BLINC: Multilevel Traffic Classification in the Dark

Thomas Karagiannis
UC Riverside
tkarag@cs.ucr.edu

Konstantina Papagiannaki
Intel Research, Cambridge
dina.papagiannaki@intel.com

Michalis Faloutsos
UC Riverside
michalis@cs.ucr.edu

ABSTRACT

We present a fundamentally different approach to classifying traffic flows according to the applications that generate them. In contrast to previous methods, our approach is based on observing and identifying patterns of host behavior at the transport layer. We analyze these patterns at three levels of increasing detail (i) the social, (ii) the functional and (iii) the application level. This multilevel approach of looking at traffic flow is probably the most important contribution of this paper. Furthermore, our approach has two important features. First, it operates *in the dark*, having (a) no access to packet payload, (b) no knowledge of port numbers and (c) no additional information other than what current flow collectors provide. These restrictions respect privacy, technological and practical constraints. Second, it can be tuned to balance the accuracy of the classification versus the number of successfully classified traffic flows. We demonstrate the effectiveness of our approach on three real traces. Our results show that we are able to classify 80%-90% of the traffic with more than 95% accuracy.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Measurement, Algorithms

Keywords: Traffic classification, host behavior, transport layer.

1. INTRODUCTION

Classifying traffic flows according to the applications that generate them is an important task for (a) effective network planning and design, and (b) monitoring the trends of the applications in operational networks. However, an accurate method that can reliably identify the generating application of a flow is still to be developed. In this work, we address the problem of traffic classification; the ultimate goal is to offer a tool to network operators that will provide a meaningful classification per application, and if this is infeasible, with useful insight into the traffic behavior. The latter may facilitate detection of abnormalities in the traffic, malicious behavior or identification of novel applications.

Currently, application classification practices rely to a large extent on the use of transport-layer port numbers. While this practice may have been effective in the early days of the Internet, port numbers currently provide limited information. Often, applications and users are not cooperative and intentionally or not use inconsistent ports. Thus, "reliable" traffic classification requires the packet-payload examination, which is scarcely an option due to: (a) hardware and complexity limitations, (b) privacy and legal issues, (c) payload encryption by the applications.

Taking into account empirical application trends [8, 20] and the increasing use of encryption, we conjecture that traffic classifiers of the future will need to classify traffic "in the dark". In other words, we need to address the traffic classification problem with the following constraints: (i) no access to user payload is possible, (ii) well-known port numbers cannot be assumed to indicate the application reliably, and (iii) we can only use the information that current flow collectors provide. Clearly, there may be cases where these constraints may not apply, which would make the classification easier. However, we would like to develop an approach that would be applicable and deployable in most practical settings.

Recently, some novel approaches treat the problem of application classification as a statistical problem. These approaches develop discriminating criteria based on statistical observations and distributions of various flow properties in the packet traces. Typically, such discriminating criteria refer to the packet size distribution per flow, the inter-arrival times between packets etc. However, for the most part, these methods do not exploit network-related properties and characteristics, that we believe contain a lot of valuable information. In addition, the validation of a classification method is a challenge. The effectiveness of most of the current approaches has not been validated in a large scale, since there does not exist a reference point or a benchmark trace with known application consistency.

In this work, we propose a novel approach for the flow classification problem as defined above, which we call **BLINd Classification** or ***BLINC*** for short. The novelty of our approach is twofold. First, we shift the focus from classifying individual flows to associating Internet hosts with applications, and then classifying their flows accordingly. We argue that observing the activity of a host provides more information and can reveal the nature of the applications of the host. Second, *BLINC* follows a different philosophy from previous

SIGCOMM'05, August 21–26, 2005, Philadelphia, Pennsylvania, USA.

methods attempting to capture the inherent behavior of a host at three different levels: (a) social level, (b) network level, and (c) the application level.

Combining these two key novelties, we classify the behavior of *hosts* at three different levels. While each level of classification provides increasing knowledge of host behavior, identifying specific applications depends on the unveiled "cross-level" characteristics (i.e., a single level *cannot* reveal the generating application by itself).

- At the *social level*, we capture the behavior of a host as indicated by its interactions with other hosts. First, we examine the popularity of a host. Second, we identify communities of nodes, which may correspond to clients with similar interests or members of a collaborative application.
- At the *functional level*, we capture the behavior of the host in terms of its functional role in the network, namely whether it acts as a provider or consumer of a service, or both, in case of a collaborative application. For example, hosts that use a single port for the majority of their interactions with other hosts are likely to be providers of the service offered on that port.
- At the *application level*, we capture the transport layer interactions between particular hosts on specific ports with the intent to identify the application of origin. First, we provide a classification using only 4-tuples (source address, destination address, source port, and destination port). Then, we refine the classification further by exploiting other flow characteristics such as the transport protocol or the average packet size.

Tunability. A key feature of our methodology is that it provides results at various levels of detail and accuracy. First, *BLINC* analyzes traffic at the aforementioned three levels. Second, the classification criteria are controlled by thresholds, that when relaxed or tightened, achieve the desired balance between an aggressive and a conservative classification. The level of accuracy and detail may be chosen according to: (a) the goal of the study, and (b) the amount of exogenous information (e.g., application specifications).

The highlights of our work can be summarized in the following points:

- *Developing a classification benchmark.* We provide a comparison benchmark for flow classification. We collect *full* payload packet traces, and we develop a payload classification methodology. While this methodology could be of independent interest, we use it here to evaluate *BLINC*, which is the focus of this work.
- *Identifying patterns of behavior.* We identify "signature" communication patterns, which can help us identify the applications that a host is engaged in. Using these patterns, we develop a systematic methodology to implement our multilevel approach.
- *Highly accurate classification.* We successfully apply our approach to several real traces. While training was based on one of our datasets, our approach manages to classify successfully 80%-90% of the total traffic with more than 95% accuracy in *all* our traces.
- *Detecting the "unknown".* We show how our approach can help us detect: (a) unknown applications, such as a new p2p protocol, and (b) malicious flows, which emerge as deviations from the expected behavior. Note that these cases cannot be identified by payload or port-based analysis.

Our work in perspective. To the best of our knowledge, this is the first work to advocate the shift from characterizing flows by application to associating hosts with applications. Our methodology is a first attempt at exploring the benefits and limitations of such an approach. Given the quality of our results, we feel that our approach shows great promise and opens interesting new directions for future research.

The remainder of the paper is structured as follows. In Section 2, we motivate the problem and describe related work. In Section 3, we present our payload-based classification technique. *BLINC* is presented in Section 4 and its performance results are shown in Section 5. Section 6 discusses implementation details, limitations and future extensions to *BLINC*. Finally, section 7 concludes our paper.

2. BACKGROUND

Analysis of the application traffic mix has always been one of the major interests for network operators. Collection of traffic statistics is currently performed either by flow monitors, such as Cisco NetFlow, or by sophisticated network monitoring equipment, that captures one record for each (sampled) packet seen on a link. The former produces a list of flow records capturing the number of bytes and packets seen, while the latter produces a list of packet records that can also be aggregated into 5-tuple flows (e.g., with the same source, destination IP address, source, destination port, and protocol). The subsequent mapping of flows, however, to application classes is not as straightforward and has recently attracted attention in the research community.

While port numbers were always an approximate yet sufficient methodology to classify traffic, port-based estimates are currently significantly misleading due to the increase of applications tunneled through HTTP (e.g., chat, streaming), the constant emergence of new protocols and the domination of peer-to-peer (*p2p*) networking. Indeed, studies have confirmed the failure of port-based classification [14].

To address the inefficiency of port-based classification, recent studies have employed statistical classification techniques to probabilistically assign flows to classes, e.g., machine learning [12] or statistical clustering [18, 15]. In such approaches, flows are grouped in a predetermined number of clusters according to a set of discriminants, that usually includes the average packet size of a flow, the average flow duration, and the inter-arrival times between packets (or the variability thereof). Studies have also examined how the exact timing and sequence of packet sizes can describe specific applications in the slightly different context of generating realistic application workloads [7].

