Efficient Application Identification and the Temporal and Spatial Stability of Classification Schema

Wei Li

Computer Laboratory, University of Cambridge

Marco Canini

DIST, University of Genoa

Andrew W. Moore

Computer Laboratory, University of Cambridge

Raffaele Bolla

DIST, University of Genoa

Abstract

Motivated by the importance of accurate identification for a range of applications, this paper compares and contrasts the effective and efficient classification of network-based applications using behavioral observations of network-traffic and those using Deep-Packet Inspection.

Importantly, throughout our work we are able to make comparison with data possessing an accurate, independently-determined ground-truth that describes the actual applications causing the network-traffic observed.

In a unique study in both the spatial-domain: comparing across different network-locations and in the temporal-domain: comparing across a number of years of data, we illustrate the decay in classification accuracy across a range of application-classification mechanisms. Further, we document the accuracy of spatial classification without training-data possessing spatial diversity.

Finally, we illustrate the classification of UDP traffic. We use the same classification approach for both stateful flows (TCP) and stateless flows based upon UDP. Importantly, we demonstrate high levels of accuracy: greater than 92% for the worst circumstance regardless of the application.

Key words: Traffic classification, Application identification, Deep packet inspection, Machine learning, Temporal decay, Spatial stability.

1 Introduction

Network-security, accounting, traffic engineering, and new class-of-service offerings — each is an example of a network-service facility that is made possible by the accurate identification of network-based traffic. Another example of the need for accurate identification would be the desire of some Internet Service Providers to cope with the continual rise of peer-to-peer (P2P) usage by throttling network traffic identified as P2P file-downloading. The challenges posed for organizations faced with the development of new types of network usage focuses attention on the use of application-identification and the need to do so accurately.

In this paper we explore the effective and efficient classification of network-based applications using only the observed network-traffic. We conduct a four-way comparison of application-identification methods: contrasting common industry-standard methods such as known port numbers and Deep-Packet Inspection with Naïve Bayes and the C4.5 decision-tree method.

We conduct an assessment of our method using real traffic collected over a number of years on two different sites. This enables both assessment of the individual accuracy and, more usefully, the temporal and spatial stability of our models. The temporal stability measures how accurately traffic is being identified for each period, thereby revealing how accurate a method will remain over time. The spatial stability describes how models will perform across different sites. Both assessments serve to provide insight into what is required to build models that will remain accurate regardless of such time and location heterogeneity.

1.1 Motivation

Traffic classification remains a fundamental problem in the network community. As noted above, this supports numerous network activities from network management, monitoring, Quality-of-Service, to providing traffic models and data for simulation, forecast and application-specific investigations.

However, the continual growth in the diversity of applications, number of hosts and traffic volume on the Internet has been a challenge for methods to classify network traffic according to the applications. The level of continuous development is only predicted to continue to grow.

Despite a number of approaches on traffic classification in the past, for example,

Email addresses: wei.li@cl.cam.ac.uk (Wei Li), marco.canini@unige.it (Marco Canini), andrew.moore@cl.cam.ac.uk (Andrew W. Moore), raffaele.bolla@unige.it (Raffaele Bolla).

[1] This work was done while Marco Canini was visiting the Computer Laboratory, University of Cambridge.

[1,2,3,4,5,6], the problem continues to be a significant challenge. In large part this is because:

(1) Internet traffic is not easily classified using the standard International Assigned Number Authority (IANA) port number list [7]. Emerging applications and proxies often avoid the use of standard port numbers.
(2) Further, port numbers and protocol signatures may be insufficient to determine the actual application. There is, in principle, no rigid bound between applications and the underlying protocols. For example, applications such as MSN Messenger, BitTorrent and Gnutella may explicitly use the HTTP protocol and port 80; while Skype clients may operate servers on ports 80 and 443.
(3) There is a growing proportion of encrypted or encapsulated traffic. Examples include SOCKS proxies, VPN and VPN-like solutions, tunneling and applications that encapsulate data using a different protocol (e.g., GetByMail [8]). Encapsulation will change the patterns in the original protocol, while packet encryption renders the payload inspection mechanisms unusable.

1.2 Classification of Network-based Applications

While relevant to all classification work, a discussion of the precise nature of classification and the many applications of classification is beyond the scope of this paper. However, we feel it appropriate to mention that the classification may take a variety of forms — often related to how the class information is being used.

The uses of classification-systems may also have different requirements dependent upon different fidelity of recall (different numbers of classes) and different levels of precision. An operation-console system may be tolerant of a 10% error rate but a scheme for application-differential billing may need a large number of classes and a high precision.

A motivation of this work is the classification of network application traffic into discrete application categories in a real-time environment. Our specific underlying motivations for such classification is to (a) monitor the application-mix in the traffic using a low-overhead (payload-less) mechanism, (b) enable application-specific handling of traffic, and to (c) derive application-specific traffic models and study the impact of each type of applications on the network.

Further, we focus on application categories instead of a particular protocol or application. This is because: (a) a protocol only implements the design of an application but is not bounded by it; one can design a range of network protocols that implement the same functionality, the HTTP protocol is an example of this where people use HTTP as a transport service for many different purposes, and (b) there is ample opportunity for change and adaptation in the specific details of an application over time. There exists considerable variety among both the different implementations of an application and variety across the different protocol behaviors that may be used within an application. However, it is our assertion that an application exhibits a behavior or family of behaviors that are specific, and recognizable, irrespective of the implementation specifics. It is not desirable to be specific about each protocol or each application since they may change in popularity, design or simply disappear from use. Instead, we categorize each application into a modest number of related classes, which can cover all the Internet traffic according to the application's purpose (e.g., classify to CHAT rather than to "GoogleTalk"). We hope the classifiers and the models thus abstracted from the traffic will capture the inherent nature of the types of applications, and exhibit both a high level of accuracy and a predictable level of stability over time.

In our classification scheme, the basic object is a flow defined as a bi-directional session between two hosts uniquely identified by the IP five-tuple {protocol (UDP or TCP), client-IP, server-IP, client-Port and server-Port}. The TCP, being connection-oriented, cleanly defines the beginning and end of flow. A flow may be further delineated based upon which end is server or client: observation of the SYN packet is sufficient to determine this. For UDP there is no specific determinate for the start and end of a flow. A timeout mechanism, like that of Cisco NetFlow, is used to provide boundaries for flows based on the state-less UDP protocol or for flows of TCP where explicit flow-start or termination is not observed.

We define a number of mutual-exclusive application categories, such as WEB-browsing, BULK transfer, MAIL activities, etc. The first column of Table 4 reports the full list of categories. This kind of taxonomy has been previously used in [7] and [9]. It has been updated following the rise in popularity or emergence of new application types such as Voice-over-IP (VoIP) and instant messengers. It is intended that every network application would fall into one of the categories. Our goal is to be able to label each flow with the category of the application to which it belongs. The label will depend only upon the actual application that generates the traffic: for example, a flow carrying an advertisement used by Napster would be labeled as P2P, even though it may use the HTTP protocol.

1.3 Contributions

This paper provides a number of unique contributions.

- We conduct a comparison among a number of classification schemes. Significantly, we only use data where we have already established the ground-truth: the actual applications causing the network-traffic observed.
- We train accurate models from data with a high-confidence ground-truth and an efficient, small, feature set derived from the reduction of a large flow feature list using a sophisticated feature selection mechanism.
- We demonstrate the use of sub-sampling using start-of-flow to enable online classification; further, we demonstrate the resultant online classification method

has high accuracy.

- We carry out a unique temporal and spatial stability study on the four classification methods using data collected over a four year period and from two different sites.
- In contrast with much of the past work we examine the use of multiple-application classifiers for both stateful flows (TCP) and stateless flows based upon UDP.
- We document the computational complexity and measure the costs within the classification pipeline.

2 Network Traffic Data

We examine network-data from two different sites. These two sites are both *research-centric* but conduct research in very different disciplines and are located in two different countries. We refer to these two different sites as Site A and Site B. Each of Site A and Site B have over a thousand local users from a population of researchers, administrators and technical support staff. From Site A, we use three day-long data-sets taken on three weekdays in 2003, 2004, and 2006 (denoted as Day1, Day2, and Day3 respectively). From Site B, we use one data-set recorded on a weekday in late 2007 (denoted as SiteB). In both cases the sites are connected to the Internet via a Gigabit Ethernet link. Each data-set captures full-duplex traffic at the site border to the Internet.

For this paper, we selected 10 non-overlapping, randomly distributed periods, each approximately 30 minutes long, from each of Day1 and Day2. We also randomly selected two 30 minute periods from Day3, and a 30 minute periods from SiteB. Table 1 summarizes the data-sets volume breakdown by IP protocol type.

Further, as the TCP protocol semantic allows for a precise definition of start and end of a flow, we concentrate upon complete TCP flows: those for which both the SYN handshake and FIN handshake packets are observed — thus we avoid the bias from flows which have started before the selected period. We recognize the incomplete flows have included a number of long flows (e.g., FTP downloads) however, we are confident the use of such censored data has not overly impacted our com-

Data-Set	Packets [%]			Bytes [%]		
	TCP	UDP	Other	TCP	UDP	Other
Day1	93.49	1.60	4.91	98.18	0.72	1.10
Day2	97.87	1.59	0.55	99.33	0.42	0.25
Day3	96.91	2.19	0.90	98.16	1.44	0.41
SiteB	91.22	8.24	0.54	97.94	1.97	0.09

Table 1
Data-Set duration and volume breakdown by IP protocol type.

	Data-Set	Duration	Flows	Packets	Bytes
TCP	Day1	10 × 30 min	377K	42M	31GB
	Day2	10 × 30 min	175K	35M	28GB
	Day3	2 × 30 min	260K	30M	18GB
	SiteB	1 × 30 min	250K	11M	7.1GB
UDP	Day1	1 × 30 min	25K	197K	41MB
	Day3	1 × 30 min	46K	592K	242MB
	SiteB	1 × 30 min	774K	1.6M	180MB

Table 2
General workload dimensions of our data-sets.

parisons due to the small amount of total data by packet or byte that it represents. We also observe that incomplete TCP flows are often composed of various kinds of scans and unsuccessful connection attempts resulting in single SYN packets and no payload data. This is a common phenomena in the Internet, part of what may be referred to as the *Internet Background Radiation* and is well described in [10].

Table 2 lists the durations and workload dimensions of our data-sets for complete TCP traffic and UDP traffic.

TCP traffic constitutes the great majority of the traffic in our data-sets, thus in the following we focus upon TCP traffic to investigate the classification schemes. We show in Section 4.6 the classification of UDP traffic using similar techniques as those introduced for TCP.

2.1 Traffic Features

A feature is a descriptive statistic to characterize an object; and, ideally, each object exhibits different feature values depending on the category to which it belongs. Based on the features, models can be established using machine learning techniques.

The features of a flow that we use for classification are completely derived from the packet headers: UDP, TCP and IP. These features describe the general behavior of a flow, for example, the size of transferred data in either direction, the packet size and inter-arrival time distributions, entropy in the flow, and the first 10 components by FFT of packet inter-arrival times. Features include some higher level heuristics such as the data exchange behavior in forms of flow-idle, keep-alive, interactive, or transferring data from one end to the other. The complete feature set is fully described in [11], and a sample of features are listed in Table 3.

