Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation

F. Risso, M. Baldi, O. Morandi
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italy

A. Baldini, P. Monclus
Cisco Systems
San Jose, CA, USA

***Abstract*— With the ever increasing amount of traffic, scalability is probably the most important factor that differentiates several existing approaches to traffic classification. This paper focuses on payload-based classification and compares the results obtained through a "lightweight" traffic classification approach with the ones obtained with a "completely stateful" approach, demonstrating that the first approach, albeit less precise, is still appropriate for a large class of applications.**

I. INTRODUCTION

Traffic classification is one of the hottest issues in modern networks. On one side, corporate and ISPs network managers want to know the nature of the traffic in their network for better managements (e.g., QoS, security), on the other side users want to bypass network restrictions in order to re-gain their freedom.

Several traffic classification technologies have been proposed so far, ranging from the "old" payload-based methods [2][3][4] (the packet payload is inspected to determine the application-layer protocol contained in it) to new statistical methods [5][6][7][8] (some statistical behavior of the traffic, e.g. packet inter-arrival time, number and type of new sessions, etc., are used to characterize and classify the traffic). Former methods are usually more accurate, but are known to require more resources (processing and memory) and fail (always) in case of encrypted traffic and (often) in case of tunneled traffic. Statistical methods look promising for overcoming previous limitations, but currently are not yet suitable to fine-grained traffic classification. For example, they are usually able to distinguish between web and peer-to-peer traffic, but they have a hard time differentiating the traffic of a malicious peer-to-peer application from a corporate-friendly one. Moreover, current research results take into account a small set of protocols, which is far from the large number of protocols that are commonly used in today's networks. In addition, the accuracy of their classification (in terms of false positives and false negatives) is often below the limits usually acceptable. For these reasons, most of the products available nowadays are based on payload-based methods, which are able to provide much more precise results, although they are known to be affected by scalability problems. In fact, these problems are particularly evident in the last generation of "stateful" methods, which are often based on the application of a set of rules (usually application-layer signatures) against the entire payload. While this avoids common problems such as the verification of messages when they are split across several packets, it requires the complete reconstruction of the session at application-layer.

While future traffic classification mechanisms will probably implement both payload-based and statistical-based techniques in order to overcome each other issues and to take the best from both worlds, this paper focuses on payload-based techniques because of their widespread use. The aim of this paper is to demonstrate that these "stateful" techniques can be replaced with some more "lightweight" techniques[1] at least for a certain set of applications, guaranteeing a better scalability while maintaining a high level of accuracy.

This paper is organized as follows. Section II presents a taxonomy of the methods that can be used in payload based traffic classification; Section III presents some results related to the evaluation of "packet-based" vs. "message-based" technologies, Section IV presents some conclusive remarks and a brief look at the future research directions.

II. PAYLOAD-BASED TRAFFIC CLASSIFICATION

Although the problem of determining the higher-layer protocol of a packet is commonly known as "traffic classification", in case of payload-based methods often it becomes a problem of "pattern verification". In the past, applications were detected by "classifying" the packet based on the value of a field (e.g. port 25 equal to SMTP traffic) [1]. These days, ports are meaningless on a vast portion of the traffic and the classification consists in guessing the application-layer protocol and verifying that this is correct by means of some appropriate controls (e.g. if the payload contains some well-known keywords). The difference between the payload-based technologies is due to the different types of controls that can be put in place, and the different processing methods that are required to perform these checks, as shown in Figure 1. The different complexity of these methods can be seen in the processing power that increases from left to right (e.g. the simplest method requires the parsing of L2-L4 headers, while the most complex one requires the processing of the application payload in all packets) and in the corresponding memory requirements.

[1] Currently, the accuracy that can be achieved with a "stateless" technique (which is basically a port-based one) is pretty low; therefore all the techniques that are in use nowadays require keeping a limited amount of per-session state, as shown in the following.

The following sections will show the problem of traffic classification from the point of view of the possible verification methods, the (currently unsolved) problem of application classification, and a list of processing methods that can be deployed within a payload-based classifier.

