Fine-grained traffic classification with Netflow data

Dario Rossi, Silvio Valenti
Telecom ParisTech, France
INFRES Department
first.last@enst.fr

ABSTRACT

Nowadays Cisco Netflow is the de facto standard tool used by network operators and administrators for monitoring large edge and core networks. Implemented by all major vendors and recently a IETF standard, Netflow reports aggregated information about traffic traversing the routers in the form of flow-records. While this kind of data is already effectively used for accounting, monitoring and anomaly detection, the limited amount of information it conveys has until now hindered its employment for traffic classification purposes. In this paper, we present a behavioral algorithm which successfully exploits Netflow records for traffic classification. Since our classifier identifies an application by means of the simple counts of received packets and bytes, Netflow records contain all information required. We test our classification engine, based on a machine learning algorithm, over an extended set of traces containing a heterogeneous mix of applications ranging from P2P file-sharing and P2P live-streaming to traditional client-server services. Results show that our methodology correctly identifies the byte-wise traffic volume with an accuracy of 90% in the worst case, thus representing a first step towards the use of Netflow data for fine-grained classification of network traffic.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management, Network monitoring*

General Terms

Algorithms, Measurement

Keywords

Netflow, Traffic Classification, SVM

1. INTRODUCTION

As networks become increasingly complex and data transmission speed keeps growing at a steady rate, networks operators are constantly demanding efficient and dependable tools to support them in network management and monitoring tasks. In this context, a reliable traffic classification engine, able to efficiently identify the applications generating the traffic, is a very desirable tool. This would enable several important management activities like accounting, SLA compliance verification, lawful interception, etc. Unfortunately, traditional classification techniques, such as port-based classification or deep packet inspection, are ineffective with modern applications and unable to deal with the increasing amount of traffic of nowadays networks. In response to such problems, the research community has proposed a number of solutions which base the classification on different information, either the statical properties of traffic [1, 13] or the characteristics of the specific pattern of traffic generated by an application [10, 12, 20].

The use of aggregated measurements has represented an efficient way to cope with the huge amount of traffic for many network management tasks. For instance, in recent years Netflow has established itself as the main solution for flow-level measurements. Widely implemented in routers and recently standardized in a IETF draft [4], Netflow reports aggregated information on network traffic in the form of *flow-records*. Operators already employ such information successfully for accounting, anomaly detection and network monitoring, but, to date, there have been only a few tentatives [2, 11] of using it for traffic classification. It is a common belief that the amount of information carried by flow-records is not enough to support an accurate identification of network applications. Yet, with the widespread adoption and availability of Netflow we argue that a classification engine based on flow-records would be highly appreciated by ISPs. Possible applications we foresee are long-term monitoring of network usage in large networks (e.g. to investigate and rank application popularity) or charging (e.g. charge specific traffic more than other).

In this work we extend the Abacus behavioral classification algorithm, formerly presented in [20], to make it able to work with Netflow data. The Abacus classifier, designed with P2P live-streaming traffic as main target, characterizes the behavior of an application by means of the simple count of packets and bytes exchanged with other hosts in small time-windows. We found that these measurements capture characteristic properties of target applications, and so, in combination with a powerful machine learning algorithm, they enable the high classification accuracy showed by Abacus.

This work extends the classifier in several ways. First, we want the methodology to be fully compliant with Netflow data. Therefore, we consider larger time-scales, in the order of minutes, which is the usual time granularity of Netflow whereas in [20] we just used smaller time-scales for faster classification. Second, we take into account a larger set of applications, which also includes other P2P services besides P2P-TV, such as file-sharing (e.g, BitTorrent, eDonkey) and VoIP (e.g., Skype), as well as traditional client server

"*IWCMC* '10, June 28– July 2, 2010, Caen, France.

applications (e.g. DNS). Finally, we use a more robust training phase, introducing an explicit class "others" for all traffic not pertaining to target applications. In this way we simplify the classification methodology by getting rid of the rejection criterion [20] formerly needed to filter out "unknown" traffic.