Figure 1 shows how different types of service exhibit different behavior in two groups each of two features: (1) variance of total bytes in packets (client to server)

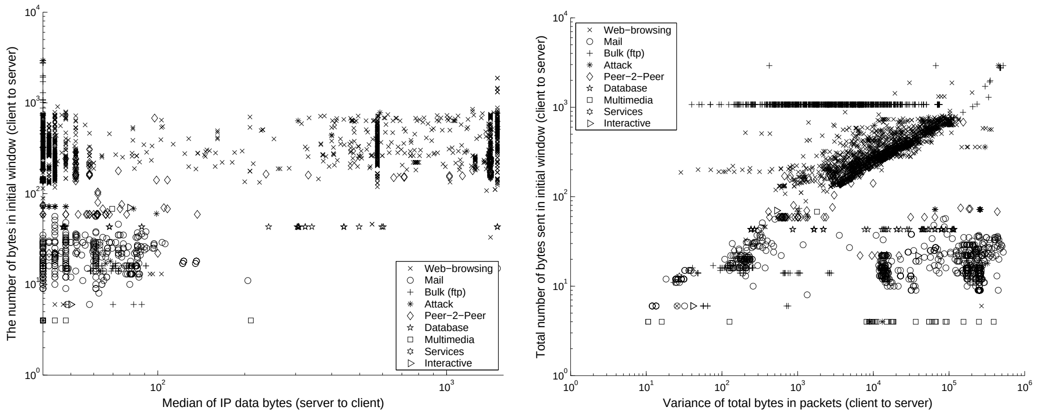


Fig. 1. Scattered plot of 5,000 random selected samples from Day1.

by the total number of bytes sent in the initial window (client to server), and (2) count of packets with PUSH flag set in the TCP header (server to client) by minimum segment size (client to server). There is a clear opportunity to discriminate between the flows of each application class using a combination of these features.

Collecting each feature from live traffic is associated with a computational cost equal or less than $O(n \times \log_2 n)$, and a memory footprint equal or less than $O(n)$, where n is the number of packets in a flow used for extracting the feature. The total cost of collecting K features is bounded to $O(K \times n \times \log_2 n)$.

In order to improve the classifier performance and to reduce the computational cost of the classification work-flow, we select a feature subset. The selection criterion chooses features that are most relevant for the discrimination of application classes while having the minimum redundancy with respect to each other. This feature-reduction process is fully described in the Section 3.5.

We applied the correlation-based filtering mechanism to each of the ten Day1 periods. We observed that the feature subsets selected by the algorithm possess moderately good stability, and we manually picked 12 features which appear in at least one third of the feature subset. Table 3 gives a list of the feature subset. The selection-criteria here is to identify the best-possible feature set which is both stable over time and independent of the location in the network. The feature subset are almost entirely dependent upon the actual applications on the end-hosts. Thus any classification model is able to maintain its accuracy over time and be applied in different network locations.

Without using the IANA port list or any prior knowledge of port-application mapping, we still use port numbers as two of the feature-subset. As our classifiers are built upon pre-classified data — for which we know the ground-truth — the port-number features used in the final classifier will maintain the association (as actually observed) between pre-classified application classes and port numbers.

Abbreviation	Description	SU	Memory Overhead	Computational Complexity
push_pkts_serv	Count of all packets with push bit set in TCP header (server to client)	0.3165	O(1)	O(n)
init_win_bytes_clnt	The total number of bytes sent in initial window(client to server & server to client)	0.2070	O(1)	O(1)
init_win_bytes_serv		0.3422	O(1)	O(1)
avg_seg_size_serv	Average segment size: data bytes divided by #packets. (server to client)	0.3390	O(1)	O(n)
IP_bytes_med_clnt	Median of total bytes in IP packet (client to server)	0.2011	O(n)	O($n \times \log_2 n$)
act_data_pkt_clnt	Count of packets with at least 1 byte of TCP data payload (client to server)	0.1722	O(1)	O(n)
data_bytes_var_serv	Variance of total bytes in packets (server to client)	0.2605	O(n)	O(n)
min_seg_size_clnt	Minimum segment size observed. (client to server)	0.2131	O(1)	O(n)
RTT_samples_clnt	Total numbers of RTT samples found (client to server), see also [11]	0.2434	O(1)	O(n)
push_pkts_clnt	Count of all packets with push bit set in TCP header (client to server)	0.2138	O(1)	O(n)
serv_port	Server Port	0.8378	O(1)	O(1)
clnt_port	Client Port	0.0760	O(1)	O(1)

Table 3
Properties of the subset of TCP features selected using 5-packet observation window. SU is the Symmetrical Uncertainty measurement, as fully described in Section 3.5.

2.2 Ground Truth

Ground-truth information is fundamental when assessing traffic classification mechanisms and to provide trustworthy results. Therefore, we have given a significant attention to obtaining accurate ground-truth for our comparison data-sets. Every flow in each data-set has been labeled with a corresponding application category. This is done using a (human) supervised semi-automated, heuristic-based, data verification process. The process is detailed in [12] however, we provide an overview of the process in this section. The use of a supervised, semi-automated, procedure does not replace the manual verification process but supplements it — allowing identification of the ground-truth without sacrificing confidence.

The procedures for the computation of ground-truth are based upon a variety of sources of the information about each flow. However, as detailed below, none of these pieces of information are used in isolation — cross-validation forms a critical part in the establishment of ground-truth. Without enumerating the derivative information: multiple-flow behavior or host-rôles, base sources of data include (in no particular order):

- packet payload contents,
- well known port numbers,
- well known host names and addresses (e.g., *ftp.kernel.org*), and
- background information about particular users, hosts, and specific application

behavior, e.g., P2P-networks.

We observed that flows belonging to the same service or application often share a subset of the IP five-tuple, notably, the {DstIp, DstPort}, {SrcIp, DstIp}, and {SrcIp, SrcPort} sub-tuples. Such an observation allows us to consider[2] flows bearing the same sub-tuples may belong to the same service or application. With this, the traffic can be classified at a high level of aggregation, which may significantly accelerate the process. The consistency between the application label and the sub-tuples was validated using two day-long traces which have been previously classified with the methodology described in [7], and using several segments of new hand-classified data. The assumption holds for most cases with the exception of (1) tunneled traffic (e.g., VPN or SOCKS proxies), (2) multiple applications between a server and a client (e.g., a server operating both mail and web services), and (3) HTTP traffic where the HTTP protocol is used to relay non-web traffic (e.g., BitTorrent signaling traffic). Aside from a VPN subsystem where the application was independently established, we didn't observe other tunneled traffic in our data. However, in future such traffic that could be grouped into its own category. For (2), we observe that the {SrcIp, DstIp} sub-tuple can complement the {DstIp, DstPort} sub-tuple after the latter has been used to identify standard services. We addressed (3) by verifying (through manually derived signatures) all HTTP requests and responses to find out what type of content they carry and what kind of applications have been causing them. This process was documented in [13].

A set of payload signatures (derived from *l7-filter* [14]) was used to provide an initial indication of an application based upon its packet content. Extensive tests were carried out to evaluate the signatures based on previously hand-classified data against several segments of new data. These results allowed us to tune the signature set making it capable of identifying 35 of the most popular protocols. Of course, the signatures only provide one piece of evidence that needs to be cross-validated within the process that determines the ground-truth.

The verification process is accelerated by exploiting a number of heuristic rules, (e.g., tuple-based rules), and by working upon larger traffic aggregations (e.g., services rather than individual flows) whenever possible. However, we followed the principle of making decisions only with very high confidence, e.g., when superior evidence from two or more mutually-independent information sources are validated.

Firstly, we consider whether the signature matching results for a specific server:port appear to be strongly consistent, in which case we can reasonably assume that we have identified a particular service on that server:port. Several criteria are used to quantitatively justify the consistency: thresholds are specified to guarantee that at least a certain percentage of flows as well as a minimum number of flows have

[2] We re-iterate that this process relies on heavy human supervision. For example, human-supervision takes the form of verification/confirmation of a sample of flows that such inferences are true and valid.

matched a specific signature. Additionally, only an exclusive subset of signatures is allowed to have matched the flows. This avoids situations where, for example, HTTP caused by BitTorrent being labeled as web traffic. We applied this heuristic widely and particularly for those applications with a well-established signature.

Based on the assumption that flows between the same IP addresses pair may be due to the same application, we can verify such situations as FTP traffic between two hosts or HTTPS traffic as in many cases a web server runs both standard and secure HTTP services. We derived similar heuristics for streaming and VoIP applications: for example, RTSP appear within a TCP control channel while the data are relayed on a unidirectional UDP stream; while a VoIP application may use a SIP session and a bi-directional UDP stream.

A great amount of information can be inferred from the host names. We base further heuristics on accumulated knowledge about particular popular services (e.g., Google, MSN, eBay): for example, we may have a heuristic that indicates HTTPS traffic to MSN servers is due to MSN messenger clients instead of browsers.

Finally, we consider the behavioral characteristics of hosts may be (subject to additional verification) due to overlay networks. This is particularly useful for the identification of P2P traffic. Although a counter-example to this rule is that SMTP traffic has a strong P2P-like behavior: SMTP servers may act as both the recipients and the senders of emails. The assumption is that if a host is an SMTP server, all the flows generated from this host towards port 25 are mail traffic. In general, this heuristic is applicable for P2P traffic as long as the information about the port number can be utilized[3], and the assumption of the heuristic can be validated. We applied this heuristic to verify a large number of eDonkey and BitTorrent flows on port 4662 and 6881 respectively. Additionally, for P2P applications that use random port numbers, we started from an initial set of identified peers and identified new peers that are contacted by previously known peers. For example, initial Skype participants were identified using the information of centralized login servers, and new Skype nodes could be inferred by considering the hosts reached by a number of (e.g., at least three) known Skype peers.

We show the application category breakdown of TCP traffic for each data-set in Table 4. As noted above, the *meta*-classes described in this table aggregate a number of specific applications — a full application breakdown is provided in [12] — although WEB deserves special comment. Throughout our work we refer to the application-class as meaning web-browsing alone. It is worth reiterating that this class of traffic does not refer to data moved via HTTP, nor to non-browser applications (such as a webmail server providing data to Outlook — such data would be classified as MAIL.)

[3] We observed that many P2P nodes still use the well-known port numbers.

Class	By flows [%] / packets [%] / bytes [%]				Applications
	Day1	Day2	Day3	SiteB	
WEB	84.558 / 22.529 / 29.438	80.198 / 16.383 / 17.623	84.077 / 27.381 / 25.456	85.557 / 70.876 / 69.456	Web browsers, web applications
MAIL	8.682 / 6.777 / 7.904	9.384 / 1.884 / 1.763	1.530 / 0.867 / 0.811	4.377 / 7.506 / 6.683	IMAP, POP, SMTP
BULK	3.800 / 69.483 / 60.569	6.146 / 80.850 / 79.372	2.058 / 67.201 / 69.881	0.223 / 8.310 / 11.905	FTP, wget
ATTACK	0.787 / 0.084 / 0.132	0.562 / 0.016 / 0.002	0.013 / 0.002 / 0.001	1.614 / 0.578 / 0.075	Port scans, worms, viruses, sql injections
CHAT	- / - / -	- / - / -	0.025 / 0.013 / 0.004	0.204 / 0.195 / 0.059	MSN Messenger, Yahoo IM, Jabber
P2P	0.589 / 0.331 / 0.567	1.572 / 0.548 / 0.777	8.571 / 0.745 / 0.433	7.188 / 10.455 / 9.735	Napster, Kazaa, Gnutella, eDonkey, BitTorrent
DATABASE	0.862 / 0.387 / 0.703	1.483 / 0.253 / 0.400	3.531 / 3.589 / 3.196	- / - / -	MySQL, dbase, Oracle
MULTIMEDIA	0.137 / 0.112 / 0.196	0.002 / 0.000 / 0.000	0.007 / 0.105 / 0.145	0.004 / 0.165 / 0.232	Windows Media Player, Real, iTunes
VOIP	- / - / -	- / - / -	0.036 / 0.018 / 0.002	0.420 / 0.483 / 0.169	Skype
SERVICES	0.555 / 0.135 / 0.194	0.633 / 0.026 / 0.009	0.027 / 0.003 / 0.000	0.187 / 0.078 / 0.041	X11, DNS, IDENT, LDAP, NTP
INTERACTIVE	0.027 / 0.156 / 0.285	0.021 / 0.040 / 0.053	0.124 / 0.076 / 0.071	0.128 / 1.261 / 1.552	SSH, TELNET, VNC, GotoMyPC
GAMES	0.002 / 0.006 / 0.013	- / - / -	- / - / -	0.060 / 0.027 / 0.007	Microsoft Direct Play
GRID	- / - / -	- / - / -	- / - / -	0.037 / 0.066 / 0.084	Grid computing

Table 4. Composition of TCP traffic in our data-sets. Applications shown are only examples for demonstration. “-” denotes non-existing traffic.