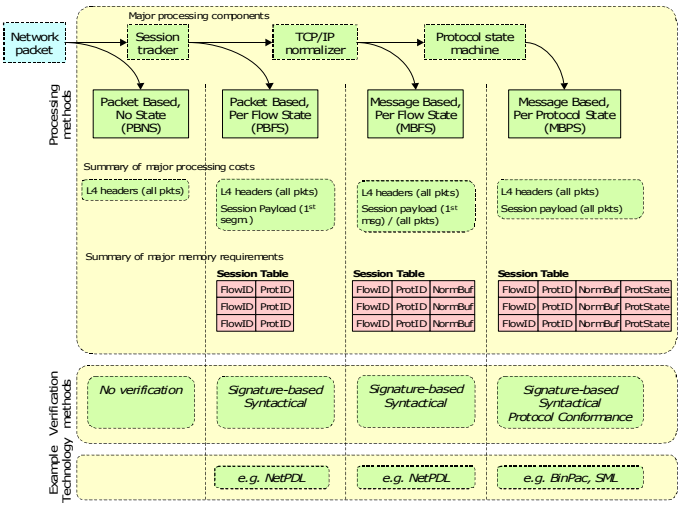

Figure 1. Payload-based traffic classification: different processing and verification methods.

A. *Verification methods*

With respect to traffic classification, we can envision four different degrees of verification.

A first degree of verification is the *signature-based* one, which aims at locating some signatures within the application-layer payload. For instance, an HTTP packet begins with a command followed by the URL and the protocol version, while most Edonkey packets start with a field containing the size of the payload. A signature-based method classifies the traffic by checking if (part) of the payload matches the signature defined for that protocol. Signatures are usually regular expressions, although they can include some more controls (e.g., a check over the length of the payload). A second degree of verification, the *syntactical* one, can be seen as a more accurate version of the signature-based one, since it aims at checking the correctness of the transmitted data from the syntactical point of view (e.g. a supposed HTTP payload must contain HTTP headers). A verifier should decode all the fields contained in the message and guarantee that the message is well formed. A third degree of verification relates to *protocol conformance*, e.g. controlling that an *HTTP GET Request* from a client is followed by a valid response from the server. This form of verification is more accurate because it can validate the run-time behavior of the protocol against its canonical state machine as defined by specifications. A fourth degree of verification refers to the *semantic* of the data, e.g., the possibility to verify whether an image object transferred by the HTTP protocol is in fact an image, or some other form of content. This verification is extremely useful to detect "smart tunneling" mechanisms, in which an application uses another protocol to transport its data. However, the authors are not aware of any technology that implements such level of verification, which is very complex indeed.

B. *Protocol vs. application classification*

One of the possible problems at this level is that users tend to mix up the concepts of protocol and application classification. For instance, it is not clear how to classify the traffic of a peer-to-peer application that uses HTTP for data exchange: the protocol is HTTP, but it is used for P2P traffic. In general, applications use protocols for their purposes: often applications define their own protocols (and this makes this problem trivial), but sometimes they use other existing protocols; in order words, as shown in Figure 2, application classification may be somewhat orthogonal to protocol classification.

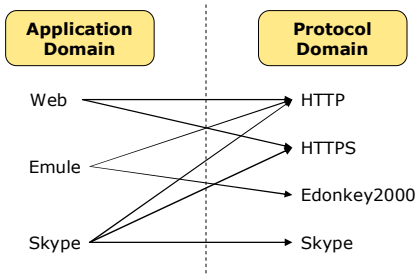

Figure 2. "Application domain" vs. "Protocol domain".

From the practical point of view, verification methods presented in the previous section are a solution to protocol classification but might not be appropriate for application classification, which, unfortunately, is what the user expects from a classification system. For example, the protocol conformance verification will fail in case of a P2P traffic exchanged through HTTP (the result will be that the traffic is HTTP, which is in fact correct from the protocol conformance point of view), while simplest methods, such as the signature-based one, can be better tuned to distinguish (e.g. by checking different values of the "user-agent" HTTP header) between web traffic and the HTTP generated by the P2P application.

While a better analysis of the problem of application classification is left to future work, the conclusion is that the verification techniques shown in the previous section, when taken alone, are not able to return the results expected by the users and must be integrated with some "tricks" that take into account the application domain. Therefore, it is not guaranteed that a technology based on the most advanced verification methods will be able to return the best results; in fact, vendors usually integrate these methods with some heuristics to classify traffic per-application instead of per-protocol.

The rest of the paper will consider systems that are able to perform application-based classification, mixing adequately the methods shown in Section II.A with heuristics.

C. *Payload-based processing methods*

A taxonomy of the possible approaches in shown in the portion of Figure 1 dedicated to processing methods. The simplest method is the Packet Based-No State (PBNS), which operates by checking the value of some fields present in each packet, such as the TCP/UDP ports. This method is very simple computationally (only packet headers until L4 have to be processed), it does not require state, but the results in terms of accuracy are inadequate on current traffic.