We validate the classifier on a large set of packet-level traces, from which we derive Netflow records used by the classification engine. Traces come either from active testbed experiments, which guarantee the best accuracy for the ground truth, or from passive measurements from operational networks, in which case the ground truth is provided by a traditional DPI classifier. Results, which consider different setting for Netflow timers, show that Abacus is able to achieve a high classification accuracy (90% in the worst-case) relying only on flow-records, so representing a first step towards fine-grained traffic classification based on Netflow data.

2. RELATED WORK

Two bodies of work are related to our effort. On the one hand, we have research works which focus on traffic classification [1, 2, 7, 9–13, 16, 20], but that with minor exceptions [2, 11] do not explicitly consider Netflow input. On the other hand, we have works which explicitly use Netflow records for other purposes [6,14,17,18], such as for traffic monitoring, accounting or anomaly detection.

The first body of literature [1, 2, 7, 9–13, 16, 20] is a clear evidence of the great interest of the research community in the topic of network traffic classification. Still, despite the number of proposed solutions, a reliable identification of the application protocol associated to network packets remains an open challenge. A first part of studies bases the classification on the inspection of *packet payload*, either by searching for known signatures [16] or by deriving a statistical description of transmitted data [7, 9]. A second part of works [1, 2, 11, 13] relies on *statistical properties* of flows generated by a given application. Finally, some studies [10, 12, 20] characterize the behavior of an application, recognizing the *pattern of traffic* generated. Our work fits in this last category but is the first that, to the best of our knowledge, is capable of fine-grained application detection based on behavioral data.

The research community has also investigated the possible uses of Netflow records for different network managing tasks. The most natural applications of flow-level data are obviously *accounting* [18, 19] and network traffic *reporting and monitoring* [6]. Other works [14, 17] present effective techniques to perform *anomaly detection* based on Netflow data. Finally a few works [2, 11] closer to ours have recently proposed *traffic classification* based on Netflow records. Both these studies belong to the family of statistical based classifiers, while our work is the first example of behavioral classifier explicitly addressing the use of Netflow records. Moreover this work differentiates from them in the aim of the classification as it is the first to distinguish among *specific applications* belonging to the same class of service.

3. NETFLOW DATA AND CLASSIFICATION ALGORITHM

3.1 Netflow data

Netflow [3] represents the de facto standard for flow level measurements in the networking operational community. Originally designed by Cisco as a cache to improve IP flow lookups in routers, it soon revealed itself to be a very useful tool for network traffic monitoring and reporting. Netflow main feature relies in the level of information it provides – more compact than packet level traces, but still more expressive than coarse-grained counters of SNMP. After several subsequent versions, with v5 being the most commonly used, Netflow v9 has finally evolved in a IETF standard, IPFIX [4], already implemented by most network equipment vendors.

A Netflow probe tracks *flows*, i.e. unidirectional sequences of packets exchanged by two endpoints. First, it extracts from each packet a *key* composed of specific header fields (for v5 it is the classical 5-tuple: IP source and destination addresses, transport-layer source and destination ports, IP protocol). This key identifies a record in memory, where the probe stores, besides the key itself, a number of *attributes*, like cumulative packets and bytes counters, flow starting and finishing timestamps, IP type of service, TCP flags, MPLS label, physical input and output interface indexes, only to cite the most important ones. Although last releases have introduced the possibility to specify custom keys and attributes, in the following we employ the flow definition of v5 for compatibility reasons.

Whenever a flow expires, the router transmits a UDP packet containing the related record to a Netflow collector, which elaborates and eventually stores this information. Different reasons can cause a flow expiration: (i) a packets explicitly terminates the flow (e.g, a TCP FIN packet); (ii) the flow has been *inactive* for a time greater than the `inactive_timeout` (iii) the flow has been *active* for a time greater than the `active_timeout`; (iv) the flow cache is full and some space needs to be freed for new flows. Default values for `inactive_timeout` and `active_timeout` are respectively 15 seconds and 30 minutes. In Sec. 4 we will see that the value of these timeouts has an important impact on classification performance.