One of the most challenging application types is *p2p* traffic. Quantifying *p2p* traffic is problematic both due to the large number of proprietary *p2p* protocols, but also because of the intentional use of random port numbers for communication. Payload-based classification approaches tailored to *p2p* traffic have been presented in [19, 9], while identification of *p2p* traffic through transport layer characteristics is proposed in [8]. In the same spirit, Dewes et al. [5] look into the problem of identifying and characterizing *chat* traffic. Our work goes beyond previous efforts aiming at classifying most of the applications that generate the majority of today's Internet traffic by describing their underlying behavior. Par-

Table 1: General workload dimensions of our traces.

Set	Date	Day	Start	Dur	Direc.	Src.IP	Dst.IP	Packets	Bytes	Aver.Util.	Aver. Flows.
GN	2003-08-19	Tue	17:20	43.9 h	Bi-dir.	1455 K	14869 K	1000M	495 G	25 Mbps	105 K
UN1	2004-01-20	Tue	16:50	24.6 h	Bi-dir.	2709 K	2626 K	2308 M	1223 G	110.5 Mbps	596 K
UN2	2004-04-23	Fri	15:40	33.6 h	Bi-dir.	4502 K	5742 K	3402 M	1652 G	109.4 Mbps	570 K

Table 2: Application specific bit-strings at the beginning of the payload."0x" implies Hex characters.

Application	*String*	*Trans. prot.*
eDonkey2000	0xe319010000	TCP/UDP
MSN messenger	"PNG"0x0d0a	TCP
IRC	"USERHOST"	TCP
nntp	"ARTICLE"	TCP
ssh	"SSH"	TCP

allel to our work, profiling of end-host communications in backbone Internet traffic has been proposed in [23].

3. PAYLOAD-BASED CLASSIFICATION

This section describes our payload classifier. Our data feature the unique property of allowing for accurate classification; our monitors capture the full payload of each packet instead of just the header as is commonly the case. Thus, we can move beyond simple port-based application classification and establish a comparison benchmark. To achieve efficient payload classification, we develop a signature-matching classifier able to classify the majority of current Internet traffic. While gathering payload data is not directly related to the design of *BLINC*, payload-based classification guides our analysis in profiling host behavior and establishes a comparison reference point to evaluate *BLINC*'s performance.

3.1 Payload packet traces

We use packet traces collected using a high speed monitoring box [13] installed on the Internet link of two access networks. We capture every packet seen on each direction of the link along with its *full* payload.

Table 1 lists general workload dimensions of our data sets: counts of distinct source and destination IP addresses, the numbers of packets, and bytes observed, the average utilization and the average number of flows per 5-minute interval. Throughout the paper, flows are defined according to their 5-tuple (source and destination IP address, source and destination port, protocol) and expire if they are idle for 64 seconds [4]. We processed traces with CAIDA's Coral Reef suite [11]. The monitored networks are the following:

Genome campus: Our traces (*GN* in table 1) reflect traffic of several biology-related facilities. There are three institutions on-site that employ about 1,000 researchers, administrators and technical staff.

Residential university: We monitor numerous academic, research and residential complexes on-site (*UN1* and *UN2* traces in table 1). Collectively we estimate a user population of approximately 20,000. The residential nature of the university reflects traffic covering a wider cross-section of applications.

The two sites and time-of-capture of the analyzed traces were selected so that our methodology could be tested against a variety of different conditions and a diverse set of applications. Indeed, the selected links reflect significantly different network "types", evident from the application mix as analyzed in the following section. In addition, the two university traces were collected both during weekdays (*UN1*) and also beginning of weekend (*UN2*) to capture possible weekday to weekend variation in application usage and network traffic patterns. Finally, the traces were captured several months apart from each other to minimize potential similarities in the offered services and client interactions.

3.2 Payload classification

Even with access to full packet payload, classification of traffic is far from trivial. The main complication lies in the fact that payload classification of traffic requires *a priori* knowledge of application protocol signatures, protocol interactions and packet formats. While some of the analyzed applications are well-known and documented in detail, others operate on top of nonstandard, usually custom-designed proprietary protocols. To classify such diverse types of traffic, we develop a signature-based classifier in order to avoid manual intervention, automate the analysis and speed-up the procedure.

Our classifier is based on identifying characteristic bit strings in the packet payload that potentially represent the initial protocol handshake in most applications (e.g., HTTP requests). Protocol signatures were identified either from RFCs and public documents in case of well-documented protocols, or by reverse-engineering and empirically deriving a set of distinctive bit strings by monitoring both TCP and UDP traffic using tcpdump [22]. Table 2 lists a small subset of such signature (bit) strings for TCP and UDP. The complete list of bit strings we used is presented in [10].

Our payload technique operates on two different time scales and traffic granularities. The short time scale operates on a per packet basis upon each packet arrival. The coarse time scale essentially summarizes the results of the classification process during the preceding time interval (we use intervals of 5 minutes throughout the paper) and assists in the identification of flows that potentially have remained unclassified during payload analysis.

Both operations make use of an {*IP, port*} pair table that contains records of the IP-port pairs that have already been classified based on past flows. These {IP, port} pairs associate a particular IP address and a specific port with a code reflecting its causal application. The {IP, port} table is updated upon every successful classification and consulted at the end of each time interval for evidence that could lead to the classification of unknown flows or the correction of flows mis-classified under the packet level operation. Since the service reflected at a specific port number for a specific IP does not change at the time-scales of interest, we use this heuristic to reduce processing overhead. To avoid increasing memory requirements by storing an immense number of {IP, port} pairs, we only keep {IP, port} pairs that reflect known services such as those described in table 3. Lastly, to further identify data transfer flows, such as passive ftp, we parse the control stream to acquire the context regarding the upcoming data transfer, i.e. the host and port number where the follow-up data connection is going to take place.

The per-packet operation simply examines the contents of each packet against our array of strings, and classifies the corresponding flow with an application-specific tag in case of a match. Successful classification of a flow on one direction leads to the subsequent classification of the respective flow in

Table 3: Categories, applications analyzed and their average traffic percentages in flows (bytes).

Category	Application/protocol	GN	UN1	UN2
web	www	32% (14.0%)	31.8% (37.5%)	24.7% (33.5%)
p2p	FastTrack, eDonkey2000, BitTorrent, Gnutella WinMx, OpenNap, Soulseek, Ares, MP2P Dirrect Connect, GoBoogy, Soribada, PeerEnabler	0.3% (1.2%)	25.5% (31.9%)	18.6% (31.3%)
data (ftp)	ftp, databases (MySQL)	1.1% (67.4%)	0.3% (7.6%)	0.2% (5.4%)
Network management (NM)	dns, netbios, smb, snmp, ntp, spamassasin, GoToMyPc	12.5% (0.1%)	9% (0.5%)	9.4% (0.2%)
mail	mail (smtp, pop, imap, identd)	3.1% (3.4%)	1.8% (1.4%)	2.5% (0.9%)
news	news (nntp)	0.1% (4.0%)	0% (0.3%)	0% (0.2%)
chat/irc (chtirc)	IRC, msn messenger, yahoo messenger, AIM	3.7% (0.0%)	1.8% (0.2%)	5.8% (0.7%)
streaming (strm)	mms (wmp), real, quicktime, shoutcast vbrick streaming, logitech Video IM	0.1% (0.8%)	0.2% (6%)	0.2% (6.8%)
gaming (gam)	HalfLife, Age of Empires, etc.	–	0.3% (0.1%)	0.3% (0.3%)
Nonpayload	–	45.3% (2.2%)	24.9% (0.5%)	30.9% (1.0%)
Unknown	–	1.3% (6.6%)	4.3% (11.9%)	7.3% (16.9%)

the reverse direction, if it exists. Previously classified flows are not examined, unless they have been classified as *HTTP*. This further examination allows identification of non-web traffic relayed over HTTP (e.g., streaming, p2p, web-chat).

At the end of each time interval, we compare all flows against our list of known {IP, port} pairs, to classify possible unknown flows or correct misclassifications (e.g., a *p2p* flow that was classified under *web*, because the only packet so far was an HTTP request or response).