A second method is the Packet Based – Per Flow State (PBFS), which requires a session table in which each record includes the session ID (the 5-tuple IP source/destination, transport protocol, port source/destination) and the corresponding application-layer protocol (Protocol ID), which accounts for some tens of bytes each. This method supports the implementation of a verification mechanism based on application-layer data, although this is limited to a per-packet inspection. For example, it is able to scan the payload looking for a specific signature (e.g. "GET * HTTP"). The limited per-session state is required because the application signature may not be present on all the packets; hence the entire session must be "marked" as soon as one packet is detected that complies with the verification process. The processing complexity of this method increases because of the necessity to manage the session table, plus the payload inspection: it is worthy noticing that the latter is required only on a subset of packets (i.e., the first one(s) of each session).

A third method is the Message Based – Per Flow State (MBFS), which operates on messages instead of packets. This requires a TCP/IP normalizer[2] module that takes care of IP fragments (i.e. putting all the IP fragments together into a single IP packet) and TCP segments (i.e. creating a "virtual payload" by putting the required segments together and reconstructing the application-layer message). Basically, MBFS technologies can perform the same checks of the PBFS ones, but operate on messages; hence their controls can be extended over the entire message instead of the first segment of data. In such case, memory requirements increase because of the additional state information that must be kept for each session (e.g. TCP sequence number) and of buffers required by the TCP/IP normalizer, although these are highly dependent on the nature of the traffic, i.e., on the quantity of fragmented packets and "abnormal" (e.g., out-of-order, overlapped, missing segments, and more) TCP sessions. Processing requirements increase as well because at least the first message (which may span across different packets) on each session must be checked. Depending on the implementation, some products can perform syntactical verification on all the messages, therefore requiring a deep processing of the entire data exchanged during each session.

The fourth method is the Message Based – Per Protocol State (MBPS), which goes one step further by interpreting exactly what each application transmits and receives. A MBPS processor understands not only the semantic of the message, but also the different phases of a messages exchange (e.g. an HTTP GET must be followed by a valid response code from the web server, while MBFS does not implement such kind of correlation between messages) because it has a full understanding of the protocol state machine. Memory requirements become even larger, because this method needs to take into account not only the state of the transport session, but also the state of each application-layer session. Also processing power is the highest because the protocol conformance analysis requires processing the entire application data, while previous methods are limited to the first packet(s) within each session.

Real implementations based on the PBFS technology usually associate some additional state to each session in order to perform a more accurate classification. For example, some applications (e.g. Skype, VoIP) can be detected by checking a pattern over some consecutive packets, and this can be implemented by storing some additional data (usually a few bytes) in the session table. However, this method cannot provide deterministic results, because although PBFS can support cross packet analysis (and, to some extent, also protocol conformance) through this small additional state, the reality is that packets can arrive out of order, may be duplicated or missing. In this case, i.e., when the application-layer data stream is "corrupted" due to some error at packet level, a TCP/IP normalizer is required, hence falling in the MBFS case. Therefore, PBFS and MBFS are much more oriented to achieve high performances at the expense of the accuracy and some decisions (e.g. "this packet belongs to an HTTP session") may be proven wrong in case the same session is analyzed with a more sophisticated method.

Existing technologies do not fit exactly in this taxonomy, since the same technology can span over several categories. Particularly, technologies are usually differentiated in packet-based (such as NetPDL [9], NBAR [12]) and "stream-based" ones (such as SML [13], BinPac [11]). However, these technologies behave differently according to the different implementation; for instance, NetPDL and NBAR can behave as PBFS or MBFS depending on the presence of a TCP/IP normalizer in the processing data path.

Obviously, when moving from the simplest PBNS to the more sophisticated MBPS, requirements in terms of memory and processing power increases, with the advantage of a more accurate protocol classification. However, it is currently unknown to what extent such higher complexity is worthy, i.e., if the additional accuracy obtainable with the more sophisticated methods is balanced by the increased complexity.

III. Experimental Evaluation

A. Technologies under evaluation

The common belief is that a more sophisticated technology allows obtaining more precise results. However, as explained in Section II, a more sophisticated technology costs more in terms of memory and processing power; therefore is not suitable for high speed links. Moreover, it is currently unclear the worsening, e.g. in terms of accuracy, when moving from a more sophisticated technology to a simpler one.