3.2 Classification methodology

In the following we will introduce our classification method and the modifications needed to couple it with Netflow. We focus more on the intuitions behind the algorithm, and refer the reader to [20] for a more rigorous and detailed presentation and to [8] for a comparison with another payload based classifier.

Abacus classifies application endpoints, i.e the pair IP address and transport layer port (IP_0, p_0). In this work, we focus on UDP traffic and applications running over it. We consider a single traffic direction and we also assume that all traffic directed to the target endpoint is observed by our engine. The classification is based on the raw count of packets and bytes that the endpoint (IP_0, p_0) receives from others parties of the communication $\mathcal{C} = \{(IP_i, p_i)\}_{i=1}^{N}$, where $\mathcal{C}$ represents the set of N neighboring endpoints which contacted (IP_0, p_0) during an interval ΔT. In more detail, we evaluate the distributions $\underline{p}$ and $\underline{b}$ of the N endpoints according to the count of exchanged packets P and bytes B respectively: notice that, by definition, $|\underline{p}| = 1$ and $|\underline{b}| = 1$. We use a logarithmic binning with base 2 as support for both distributions: indeed, logarithmic binning has the advantage of highlighting the differences for lower counts of packets and bytes (e.g., having exchanged 1 or 4 packets w.r.t 1001 or 1004 packets), as well as that of providing a more concise representation.

The final Abacus signature is simply the combination of the packet-wise and byte-wise distribution $\underline{s} := (\underline{p}, \underline{b})$. This compact description of the endpoint behavior is fed to a machine learning algorithm, namely Support Vector Machine (SVM) [5], to perform the classification.

An example will clarify the methodology. If an endpoint receives only single-packet probes from its N neighbors during ΔT, the distribution will be $\underline{p} = (1, 0, 0, 0, \ldots)$ as all peers fall in the first bin. Instead, a distribution like $\underline{p} = (\frac{1}{2}, \frac{1}{4}, 0, \frac{1}{4}, \ldots)$ means that half of the endpoints sent single-packet probes, while the remainders have sent either $n = 2$ or a number $n \in (4, 8]$ of packets to (IP_0, p_0).

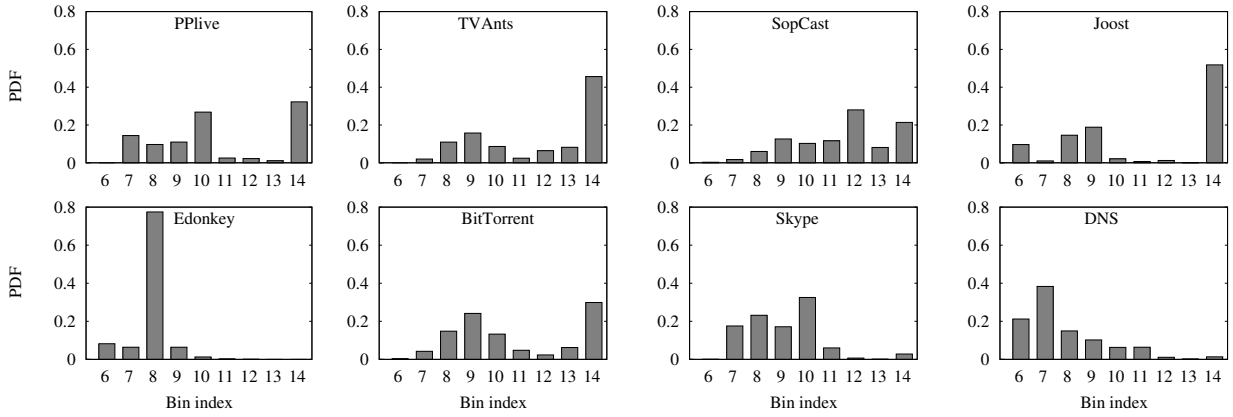

Figure 1: Mean byte-wise signature for the considered applications.