3.3 Application breakdown

We classify flows in eleven distinct application-type categories. Table 3 lists these categories, the specific applications and their share of traffic as percentage of the total number of flows and bytes in the link. The *nonpayload* category includes flows that transfer only headers and no user-data throughout their lifetime, while the *unknown* category lists the amount of traffic that could not be classified.

As expected, the two types of network (*GN* vs *UN*) appear significantly different. The *UN* network is mostly dominated by *web* and *p2p* traffic, whereas *GN* contains a large portion of *ftp* traffic reflecting large data transfers of Genome sequences. Despite the difference in the day of capture and the large interval between the two *UN* traces, their traffic mix is roughly similar, while encrypted flows (*SSH/SSL*) correspond to 1%-2% of the traffic in all traces. Other interesting observations from these traces are:

Nonpayload flows account almost for one third of all flows in both links! Examination of these flows suggests that the vast majority corresponds to failed TCP connections on ports of well-known exploits or worms (e.g., 135). Large percentage of address space scans is also implied by the large number of destination IPs especially in the *GN* trace (table 1).

Unknown flows: The existence of user payload data does not guarantee that all flows in our traces will be classified. Our analysis of the most popular applications cannot possibly guarantee identification of all applications contributing traffic to the Internet. For example, 4%-7% of all flows (10% in bytes) of the *UN* traffic cannot be identified. Note that a fraction of this unknown traffic is due to experimental traffic originating from *PlanetLab* (three *PlanetLab* nodes exist behind our monitoring point).

4. TRANSPORT LAYER CLASSIFICATION

This section describes our multi-level methodology, *BLINC*, for the classification of flows into applications without the use of the payload or "well-known" port numbers. *BLINC* realizes a rather different philosophy compared to other approaches proposed in the area of traffic classification. The main differences are the following:

- We do not treat each flow as a distinct entity; instead, we focus on the source and destination hosts of the flows. We advocate that when the focus of the classification approach is shifted from the flow to the host, one can then accumulate sufficient information to disambiguate the roles of each host across different flows, and thus identify specific applications.
- Our approach operates on flow records and requires no information about the timing or the size of individual packets. Consequently, the input to our methodology may potentially be flow record statistics collected by currently deployed equipment.
- Our approach is insensitive to network dynamics such as congestion or path changes, that can potentially affect statistical methodologies which rely heavily on inter-arrival times between the packets in a flow.

4.1 The Overview of BLINC

BLINC operates on flow records. Initially, we parse all flows and gather host-related information reflecting transport layer behavior. We then associate the host behavior with one or more application types and thus indirectly classify the flows. The host behavior is studied across three different levels:

- At the *social level*, we capture the behavior of a host in terms of the number of other hosts it communicates with, which we refer to as *popularity*. Intuitively, this level focuses on the diversity of the interactions of a host in terms of its destination IPs and the existence of user communities. As a result, we only need access to the source and destination IP addresses at this level.
- At the *functional level*, we capture the behavior of the host in terms of its functional role in the network, that is, whether it is a provider or consumer of a service, or whether it participates in collaborative communications. For example, hosts that use a single source port for the majority of their interactions are likely to be providers of a service offered on that port. At this level, we analyze characteristics of the source and destination IP address, and the source port.
- At the *application level*, we capture the transport layer interactions between hosts with the intent to identify the application of origin. We first provide a classification using only the 4-tuple (IP addresses and ports),

and then we refine the final classification, by developing heuristics that exploit additional flow information, such as the number of packets or bytes transferred as well as the transport protocol. For each application, we capture host behavior using empirically derived patterns. We represent these patterns using graphs, which we call ***graphlet*s**. Having a library of these *graphlet*s, we then seek for a match in the behavior of a host under examination.

We want to stress that throughout our approach, we treat the port numbers as indexes without any application-related information. For example, we count the number of distinct ports a host uses, but we do not assume in any way that the use of port 80 signifies web traffic.

While the preceding levels are presented in order of increasing detail, they are equally significant. Not only analysis at each level will benefit from the knowledge acquired in the previous level, but also the classification depends on the unveiled "cross-level" characteristics. However, the final classification of flows into applications cannot be achieved without examination of the application level characteristics.

A key advantage of the proposed approach is its tunability. The strictness of the classification criteria can be tailored to the goal of the measurement study. These criteria can be relaxed or tightened to provide results at different points in the trade off between the completeness of the classification versus its accuracy.

BLINC provides two types of output. First, it reports aggregate per-class statistics, such as the total number of packets, flows and bytes. Second, it produces a list of all flows (5-tuple) tagged with the corresponding application for every time interval. Furthermore, *BLINC* can detect unknown or non-conformant hosts and flows (section 5).

Throughout the remaining of this section, we will present characteristic samples of behaviors at each level as seen in our traces. For the sake of presentation, we will only demonstrate examples for a limited time-interval of one of the traces (5 or 15 minutes); however, these observations were typical for all our traces.

4.2 Classification at the social level

We identify the social role of each host in two ways. First, we focus on its *popularity*, namely the number of distinct hosts it communicates with. Second, we detect *communities* of hosts by identifying and grouping hosts that interact with the same set of hosts. A community may signify a set of hosts that participate in a collaborative application, or offer a service to the same set of hosts.

Examining the social behavior of single hosts. The social behavior of a host refers to the number of hosts this particular host communicates with. To examine variations in host social behavior, Fig. 1 presents the complementary cumulative distribution function (CCDF) of the host *popularity*. Based on payload classification from section 3, we display four different CCDFs corresponding to different types of traffic, namely *web*, *p2p*, *malware* (e.g., failed nonpayload connections on known malware ports), and *mail*. In all cases, the majority of sources appear to communicate with a small set of destination IPs.

In general, the distribution of the host *popularity* cannot reveal specific rules in order to discriminate specific applications, since it is highly dependent upon the type of network, link or even the specific IPs. However, this distribution allows us to distinguish significant differences among applications. For example, hosts interacting with a large number of other hosts in a short time period appear to either participate in a *p2p* network or constitute malicious behavior. In fact, the *malware* curve, appears flat below 100 destination IPs, denoting the presence of a large number of possible address-space scans, where a number of sources has the same number of destination IPs during a specific time interval.

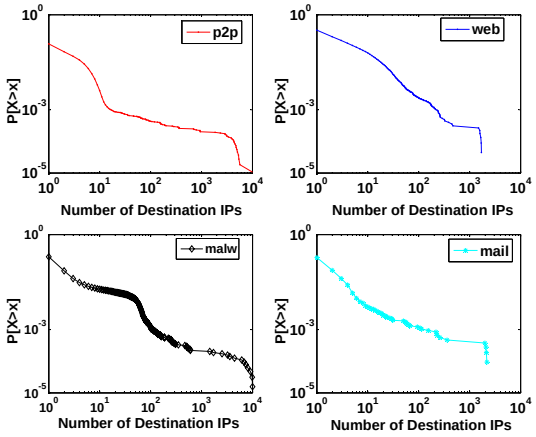

Figure 1: Complementary cumulative distribution function of destination IP addresses per source IP for 15 minutes of the *UN1* trace for four different applications.

Detecting communities of hosts. Social behavior of hosts is also expressed through the formation of communities or clusters between sets of IP addresses. Communities will appear as *bipartite cliques* in our traces, like the one shown in Fig. 2. The bipartite graph is a consequence of the single observation point. Interactions between hosts from the same side of the link are not visible, since they do not cross the monitored link. Communities in our bipartite graph can be either exact cliques where a set of source IPs communicates with the exact same set of destination IPs, or approximate cliques where a number of the links that would appear in a perfect clique is missing. Communities of interest have also been studied in [1], where a community is defined by either the popularity of a host or frequency of communications between hosts. Our definition of community is targeted to groups of hosts per application type.

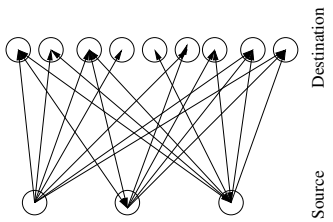

Figure 2: An example of a community in our traces: the graph appears as an approximate bipartite clique.