In order to give an answer to this question, this paper compares the PBFS and MBPS technologies with respect to traffic classification. Particularly NetPDL (available in the NetBee library [14]) will be used as an example of PBFS technology, while for MBPS two implementations have been selected: one based on the BinPac language, available in the BRO Intrusion Detection System [15][16] and the other based on the SML language, available in the Cisco SCE box [17]. All

[2] The taxonomy presented in this section presents different processing methods without taking into account implementation details. Particularly, several implementations are available for a TCP/IP normalizer; the important point is that in case of MBFS methods this module must be present and must be able to return TCP/IP packets in such a way that the following modules can operate on the entire application-layer message, even if the message is split across different packets.

the technologies are based on a description language (packet-based for NetPDL, protocol-based for BinPac and SCE) that allows describing some aspects of network protocols, i.e., protocol format and, for the most advanced ones, the protocol state machine, and are included in a product that does packet/protocol processing based on that language. Due to some practical constraints (e.g., sources are available only for NetPDL and BinPac), analysis related to PBFS technology were done using used NetPDL, while MBPS analysis was based either on BinPac or SML, depending on the requirements of the planned test. More details about these technologies can be found in [10].

Obviously, each implementation has some differences compared to the set of verification technologies presented in Section II.C. For example, the SCE box implements the MBPS technology plus some additional features (not really documented) that belong to the application classification domain, and some statistical-based methods. Vice versa, Bro is a pure MBPS technology (with respect to protocols implemented in BinPac language) and it is not capable of application classification, at least in the code available on the web site at time of testing. These characteristics must be taken in mind when discussing the outcomes of our tests.

B. *Performance indexes*

Our tests aimed at evaluating the parameters shown in Figure 3, mainly the *coverage*, i.e., the number of protocols that are classified by a given technology, the *accuracy* and the *completeness* of the classification.

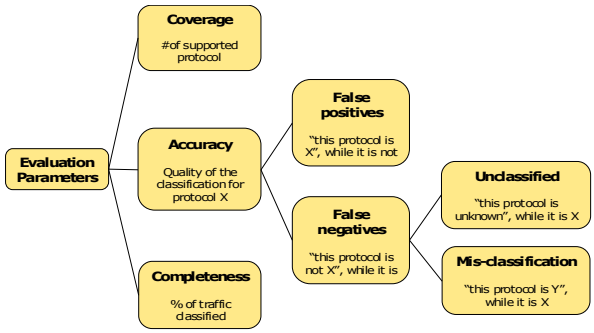

Figure 3. Parameters under evaluation.

Particularly, the accuracy is computed "per protocol" and is conveyed in terms of false positives (FP) and false negatives (FN) with a definition that is slightly different from the one found in literature, but it is more intuitive and makes easier to compare the results. In this paper a false positive for protocol X (*FP(X)*) occurs when a session contain protocol Y is erroneously classified as X. A false negative for protocol X (*FN(X)*) occurs when a session containing protocol X is erroneously classified as Y (FN-Wrong Classification), or it remains unclassified (FN-Unclassified). Particularly, the case of FN-Wrong is often more problematic because in some applications this can lead to dangerous behavior (e.g. a session that should be blocked by a firewall could get higher privileges instead), while the unclassified traffic (FN-Unclassified) is usually tolerated. With respect to the completeness, this parameter represents the amount of traffic that has been classified. This parameter must be taken with care because it does not distinguish between the traffic correctly classified and the one that is not, being equal to 1 minus the percentage of unclassified traffic.

C. *Methodology*

Tests have been made using both pre-classified traces organized per-application (each trace either contains only the traffic generated by that application, or a traffic that is similar but is generated by another application), synthetic traces (the traffic coming from a set of applications is duplicated several times, changing "randomly" IP addresses, ports and some of the most peculiar data in it, such as email addresses and more) and real traces captured on the link that connects our University to the Internet. Captured data was pre-processed in order to discard packets belonging to a session started before the beginning of the capture[3]; moreover, we discarded all the packets belonging to sessions that resulted idle for more than 5 minutes. Several tests demonstrated that this timeout was fair enough to keep the vast majority of the traffic. Table 1 shows an outcome of one of these tests, proving that an increased value for the timeout leads only to greater cost (e.g., session entries must be kept in memory for more time) without any other impact in the results of the test. Moreover, several real implementations (e.g. firewalls) use the same value as the standard inactivity timeout.