Despite the use of a simple count, furthermore relative to a single traffic direction, Abacus signatures capture distinctive properties of the observed applications. Concerning P2P applications, a number of design choices directly reflect in the signature $\underline{s}$: for instance, different peer discovery techniques (e.g., single-packet probe versus handshakes) clearly make peers fall into different bins (e.g., the first or second respectively). In the same way, even though several applications may adopt similar content-diffusion techniques (e.g., mesh-pull diffusion for BitTorrent, SopCast, PPLive) the use of specific chunk-sizes, or even packet sizes, still makes them easily recognizable (e.g., by making a specific bin more likely than others in the $\underline{p}$ and $\underline{b}$). Concerning client-server traffic, instead, we expect the length of requests to be particularly tied to the protocol (e.g., very short DNS queries, versus medium length HTTP requests, versus rather long SMTP message transmissions).

During our experiments we empirically found that Abacus signatures calculated on the downlink traffic are more stable over time and also able to better differentiate the target applications. We can individuate some reasons behind such a finding. For instance, P2P-TV applications strive to download an almost constant bitrate from other peers to ensure a smooth video playback, so this direction of traffic exhibits a rather steady downlink throughput. Other P2P applications do not have such constraints and are usually characterized by multi-modal behaviors on both directions, so there is no real advantage in considering one direction over the other. For client-server traffic, since we focus on the classification of endpoints talking with many hosts, we are actually concentrating on servers, and, as already said, we expect the incoming direction (i.e. directed to the server) to be more representative.

To support our intuition, we report in Fig. 1 a pictorial representation of the average byte-wise signature $\underline{b}$ of all the applications considered in the experiments. We represent signatures as histograms, where each bar corresponds to the value of the component of the distribution identified by the index on the x-axis. Notice that each application generates a different and characteristic pattern. For instance, DNS adopts short exchanges of data for the majority of communications, which contribute to larger values of lower bins (e.g. most DNS queries fall into bin number 7, corresponding to single packets having size in the 128–255 Bytes range). Instead all P2P-TV applications (top row) together with BitTorrent show a considerable percentage of peers falling in the last bin (corresponding to hosts which are contributing with most of the video or data stream), but still differentiate among themselves thanks to lower bins.

We had to tweak the original Abacus classifier to have it work smoothly with Netflow data. First, in our previous work we used a duration of 5 s for the endpoint observation window, which is incompatible with the time granularity of Netflow: hence, we now consider values of ΔT in the time-scale of minutes. A second difference derives from the fact that flow-records exports are not synchronous. In other words we cannot ask Netflow to terminate and emit all flow-records at a given time (e.g. at the end of ΔT). As a result, we may end with some flow spanning over two windows. In this case we simply divide the flow in two parts, by prorating the number of packets and bytes (i.e., we split the flow-record at the end of the current ΔT and divide the packets and bytes between the two segments proportionally to their duration), as already done in [19].

4. EXPERIMENTAL RESULTS

4.1 Dataset and Methodology

We validate our classification engine on a large set of traces, collected in different environments and whose main characteristics are reported in Tab. 1. We concentrate on UDP traffic, which is not only the preferred transport-layer protocol of P2P live-streaming applications and VoIP, but has also recently become the choice of the most popular file-sharing application, namely BitTorrent.

The traces containing P2P live-streaming traffic were captured during a series of experiments performed in the context of the NapaWine project [15]. All partners participated to the experiments with a number of machines (having different ADSL or high bandwidth access), running different popular P2P-TV applications (PPLive, SopCast, TVAnts, Joost) and capturing the generated traffic. BitTorrent traces were instead collected by running the official client connected to the Internet through an ADSL access. Using traces from controlled active scenarios has the advantage of providing a reliable ground truth, as there is no doubt on the application generating the traffic.