3 Classification Methodologies

Several methods exist for classifying network-traffic and all of them fall into two broad classes: deterministic (hard) and probabilistic (soft) classification. As the name suggests, deterministic classification assigns data points to one of several mutually-exclusive traffic-classes. This is done by considering some metric that defines the distance between data points and by defining the class boundaries. On the other hand, probabilistic classification method classifies data by assigning it a per-class membership-probability. The per-class membership-probability may be based on an artificial allocation of probabilities or be based upon *a priori* experience. For example after trying to classify a flow using this method, it could be observed that with a probability of 0.8 this flow belongs to the MAIL class, with probability 0.1 to WEB class and with probability 0.1 to BULK class. Class assignment is done by considering the class with the largest probability. In the example, the flow will then be considered a member of the MAIL class.

In this paper, we document four methods for network-traffic classification. Two methods: Port-number and L7, a classifier based upon deep-packet inspection, are both deterministic. While two alternatives: Naïve Bayes and the use of a C4.5 are probabilistic in nature.

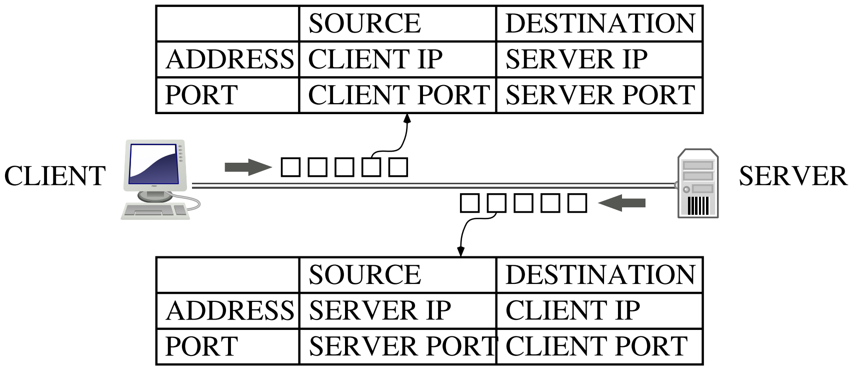
We primarily focus on using C4.5 to capitalize on the following properties of the decision tree algorithm:

(1) Sufficient accuracy. It is suggested in [6] that C4.5 is among the most accurate method for the problem of traffic classification. Further, we find it the most accurate one in a range of most popular supervised learning algorithms, using our particular feature set and observation window setup.

(2) Extremely low computational cost for classification. Its classification process only involves several conditional statements, which is the simplest form that a classifier can be. This property ideally supports time and resource-critical tasks such as real-time application operations.

Alongside the two probabilistic classification schemes, we also utilize a feature-reduction mechanism: the Fast Correlation-Based Filter. This mechanism allows us to significantly reduce the computational cost in collecting the features and also helps to overcome the overfitting problem in each of the probabilistic schemes.

We also note that our traffic classification work relies upon the aggregation of packets into flows based upon the IP five-tuples. We are aware of sophisticated data structure and algorithm work such as [15]. Further, work in [16] shows that minimal hardware can permit tractable implementation for handling 10 Gigabit/s live traffic. While not within the scope of this paper, such approaches combined with our methodology would effectively yield a line-rate real-time traffic classification solution.



	SOURCE	DESTINATION
ADDRESS	CLIENT IP	SERVER IP
PORT	CLIENT PORT	SERVER PORT

	SOURCE	DESTINATION
ADDRESS	SERVER IP	CLIENT IP
PORT	SERVER PORT	CLIENT PORT

Fig. 2. Relationship between client and server and source and destination port numbers.

3.1 Port-number

The port-based method relies on the use of well known ports: the server port number maybe used to identify traffic associated with a particular application.

This method sees common use as state of the art in many operational environments and requires access only to the part in the packet header that contains the port numbers.

As illustrated in Figure 2, one should note that the server port number is not equivalent to the destination port number of the packets. They coincide only for packets sent in the client to server direction. For example, the destination port number is 80 for a packet directed to a Web server, while the source port number is 80 for a packet sent from a Web server. However, port based classification uses the server port number which is determined by observing the first packet of a flow in the client to server direction.

In this paper, we use a list of well known ports derived from the official port assignments established by IANA. In particular, we consider only the official ports for port numbers $\leq$ 1023. In this subset only 35 distinct ports appear in our traffic traces, and for just 16 of them the traffic is actually using the protocol associated to the official port assignment.

3.2 Deep-Packet Inspection

The Deep-Packet Inspection (DPI) method examines whether a flow carries a well known signature or follows well known protocol semantics. Such operations are accompanied by higher complexity and may require access to more than a single packet's payload. According to [7], specific flows may be classified positively from their first packet (with payload data) alone. Nonetheless, other flows may need to be examined in more detail and a positive identification may only be feasible once up to 1 KByte of payload data has been observed.

Signatures can be obtained from protocol specifications. This is relatively easy for open and published protocols. However, proprietary protocols are often neither open nor published, thus signatures must be derived from reverse engineering of the protocols, (e.g., [17]). Such a process is arguably going to produce signatures that do not fully capture the underlying protocol semantics, yielding to inaccurate estimates of the traffic associated with those protocols.

Aside from the need for payload access, a major drawback of the payload-inspection method is that it can not handle traffic with encrypted payloads. This has become increasingly problematic as common and increasingly popular applications turn to the use of encryption to evade detection by deep-packet inspection techniques.

To classify the traffic in our traces we use the identification mechanisms of the open source DPI tool *l7-filter* [14]. Because this tool is not intended as an off-line, trace processing tool (it is intended to be deployed as part of the Linux iptables firewall for traffic shaping purposes), we use a user-space version of this tool [18]. We refer to this off-line version as L7. This classifier re-assembles the data content of a flow and identifies the application via pattern matching using regular expressions. A flow is marked as identified as soon as a known pattern is found in at least one of its directions. Only the first 10 packets of each flow are considered.

3.3 Naïve Bayes with kernel estimation

In order to describe Naïve Bayesian classification it is useful to consider a data sample $\mathbf{x} = (x_1, \ldots, x_n)$. This is a realization of $\mathbf{X} = \{X_1, \ldots, X_n\}$ such that each random variable X_i is described by m attributes $\{A_1, \ldots, A_m\}$ (referred to as features) that can take numeric or discrete values. $X_i = \left(A_1^{(i)}, \ldots \;\ldots, A_m^{(i)}\right)^T$ is then a random vector. As an example, for Internet traffic, $A_j^{(i)}$ may represent the mean inter-arrival time of packets in the flow i.

Assume now that there are k known classes of interest. Let $\mathcal{C} = \{c_1, \ldots, c_k\}$ represent the set of all known classes. For each observed instance x_i in x, there is a known mapping $C : \mathbf{x} \to \mathcal{C}$ representing the membership of instance x_i to a particular class of interest. The notation $C(x_i) = c_j$ stands for “the instance x_i belongs to the class c_j”.

Bayesian statistical conclusions about the class c_j of an unobserved flow y are based on probability conditional on observing the flow y. This is called the posterior probability and is denoted by $p(c_j \mid y)$. The Bayes rule gives a way of calculating this value:

$$p(c_j \mid y) = \frac{p(c_j) f(y \mid c_j)}{\sum\limits_{c_j} p(c_j) f(y \mid c_j)} \tag{1}$$

where $p(c_j)$ denotes the probability of obtaining class c_j independently of the observed data (prior distribution), $f(y \mid c_j)$ is the distribution function (or the probability of y given c_j) and the denominator acts as a normalizing constant.

The goal of the supervised Bayes classification problem is to estimate $f(y \mid c_j)$, $j = 1, \dots, k$ given some training set x. To do that, Naïve Bayes makes certain assumptions on $f(\cdot \mid c_j)$ such as the independence of A_i's as well as the standard Gaussian behavior of them. The problem is then reduced to simply estimating the parameters of the Gaussian distribution and the prior probabilities of c_j's. In fact, Naïve Bayes is also capable of dealing with discrete random features, which could represent the state of some flag of a flow, by treating them independently and using the frequencies of occurrences to estimate $f(\cdot \mid c_j)$, $j = 1, \dots, k$.

3.4 C4.5 Decision Tree

C4.5 is well-known as a discriminative decision tree algorithm where the classification will be definitive (to assign each data point one of the mutual-exclusive classes).

Input to C4.5 consists of a collection of training cases, each having a tuple of values for a fixed set of features $F = F_1, F_2, ..., F_k$ and a class. A feature F_a can be described as continuous or discrete according to whether its values are numeric or nominal. The class C is discrete and has values $C_1, C_2, ..., C_x$.

The goal is to learn from the training cases a function

$$\mathrm{DOM}(A_1) \times \mathrm{DOM}(A_2) \times \ldots \times \mathrm{DOM}(A_k) \rightarrow \mathrm{DOM}(C).$$

that maps from the feature values to a predicted class.

As such the decision tree is a recursive structure where:

- a leaf node is labeled with a class value, or
- a test node that has two or more outcomes, each linked to a subtree

To classify an object using C4.5, imagine the object to be classified is initially at the top (root) of the tree. The object will go iteratively into a subtree as below, until it reaches a leaf node:

- if it is at a leaf node, the label associated with that leaf becomes the predicted class;
- if it is at a test node, when the outcome of the test is determined, it is moved to the top of the subtree for that outcome.

When training a model, the C4.5 learner uses information gain ratio to decide which feature goes into a test node. The information gain ratio is defined as the normalized information gain (2), which is based on the entropy (5) of the random variables. It measures the correlation between two random variables: a feature and a class label in this case.

Given discrete random variables X and Y:

$$\mathrm{GAINRATIO}(X \mid Y) = \frac{H(X) - H(X \mid Y)}{H(X)}. \tag{2}$$

$$\mathrm{IG}(X \mid Y) = H(X) - H(X \mid Y). \tag{3}$$

where

$$H(X \mid Y) = -\sum_j p(y_j) \sum_i p(x_i \mid y_j) \log_2 p(x_i \mid y_j), \tag{4}$$

and

$$H(X) = -\sum_{x_i} p(x_i) \log_2 p(x_i), \tag{5}$$

where $p(x_i) = P[X = x_i]$, $p(y_j) = P[Y = y_j]$ and $p(x_j \mid x_j) = P[X = x_i \mid Y = y_j]$.