Identifying the communities is far from trivial, since identifying maximal cliques in bipartite graphs is an NP-Complete problem. However, there exist polynomial algorithms for identifying the *cross-associations* in the data mining context [3]. Cross-association is a joint decomposition of a binary matrix into disjoint row and column groups such that the rectangular intersections of groups are homogeneous or formally approximate a bipartite clique. In our case, this bi-

nary matrix corresponds to the interaction matrix between the source and destination IP addresses in our traces.

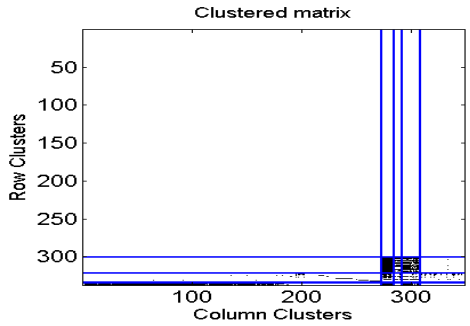

Figure 3: Communities of on-line game players appear as highly connected clusters in the interaction matrix after applying the cross-associations algorithm (*UN1* trace, 5mins).

To showcase how communities can provide interesting features of host behavior, we apply the cross association algorithm to gaming traffic for a small time period in one of our traces (a 5-minute interval of the *UN1* trace) and we successfully identify communities of gamers. Specifically, Fig. 3 presents the interaction matrix after the execution of the cross-association algorithm. The axes present source (*x-axis)* and destination (*y-axis*) IPs (350 total IPs), while the matrix is essentially the original interaction matrix shifted in such a way so that strongly connected components appear clustered in the same area. The horizontal and vertical lines display the boundaries of the different clusters. Specifically, we observe two major clusters: First, three source IPs communicating with a large number of destination IP addresses although not an exact clique (at the bottom of Fig. 3, *x-axis:0-280*, *y-axis:347-350*). Second, an exact clique of five hosts communicating with the exact same 17 destination IPs (*x-axis:280-285*, *y-axis*:300-317).

In general, we study three different types of communities, according to their deviation from a perfect clique:

"Perfect" cliques: a hint for malicious flows. While the previous example displays a perfect clique in gaming traffic, we find that perfect cliques are mostly signs of malicious behavior. In our analysis, we identify a number of hosts communicating with the exact same list of IP addresses (approximately 250 destination IPs in 15 minutes). Further analysis revealed that these cases represented malicious traffic, such as flows for the Windows RPC exploit and Sasser worm.

Partial overlap: collaborative communities or common interest groups. In numerous cases, only a moderate number of common IP addresses appear in the destination lists for different source IPs. These cases correspond to peer-to-peer sources, gaming and also clients that appear to connect to the same services at the same time (e.g., browsing the same web pages, or streaming).

Partial overlap within the same domain: service "farms". Closer investigation of partial overlap revealed hosts interacting with a number of IP addresses within the same domain, e.g., IP addresses that differ only in the least significant bits. Payload analysis of these IPs revealed that this behavior is consistent with service "farms": multi-machine servers that load-balance requests of a host to servers within the same domain. We find that service "farms" were used to offer *web*, *mail*, *streaming*, or even *dns* services.

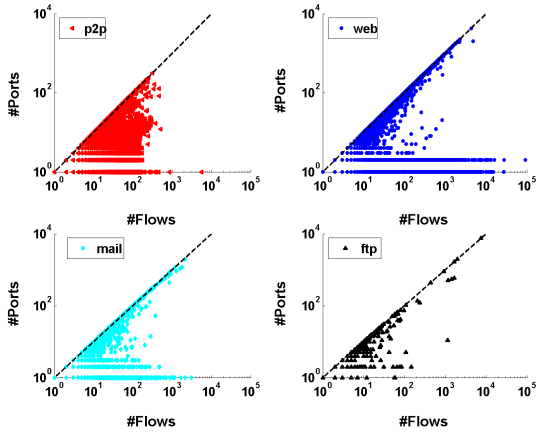

Figure 4: Number of source ports versus number of flows per source IP address in the *UN1* trace for a 15-minute interval for four different applications. In client-server applications (web,ftp,mail), most points fall on the diagonal or on horizontal lines for small values in the *y-axis* (number of used ports). In p2p, points are clustered in-between the diagonal and the *x-axis*.

The richness of the information that we discover at this level and the social role of a host is an interesting topic in its own sake. However, further analysis of social behavior and its implications is out of the scope of this work.

Conclusion and Rules: Based on the above analysis, we can infer the following regarding the social behavior of network hosts. First, "neighboring" IPs may offer the same service (e.g., server farms). Thus, identifying a server might be sufficient to classify such "neighboring" IPs under the same service (if they use the same service port). Second, exact communities may indicate attacks. Third, partial communities may signify p2p or gaming applications. Finally, most IPs act as clients having a minimum number of destination IPs. Thus, focusing on the identification of the small number of servers can retrospectively pinpoint the clients, and lead to the classification of a large portion of the traffic, while limiting the amount of associated overhead. Identification of server hosts is accomplished through the analysis of the functional role of the various hosts.

4.3 Classification at the functional level

At this level, we identify the functional role of a host: hosts may primarily offer services, use services, or both. Most applications operate with the server-client paradigm. However, several applications interact in a collaborative way, with *p2p* networks being the prominent example.

We attempt to capture the functional role by using the number of *source* ports a particular host uses for communication. For example, let us assume that host A provides a specific service (e.g., web server) and we examine the flows where A appears as a source. Then, A is likely to use a single source port in the vast majority of its flows. In contrast, if A were a client to many servers, its source port would vary across different flows. Clearly, a host that participates in only one or a few flows would be difficult to classify.

To quantify how the number of used source ports may distinguish client from server behavior, we examine the distribution of the source ports a host uses in our traces. In Fig. 4, we plot the number of flows (x-axis) versus the number of

source ports (y-axis) each source IP uses for 15 minutes of our *UN1* trace[1]. Each subplot of Fig. 4 presents traffic from a different application as identified by payload analysis. We identify three distinct behaviors:

Typical client-server behavior: Client-server behavior is most evident for *web* traffic (Fig. 4, top-right), where most points fall either on the diagonal or on horizontal lines parallel to the *x-axis* for small values of y (less or equal to two). The first case (where the number of ports is equal to the number of distinct flows) represents clients that connect to web servers using as many ephemeral source ports as the connections they establish. The latter case reflects the actual servers that use one ($y = 1$, port 80, HTTP) or two ($y = 2$, port 80, HTTP and 443, HTTPS) source ports for all of their flows.

Typical collaborative behavior: In this case, points are clustered between the *x-axis* and the diagonal, as shown in the *p2p* case in Fig. 4 (top-left), where discrimination between client and server hosts is not possible.

Obscure client-server behavior: In Fig. 4, we plot the behavior for the case of *mail* and *ftp*. While *mail* and *ftp* fall under the client-server paradigm, the behavior is not as clear as in the web case for two reasons:

- *The existence of multiple application protocols supporting a particular application*, such as *mail*. *Mail* is supported by a number of application protocols, i.e., SMTP, POP, IMAP, IMAP over SSL, etc., each of which uses a different service port number. Furthermore, mail servers often connect to *Razor* [17] databases through *SpamAssassin* to report spam. This practice generates a vast number of small flows destined to *Razor* servers, where the source port is ephemeral and the destination port reflects the SpamAssassin service. As a result, mail servers may use a large number of different source ports.
- *Applications supporting control and data streams*, such as *ftp*. Discriminating client-server behavior is further complicated in cases of separate control and data streams. For example, passive *ftp*, where the *ftp* server uses as source ports a large number of ephemeral ports different than the service ports (20,21), will conceal the ftp server.

Conclusion and Rules: If a host uses a small number of source ports, typically less or equal to two, for every flow, then this host is likely providing a service. Our measurements suggest that if a host uses only *one* source port number, then this host reflects a web, a chat or a SpamAssassin server in case of TCP, or falls under the Network Management category in case of UDP.