For the remaining applications, we resorted to a day-long trace passively collected at the POP of a large Italian Internet provider. By running a traditional DPI classifier we were able to extract flows pertaining to eDonkey, Skype and DNS, as representative of P2P file-sharing, P2P VoIP and client-server traffic respectively. Finally we built the "other" class with all the traffic contained in the first hour of this trace and not related to any of the previous protocols. By providing SVM with a description of the background traffic, the machine will correctly handle traffic not belonging to any of

Table 2: Confusion matrix: Classification accuracy as a percentage of signatures (%S) and percentages of bytes (%B)

		Classification outcome																	
		PPLive		TVAnts		SopCast		Joost		eDonkey		BitTorrent		Skype		DNS		Other	
		%S	%B	%S	%B	%S	%B	%S	%B	%S	%B	%S	%B	%S	%B	%S	%B	%S	%B
Real label	PPLive	**63.6**	**96.0**	1.0	3.2	0.7	0.3	0.1	-	-	-	0.1	0.4	2.9	-	9.4	-	22.3	-
	TVAnts	3.1	6.8	**54.4**	**92.9**	1.0	0.3	0.2	-	-	-	0.2	-	7.4	-	9.5	-	24.3	-
	SopCast	0.7	0.2	0.4	0.4	**49.7**	**99.4**	-	-	0.1	-	0.3	-	4.8	-	15.9	-	28.1	-
	Joost	0.2	-	-	-	-	-	**53.2**	**99.9**	0.3	-	0.2	-	4.5	-	19.1	-	22.5	-
	eDonkey	-	-	-	-	-	-	-	-	**94.4**	**98.9**	-	-	-	-	0.7	0.2	4.8	0.9
	BitTorrent	0.6	-	0.5	0.1	0.8	0.8	0.3	1.9	-	-	**12.5**	**89.1**	5.2	1.7	61.3	5.8	18.8	0.6
	Skype	-	-	-	-	0.1	0.3	-	-	-	-	0.2	0.4	**86.1**	**90.5**	5.8	2.5	7.8	6.4
	DNS	0.1	-	-	-	0.1	0.3	-	0.2	0.3	0.9	-	0.5	6.5	3.9	**63.9**	**91.2**	29.1	2.9
	Other	0.1	-	-	-	-	-	-	-	0.4	0.1	-	0.1	3.5	-	8.3	-	**87.6**	**99.8**

Table 1: Summary of the dataset

Category	Application	Packets	Bytes
P2P TV	PPlive	7.3 M	1.13 G
	Sopcat	3.2 M	0.45 G
	TVAnts	2.4 M	2.56 G
	Joost	3.4 M	2.14 G
P2P File-sharing	eDonkey	22.4 M	6.93 G
	BitTorrent	1.4 M	0.74 G
P2P VoIP	Skype	6.1 M	2.91 G
Naming	DNS	1.5 M	103 M
Remaining UDP	Other	10 M	10.9 G

the target applications and label it as "other".

We implemented a custom tool similar to [6], which converts traffic traces into Netflowrecords, from which we build the Abacus signatures. We randomly sample 10% of the signatures of each protocol that we use to train the SVM model. The trained model is then applied to the remaining signatures, used as validation set. This process is repeated 10 times, varying the train and validation set each time.

4.2 Classification results

In Tab. 2, we show the classification results obtained by setting the `active_timeout` and ΔT to 120 s, while we leave the `idle_timeout` to its default value of 15 s (we will discuss this parameter choice later). The table adopts the classical confusion matrix representation: rows correspond to the real traffic, while columns show the possible classification outcomes. Classification performance is expressed both as percentage of signatures ($\%S$) and as percentage of bytes ($\%B$). In fact by adding the bytes transferred by all flows seen in a ΔT, we can associate a precise amount of data to each signature. we expect ISP to prefer the bite-wise metric, as it refers to the bulk of the traffic volume. For instance, the first value on the first row tells us that 63.6% of all PPLive signatures were correctly classified as PPLive, and these signatures correspond to the 96% of the total PPLive traffic in terms of bytes. Values in bold on the diagonal represent the percentage of correctly classified traffic. All other values represent false negatives, in particular in the last column we report the percentage of traffic classified as "other". Finally it is worth saying that diagonal values can be regarded as the *recall* of that class, since they are calculated as the ratio of true positives over the sum of true positives and false negatives.