In principle, the learner iteratively looks for the best feature to partition the data points in a node. The one with highest information gain ratio will be used to make the decision in the node. The division continues until the node becomes a leaf node, or the number of training data points in the node is smaller than a given number.

Moreover, C4.5 has incorporated a large number of improvements such as error-reduced pruning, avoiding over-fitting, and dealing with missing values. We refer the reader to [19] for further reference.

3.5 Feature-space reduction

We use Fast Correlation-Based Filter (FCBF) of [20] along with a variation of a wrapper method in determining the value of the threshold (described later in this Section). The FCBF filter method performs very well in improving the performance of Naïve Bayes when contrasted with other related techniques [20].

The correlation measure used in FCBF is based on the symmetrical uncertainty. Using equations (3) and (5), *symmetrical uncertainty* is defined in the following way:

$$\mathrm{SU}(X, Y) = 2\left[\frac{IG(X \mid Y)}{H(X) + H(Y)}\right]. \tag{6}$$

Symmetrical uncertainty takes values in $[0, 1]$, where the value 1 means that the knowledge of either X or Y can induce the other, while 0 suggests that features X and Y are wholly independent. By this point, Equation (6) has only been defined for nominal feature values [4], therefore FCBF will discretize continuous features before the core analysis [20].

The FCBF algorithm selects good features via a two stage process by identifying:

- the relevance of a feature, and
- the redundancy of a feature with respect to other features.

To describe these concepts mathematically, let C denote the random variable of traffic classes taking values in $\mathcal{C}$. Further, let $SU_{i,c}$ and $SU_{i,j}$ denote the value of the symmetric uncertainty between A_i and C and between A_i and A_j respectively. A feature A_i is believed to be relevant if $SU_{i,c} \geq \delta$, where δ is some threshold value to be determined by the user.

Identification of redundancy is often done by computing the pairwise cross correlations between features. However, [20] note that this method is quite computationally expensive and so the solution they propose considers SU values, because symmetrical uncertainty captures pairwise cross correlation information. As a result, FCBF works in the following way. Initially, $SU_{j,c}$, $1 \leq j \leq m$ are calculated and features are ordered in descending order according to the values of SU. A set S is created, containing A_js that satisfy $SU_{j,c} \geq \delta$. Then, the feature with the largest $SU_{j,c}$ (call it A_p) is compared to $SU_{j,q}$, where $A_q \in S \setminus A_p$. If $SU_{j,q} \geq SU_{p,c}$, the feature A_q is considered redundant and is therefore removed from S. The procedure is repeated for all A_p's in S. The complexity of this algorithm is $O(nm \log m)$.

At last, the question arises as to how to determine the optimal value of the threshold δ. To overcome this difficulty, we use a wrapper method based upon Naïve Bayes algorithm, i.e. computational results of the Naïve Bayes algorithm will be used to estimate the optimal value of the threshold. This approach has the goal of maximizing some measure of accuracy (e.g., percentage of correctly classified flows). The advantage of this approach is that it is less computationally expensive than the "forward selection" or "backward elimination", since only m cases are needed to be checked compared to $2^m - 1$. In addition, this method significantly improves the predictive capability of Naïve Bayes technique, and may also improve the accuracy of other machine learning mechanisms.

The following procedure is used to identify the best number of features to be used for a particular training set.
1. All features are ranked in order of importance as calculated by the FCBF method.
2. We now wish to identify the most valuable features, to do this an independent set of test data is chosen and it is used to evaluate the performance of Naïve Bayes classifier trained on different number of features.
3. We train Naïve Bayes on the training set with n, where $n \in 1 \ldots m$ (recall m is the total number of features) and evaluate the resulting classifier on the test set.
4. Finally, we select the optimum value for n such that it provides maximum clas-

[4] Although it is possible to define it for continuous random variables; the estimation of probabilities is then much harder.

sification accuracy while minimizing the total number of features required.

This algorithm uses both filter and wrapper methods to determine the optimal set of features. [21]

4 Results and Discussion

Using information collected from the first several packets of the flow, we aim at building real-time classification models. These models are trained offline due to the need of hand-classified ground-truth data. The testing phase however, can be automated and involves the following three-phase pipeline:

(1) grouping packets into flows,
(2) calculating features from the flows, and
(3) classifying the flows using the features and labeling the flows.

Apart from accuracy, there are two further objectives in the design of such a classification system:

Latency Firstly, for certain tasks such as application-specific routing, monitoring or anomaly detection, it is desirable for the latency prior to identifying a flow be as low as possible.

Overhead Both the memory footprint in aggregating the packets and the computational complexity in calculating features increase proportionally with the number of packets in each flow object. Thus it behooves an architect to minimize the quantity of link-data required. Collecting all packets of live traffic on a high-speed link will quickly exhaust the memory space for any commodity hardware. However, by keeping less packets of a flow in the memory, the system will be able to exchange a small error for a higher throughput, thereby enabling the classification systems' operation on a high-speed link.

4.1 Observation Window

Theoretically, a flow classifier will be more accurate when given more relevant information collected from each flow. However, the latency and computational overheads in collecting features will increase in proportion to the observation window size. This observation window may be bounded in time, in the total number of packets, or both.

To provide a reasonable trade between accuracy, latency and throughput, one can choose to limit the number of packets at the beginning of a flow from which to collect the set of features. Formally, we define an observation window as a fixed window of a given number of packets from the first packet in a flow (i.e., a SYN packet). For example, the feature "average payload bytes" is calculated using the sum of payload bytes in these packets divided by the number of packets seen in the observation window.

To demonstrate how one might attend the accuracy/latency/throughput trade-off and to evaluate the performance of such a classification mechanism, we use a subset of 12 features and the C4.5 algorithm to identify an appropriate observation window size. When collecting features we also use a 15 seconds timeout alongside the limit to the number of packets, that is, the observation window terminates when either the number of packets sums to the window size or the timeout occurs.

The classifier accuracy for different observation window sizes varying from four to ten is shown in Figure 3. For these results and in common throughout this work, we perform our C4.5 and Naïve Bayes evaluations using the Weka [22] toolkit. For all data-sets, the accuracy achieves a high value with a window size of five packets. The accuracy is comparable to the accuracy of using complete flows or larger observation windows. This indicates that the behavior shown in the first five packets contains rich information to classify the flows.

The idea of early-flow "fingerprinting" is also used by Bernaille *et al.* [1] where they use a feature set composed of the sizes and directions of the first four data packets of each flow. In comparison, our method collects more information from fewer packets: our five observed packets include the three start-of-flow packets common at the beginning of each TCP flow. This results in lower overheads and lower latency for our methods. We choose a window size of five packets for the results presented in the rest of this paper.

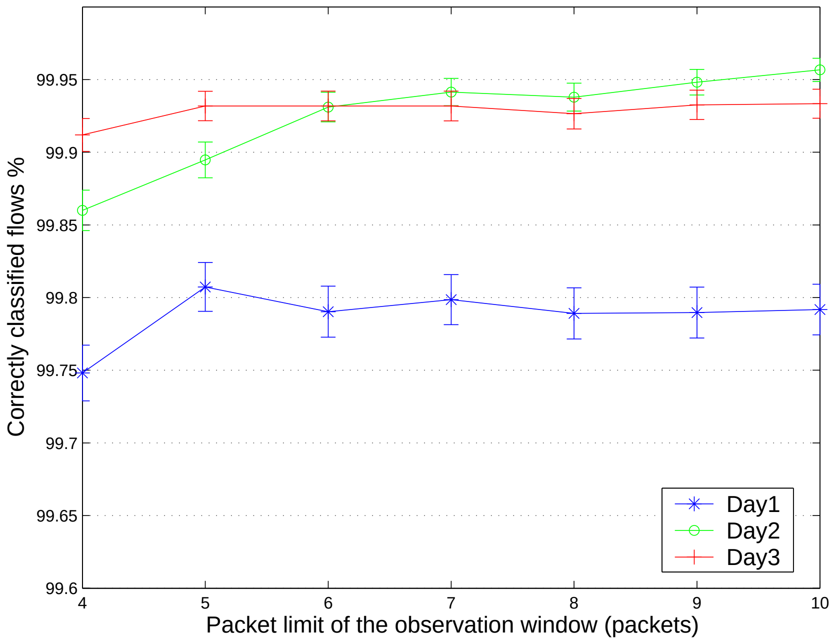


Fig. 3. Relationship between accuracy and packet-limit using C4.5.

4.2 Methodology

In Section 4, we show both a range of results and comparisons. These comprise:
1. a 4-way accuracy comparison between the classification methods (Table 5),
2. per-class accuracy on Day1, Day2, Day3, and SiteB, for C4.5 algorithm (Table 6),
3. a 4-way general accuracy comparison between the temporal decay property of classification methods (Figures 4, 5 and 6),
4. per-class temporal decay between Day1-Day2, Day1-Day3, and Day2-Day3, for C4.5 algorithm (Table 7),
5. a 4-way comparison between the spatial stability of classification methods with cross-site (Table 8) and multi-site (Table 10) training on Day3 and SiteB data-sets, along with per-class accuracy for C4.5 algorithm (Tables 9 and 11),
6. a 4-way accuracy comparison on UDP traffic (Table 14), with detailed per-class results (Table 15) and temporal and spatial stability (Tables 16 and 17), and
7. a comparison of training and testing time between C4.5, Naïve Bayes and L7 methods (Table 18).

To evaluate the accuracy of the classifier on the same day and same site (Subsection 4.3), for each flow we randomly place it into one of two data-sets (for example, with a probability of 0.5 to create comparable-sized data-sets). One data-set serves as training set and the other data-set as testing set. We can repeat the division of data-sets using a different seed for the random-number generator This allows a repeating of the process as many times as required, for example, in the computation of standard-deviation or confidence intervals. The accuracy figures represent an average of each experiment conducted where each experiment accuracy figure is the total number of correctly identified flows divided by the total number of flows within that particular data-set.

To evaluate the temporal stability (Subsection 4.4), we train the model using the Day1 (2003) and Day2 (2004) data-sets, and test the model on each of Day2 and Day3 data-sets. Recall that Day2 and Day3 were collected in 2004 and 2006 respectively.

To evaluate the spatial stability (Subsection 4.5), we first train the model using a subset of each of the Day3 and SiteB data-sets. We then validate each model against the remaining data-set from that site.

To create a multi-site model we combine one half of each of the Day3 and SiteB data-sets to use as a training set producing the multi-site model. We evaluate this model on the remaining half of each data-set. Using the same (random) mechanism as above, this process is repeated to obtain standard deviation and confidence intervals. In this way the results show the overall accuracy for both halves for Day3 and SiteB respectively. This represents the spatial stability of the multi-site model on specific, different data-sets.

It is important to note that in each case, there is no overlap between the training set and the testing set — we do not test the accuracy on the same data as was used to train the model. Further, each of the experiments is repeated multiple times using multiple different random-number seeds to divide data-sets into different training-sets and testing-sets. In each case we present the overall accuracy along with confidence intervals where these are significant.

The following metrics are used to quantify the accuracy of classification models:

- *Overall Accuracy* — the percentage of correctly classified instances over the total number of instances,
- *Precision* — the number of class members classified correctly over the total number of instances classified as class members for a given class, and
- *Recall* — the number of class members classified correctly over the total number of class members for a given class.