4.4 Classification at the application level

In this level, we combine knowledge from the two previous levels coupled with transport layer interactions between hosts in order to identify the application of origin. The basic insight exploited by our methodology is that interactions between network hosts display diverse patterns across the various application types. We first provide a classification using only the 4-tuple (IP addresses and ports), and then, we refine it using further information regarding a specific flow, such as the the protocol or the average packet size.

[1]The source {IP, port} pair is used without loss of generality. Observations are the same in the destination {IP, port} case.

We model each application by capturing its interactions through empirically derived signatures. We visually capture these signatures using *graphlet*s that reflect the "most common" behavior for a particular application. A sample of application-specific *graphlet*s is presented in Fig. 5. Each *graphlet* captures the relationship between the use of source and destination ports, the relative cardinality of the sets of unique destination ports and IPs as well as the magnitude of these sets.

Having a library of these *graphlet*s, allows us to classify a host by identifying the closest matching behavior. Since unknown behavior may match several *graphlet*s, the success of the classification will then have to rely on operator-defined thresholds to control the strictness of the match.

In more detail, each *graphlet* has four columns corresponding to the 4-tuple source IP, destination IP, source port and destination port. We also show some *graphlet*s with 5 columns, where the second column corresponds to the transport protocol (TCP or UDP) of the flow. Each node[2] presents a *distinct* entry to the set represented by the corresponding column, e.g., 135 in *graphlet* 5(a) is an entry in the set of destination ports. The lines connecting nodes imply that there exists at least one flow whose packets contain the specific nodes (field values). Dashed lines indicate links that may or may not exist and are not crucial to the identification of the specific application. Note that while some of the *graphlet*s display port numbers, the classification and the formation of *graphlet*s **do not associate in any way a specific port number with an application**.

The order of the columns in our visual representation of each *graphlet* mirrors the steps of our multilevel approach. Our starting field, the source IP address, focuses on the behavior of a particular host. Its *social* behavior is captured in the fanout of the second column which corresponds to all destination IPs this particular source IP communicates with. The functional role is portrayed by the set of source port numbers. For example, if there is a "knot" at this level the source IP is likely to be a server as mentioned before. Finally, application types are distinguished using the relationship of all four different fields. Capturing application-specific interactions in this manner can distinguish diverse behaviors in a rather straightforward and intuitive manner as shown in Fig. 5.

Let us highlight some interesting cases of *graphlet*s. The top row of Fig. 5 displays three types of attacks (*graphlet*s (a)(b)(c)). Fig. 5(a) displays a typical attack where a host scans the address space to identify vulnerability at a particular destination port. In such cases, the source host may or may not use different source ports, but such attacks can be identified by the large number of flows destined to a given destination port. A similar but slightly more complicated type of attack common in our traces involves hosts attempting to connect to several vulnerable ports at the same destination host (Fig. 5(b)). Similarly, we show the *graphlet* of typical port scan of a certain destination IP in Fig. 5(c).

The power of our method lies in the fact that we do not need to know the particular port number ahead of time. The surprising number of flows at the specific port will raise the suspicion of the network operator. Such behaviors are also identifiable by tools like *AutoFocus* [6], which however do not target traffic classification.

[2]The term node indicates the components of a *graphlet*, while the term host indicates an end-point in a flow.

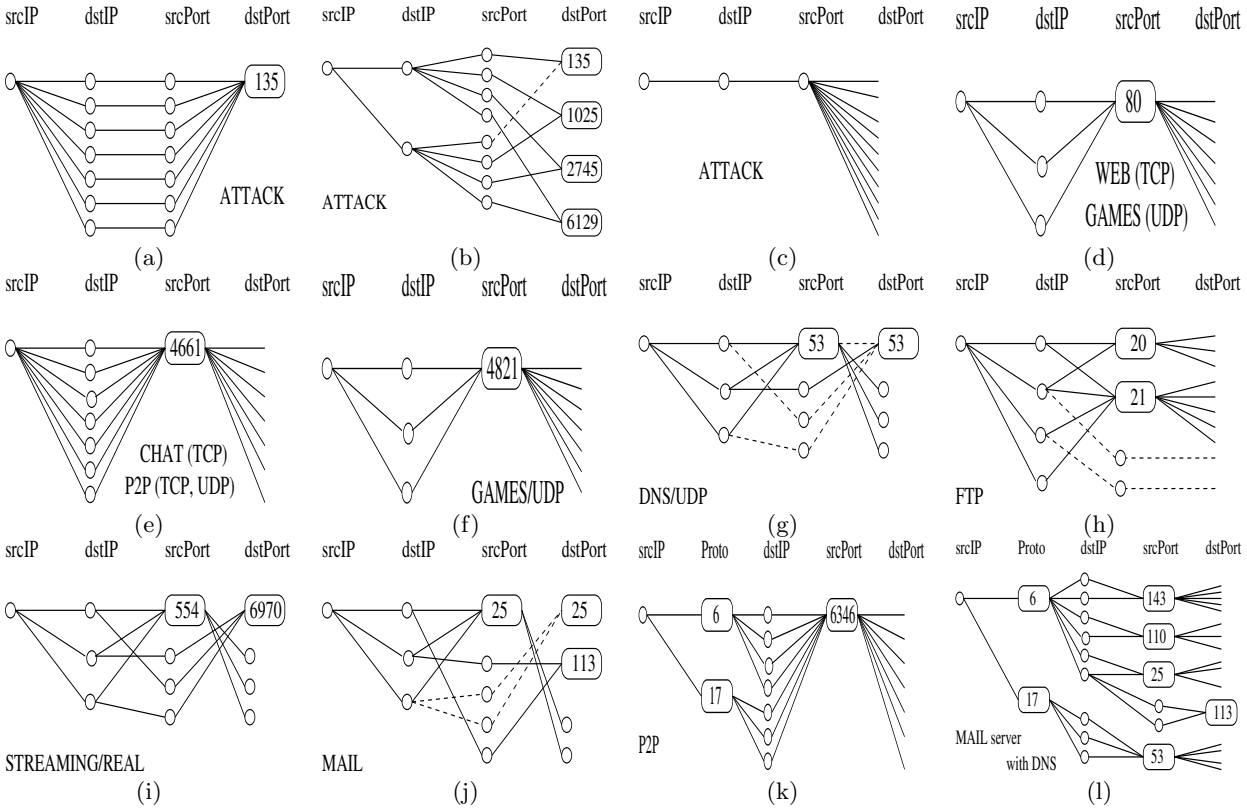

Figure 5: Visual representation of transport-layer interactions for various applications: port numbers are provided for completeness but are not used in the classification.

In some cases, hosts offering services on certain ports exhibit similar behavior. For instance, *p2p* (the server side), *web*, and *games* all result in the same type of *graphlet*: a single source IP communicates with multiple destinations using the same source port (the service port) on several different destination ports. In such cases, we need further analysis to distinguish between applications. First, we can use quantitative criteria such as the relative cardinality of the sets of destination ports versus destination IPs. As we will describe later in the section, the use of the transport protocol, TCP versus UDP, can further help to discriminate between applications with similar or complicated *graphlet*s.

Applications such as *ftp*, *streaming* or *mail* present more complicated *graphlet*s, exhibiting "cris-cross" flow interactions (Fig. 5(h)(i)(j)). These *graphlet*s have more than one service ports, or have both source and destination service ports. In the case of *ftp*, the source host provides the service at two main ports (control and data channel), whereas other source ports represent the case of *passive ftp*. *Streaming* on the other hand uses specific port numbers both at the source and the destination side. Streaming users (destination IPs in our case) connect at the service port (TCP) of the streaming server (control channel), while the actual streaming is initiated by the server using an ephemeral random source port to connect to a pre-determined UDP user port. Similarly *mail* uses specific port numbers at the source and destination side, yet all mail flows are TCP. *Mail* servers may further use port 25 both as source or destination port across different flows while connecting to other mail servers to forward mail. As previously noted, the specific port numbers are only listed to help with the description of these *graphlet*s and they are in no way considered in our algorithm.