TABLE 1. PACKETS DISCARDED WITH DIFFERENT VALUES OF THE INACTIVITY TIMEOUT FOR TCP SESSIONS

Timeout	Total packets	TCP packets	TCP packets dropped (#)	TCP packets dropped (%)
1 min	14979286	14073381	3590186	25.51%
2 min	14979286	14073381	792968	5.63%
4 min	14979286	14073381	770051	5.47%
5 min	14979286	14073381	756899	5.38%
6 min	14979286	14073381	696864	4.95%
8 min	14979286	14073381	676638	4.81%
10 min	14979286	14073381	674401	4.79%

Pre-classified traces had a very limited amount of encrypted traffic, which cannot be classified with pure payload-based technologies. In all trace files, applications can use any port, e.g. it may happen to have some HTTP sessions on port other than 80; in addition, real traces may contain malicious traffic (although we did not introduce it explicitly). The technologies under evaluation are not required to detect a security attack (e.g. SYN flooding), but must be able to recognize traffic also on non-standard ports.

D. *Coverage*

The number of protocols supported by the technologies under testing is extremely different. NetBee currently supports about 100 application-level protocols. Among the MBPS technologies, the SCE supports more than 600 applications, while BRO supports a very limited number of applications,

[3] In fact, this is valid only for TCP traffic, in which we can easily detect the beginning of a session. In this preprocessing stage, all non-TCP traffic was left untouched.

namely DNS, RPC, DCE_RPC, HTTP, NCP, SMB[4]. Among these, only HTTP is classified even if traffic is not on the standard port; hence only the HTTP protocol will be suitable for our tests. With the exception of BRO, the other technologies support the vast majority of IETF protocols and a fair number of "tough" applications such as VoIP (e.g. SIP, H.323, Skype, Vonage, etc.) and P2P (such as Kazaa, Gnutella, Edonkey). From the point of view of coverage, MBPS and PBFS are mostly equivalent, provided that the current objective is application-layer classification and not application-layer protocol decoding, in which treating packets or messages makes a big difference.

E. Accuracy

The first set of tests involves the analysis of pre-classified traces (about 200MB of data), containing each one a set of packets belonging to a given protocol. These pre-classified traces have been analyzed with the SCE box and with NetBee. Results were a 100% accuracy for the SCE box, and near-100% accuracy for NetBee (details are shown in Table 2), with no false positives (which is expected, since pre-classified traces contained only packets belonging to the selected protocol) and a negligible amount of false negatives, almost all unclassified traffic. Among the unclassified packets, there was some encrypted traffic, some packets whose signature was split across different packets, or some part of the protocol not supported by the signature base (e.g., some Kazaa packets). Finally, some packets were unclassified because of the necessity to analyze several consecutive packets to detect the protocol (e.g. RTP); in this case, the first packets of the session appear as unclassified.

TABLE 2. ACCURACY WITH PREFLASSIFIED TRACES, USING NETBEE

Application	# Files	# Packets	% Packets Detected	FP	FN-W	FN-U
Edonkey	26	41426	99.04%	0.00%	0.77%	0.19%
Kazaa	27	38888	98.89%	0.00%	0.02%	1.09%
Gnutella	21	16407	99.96%	0.00%	0.00%	0.04%
HTTP	97	81290	99.87%	0.00%	0.00%	0.13%
FTP	25	49024	99.82%	0.00%	0.14%	0.04%
RTP	19	90643	99.21%	0.00%	0.00%	0.79%
SIP + RTP	27	467267	99.98%	0.00%	0.00%	0.02%

With respect to NetBee, we ran the tests several times, each time improving the signature base. This lead to the conclusion that the results are highly dependent on the quality of the signature (for example, NetBee did not implement a signature for some binary messages rarely used by Kazaa, and in fact the accuracy related to this protocol is the lowest among the tested set), and these results can be further improved with an appropriate tuning of the signatures. Protocols under testing were some among the most significant ones (peer-to-peer, voice, client-server); it is the authors' belief that a larger number of protocols will not change these results substantially, since the quality of the classification in NetBee (hence in PBFS technologies) depends mostly on the quality of the signatures.

[4] BRO supports a larger number of applications, but only these ones are supported through the BinPac language; therefore, only these application-layer protocols are checked with the various stages of verification until Protocol Conformance.