4.3 Classification Accuracy (Same Site, Same Day)

In this subsection, we present results contrasting the overall accuracy across four classification methods: C4.5, Naïve Bayes, IANA port numbers and L7 signatures. The results are shown in Table 5. Note that C4.5 and Naïve Bayes classifiers are using the features computed with the five-packet observation window; instead, for ideal effectiveness, L7 classifier is searching the payload content for known signatures in up to the first 10 data packets of a flow.

It is well known that the port-based classifier is likely to lead to inaccurate estimates of the amount of traffic carried by different applications given that certain protocols, such as HTTP, are frequently used to relay other types of traffic, e.g., MSN Messenger over HTTP, whereas other protocols make use of parallel connec-

Data-Set		C4.5	Naïve Bayes	IANA ports	L7 Signature	L7 Signature
					Set (2008)	Set (2003)
Day1	% flows	99.807±0.021	96.663±0.064	95.29	88.93	72.27
	% packets	99.711±0.026	82.050±0.093	34.25	30.18	27.13
	% bytes	99.714±0.025	83.911±0.091	31.72	27.66	25.89
Day2	% flows	99.895±0.012	95.845±0.066	91.79	87.63	70.63
	% packets	99.886±0.013	86.152±0.089	23.96	23.39	21.24
	% bytes	99.884±0.013	86.341±0.088	16.21	16.31	15.39
Day3	% flows	99.937±0.010	98.301±0.049	84.66	78.03	60.14
	% packets	99.843±0.016	80.930±0.095	21.07	19.94	17.93
	% bytes	99.842±0.016	80.203±0.107	17.55	16.51	16.22
SiteB	% flows	99.665±0.021	97.630±0.055	89.67	94.28	56.08
	% packets	99.441±0.029	94.570±0.068	79.39	86.61	57.87
	% bytes	99.422±0.033	93.700±0.077	77.53	84.59	61.13

Table 5
Overall accuracy comparison across 4 classification methods.

Class	Day1		Day2		Day3		SiteB	
	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
WEB	99.878	99.945	99.985	99.989	99.955	99.992	99.771	99.962
MAIL	99.954	99.982	99.642	99.648	99.899	99.899	99.862	99.871
BULK	99.003	99.654	99.861	99.880	99.627	99.794	97.830	97.478
ATTACK	95.845	81.668	94.147	94.529	60.000	8.571	99.950	99.751
CHAT	-	-	-	-	95.313	92.424	96.056	81.818
P2P	96.456	96.595	99.201	98.914	99.892	99.861	99.204	99.154
DATABASE	99.892	99.463	99.808	99.693	99.967	99.978	-	-
MULTIMEDIA	97.206	98.185	0.0	0.0	100.0	94.737	100.0	72.727
VOIP	-	-	-	-	88.095	79.570	90.523	87.920
SERVICES	100.0	99.502	99.641	99.910	100.0	45.714	71.823	27.957
INTERACTIVE	100.0	100.0	97.222	97.222	100.0	99.071	97.792	97.792
GAMES	100.0	40.000	-	-	-	-	100.0	100.0
GRID	-	-	-	-	-	-	96.875	100.0

Table 6
Per-class accuracy for online classification using C4.5.

tions which use dynamically assigned port numbers, e.g., FTP transfers in PASV mode. In addition, emerging services typically avoid the use of well known ports, e.g., some P2P applications, while other popular applications deliberately use well know ports to masquerade their traffic through firewalls, e.g., Skype. These have led to only 31% of byte accuracy in Day1 and further 16% and 17% byte accuracy in Day2 and Day3 respectively.

The general performance of payload signature matching is even worse because (a) even the most up-to-date signature set is far not enough to cover all the applications, (b) some of the signatures do not match sufficiently well while (c) others are significantly over-matching.

Compared to conventional port-based and signature-based classifiers, the overall accuracy of our method is much higher in terms of flows, packets and bytes. To fully describe its performance, Table 6 details the per-class performance for each application class. The accuracy is high (>99%) for the major application classes. However, because the classifier tends to converge towards the highest overall accuracy, classes containing a small number of members may present a lower accuracy as the class distribution of the training data-set is not uniform.

Also, some of the classes are presumably more difficult than others. The ATTACK class comprises of a variety of malicious activities targeting at different vulnerabilities. This complex variety leads to a number of false predictions to and from other classes, e.g., some of them are classified as WEB in Day1 and Day3, while in Day2 some MAIL flows are classified as ATTACK. However, the results show that these models may still achieve either good precision or good recall for ATTACK class within Day1, Day2 and SiteB.

Ideally, the accuracy obtained by two-fold cross-validation in the same data-set can be an indication of accuracy for classifying the traffic of the same day or in a short period after the model is built.

We are also interested in seeing how the classifier decays over time, which is presented in the next subsection; and how the classifier performs on a totally different network or across various networks, which is presented in Subsection 4.5.

4.4 Temporal Decay of Model Accuracy

In Section 3 we note that each classification scheme involves the creation of a model that provides an identification of each network-based application. Such models may be the table of port-numbers relating to (groups of) applications, the signature set for specific applications, or the probabilistic priors grouping flow-features as used by machine-learning algorithms. In each case a model will define how traffic is classified.

With changing applications and network-host behavior, the accuracy of such a model for traffic classification will change over time, as we illustrate in the following for several different methods.

We study the temporal decay of the model accuracy by applying the models created from Day1 and Day2 data-sets to classify the Day2, Day3 and Day3 respectively. The gap between Day1 and Day2 is one year (2003 to 2004), in which the variety of application and composition of traffic did not change significantly; whereas between Day2 and Day3 it is roughly two years (2004 to 2006) when there is a rise on the number of different applications (e.g., more kinds of P2P clients, web applications, instant messengers and Voice-over-IP clients). The composition of traffic also changed accordingly.

Figures 4 and 5 show a 4-way comparison of the decay of the overall accuracy from 2003, 2004 to 2006 for correctly classified flows, and packets and bytes respectively. We used the same IANA port number list as before, but added an old version of L7 signature set (September 2003) alongside the most up-to-date L7 signature set. The two signature sets would represent the decay of signature-based mechanisms under different assumptions. Besides those, we show the results using C4.5 and Naïve Bayes models built upon the Day1 data-set to classify Day2 and Day3 traffic.

It is shown that while conventional classifiers such as port-based and signature-based degrade by around 5% of flow accuracy each year, the C4.5 classifier has only a decay of less than 2% each year in all three measures of flows, packets and bytes. The standard deviation in the results is relatively small. For example, the standard deviation of C4.5 Day1 vs. Day3 is 0.079 and Naïve Bayes Day1 vs. Day3 is 0.130. These values are invisible in the resolution of the figure therefore the error bar is not shown.

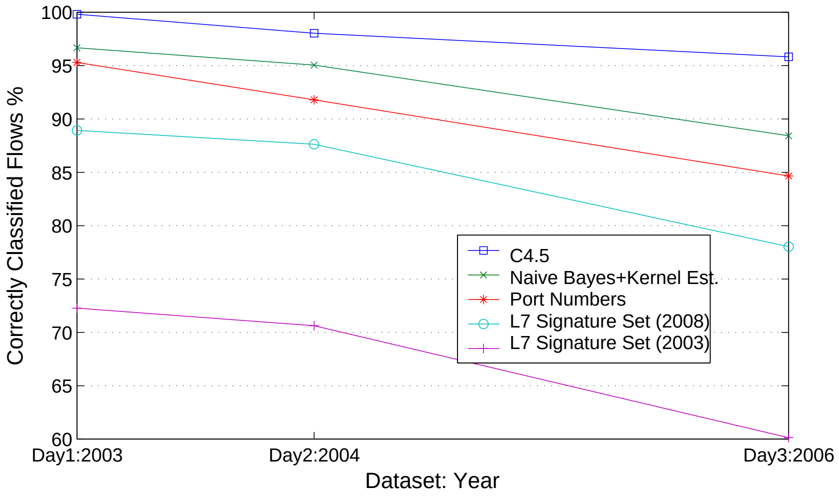


Fig. 4. A 4-way accuracy comparison of correctly classified flows for Day1 vs. Day2 and Day1 vs. Day3.

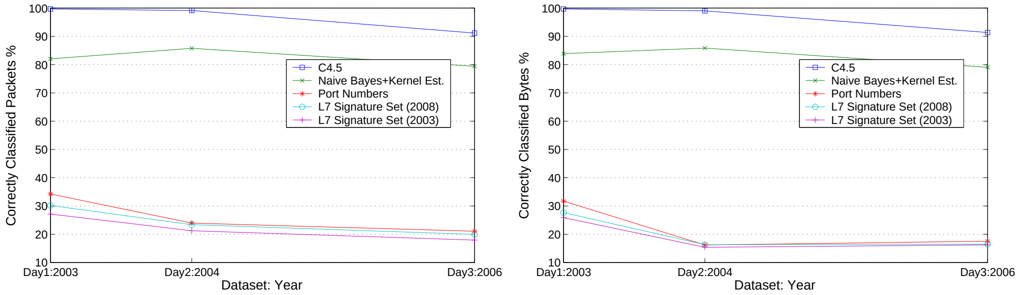


Fig. 5. A 4-way accuracy comparison of correctly classified packets and bytes for Day1 vs. Day2 and Day1 vs. Day3.

Further, Figure 6 shows a comparison Day2 vs. Day2 and then Day2 vs. Day3. From this we can see the decay of C4.5 and Naïve Bayes classifiers, between 2004 to 2006. Results are given in terms of flows, packets and bytes. The error bar is not shown due to the same reason as described above. It is clear that there is significant decay in the quality of classification provided by the Naïve Bayes — still better than the L7 and port methods (Figure 4) but significant nonetheless. In contrast the C4.5 loses less than 3% of accuracy for all metrics.

Finally, Table 7 presents the precision and recall values of the results using C4.5.

Most of the classes are very accurate from 2003 to 2004, and many classes are still very accurate from 2003 to 2006 and from 2004 to 2006, e.g., WEB, MAIL, BULK and P2P.

We also observe that in certain cases the precision is significantly higher than recall. This indicates that some new types of applications of this class may have emerged during this period. These applications are difficult to be recognized by the model trained using an earlier data-set. For example, a reason for the recall of MAIL class losing 11% accuracy is that more mail servers tunnel IMAP communications

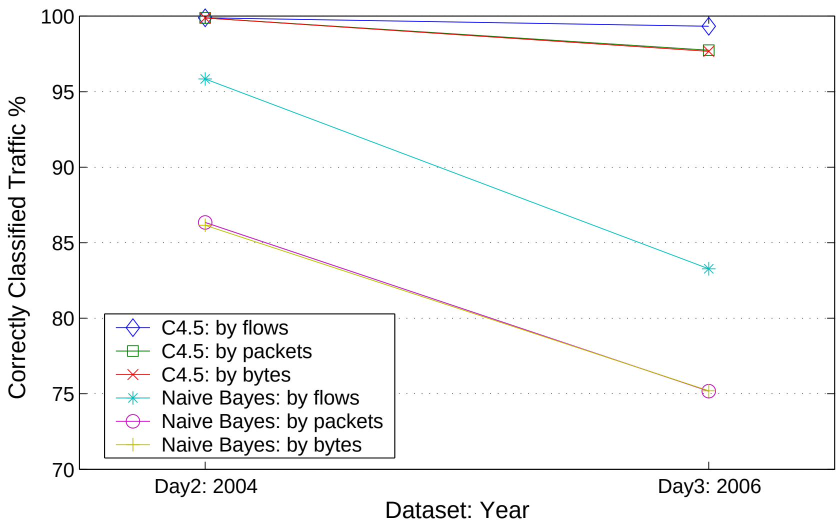


Fig. 6. Accuracy comparison of correctly classified flows, packets and bytes for Day2 vs. Day2 compared with Day2 vs. Day3.