For the sake of simplicity, let us consider different families of applications separately. We first observe the performance related to P2P-TV services, which were the original target of the Abacus classifier in [20]. Byte accuracy is extremely high for all four applications, always greater than 90%, with Joost and SopCast exceeding the 99%. Instead, the percentage of correctly classified signatures is lower (not even approaching the 65%), with a large fraction of classifications falling in either the DNS or the "other" class. From this, we can conclude that heavy signatures (i.e., carrying more bytes) are more robust to the classification, which is a positive finding as ISPs are interested in classifying the bulk of the traffic. This is also somewhat expected, as signatures carrying less traffic (e.g., few flows in the unit of time ΔT) are also statistically less significant.

Byte accuracy remains high also for the two P2P file-sharing applications, with eDonkey showing also very high values at a signature level. BitTorrent signatures are instead harder to classify in general, except those carrying the greatest portion of traffic (bytes accuracy approaches 90%). Again, it seems that the classifiers correctly identifies BitTorrent traffic when the application is actively downloading from other peers, while it has some problems under different conditions (e.g., when just discovering new peers or performing signaling). Moreover, we point out that due to the recent evolution of its protocol, we have collected just a limited dataset, so the poor performance may also be caused by the incomplete representation of the protocol in the training set. Interestingly, notice also that BitTorrent, which adopts swarming algorithms similar to P2P-TV ones, is sometimes misclassified as P2P-TV, whereas eDonkey is confused rather with DNS.

Similar considerations apply also to Skype and DNS, both performing better when considering bytes instead of signatures. DNS, the only non-P2P applications considered in this work, is the protocol which shows the greatest portion of signatures labeled as "other". This issue, which we aim at inspecting further in future work, is likely due to the presence of other client-server applications in the background class, which, having a behavior similar to DNS, cause the misclassification.

In the last row of the table we instead evaluate how the classifier deals with traffic other than the target applications. Again, the signatures corresponding to the majority of bytes are correctly assigned to the "other" class, showing the effectiveness of this strategy for handling any kind of traffic (a limited percentage of signatures is still misclassified as DNS, which further confirms our previous considerations on the similarities between these classes).

Besides the results reported in Tab. 2, we conducted a series of tests to investigate the effect of different settings of Netflow timeouts on classification performance, that we only briefly report here for lack of space. Fig. 2 depicts the byte accuracy of the different applications for increasing values of the `active_timeout`. As a reference, the leftmost set reports the performance of [20] gathered for $\Delta T = 5$ s (not applicable to real Netflow). Coherently with our previous work, this small interval provides the best performance for P2P applications, but deals poorly with DNS and the "other"

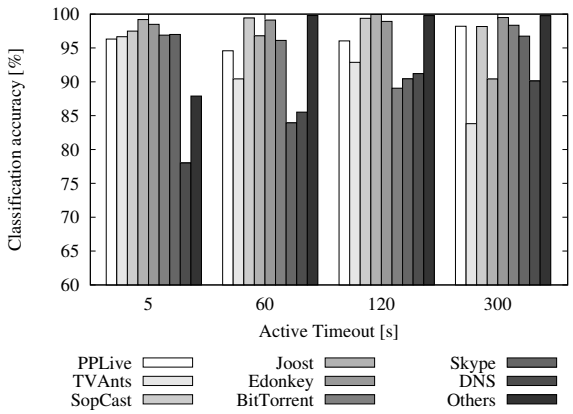

Figure 2: Byte accuracy for different values of active timeout.

classes. From the picture, it can be gathered that $\Delta T = 120$ s represents a good compromise between smaller intervals, which enhance P2P-TV classification accuracy, and larger ones, which perform better for other applications.