Lastly, *graphlet*s become even more complex when services are offered through multiple application and/or transport protocols. As an example, Fig. 5(l) presents a mail server supporting IMAP, POP, SMTP, and ident, while also acting as a DNS server. Knowledge of the role of the host may assist as corroborative evidence on other services offered by the same host. For instance identifying a host as an SMTP server suggests that the same host may be offering POP, IMAP, DNS (over UDP) or even communicate with SpamAssassin servers.

4.5 Heuristics

Here, we present a set of final heuristics that we use to refine our classification and discriminate complex or similar cases of *graphlet*s. This set of heuristics has been derived empirically through inspection of interactions present in various applications in our traces.

Heuristic 1. Using the transport layer protocol. One criterion for such a distinction is the transport layer protocol used by the flow. The protocol information can distinguish similar *graphlet*s into three groups using: (a) *TCP*, which includes *p2p*, *web*, *chat*, *ftp* and *mail*, (b) *UDP*, which includes *Network Management traffic* and *games* and (c) both protocols, which includes *p2p*, *streaming*. For example, while *graphlet*s for *mail* and *streaming* appear similar, mail interactions occur only on top of TCP. Another interesting case is shown in Fig. 5(k), where *p2p* protocols may use both TCP and UDP with a single source port for both transport protocols (e.g., *Gnutella*, *Kazaa*, *eDonkey* etc.). With the exception of *dns*, our traces suggest that this behavior is unique to *p2p* protocols.

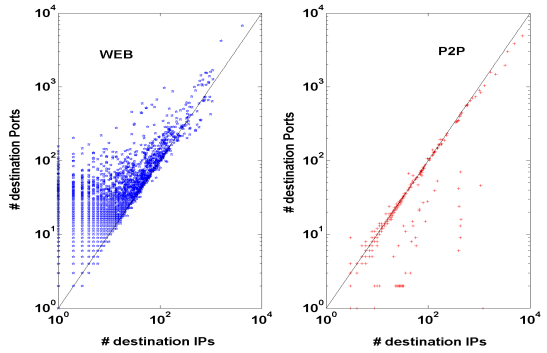

Figure 6: Relationship between the number of destination IP addresses and ports for specific applications per source IP. The cardinality of the set of destination ports is larger than the one of destination IPs reflected in points above the diagonal for web. On the contrary, points in the p2p case fall either on top or below the diagonal.

Heuristic 2. Using the cardinality of sets. As discussed earlier, the relative cardinality of destination sets (ports vs IPs) is able to discriminate different behaviors. Such behaviors may be *web* versus *p2p* and *chat*, or *Network Management* versus *gaming*. Fig. 6 presents the number of distinct destination IPs versus the number of distinct destination ports for each source IP in 15 minutes of our *UN2* trace, for *web* and *p2p*. In the *web* case, most points concentrate above the diagonal representing parallel connections of mainly simultaneous downloads of web objects (many destination ports correspond to one destination IP). On the contrary, most points in the *p2p* case are clustered either close to the diagonal (the number of destination ports is equal to the number of destination IPs) or below (which is common for UDP *p2p* communications, where the destination port number is constant for some networks).

Heuristic 3. Using the per-flow average packet size. A number of applications displays unique behavior regarding packet sizes. For instance, the majority of *gaming*, *malware* or SpamAssassin flows are characterized by a series of packets of constant size. Thus, constant packet size can discriminate certain applications. Note that it is not the actual size that is the distinctive feature, but instead the fact that packets have the same size across all flows; in other words, we simply need to examine whether the average packet size per flow (e.g. the fraction of total bytes over the number of packets) remains constant across flows.

Heuristic 4. Community heuristic. As discussed in the social behavior of network hosts, communities offer significant knowledge regarding interacting hosts. Thus, examining IP addresses within a domain may facilitate classification for certain applications. We apply the community heuristic to identify "farms" of services by examining whether "neighboring" IPs exhibit server behavior at the source port in question.

Heuristic 5. Recursive detection. Hosts offering specific types of services can be recursively identified by the interactions among them (variation of the community heuristic). For example *mail* or *dns* servers communicate with other such servers and use the same service port both as source or destination port across different flows. Also, SpamAssassin servers should only communicate with mail servers.

Heuristic 6. *Nonpayload* flows. Nonpayload or failed flows usually point to attacks or even *p2p* networks (clients often try to connect to IPs that have disconnected from the *p2p* network). The magnitude of failed flows can hint toward types of applications.

As previously mentioned, the strictness of classification depends on operator-defined thresholds. These thresholds, which implicitly originate from the structure of the *graphlet*s and the heuristics, include:
a) The minimum number of distinct destination IPs observed for a particular host (T_d) required for *graphlet* matching (e.g., at least T_d IPs are needed for a host to match the *p2p graphlet*).
b) The relative cardinality of the sets of destination IPs and ports (T_c) (e.g., for the *p2p graphlet*, it will define the maximum difference between the cardinalities of the two sets so that the *graphlet* is allowed to match - $T_c = 0$ indicates that the cardinalities must be equal).
c) The number of distinct packet sizes observed (T_s) (defines the maximum number of distinct average packet sizes per flow below which Heuristic 3 is considered).
d) The number of payload versus nonpayload flows (T_p) (defines the maximum ratio for Heuristic 6 to be considered).
Note that these thresholds may be specific to the *graphlet*.

5. CLASSIFICATION RESULTS

Here, we demonstrate the performance of our approach when applied to the traces described in section 3. Overall, we find that *BLINC* is very successful at classifying accurately the majority of the flows in all our traces.

We use two metrics to evaluate the success of the classification method. The **completeness** measures the percentage of the traffic classified by our approach. In more detail, completeness is defined as the ratio of the number of classified flows (bytes) by *BLINC* over the total number of flows (bytes) indicated by payload analysis. The **accuracy** measures the percentage of the classified traffic by *BLINC* that is correctly labeled. In other words, accuracy captures the probability that a classified flow belongs to the class (according to payload) that *BLINC* indicates. Note that both these metrics are defined for a given time interval, which could be either in the time scales of minutes or the whole trace, and can be applied to each application class separately or to the entire traffic.

The challenge for any method is to maximize both metrics, which however exhibit a trade-off relationship. The number of misclassifications will increase depending on how aggressive the classification criteria are. These criteria refer to the thresholds discussed in the previous section and can be tuned accordingly depending on the purpose of the measurement study. In this work, the thresholds have been tuned in the *UN1* trace and applied as such in the rest of the traces. We examine the sensitivity of our approach relative to the classification thresholds in section 5.2.

We use the payload classification as a reference point (section 3) to evaluate *BLINC*'s performance. Given that the payload classifier has no information to classify *nonpayload* flows, such flows need to be excluded from the comparison to level the field. Further, we have no way of characterizing "unknown" flows according to payload analysis. Consequently, the total amount of traffic used to evaluate *BLINC* for each trace, does not include *nonpayload* and *unknown* (according to payload) flows, which are discussed separately

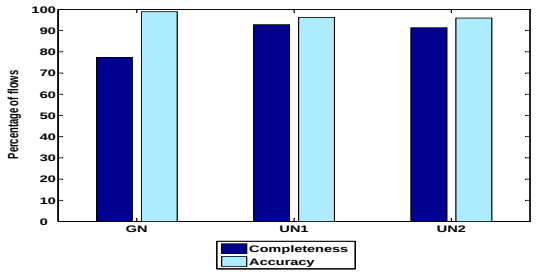

Figure 7: **Accuracy and completeness of all classified flows. For UN traces more than 90% of the flows are classified with approximately 95% accuracy. In GN trace, we classify approximately 80% of the flows with 99% accuracy.**

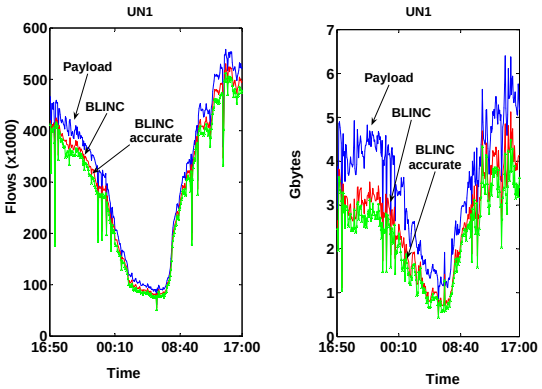

Figure 8: **Accuracy and completeness of *BLINC* in *UN1* trace in time (5-min intervals). The top line presents flows (bytes) classified using the payload, the middle line flows (bytes) classified by *BLINC*, and the bottom line presents flows (bytes) classified correctly by *BLINC*. The three lines coincide visually indicating high completeness and accuracy.**

at the end of this section. However, our approach is even able to characterize flows where payload analysis fails.