Class	Day1 vs. Day2		Day1 vs. Day3		Day2 vs. Day3	
	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
WEB	99.917	99.868	99.851	99.713	99.941	99.866
MAIL	94.362	100.0	99.606	88.914	100.0	87.934
BULK	83.545	99.259	35.101	92.543	86.623	97.178
ATTACK	0.0	0.0	0.0	0.0	0.0	0.0
CHAT	-	-	0.0	0.0	0.0	0.0
P2P	92.690	97.791	96.219	99.578	98.471	98.856
DATABASE	96.893	19.148	51.741	4.531	100.0	99.967
MULTIMEDIA	0.0	0.0	0.0	0.0	0.0	0.0
VOIP	-	-	0.0	0.0	0.0	0.0
SERVICES	99.541	97.570	43.750	30.000	4.636	30.000
INTERACTIVE	97.222	97.222	100.0	8.050	100.0	8.050

Table 7
Per-class accuracy temporal stability for online classification using C4.5.

through SSL in Day2 and Day3. Moreover, we can see different database management system servers on the site increased between year 2003 and 2004: the recall is low for Day1 vs. Day2 and Day1 vs. Day3, while nearly all the database flows are correctly identified in Day2 vs. Day3. Similarly, it also failed to identify the network monitoring applications built upon HTTP (they are categorized into SERVICES) and several remote control applications in the INTERACTIVE class, such as PCAnywhere and GotoMyPC that appeared between 2004 and 2006. The CHAT and VOIP traffic are not successfully classified because these classes do not even exist in the training sets. This observation also raises an intersting problem of how to identify a whole new class of traffic which is previously unobserved. Further, poor temporal stability is shown for ATTACK as the malicious traffic is of totally different types in the training and testing data-sets.

4.5 *Spatial Stability of Model Accuracy*

Our spatial stability study comprises of two experiments: the first is using the model trained on one site to measure the accuracy on another site; the second is building a model on the traffic from both sites and test on them separately.

In the first experiment, we measure the accuracy of a model being applied to classify the traffic in a totally different network where no prior information is available. In the second experiment we are measuring the effectiveness of a generic model across different, specific networks. Both experiments serve to illustrate the spatial property of the classification methodology.

4.5.1 *Training on One Site and Testing on Another*

The accuracy here shown are for models trained on Day3 and SiteB respectively and applied to test the accuracy on the other data-set. Table 8 compares the model accuracy with L7 and port numbers. The results under "Day3: SiteB" is using Day3 as training set and SiteB as testing set; the results under "SiteB: Day3" is using SiteB as training set and Day3 as testing set.

Since the majority of traffic in SiteB is web-browsing traffic and the HTTP signature works relatively well, the L7-signatures yield good results. However, in all cases the result using C4.5 is advantageous compared with other methodologies. Notably, the Day3 C4.5 model shows better results in terms of flows, packets and bytes on SiteB which is from a totally different network 19 months later.

Table 9 reports the results for each application class using C4.5. Several major classes, namely WEB, MAIL, BULK and MULTIMEDIA have shown outstanding recall values on both directions; this means the model is very effective for most traditional services. However, we notice that there are drops in some other traffic classes.

Model: Data-Set		C4.5	Naïve Bayes	IANA ports	L7 Signature Set (2008)
Day3: SiteB	% flows	94.466±0.095	89.821±0.122	89.67	94.28
	% packets	93.294±0.103	84.417±0.263	79.39	86.61
	% bytes	94.035±0.094	84.725±0.394	77.53	84.59
SiteB: Day3	% flows	95.790±0.080	84.454±0.149	84.66	78.03
	% packets	94.967±0.086	73.226±0.337	21.07	19.94
	% bytes	95.345±0.082	73.162±0.340	17.55	16.51

Table 8
Accuracy of models trained from one site and applied to each other site, between Day3 (2006, Site A) and SiteB (2007, Site B).

Class	Day3: SiteB		SiteB: Day3	
	Precision%	Recall%	Precision%	Recall%
WEB	98.586	99.748	99.917	99.526
MAIL	99.697	99.981	91.270	99.346
BULK	11.307	91.172	37.421	98.206
ATTACK	0.000	0.000	0.000	0.000
CHAT	50.629	63.636	91.379	80.303
P2P	88.610	58.355	98.172	99.520
DATABASE	-	-	0.000	0.000
MULTIMEDIA	0.578	90.909	100.000	94.737
VOIP	33.019	26.845	3.022	11.828
SERVICES	0.079	0.430	0.333	1.429
INTERACTIVE	52.917	80.126	17.266	7.430
GAME	0.000	0.000	-	-
GRID	0.000	0.000	-	-

Table 9
Per-class accuracy of C4.5 models trained from one site and applied to each other site.

- ATTACK: because of the difference in both location and time, the attack traffic in the two data-sets are totally of different types, which caused that none of the models can identify the ATTACK traffic in the other data-set. [5]
- P2P: the identification is highly accurate from SiteB to Day3, but it drops to 58% when using Day3 model to test on SiteB. As a matter of fact, we have only found 9 internal hosts in Day3 running a P2P application (Gnutella, eDonkey or Azureus). However in SiteB there were 30 internal hosts and the variety of the clients are larger (eDonkey, BitTorrent, DirectConnect, Joltid, Gnutella and Pando) and are probably of newer versions. Further to our knowledge, a major part of Site A is running a firewall that severely throttles P2P traffic. Such throttling produced a lot of failed attempts for connection, which may lead to some skewness in the model.
- DATABASE and GRID: there is no networked database traffic in SiteB, which caused the failure in identifying them in Day3. Similarly, there is no Grid computing traffic in Day3 which caused the failure in identifying them in SiteB.
- VOIP: neither of the two data-sets have many samples of Skype flows in TCP. Also, the result is not ideal due to the different Skype versions (version 2 in Day3 and version 3 in SiteB).

[5] In Day3 the ATTACK class contains a small number of flows by SQL injection over HTTP while in SiteB it contains traffic generated by botnets and a MS-RPC worm.

Data-Set		C4.5	Naïve Bayes	IANA ports	L7 Signature Set (2008)
Day3	% flows	99.919±0.008	97.986±0.052	84.66	78.03
	% packets	99.631±0.025	79.281±0.309	21.07	19.94
	% bytes	99.627±0.027	78.515±0.319	17.55	16.51
SiteB	% flows	99.662±0.032	97.884±0.053	89.67	94.28
	% packets	99.558±0.035	94.683±0.312	79.39	86.61
	% bytes	99.600±0.033	93.906±0.456	77.53	84.59

Table 10
Accuracy of the dual-site model trained on half of the two data-sets and tested on the other half of each data-set.

- SERVICES: this class comprises of only a small amount of flows, and in Day3 many of them is network monitoring traffic over HTTP. This leads to a number of false negatives and false positives on both ways.
- INTERACTIVE: this class incoporates ssh connections and also several remote access software, e.g. GotoMyPC/expertcity. While ssh is easier to identify from packet stream behavior, the INTERACTIVE traffic in the two traces are significantly different in their behavior.

4.5.2 Multi-site Training

Now we evaluate the model trained with combined data from two different sites. In this experiment, a half of each data-set is randomly selected and combined together as the training set to train the model. Then the model is tested on the other half of each data-set.

Table 10 shows the overall accuracy comparison and Table 11 shows detailed per-class accuracy on each data-set. The resultant accuracy only slightly decreases in comparison to models trained specifically for a given site. This indicates that there are very little conflict when combining the data-sets in order to train a model for multiple sites; and the features, as we discussed before, are faithfully representing the behavior of the applications rather than specific networks or different situations in the network communication.

Most of the classes are very accurate except few classes which only have a very small amount of flows. This is because the classifier tends to converge toward higher overall accuracy during training, and the number of samples in these classes might be insufficient to build a good model. There are methods to trade-off between the accuracy of a certain class with the overall accuracy, and we refer readers to related machine learning literature.

Class	Day3		SiteB	
	Precision%	Recall%	Precision%	Recall%
WEB	99.964	99.982	99.746	99.968
MAIL	99.924	99.949	99.963	99.862
BULK	99.663	99.495	96.654	98.919
ATTACK	29.411	14.285	99.626	99.825
CHAT	94.736	81.818	97.852	81.028
P2P	99.668	99.887	99.314	98.964
DATABASE	99.923	99.934	-	-
MULTIMEDIA	100.0	94.736	90.909	90.909
VOIP	80.303	56.989	89.509	89.166
SERVICES	71.428	50.000	72.992	21.505
INTERACTIVE	99.047	96.594	97.763	96.530
GAME	-	-	99.337	100.0
GRID	-	-	95.876	100.0

Table 11
Per-class accuracy of C4.5 dual-site model trained on half of the two data-sets and tested on the other half of each data-set.

4.6 UDP Classification Accuracy and Stability

Although in recent years the UDP traffic has not increased too much in the proportion of total traffic volume, we have seen a significantly increased variety of applications over UDP, such as VoIP communications, multimedia applications, P2P file-downloading and networked games. Accordingly there is an increasing need to understand and technically support such variety of applications.

For this study, we selected three 30 minutes traces of UDP traffic from Day1, Day3 and SiteB respectively (as detailed in Section 2). All the three traces are collected at around 10:30 AM of a weekday.

Since the UDP flows do not maintain a state machine like in TCP, we use an inactivity time-out. We elect to use the default inactivity time-out of Cisco Netflow to aggregate the flows, with the time-out value set to be 60 seconds. We acknowledge this may not be the ideal value for all types of traffic and would suggest that a multiple time-resolution approach: using several different time-out values, would make interesting future work.

The ground truth in the UDP data-sets is derived in a similar way as in the TCP data-sets. The actual application breakdown in the UDP data-sets is shown in Table

Class	By flows [%] / packets [%] / bytes [%]			Applications
	Day1	Day3	SiteB	
ATTACK	10.554 / 1.510 / 0.987	14.933 / 1.158 / 1.843	42.433 / 20.305 / 15.963	Port scans, ms-sql worms
SERVICES	88.889 / 76.807 / 62.740	57.073 / 25.167 / 14.084	11.968 / 26.532 / 26.339	DNS, LDAP, NTP, SNMP, middleware
P2P	0.036 / 0.022 / 0.007	13.861 / 28.109 / 23.862	7.062 / 7.560 / 7.280	Kazaa, Gnutella, eDonkey, BitTorrent
MULTIMEDIA	0.522 / 21.661 / 36.266	1.268 / 42.521 / 59.498	0.000 / 0.396 / 2.287	Windows Media Player, Realmedia
VOIP	- / - / -	12.865 / 3.045 / 0.712	38.536 / 44.482 / 45.297	Skype
GAME	- / - / -	- / - / -	0.001 / 0.726 / 2.830	Second Life

Table 12
Composition of UDP traffic in our data-sets. Applications shown are only examples for demonstration. “-” denotes non-existing traffic.

Abbreviation	Description	SU	Memory Overhead	Computational Complexity
num_pkts	Number of packets seen on both directions	0.3333	O(1)	O(n)
min_pbyte_clnt	Minimum payload bytes seen (client to server)	0.3897	O(1)	O(n)
min_pbyte_serv	Minimum payload bytes seen (server to client)	0.4114	O(1)	O(n)
max_pbyte_clnt	Maximum payload bytes seen (client to server)	0.4032	O(1)	O(n)
max_pbyte_serv	Maximum payload bytes seen (server to client)	0.3803	O(1)	O(n)
ini_pbyte_clnt	Payload bytes sent from client to server before the first packet coming back	0.3353	O(1)	O(n)
max_csct_pkts_clnt	Maximum number of consecutive packets (client to server)	0.3064	O(1)	O(n)
serv_port	Server Port	0.6424	O(1)	O(1)
clnt_port	Client Port	0.3676	O(1)	O(1)

Table 13
Properties of the subset of UDP features selected using five-packet observation window. SU is the Symmetrical Uncertainty measurement, as fully described in Section 3.5.