Figure 4 compares the classification results obtained by the SCE box and NetBee on four different sets of synthetic traces. Our tests show that these technologies have almost the same results, with the most part of the traffic (usually around 85%) either classified in the same way or unclassified. Differences (the three upper sections) are very limited, and mostly related to sessions that goes unclassified in one of these two technologies, bringing two different results only in about an average of 2% of the sessions. While this may be related to the quality of the signature database in NetBee, we have to say that, surprisingly, a manual inspection showed that most of the errors were in the SCE box.

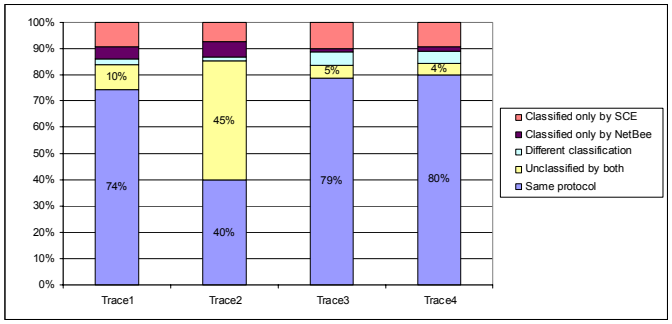

Figure 4. Comparing classification results between NetBee and SCE.

F. Robustness

One of the problems of the MBPS technology is that it does not tolerate losses of packets. This is the case of a network-related packet loss, but it may be the case of an overloaded classification device (e.g. some device on the market start discarding packets when the processing load approaches 100%), or in case of asymmetric routing, where packets within the same session follow different paths in the two directions.

A test was performed to demonstrate this problem. Traces containing HTTP sessions were analyzed with Bro and NetBee, and then results were compared "session-by-session". Sessions detected by both tools were assumed being HTTP without any further control, while sessions detected only by one tool were manually inspected to decide who was wrong.

The most important result was that, globally, NetBee was more accurate than Bro, and perhaps the most important reasons is related to the better robustness of the PBFS technology, as shown in the first two sections of Table 3. In fact, Bro showed a fair number of false negatives due to asymmetric routing (respectively, 17 and 29) that caused sessions to be captured only in one direction. This behavior is correct indeed from the point of view of MBPS technologies (if a packet is missing, the protocol state machine cannot be validated), but this points out that the physical deployment of these technologies must be done with care. For example, a MBPS classifier cannot be deployed in the backbone, or in networks with multiple exit gateways. A second problem was due to packet losses. Our sniffer was dropping a limited amount of packets during the capture, resulting in some "corrupted" HTTP sessions. NetBee correctly classified these sessions (3 in each test), while (as we may expect) Bro was not

able to mark them as HTTP[5]. On the other side, Bro was able to classify two HTTP sessions more than NetBee, due to the presence of a signature split across two packets. While, again, this is a known drawback of the PBFS technology, it is interesting to note that the number of sessions showing this problem was extremely limited (only four sessions) and, more important, two of them were correctly classified by NetBee. In fact, the last two sessions were related to the version 1.1. of the HTTP protocol, which can handle several transactions (request/response) in the same TCP session. It turned out that the first request was split across two packets, while the second one was contained within a single packet, and NetBee classified this session as HTTP starting from that point[6].

This result must be taken with care because the current analysis is far from being exhaustive, but the general impression is that this problem may be less important than we did expect. First, the number of sessions with split signatures may vary in different captures; second, this problem may be more evident in applications in which the signature is particularly complex, or in case of malicious (e.g. short) packets such as the ones used in some attacks. Another point is that this problem is exacerbated when only a snapshot of each packet is analyzed in order to decrease processing costs, and for these reasons we believe that this is an area in which some more investigations will be required.

It is worth noticing that all the false negatives in Table 3 were related to unclassified packets; in other words, no misclassifications were present in these files.

TABLE 3. CLASSIFICATION OF HTTP SESSIONS IN BRO AND NETBEE

	Trace1	Trace2	Comments
# HTTP sessions	22897	38018	
Classified by both	22875	37986	
False Negatives			
Bro	20	32	Asymmetric routing (17/29), drops (3/3)
NetBee	2	0	Signature split across packets (2/0)
False Positives			
Bro	7	13	Skype (6/11), Edonkey (1/2)
NetBee	0	0	

G. Application vs. protocol classification

The fact that the MBPS technology alone is not able to perform application classification (Section II.B) is evident from the last section of Table 3. In fact, Bro showed some false positives due to web-like traffic generated by peer-to-peer applications, which was classified as web traffic. Particularly, 7 (in the first test) and 13 sessions (in the second test) that were in fact generated either by Edonkey or Skype and encapsulated in HTTP, were classified by Bro as web traffic. This demonstrates our statement that a pure MBPS technology is still not enough in current application-based traffic classification because this technology is able to check only protocol conformance, while application classification requires some more controls.

