able to correctly handle unknown applications, raising only 0.1% of false alarms.

We believe this work to be a first step toward accurate, fine-grained, behavioral classification: several aspects remains indeed uncovered (e.g., byte-wise vs packet-wise signatures, more P2P applications, TCP traffic, training set selection etc.), which we plan to address in the future.

Acknowledgements

This work was funded by EU under the FP7 Collaborative Project "Network-Aware P2P-TV Applications over Wise-Networks" (NAPAWINE).

References

1. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: multilevel traffic classification in the dark. ACM Communication Review 35(4) (2005)
2. Xu, K., Zhang, Z., Bhattacharyya, S.: Profiling internet backbone traffic: behavior models and applications. In: ACM SIGCOMM 2005, Philadelphia, PA, August 2005, pp. 169–180 (2005)
3. Sen, S., Spatscheck, O., Wang, D.: Accurate, scalable in-network identification of p2p traffic using application signatures. In: WWW 2004, NY (May 2004)
4. Moore, A.W., Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 41–54. Springer, Heidelberg (2005)
5. Roughan, M., Sen, S., Spatscheck, O., Duffield, N.: Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification. In: ACM IMC 2004 (October 2004)
6. Moore, A.W., Zuev, D.: Internet traffic classification using bayesian analysis techniques. In: ACM SIGMETRICS 2005 (2005)
7. Bernaille, L., Teixeira, R., Salamatian, K.: Early Application Identification. In: Conference on Future Networking Technologies (CoNEXT 2006), Lisboa, PT (December 2006)
8. Crotti, M., Dusi, M., Gringoli, F., Salgarelli, L.: Traffic Classification through Simple Statistical Fingerprinting. ACM Computer Communication Review 37(1) (January 2007)
9. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing Skype Traffic: when Randomness Plays with You. In: ACM SIGCOMM, Kyoto, Japan (August 2007)

10. Cristianini, N., Shawe-Taylor, J.: An introduction to support Vector Machines and other kernel-based learning methods. Cambridge University Press, New York (1999)
11. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A Measurement Study of a Large-Scale P2P IPTV System. In: IEEE Transactions on Multimedia (December 2007)
12. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by probability distributions. Bull. Calcutta Math. Soc. 35, 99–109 (1943)
13. NAPA-WINE, `http://www.napa-wine.eu`
14. Kulbak, Y., Bickson, D.: The eMule protocol specification. Tech. Rep. Leibniz Center TR-2005-03 (2005)