12, comprising of 6 major classes [6] .

Following the approach described in Section 4.2, a machine learning approach is applied to classify the UDP traffic. Like the TCP traffic, we use an observation window that limits us to the first five packets.

As the UDP header contains different information from the TCP header, it is necessary to select a different feature set for UDP. Therefore, a complete set of features similar to the TCP feature set (with the TCP-specific ones removed and a few others changed) are collected and then, applying the FCBF-based approach described in Section 3.5, we select an optimal subset of features. The resultant feature set contains nine features and are shown in Table 13 below.

[6] In theory there may also be instant messengers traffic (CHAT class) over UDP, but it is not found in these traces.

Data-Set		C4.5	Naïve Bayes	IANA ports	L7 Signature Set (2008)
Day1	% flows	99.956±0.011	86.879±0.208	88.91	73.43
	% packets	99.969±0.010	88.567±0.163	79.44	72.55
	% bytes	99.977±0.010	90.529±0.157	64.38	59.58
Day3	% flows	99.627±0.034	97.812±0.077	56.44	65.97
	% packets	98.745±0.043	94.389±0.101	18.54	20.03
	% bytes	94.750±0.069	95.121±0.098	5.28	5.66
SiteB	% flows	99.889±0.019	97.964±0.075	10.50	53.15
	% packets	99.052±0.036	96.224±0.101	22.96	71.30
	% bytes	98.937±0.037	92.250±0.100	23.35	72.34

Table 14
Overall accuracy comparison across 4 classification mechanisms.

Class	Day1		Day3		SiteB	
	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
ATTACK	99.772	99.924	99.782	100.0	99.994	100.000
SERVICES	99.996	99.964	99.905	99.958	99.930	99.923
P2P	100.0	88.889	99.115	98.601	99.147	99.332
MULTIMEDIA	97.015	100.0	97.789	98.797	0.0	0.0
VOIP	-	-	98.950	98.916	99.896	99.859
GAME	-	-	-	-	0.0	0.0

Table 15
Per-class accuracy for UDP classifier using C4.5.

Table 14 compares the overall classification accuracy across the classification methodologies: C4.5 and Naïve Bayes using five-packet observation window, IANA port numbers and L7 signatures, using the same criteria as in previous Table 5. It shows that C4.5 achieves very good accuracy and works far better than IANA port numbers and L7-signatures.

Table 15 shows the precision and recall results for each class in each data-set. Most of classes are approaching 100% recall and precision, except two minor classes in SiteB with only few instances, which are probably too small to be effectively modeled.

We further evaluate the temporal and spatial performance of the UDP classifier and the results are shown in Table 16.

Class	Temporal		Spatial			
	Day1 vs. Day3		Day3: SiteB		SiteB: Day3	
	Precision%	Recall%	Precision%	Recall%	Precision%	Recall%
ATTACK	36.493	22.053	99.986	96.516	99.965	81.465
SERVICES	72.221	99.862	95.366	96.577	99.200	99.501
P2P	24.625	1.289	67.106	63.018	68.973	86.469
MULTIMEDIA	10.204	92.955	0.256	50.000	0.0	0.0
VOIP	0.0	0.0	91.225	95.252	84.958	87.519
GAME	-	-	0.0	0.0	-	-
Overall Flows%	61.428±0.358		92.149±0.157		93.670±0.144	
Overall Pkts%	70.086±0.298		52.955±0.375		92.553±0.148	
Overall Bytes%	65.275±0.334		36.772±0.434		89.725±0.161	

Table 16
Per-class and general accuracy for temporal and spatial evaluation of the UDP models trained with C4.5.

The temporal study indicates that the diversity and variety of applications over UDP have significantly increased between Day1 and Day3. The traditional services: DNS, NTP, SNMP and Windows Media Player traffic are correctly identified. However, the new types of attacks, P2P filedownloading and VoIP applications are not able to be identified using the previous model.

Despite the Day3 and SiteB UDP date-sets are highly different in the traffic composition and the application-set, most of the classes are correctly identified. The result for P2P class is lower than SERVICES and VOIP due to the different application variety on each data-set and the impact from firewall policies in Day3. Also, there are only 2 multimedia flows in SiteB which is insufficient to build the model to classify the MULTIMEDIA traffic in Day3.

Finally, a multi-site model is trained using combined UDP data from two different sites. A half of each data-set is randomly selected and combined together as the training set to train the model. Then the model is tested on the other half of each data-set. As shown in Table 17,

Again the result is much better than applying model from one site to the other. A very small decrease is seen in comparison to models trained specifically for a given site. However, the overall accuracy is still very good, and the recall value of some classes such as VOIP and MULTIMEDIA in SiteB have even improved, probably because the additional training samples from Day3 complemented those from SiteB.

Class	Day3		SiteB	
	Precision%	Recall%	Precision%	Recall%
ATTACK	99.971	100.0	99.989	99.995
SERVICES	99.939	99.958	99.912	99.917
P2P	98.449	97.768	99.221	99.270
MULTIMEDIA	99.648	97.251	7.143	50.000
VOIP	97.767	98.611	99.884	99.864
GAME	-	-	0.0	0.0
Overall Flows%	99.453±0.043		99.884±0.019	
Overall Pkts%	98.150±0.324		98.937±0.101	
Overall Bytes%	97.815±0.500		95.888±0.572	

Table 17
Per-class and general accuracy of the dual-site model trained on half of the two UDP data-sets and tested on the other half of each data-set.

4.7 Complexity and Memory Footprint

The cost in the online classification pipeline can be broken down into three parts. Suppose we drop the later packets of a flow after classification of the flow. Then if we denote M as total flows currently in the memory and n as the number of packets in the observation window, then the total computational overhead would comprise all the following (also note that roughly M has a linear relationship with n):

(1) For capturing packet headers and aggregating packets into flows, there will be a memory footprint of $O(M)$ to store M flows in the memory and computational complexity of at least $O(log_2 M)$ for each packet captured to find the flow it belongs to. Assuming M is proportional to n, for each flow, the complexity in aggregating the packets is roughly $O(n \times log_2 n)$.

(2) For feature collection and calculation of each flow, different features in the complete set would cause a memory footprint varied from $O(1)$ to $O(n)$ and computational complexity varied from $O(1)$ to $O(n \times log_2 n)$. Roughly, the total cost of feature collection of one flow would have a super-linear relationship with the number of features in the feature set.

(3) Presuming C4.5 is being used as the classifier, for classification of each flow, the computational complexity in the classifier is equal to the average depth of the decision tree, which is similar to the complexity of simple port-based rule-sets.

Clearly, we know from the cost breakdown above, as the complexity of the classifier is $O(1)$, the bottleneck in this pipeline may be in reassembling the flows and calculating the features, instead of the calculations in the classifier. However, the total cost of classifying a flow can be bounded within $O(n \times log_2 n + n \times K)$ where n is the number of packets in the observation window and K is the number of features collected. Such cost is considerably lower than a flow-based string signature matching system when n and K are both small.

Besides the complexity analysis, empirical results on the CPU time required for training and testing are provided below.

4.7.1 CPU Time Results

The experiments are run on a Dell Poweredge 2850 equipped with 2 Intel Xeon 3.5GHz CPU and 4GB of RAM. L7 signature matching and feature collection are implemented in C++, whereas the training and testing of the machine-learning models are performed using the Weka toolkit.

The CPU times are evaluated using data-sets on three different scales: (1) one-tenth of a 30-minute trace (2,476 flows), (2) one 30-minute trace (23,810 flows), and (3) ten 30-minute traces (377,000 flows).

Three different classification methods: C4.5, Naïve Bayes and L7 are separately executed on these data-sets. The time in offline training, online feature collection and classification are measured separately for each of the machine learning approaches, whereas for L7 we measure the time spent in classification alone. For each result of machine learning methods, we classify each data-set using the model trained from a data-set of the same time scale. Each experiment is repeated ten times. Table 18 lists the CPU times normalized by the number of flows in the data as the size of the trace varies, along with the standard deviations. The time for aggregating packets into flows is included in "online feature collection / classification".

The offline training time is less relevant to the online classifier performance: it shows how long it takes for the model to be built up using a certain data-set size.

The feature collection time for machine learning algorithms, and the classification time for L7 are both nearly constant when normalized. However, the feature collection time is about one-third of the time spent in signature-matching by L7 (both implementations are not optimized). The classification time using C4.5 is very low: 13 μs per flow for the largest data-set. Recall that the average computation needed

	Data-Set Size	C4.5	Naïve Bayes	L7
Offline training time [μs]	1/10	186.06 ± 25.66	103.74 ± 20.49	-
	1	130.20 ± 8.19	24.65 ± 1.97	-
	10	337.71 ± 2.88	32.28 ± 0.53	-
Online feature collection / classification [μs]	1/10	673 ± 49 / 67.79±11.09	673 ± 49 / 610.25±40.43	1666 ± 70
	1	692 ± 101 / 19.36±1.05	692 ± 101 / 1331.8±13.76	1540 ± 198
	10	540 ± 8 / 13.73±0.34	540 ± 8 / 4276.7±375.76	1814 ± 63

Table 18
Comparison of training and testing times between C4.5, Naïve Bayes and L7 schemes.

in a C4.5 decision tree is the average depth of the tree. As the size of the training set increases, the average tree-depth in all our models remains between 4 and 5, due to the effective pruning process of this algorithm. And so the normalized classification time should remain constant [7]. In contrast, the normalized classification time is not constant for Naïve Bayes with kernel estimation as the CPU time grows from 610 μs per flow (1/10 data-set) to 4276 μs per flow (ten data-sets). It is determined that training on larger data-sets results in a higher complexity of the model.

5 Related Work

The challenge in classifying the Internet traffic has aroused an ever-widening interest — whether among users concerned about their data, network-operators and nations keen to embargo certain applications [23], or researchers wanting to further their understanding [24], there has been a rich literature that drafted a range of proposals and conducted a number of preliminary experiments to address this interest.

The work in [7] quantitatively addressed the problem for conventional classification methods based on port numbers to effectively deal with current Internet traffic. It also described a content-based work to obtain the ground truth application information for each flow in the traffic.

Traditionally another type of solutions came from the intrusion detection community, examples of this include Snort [25] and Bro [26]. Each of these systems can identify the application layer protocols through the use of content signatures or connection patterns. The L7-filter, the deep-packet inspection filter described in [14], is a signature matching system able to identify the application layer protocols so as to enable per-application traffic management within the scope of the Linux `iptables` system.

There have been a number of previous works on statistical characterization and classification of different application traffic. Fundamental to these are papers to characterize a network flow using a different variety of features to describe its behavior [27, 11]. Their contribution provides an exploration of the different features of application traffic. Such rich sets of features allow us to consider the ability to perform behavior-based classification by providing the input parameters for traffic classification for a range of classification approaches as well as inputs to application-specific modeling and simulation.