[5] This demonstrates the higher robustness of the PBFS technology, which does not mean absolute robustness. For instance, PBFS cannot classify a session if the packet containing the signature is lost. The difference is that PBFS fails only if this packet is missing, while MBFS or MBPS will fail whenever a single packet within the session is lost, whatever this will be.

[6] The current implementation of NetBee analyzes all the packets within a session until a valid application-layer is found. Therefore, in case a session is still unclassified, all its packets are inspected looking for a known signature.

H. Completeness

Given the results that proved the accuracy of the PBFS technology (at least in case of the pre-classified traces), we ran a couple of tests on real traces to test the completeness of the classification. Results were extremely interesting: in both traces (captured in different days, and accounting for different groups of users) almost 97% of the traffic was classified (see Table 4), although the accuracy is unknown. It is worthy noticing that another capture done in February 2007 showed a completeness of about 70%, but a large part of the unclassified traffic was generated by Emule using obfuscation techniques (i.e., it was encrypted traffic), and this traffic cannot be classified with payload-based technologies.

TABLE 4. TRAFFIC CLASSIFIED BY NETBEE WITH TWO DIFFERENT REAL TRACES

	Classified	Unclassified
November 2006, 9.5 GB, 9.5 hours	96.80%	3.20%
May 2007, 7.8 GB, 22 minutes	96.55%	3.45%
December 2007, 461 GB, 12.5 hours	92.83%	7.17%

Figure 5 shows the results of the classification on the last trace, covering the traffic of the entire campus and spanning over about 12 hours. Results show an increase of the unclassified traffic, but we guess that a large part of this traffic is encrypted edonkey, which is unexpectedly low in our measurements, while we know (from other sources) that it should have higher share. Also interesting is the amount of SSH traffic, which is probably an indication of some tunneling mechanisms used to bypass network restrictions.

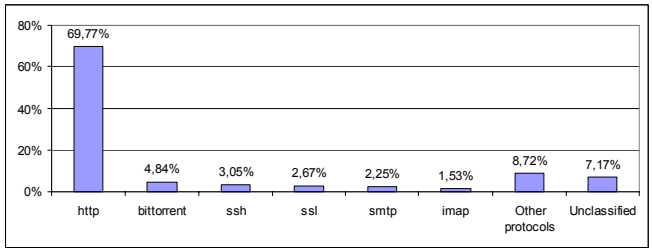

Figure 5. Protocol distribution (in number of bytes), with NetBee.

I. Scalability

While MBPS technology is known to suffer from the scalability point of view, our findings demonstrate that also the requirements of PBFS may be problematic in case of high-speed links. For instance, Figure 6 shows the cumulative distribution function (CDF) related to the number of TCP session entries required to classify some real capture traces, with the 5-min inactivity timeout defined in Section III.C. A couple of capture traces with an average traffic load of 2Mbps and about 100 subscribers required about 2500 entries in the

TCP session table for being able to classify the entire data. While this result does not have a general validity (and further analysis is required), it turns out that a multi-gigabit pipe may require millions of entries in the TCP session table, which may be far beyond the capability of any hardware platform. A more detailed study about this problem and possible solutions to mitigate these effects are left to future work.

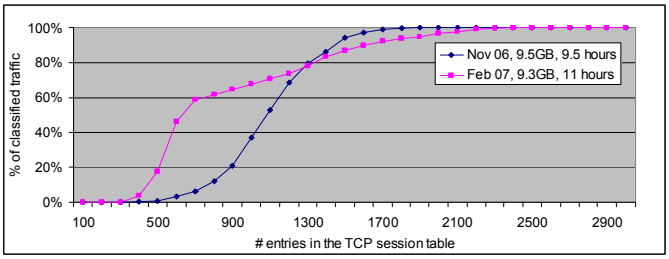

Figure 6. Number of TCP session entries required to classify the traffic.