5. CONCLUSIONS

This paper described a behavioral classification engine able to achieve an accurate fine-grained classification of network traffic, by exploiting Netflow records. Our solution characterizes the behavior of an application by a simple count of packets and bytes exchanged with other hosts, which is directly found in standard Netflow records. We tested our methodology on a large set of traces, ranging from P2P applications to traditional client-server services. Results show that the proposed technique, albeit not infallible in term of individual signatures, classifies all the target applications with high byte accuracy, and also correctly handles non-relevant traffic.

We believe that this work represents a first step toward fine-grained traffic classification using Netflow data. However, a number of aspects deserve a deeper investigation: for instance the portability of signatures should be tested by running the classifier on traces collected in different place and time from the ones used for training. Also the applicability of this methodology to TCP traffic would be an interesting point, which we plan to address in our future work.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *ACM CoNEXT'06*, Lisboa, PT, Dec 2006.

[2] V. Carela-Español, P. Barlet-Ros, and J. Solé-Pareta. Traffic classification with sampled netflow. *Tech. Rep., UPC-DAC-RR-CBA-2009-6*, Feb 2009.

[3] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), Oct 2004.

[4] B. Claise. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Standards Track), Jan 2008.

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

[6] L. Deri. nProbe: an Open Source NetFlow Probe for Gigabit Networks. In *In Proc. of Terena TNC 2003*, Zagreb, Croatia, 2003.

[7] A. Finamore, M. Mellia, M. Meo, and D. Rossi. KISS: Stocastic Packet Inspection. In *Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537*, May 2009.

[8] A. Finamore, M. Meo, D. Rossi, and S. Valenti. Kiss to Abacus: a comparison of P2P-TV traffic classifiers. In *Traffic Measurement and Analysis (TMA'10) Workshop colocated with PAM'10*, Zurich, Switzerland, Apr 2010.

[9] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *ACM SIGCOMM Workshop on Mining Network Data (Minenet'05)*, Philadelphia, PA, August 2005.

[10] M. Iliofotou, H. Kim, P. Pappu, M. Faloutsos, M. Mitzenmacher, and G. Varghese. Graph-based p2p traffic classification at the internet backbone. In *12th IEEE Global Internet Symposium (GI2009)*, Rio de Janeiro, Brazil, Apr 2009.

[11] H. Jiang, A. W. Moore, Z. Ge, S. Jin, and J. Wang. Lightweight application classification for network management. In *ACM SIGCOMM Internet network management (INM '07)*, Kyoto, Japan, Aug 2007.

[12] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, 35(4):229–240, 2005.

[13] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *ACM CoNEXT'08*, Madrid, Spain, Dec 2008.

[14] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM '04*, Portland, OR, USA, 2004.

[15] E. Leonardi, M. Mellia, A. Horvart, 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 Communications Magazine*, 46:20–22, April 2008.

[16] Moore, Andrew. W. and Papagiannaki, Konstantina. Toward the Accurate Identification of Network Applications. In *Passive and Active Measurement (PAM'05)*, Boston, MA, US, March 2005.

[17] V. P. Roche and U. Arronategui. Behavioural characterization for network anomaly detection. *Transactions on Computational Science IV: Special Issue on Security in Computing*, pages 23–40, 2009.

[18] S. Romig, M. Fullmer, and L. Luman. The osu flow-tools package and cisco netflow logs. In *USENIX LISA'00*, New Orleans, LA, US, Dec 2000.

[19] R. Sommer and A. Feldmann. Netflow: information loss or win? In *ACM SIGCOMM Internet Measurement workshop (IMW '02)*, Marseille, France, Nov 2002.

[20] S. Valenti, D. Rossi, M. Meo, M. Mellia, and P. Bermolen. Accurate, Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets. In *Traffic Measurement and Analysis (TMA), Springer-Verlag LNCS 5537*, May 2009.