Meanwhile, several ground-breaking approaches emerged for classification of current Internet traffic. Roughan *et al.* [9] proposed a method, representing a traf-

[7] However, there is a non-negligible model-loading time in our experiments, which leads to higher normalized values for smaller data-sets. The normalized value for the largest data-set is closest to the actual cost in the classification.

fic flow with some features (referred to as “descriptors”) such as packet size and connection duration. They applied two supervised learning techniques (k-NN and LDA) to define the relationship between different values of these descriptors and different Class-of-Service (CoS) categories. However, in common with a number of authors of the time, Roughan *et al.* created ground-truth data using IANA port numbers as a criteria, i.e., 80=HTTP, 25=SMTP, 22=SSH. Alongside this overloading of the port information there is an assumption about the conventional protocol-application mapping such as HTTP=web browser, a mapping which is now demonstrated as wrong. In Section 2 we summarize the process by which we calculate ground-truth information, by making heavy use of manual verification our process breaks that protocol-application mapping and places no direct reliance on port number information.

Another interesting approach by Li *et al.* [28] used a significant number of machines to create a known albeit artificial ground-truth. We acknowledge the great effort of the authors however, their approach can only serve to compliment the capture of spatial heterogeneity among different networks and sites, as what we have done in this paper.

In a precursor of this paper, Moore & Zuev in [5] presented a statistical approach to classify the traffic into different types of services. A Naïve Bayes classifier, combined with kernel estimation and a correlation-based filtering algorithm, was used to solve the classification problem on offline TCP traces. The resulting accuracy up to 96% demonstrated the discriminative power of a combination of 12 flow-behavior features with a classical machine-learning algorithm. Li & Moore [29] extend this with the application of the C4.5 algorithm and the use of an “observation window”: to limit the total number of required packets from the beginning of each flow. These extensions allowed them to build an increased accuracy while also enabling online (real-time) classification. In contrast with the earlier work of Moore & Zuev, we have also focused upon algorithms that return high accuracy and lend themselves to use in real-time.

Williams *et al.* [6] carried out a comparison of five widely-utilized machine learning algorithms for the task of traffic classification. Among these algorithms, AdaBoost+C4.5 was demonstrated with the highest accuracy. Intended only as a guidebook for algorithms, their feature set is relatively unsophisticated, containing only packet lengths, total bytes, total packets, inter-arrival times, flow duration and protocol. While a small feature set has computation and storage advantages we demonstrate in Section 4 that by considering a much wider range of features, and then minimizing the feature set on the basis of quality of information, computation and storage advantages can be maintained while accuracy and effectiveness is improved.

Bernaille *et al.* [1] presented an approach to identify applications using start-of-flow information. The authors utilized the packet size and direction of the first 4 data packets in each flow as the features with which they trained Gaussian and Hidden Markov Models respectively. These models achieved 98.7% overall accuracy when assisted by an expanded port-number list, and 93.7% overall accuracy using simple prediction heuristics. The authors have further extended their work for the identification of encrypted traffic in [30].

There have been a number of works that attempted the same problem with different machine learning algorithms and feature set such as unsupervised clustering [4, 31] and maximum likelihood with the first order Markov chain for the state space of TCP control packet of a TCP connection [3]. A recent statistical-fingerprinting mechanism was proposed by the authors of [2]. This work includes the description of features (described as as application-fingerprints) and included the size of the IP packets, the inter-arrival time and the order of the packets seen on the link. While useful in some circumstance, without extra work, such mechanism are not best suited to the classification of the majority of Internet applications. In contrast, in Section 3, we present results from a group of general-purpose classification systems.

Differing from the per-flow classification approaches, other work utilizes information retrieved from a higher level of traffic aggregation — the end host. Karagiannis *et al.* [32] combined information from multiple levels such as the interaction between hosts on the host level, protocol usage and per-flow average packet size on the flow level. The results show an ability of classifying 80%-90% of the traffic with 95% accuracy. In [33] the authors present a related mechanism to profile user-activity and the behaviors of the end-hosts, as well as an analysis of the dynamic characteristics of host behaviors.

Notably there is also a lot of work focused on specific kinds of traffic or applications of interest. For example, [17] showed an interesting approach specifically intended to identify Skype traffic by recognizing special characteristics of Skype and Voice-over-IP traffic. While a number of papers have been published focussing on the identification of P2P applications: [34] effectively identified P2P traffic using several sources of information taken from the IP-tuple connection graph. Further, [35] proposed a way to identify P2P activities through the observation of the connection graph property and the client-server property of an end host.

6 Conclusion

Motivated by the importance of accurate identification for a range of applications, we have explored the effective and efficient classification of network-based applications using only the observed network-traffic.

We presented results of a comparison among a number of classification schemes. Significantly, we only used data for which we have established the ground-truth: the actual applications causing the network-traffic observed. Despite the burden of computing accurate ground-truth, we summarize a method we have used to accurately assess many hundreds of Gigabytes of raw data.

We demonstrated the accurate training of models from data with a high-confidence ground-truth and the use of an efficient, small, feature set derived from the reduction of a large flow feature list using a sophisticated feature selection mechanism. In combination with a small feature list, we further show the return of good results with the use of sub-sampling based upon a limited observation window. We conclude the combination of an observation-window and the machine-learning method (C4.5) is transformative: enabling this approach to make real impact in online, real-time, classification applications.

We have documented a unique study in both the temporal and spatial domains: comparing classification performance across several years at one location and across several locations. These results are unequivocal in demonstrating the effectiveness of C4.5 mechanisms, the brittleness of packet-content methods and the need for samples of traffic from multiple sites to provide for accurate multi-site classification.

We demonstrated the computational complexity and measured the costs within the classification pipeline, noting the non-linear trade-offs incurred by some methods, the limitations of deep-packet inspection, and the need for observation-window based mechanisms.

Finally, while some past publications had investigated UDP traffic classification within the scope of a single application, in contrast with them, we illustrate the classification of UDP (and TCP) traffic using an application-independent classifier. We use the same classification approach for both stateful flows (TCP) and stateless flows based upon UDP. Importantly, we demonstrate high levels of accuracy: greater than 92% for the worst circumstance regardless of the application.

Data

While anonymity needs and legal obligations limit an out-right release of our raw data-sets; we make available to the community anonymized data files as well as software at http://www.cl.cam.ac.uk/research/srg/netos/brasil/

Thanks

This work is supported by the Engineering and Physical Sciences Research Council through grant GR/T10510/02. We thank the numerous network-operations groups who have supported this research, they include JaNet, Dante, KAREN, AT&T and GARR; this work would have been impossible with their support and assistance.

We thank the anonymous reviewers for their many helpful comments and suggestions, we also thank Jon Crowcroft, Damien Fay and Ralphe Neill for their helpful comments on drafts of this work.

References

[1] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proceedings of the 2006 ACM Conference on Emerging Network Experiment and Technology (CoNEXT'06)*, December 2006.

[2] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic classification through simple statistical fingerprinting. *SIGCOMM Computer Communication Review*, January 2007.

[3] H. Dahmouni, S. Vaton, and D. Rosse. A markovian signature-based approach to IP traffic classification. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data (MineNet'07)*, June 2007.

[4] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data (MineNet'06)*, September 2006.

[5] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the 2005 ACM SIGMETRICS*, pages 50–60, 2005.

[6] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *SIGCOMM Computer Communication Review*, October 2006.

[7] A. W. Moore and D. Papagiannaki. Toward the accurate identification of network applications. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM'05)*, volume 3431. Springer-Verlag LNCS, March 2005.

[8] Getbymail: Remote access & file sharing by mail. http://www.getbymail.com/.

[9] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A statistical signature-based approach to IP traffic classification. In *Proceedings of the 2004 ACM SIGCOMM Internet Measurement Conference (IMC'04)*, October 2004.

[10] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *Proceedings of the 2004 ACM SIGCOMM Internet Measurement Conference (IMC'04)*, October 2004.

[11] A. W. Moore, D. Zuev, and M Crogan. Discriminators for use in flow-based classification. Technical Report RR-05-13, Department of Computer Science, Queen Mary, University of London, September 2005.

[12] Marco Canini, Wei Li, Andrew W. Moore, and Raffaele Bolla. GTVS: Boosting the collection of application traffic ground truth, 2009. In submission (submitted 2008).

[13] Wei Li, Andrew W. Moore, and Marco Canini. Classifying HTTP traffic in the new age. In *ACM SIGCOMM 2008, Poster*, August 2008.

[14] Application layer packet classifier for linux. http://l7-filter.sourceforge.net/.

[15] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better NetFlow. In *Proceedings of the 2004 ACM SIGCOMM*, August 2004.

[16] Marco Canini, Damien Fay, David J. Miller, Andrew W. Moore, and Raffaele Bolla. Per flow packet sampling for high-speed network monitoring, 2009. In submission (submitted 2008).

[17] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing skype traffic: when randomness plays with you. In *Proceedings of the 2007 ACM SIGCOMM*, August 2007.

[18] Raffaele Bolla, Marco Canini, Riccardo Rapuzzi, and Michele Sciuto. On the double-faced nature of P2P traffic. In *Proceedings of the Sixteenth Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'08)*, pages 524–530, 2008.

[19] J. R. Quinlan. *C4.5: Program for Machine Learning.* Morgan Kaufman, 1993.

[20] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, 2003.

[21] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons Inc., 2001.

[22] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann Publishers, 2005.

[23] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kimberly C. Claffy, and Michalis Faloutsos. Is P2P dying or just hiding? In *Proceedings of the 2004 IEEE GLOBECOM: Global Internet and Next Generation Networks*, November 2004.

[24] David A. Patterson, David D. Clark, Anna Karlin, Jim Kurose, Edward D. Lazowska, David Liddle, Derek McAuley, Vern Paxson, Stefan Savage, and Ellen W. Zegura. *Looking Over the Fence at Networks: A Neighbor's View of Networking Research*. Computer Science and Telecommunications Board, National Academy of Sciences, Washington, DC, 2001.

[25] Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the Thirteenth USENIX conference on System Administration (LISA'99)*, 1999.

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

[27] A. De Montigny-Leboeuf. Flow attributes for use in traffic characterization. Technical report, Communications Research Centre Canada, December 2005.

[28] Li Jun, Zhang Shunyi, Lu Yanqing, and Zhang Zailong. Internet traffic classification using machine learning. In *Proceedings of Second International Conference on Communications and Networking in China, 2007. (CHINACOM '07)*, pages 239–243, Aug. 2007.

[29] Wei Li and Andrew W. Moore. A machine learning approach for efficient traffic classification. In *Proceedings of the 2007 IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'07)*, October 2007.

[30] L. Bernaille and R. Teixeira. Early recognition of encrypted applications. In *Proceedings of the Eighth Passive and Active Measurement Conference (PAM'07)*, April 2007.

[31] K. Gopalratnam, S. Basu, J. Dunagan, and H. Wang. Automatically extracting fields from unknown network protocols. In *Proceedings of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML'06)*, June 2006.

[32] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: multilevel traffic classification in the dark. In *Proceedings of the ACM SIGCOMM 2005*, pages 229–240, 2005.

[33] Thomas Karagiannis, Konstantina Papagiannaki, Nina Taft, and Michalis Faloutsos. Profiling the end host. In *Proceedings of the Eighth Passive and Active Measurement Conference (PAM'07)*, April 2007.

[34] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kimberley C. Claffy. Transport layer identification of P2P traffic. In *Proceedings of the 2004 ACM SIGCOMM Internet Measurement Conference (IMC'04)*, October 2004.

[35] F. Constantinou and P. Mavrommatis. Identifying known and unknown peer-to-peer traffic. In *Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, 2006.