5.1 Overall completeness and accuracy

BLINC *classifies the majority of the traffic with high accuracy.* In Fig. 7, we plot the completeness and accuracy for the entire duration of each trace. In the *UN* traces, *BLINC* classifies more than 90% of the flows with approximately 95% accuracy. For the *GN* trace, *BLINC* classifies approximately 80% flows with 99% accuracy.

BLINC *closely follows traffic variation and patterns in time.* To stress test our approach, we examine the classification performance across smaller time intervals. In Fig. 8, we plot flows (left) and bytes (right) classified with *BLINC* versus the payload classifier, computed over 5-minute intervals for the *UN1* dataset. The top line presents all classified flows as identified by the payload classifier, the middle line represents flows classified by *BLINC*, and the bottom line flows classified correctly. The performance seems consistently robust over time. In terms of bytes, completeness ranges from 70%-85% for the *UN* traces and 95% for the *GN* trace with more than 90% accuracy. It is interesting to note that the difference between *BLINC* and payload in terms of bytes is due to a small number of large volume flows. In these flows, both source and destination hosts do not present sufficient number of flows in the whole trace and thus cannot be classified with *BLINC* without compromising the accuracy.

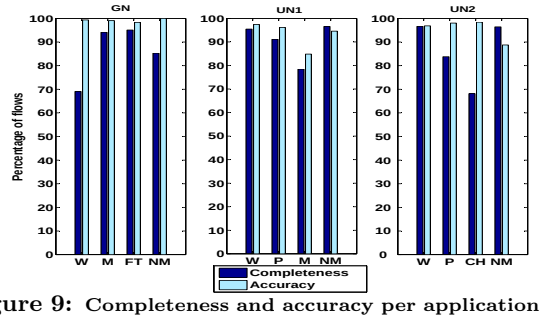

Figure 9: **Completeness and accuracy per application type. For each trace, we show the four most dominant applications, which contribute more than 90% of the flows. W:web, P:p2p, FT:ftp, M:mail, CH:chat, NM: network management.**

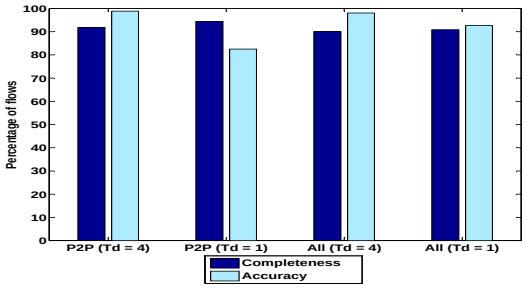

Figure 10: **Trade-off of accuracy versus completeness for *p2p* and the total number of flows. Decreasing the number of samples required to detect *p2p* behavior increases completeness but decreases accuracy.**

High per-application accuracy. Fig. 9 presents the accuracy and completeness for each of the four dominant applications of each trace, collectively representing more than 90% of all the flows. In all cases, accuracy is approximately 80% or more and completeness in most cases exceeds 80%. Note that per-class accuracy and completeness depends on the total amount of traffic in each class. For example, *web*-related metrics always exceed 90% in *UN* traces since *web* is approximately one third of all traffic. In *GN* where *web* is approximately 15% of the total bytes, completeness is approximately 70% (99% accuracy).

5.2 Fine-tuning BLINC

The trade-off between accuracy and completeness directly relates to the "strictness" of the classification criteria as discussed in section 4. Here, we study the effect of one of the thresholds we use in our approach. In classifying a host as a *p2p* candidate, we require that the host participates in flows with at least T_d distinct destination IPs. Setting T_d to a low value will increase completeness since *BLINC* will classify more hosts and their flows as *p2p*. However, the accuracy of the classification may decrease.

In Figure 10, we plot the accuracy and completeness for *p2p* flows (left columns) and the total number of classified flows (right columns) for two different values of T_d: $T_d = 1$ and $T_d = 4$. We observe that by reducing the threshold, the fraction of classified flows increases, whereas the fraction of correctly identified flows drops from 99% to 82%. Note that the total accuracy is also affected (as previously "unknown" flows are now (mis)classified) but the decrease for total accuracy is much smaller than in the *p2p* case dropping from approximately 98% to 93%. In all previous examples, we have used a value of $T_d = 4$ opting for accuracy.

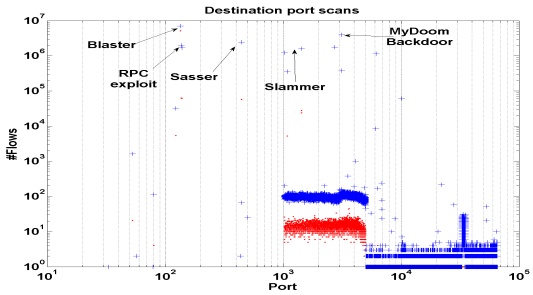

Figure 11: Histogram of destination ports for flows classified under address space scans for *GN* and *UN2* traces. *BLINC* successfully discriminates major address space scans at ports of "well-known" worms or exploits.

This flexibility is a key advantage of our approach. We claim that, for a network operator, it may be more beneficial if *BLINC* opts for accuracy. Misclassified flows are harder to detect within a class of thousands of flows, whereas unknown flows can potentially be examined separately by the operator by using additional external information such as *BLINC*'s social and functional role reports, or application specifications and consultations with other operators.

5.3 Characterizing "unknown" flows

In some cases, our approach goes beyond the capabilities of the payload classification. Although we unavoidably use payload analysis as benchmark, payload classification fails in two cases: a) it cannot characterize nonpayload flows (zero payload packets), and b) it cannot profile traffic originating from applications that are not analyzed a priori. In contrast, *BLINC* has the ability to uncover transport layer behavior that may allow for the classification of flows originating from previously unknown applications that fall under our *graphlet* modeled types (e.g., a new *p2p* protocol).

Nonpayload flows: The multilevel analysis of *BLINC* highlighted that the vast majority of nonpayload flows were due to IP address scans and port scans. Fig. 11 presents the histogram of destination ports in the flows that were classified as address space scans for two different traces using the attack *graphlet*s (fig. 5 (a),(b),(c)). Inspecting the peaks of this histogram shows that *BLINC* successfully identified destination ports of well-known worms or exploits, some of which are highlighted in the plot for visualization purposes. In total, *BLINC* classified approximately 26M flows as address space scans in our *UN2* trace. In addition, we classified approximately 100K flows as port scanning on 90 IP addresses in the same trace. Note that we did not need to use the port number of the exploits or any other external information. On the contrary, *BLINC* helped us identify the vulnerable ports by showing ports with unusually high traffic targeted at many different destinations IPs. However, our approach cannot in any way replace IDS systems such as *SNORT* [21] or *Bro* [2]. *BLINC* can only provide hints toward malicious behavior by detecting destination ports with high activity of failed flows.

Unknown applications: *BLINC* has the ability to identify previously "unknown" protocols and applications, since it captures the underlying behavior of application protocols. Indeed, during our analysis, *BLINC* identified a new *p2p* protocol (classified as unknown with payload analysis) running on the *PlanetLab* network (three *PlanetLab* nodes are behind our monitoring point). This *p2p* application corresponded to the *Pastry* project [16], which we identified after inspecting the payload, while we were examining our false positives. *BLINC* also identified a large number of gaming flows which were classified as unknown by payload analysis.

6. DISCUSSION

Implementing *BLINC* is not as straightforward as the presentation may have let us believe. We now present the implementation challenges and issues and discuss *BLINC*'s properties and limitations.

6.1 Implementation issues

We highlight here two major functions of the implementation: (a) the generation of *graphlet*s, and (b) the matching process of an unclassified host against the *graphlet*s.