IV. CONCLUSIONS

This paper brings two contributions. First, it presents a taxonomy of payload-based classification approaches, showing their characteristics, their strength and their weaknesses, and their processing and memory requirements. Second, it compares one implementation following the Packet Based – Flow State approach (often known as "lightweight" technology) with a couple of implementations following the Message Based – Per Protocol State approach (often known as "stateful" technology). Results show that, although the former cannot probably achieve the precision required in security appliances, differences in terms of coverage, accuracy and completeness are very limited at a fraction of the complexity (memory requirement, processing limited to the first few packets within each session) of the latter. In any case, MBPS technologies are not able to achieve the expected results in terms of application classification, which is one step further than protocol conformance and that requires specific mechanisms (e.g. application-specific signature-based classification). Moreover, tests pointed out a couple of additional interesting results. First, tests showed that some known problems of the per-packet approach (such as signature split across two packets) might be negligible on real traffic. Second, they showed that the packet-based technology is more robust (e.g. packet losses, asymmetric routing), and that the deployment (e.g. in terms of physical location) of "stateful" technologies must be done with care.

The tuning of the signature database is perhaps the most problematic issue that emerges from this work. In fact, all these packet classification technologies were increasingly relying on signatures for protocol detection, and the accuracy depends on the quality of these signatures. However, some protocols are not well documented (or not documented at all) and the choice of the best signature often makes the difference.

This paper answers to some question, but several new areas of investigations are emerging from current results. Among the others, a more detailed evaluation of the accuracy in case of real traces, the problem of further decreasing the state in PBFS technology, the accuracy obtainable when decreasing the amount of data processed (e.g. analyzing only a snapshot of the packet) in PBFS, and the problem of application (and not protocol) classification.

V. ACKNOWLEDGMENT

The authors wish to thank Flavio Bonomi, Satish Gannu, Bob Olsen, Ravid Sagy, Mario Ullio and Vinicio Vercellone, for their comments and their many suggestions, and Niccolò Cascarano, Christian Novello and Gianpaolo Surfaro who spent part of their graduation thesis working on this topic.

REFERENCES

[1] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy. The CoralReef Software Suite as a Tool for System and Network Administrators. In *Proceedings of the 15th USENIX conference on Systems Administration (LISA 01)*, pages 133–144, San Diego, CA, USA, December 2001.

[2] A. W. Moore, K. Papagiannaki. Toward the Accurate Identification of Network Applications. In *Proceedings of the 6th International Workshop on Passive and Active Network Measurement (PAM 2005)*, Boston, MA, USA, March 2005, Pages 41-54.

[3] S. Sen, O. Spatscheck, D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web (WWW 04)*, pages 512–521, New York, NY, USA, 2004.

[4] P. Haffner, S. Sen, O. Spatscheck, D. Wang. ACAS: Automated Construction of Application Signatures. In *Proceeding of the 2005 ACM SIGCOMM workshop on Mining network*, pages 197 – 202, Philadelphia, PA, USA, 2005

[5] A. W. Moore, D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 50–60, Banff, Alberta, Canada, June 2005.

[6] J. Erman, M. Arlitt , A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281-286, September 2006, Pisa, Italy.

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

[8] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of ACM SIGCOMM*, pages 229–240, Philadelphia, PA, August, 2005.

[9] M. Baldi, F. Risso. NetPDL: An Extensible XML-Based Language for Packet Header Description. In *Elsevier Computer Networks Journal (COMNET)*, Volume 50, Issue 5, Pages 688-706, April 2006.

[10] F. Risso, A. Baldini, F. Bonomi. Extending the NetPDL Language to Support Traffic Classification. In *Proceedings of IEEE Globecom 2007*, Washington, D.C, USA, November 2007.

[11] R. Pang, V. Paxson, R. Sommer, L. Peterson. Binpac: a yacc for writing application protocol parsers. In *Proceedings of the 6th ACM SIGCOMM on Internet Measurement*, pages 289-300, Rio de Janeiro, Brazil, October 2006.

[12] Cisco Systems. *Network Based Application Recognition (NBAR)*. Available at http://www.cisco.com/en/US/products/ps6616/products_ios_protocol_group_home.html.

[13] O. Reviv. Inside network programming with SML. *EE Times*, August 2003. Available at http://www.eetimes.com/story/OEG20030818S0077.

[14] Computer Networks Group (NetGroup) at Politecnico di Torino. *The NetBee Library*. August 2004. Available at http://www.nbee.org/.

[15] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, 1999.

[16] Lawrence Berkeley National Laboratory. *Bro Intrusion Detection System*. Available at http://www.bro-ids.org/.

[17] Cisco Systems. *SCE 2000 Series Service Control Engine*. Product literature at http://www.cisco.com/en/US/products/ps6151/index.html.