A. Creating the graphlet*s.* In developing the *graphlet*s, we used all possible means available: public documents, empirical observations, trial and error. An automated way of defining new *graphlet*s is an interesting and challenging problem that is left for future work. We typically followed the steps below to create the majority of our *graphlet*s: (i) detection of the existence of a new application (which could be triggered from unusual amounts of unknown traffic), (ii) manual identification of the hosts involved in the unknown activity, (iii) derivation of the *graphlet* according to the interactions observed, and (iv) verification using human supervision and partially *BLINC*. *Graphlet* generation should be executed periodically in an off-line fashion.

B. The matching process among different graphlet*s.* To classify unknown behavior, we attempt to match *graphlet*s starting from the more specific to the more general ones. This matching policy resolves conflicts originating from similar *graphlet*s efficiently and speeds up the classification.

C. Extensibility: adding new graphlets. As mentioned in previous sections, *BLINC* is extensible by design. However, the addition of a *graphlet* requires careful consideration, since one needs to eliminate race conditions or overlaps between new and existing *graphlet*s. First, the *graphlet* matching order must be carefully examined before inserting a new *graphlet*. Second, if the new *graphlet* presents significant similarities with existing *graphlet*s, additional distinguishing features need to be derived.

Currently, our implementation of *BLINC* utilizes three special purpose data structures that capture the diverse application behavior across the *graphlet*s in the library. Due to space limitations, we describe the data structures along with the pseudocode that performs the actual mapping of flows into applications in the technical report [10].

Computational Performance. Our first version of *BLINC* appears sufficiently efficient to allow for a real-time implementation alongside currently available flow collectors. Despite the fact that the current *C++* implementation has hardly been optimized, *BLINC* classified our largest and longest (34-hour) *UN2* trace in less than 8 hours (flow tables were computed over 5 minute intervals); processing took place on a DELL PE2850 with a Xeon 3.4GHz processor and 2GB of memory, of which maximum memory usage did not surpass 40%.

6.2 Limitations

Classifying traffic "in the dark" has several limitations. Note that many of those limitations are not specific to our approach, but are inherent to the problem.

BLINC *cannot identify specific application sub-types:* Our technique is capable of identifying the type of an application but may not be able to identify distinct applications. For instance, we can identify *p2p* flows, but it is unlikely that we can identify the specific *p2p* protocol (e.g., *eMule* versus *Gnutella*) with packet header information alone. Naturally, this limitation could be easily addressed, if we had additional information, such as the specifications of the different protocols, or in the presence of distinctive behavior at the transport layer. We believe that for many types of studies and network management functions, this finer classification may not be needed. For example, the different instances of the same application type may impose the same requirements on the network infrastructure.

Encrypted transport layer headers: Our entire approach is based on relationships among the fields of the packet header. Consequently, our technique has the ability to characterize encrypted traffic as long as, the encryption is limited to the transport layer payload. Should layer-3 packet headers be also encrypted, our methodology cannot function. However, this is probably true for most classification methods.

Handling NATs: BLINC may require modification to classify traffic crossing Network Address Translators (NATs). Since classification in *BLINC* is mostly based on identifying the servers (profiling the flow patterns on the server port in graphlets), NATs should not affect the classification as long as the server is not behind the NAT. Even in this case, *BLINC* should be able to differentiate between multiple services behind NATs through the service port number. A particularly challenging scenario is when NATed hosts offer simultaneously more than one services with control and data streams (e.g., ftp and streaming service). Further analysis of *BLINC*'s performance when NATs are present would be an interesting extension of this work.

Point of observation: In this work, *BLINC* is evaluated in traces collected at the edge of the network. Intuitively, *BLINC* is designed to describe generic protocol behavior and should not be affected by the monitoring point. Nevertheless, applying *BLINC* at other points of the network, e.g., the backbone, may present different challenges, advantages and complications. For example, larger traffic sample per host will facilitate accurate classification of servers, popular hosts and large communities. On the other hand, individual user behavior might be harder to discern, while backbone data may reveal complex behavior not seen towards the edge. To evaluate *BLINC*'s performance under such conditions we would require publicly available payload data from multiple points on the Internet. Since no such data is currently available, we leave this task for future work.

7. CONCLUSIONS

In this paper, we propose *BLINC*, a traffic classification approach with significantly different philosophy compared to existing efforts. The novelty of *BLINC* lies in two key features: First, we classify hosts by capturing the fundamental patterns of their behavior at the transport layer. Second, our methodology defines and operates at three levels of host behavior: (i) the social level, (ii) the functional level, and (iii) the application level. Additionally, *BLINC* is tunable striking the desired point of balance in the trade-off between the *percentage of classified traffic* and its *accuracy*.

We applied *BLINC* on three real traces with very promising results. *BLINC* classified approximately 80%-90% of the total number of flows in each trace with 95% accuracy. In terms of individual application types, *BLINC* classified correctly more than 80% of the flows of each dominant application in our traces with an accuracy of at least 80%. Finally, *BLINC* identified malicious behavior or previously "unknown" applications without having a priori knowledge or port specific information.

Practical impact and the grand vision. We envision our approach as a flexible tool that can provide useful information for research or operational purposes in an evolving network with dynamic application behavior. By focusing on the fundamental communication behavior, our approach provides the first step towards obtaining understanding of traffic traces that transcends the technical specifications of the applications.

Acknowledgments

The authors are thankful to Dr. Andrew Moore for facilitating this study and Dr. Petros Faloutsos for his valuable suggestions and constructive criticism.

8. REFERENCES

[1] B. Aiello, C. Kalmanek, P. McDaniel, S. Sen, O. Spatscheck, and J. Van der Merwe. Analysis of Communities Of Interest in Data Networks. In *PAM*, 2005.
[2] Bro. http://bro-ids.org/.
[3] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully Automatic Cross-associations. In *KDD*, August 2004.
[4] K. Claffy, H.-W. Braun, and G. Polyzos. A Parametrizable methodology for Internet traffic flow profiling. In *JSAC*, 1995.
[5] C. Dewes, A. Wichmann, and A. Feldmann. An analysis of Internet chat systems. In *ACM/SIGCOMM IMC*, 2003.
[6] C. Estan, S. Savage, and G. Varghese. Automatically Inferring Patterns of Resource Consumption in Network Traffic. In *SIGCOMM*, 2003.
[7] F. Hernandez-Campos, A. B. Nobel, F. D. Smith, and K. Jeffay. Statistical Clustering of Internet Communication Patterns. *Computing Science and Statistics*, 35, July 2003.
[8] T. Karagiannis, A.Broido, M. Faloutsos, and kc claffy. Transport layer identification of P2P traffic. In *ACM/SIGCOMM IMC*, 2004.
[9] T. Karagiannis, A.Broido, N.Brownlee, kc claffy, and M.Faloutsos. Is P2P dying or just hiding? In *IEEE Globecom 2004, GI.*
[10] T. Karagiannis, D. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. Technical report, 2005. http://www.cs.ucr.edu/∼tkarag/papers/BLINC_TR.pdf.
[11] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and k. claffy. The architecture of the CoralReef: Internet Traffic monitoring software suite. In *PAM*, 2001.
[12] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow Clustering Using Machine Learning Techniques. In *PAM*, 2004.
[13] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt. Architecture of a Network Monitor. In *PAM*, 2003.
[14] A. Moore and K. Papagiannaki. Toward the Accurate Identification of Network Applications. In *PAM*, March 2005.
[15] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *ACM SIGMETRICS*, 2005.
[16] Pastry. http://research.microsoft.com/∼antr/Pastry/.
[17] Razor. http://razor.sourceforge.net/.
[18] 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 *ACM/SIGCOMM IMC*, November 2004.
[19] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In *WWW*, 2004.
[20] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. In *ACM/SIGCOMM IMW*, 2002.
[21] SNORT. http://www.snort.org/.
[22] tcpdump. http://www.tcpdump.org/.
[23] K. Xu, Z. Zhang, and S. Bhattacharya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *SIGCOMM*, 2005.