Transport Layer Identification of P2P Traffic

Thomas Karagiannis
UC Riverside

Andre Broido
CAIDA, SDSC

Michalis Faloutsos
UC Riverside

Kc claffy
CAIDA, SDSC

ABSTRACT

Since the emergence of peer-to-peer (P2P) networking in the late '90s, P2P applications have multiplied, evolved and established themselves as the leading 'growth app' of Internet traffic workload. In contrast to first-generation P2P networks which used well-defined port numbers, current P2P applications have the ability to disguise their existence through the use of arbitrary ports. As a result, reliable estimates of P2P traffic require examination of packet payload, a methodological landmine from legal, privacy, technical, logistic, and fiscal perspectives. Indeed, access to user payload is often rendered impossible by one of these factors, inhibiting trustworthy estimation of P2P traffic growth and dynamics. In this paper, we develop a systematic methodology to identify P2P flows at the transport layer, i.e., based on connection patterns of P2P networks, and without relying on packet payload. We believe our approach is the first method for characterizing P2P traffic using only knowledge of network dynamics rather than any user payload. To evaluate our methodology, we also develop a payload technique for P2P traffic identification, by reverse engineering and analyzing the nine most popular P2P protocols, and demonstrate its efficacy with the discovery of P2P protocols in our traces that were previously unknown to us. Finally, our results indicate that P2P traffic continues to grow unabatedly, contrary to reports in the popular media.

Categories and Subject Descriptors

C.2.5 [Computer-Communication Networks]: Local and Wide-Area Networks

General Terms

Algorithms, Measurement

Keywords

Peer-to-peer, Measurements, Traffic classification

1. INTRODUCTION

Over the last few years, peer-to-peer (P2P) file-sharing has relentlessly grown to represent a formidable component of Internet traffic. P2P volume is sufficiently dominant on some links to incent increased local peering among Internet Service Providers [24], to observable yet unquantified effect on the global Internet topology and routing system not to mention competitive market dynamics. Despite this dramatic growth, reliable profiling of P2P traffic remains elusive. We no longer enjoy the fleeting benefit of first-generation P2P traffic, which was relatively easily classified due to its use of well-defined port numbers. Current P2P networks tend to intentionally disguise their generated traffic to circumvent both filtering firewalls as well as legal issues most emphatically articulated by the Recording Industry Association of America (RIAA). Not only do most P2P networks now operate on top of nonstandard, custom-designed proprietary protocols, but also current P2P clients can easily operate on any port number, even HTTP's port 80.

These circumstances portend a frustrating conclusion: robust identification of P2P traffic is only possible by examining user payload. Yet packet payload capture and analysis poses a set of often insurmountable methodological landmines: legal, privacy, technical, logistic, and financial obstacles abound, and overcoming them leaves the task of reverse engineering a growing number of poorly documented P2P protocols. Further obfuscating workload characterization attempts is the increasing tendency of P2P protocols to support payload encryption. Indeed, the frequency with which P2P protocols are introduced and/or upgraded renders packet payload analysis not only impractical but also glaringly inefficient.

In this paper we develop a systematic methodology to identify P2P flows at the transport layer, i.e., based on flow connection patterns of P2P traffic, and without relying on packet payload. The significance of our algorithm lies in its ability to identify P2P protocols without depending on their underlying format, which offers a distinct advantage over payload analysis: we can identify previously unknown P2P protocols. In fact during our analysis we detected traffic of three distinct P2P protocols previously unknown to us. To validate our methodology we also developed a payload-based technique for P2P traffic identification, by reverse engineering and analyzing the nine most popular P2P protocols.

Specifically, the highlights of our paper include:

- We develop a systematic methodology for P2P traffic profiling by identifying flow patterns and character-

Table 1: Bulk sizes of OC-48 datasets

Set	Bb	Date	Day	Start	Dur	Dir	Src.IP	Dst.IP	Flows	Packets	Bytes	Aver.Util.	Ut.%
D09N	2	2003-05-07	Wed	10:00	2 h	Nbd (1)	904 K	2992 K	56.7 M	930.4 M	603 G	651 Mbps	26.2
D09S	2	2003-05-07	Wed	10:00	2 h	Sbd (0)	466 K	2527 K	47.3 M	624.2 M	340 G	376 Mbps	15.1
D10N	2	2004-01-22	Thu	14:00	60 m	Nbd (1)	812 K	2181 K	23.6 M	412.7 M	288 G	638.9 Mbps	25.7
D10S	2	2004-01-22	Thu	14:00	60 m	Sbd (0)	279 K	4177 K	18.6 M	252.7 M	117 G	260.4 Mbps	10.5
D11S	2	2004-02-25	Wed	10:00	2 h	Sbd (0)	410 K	7465 K	25.3 M	249.6 M	98.5 G	109.4 Mbps	4.4
D13N	2	2004-04-21	Wed	20:00	122 m	Nbd (1)	1971 K	6956 K	86.4 M	1263 M	852 G	930.6 Mbps	37.4
D13S	2	2004-04-21	Wed	20:00	122 m	Sbd (0)	306 K	10847 K	27.8 M	266.4 M	106 G	115.5 Mbps	4.6

istics of P2P behavior, without examination of user payload.

- Our methodology effectively identifies 99% of P2P flows and more than 95% of P2P bytes (compared to payload analysis), while limiting false positives to under 10%.
- Our methodology is capable of identifying P2P flows missed by payload analysis. Using our methodology we identify approximately 10% additional P2P flows over payload analysis.
- Using data collected at an OC48 (2.5Gbps) link of a Tier1 Internet Service Provider (ISP), we provide realistic estimates and trends of P2P traffic in the wide-area Internet over the last few years. We find that in contrast to claims of a sharp decline, P2P traffic has been constantly growing.

Our methodology can be expanded to support profiling of various types of traffic. Since mapping applications by port numbers is no longer substantially valid, a generalized version of our algorithm can support traffic characterization tasks beyond P2P workload. Indeed, to minimize false positives in P2P traffic identification, we assess, and then filter by, connection features of numerous protocols and applications (such as mail or DNS).

The rest of this paper is structured as follows: Section 2 describes our backbone traces, which span from May 2003 to April 2004. Section 3 discusses previous work in P2P traffic estimation and analysis. Sections 4 and 5 describe in detail our payload and nonpayload methodologies for P2P traffic identification. Section 6 presents an evaluation of our algorithm by comparing the volume of P2P identified by our methods. In section 7 we challenge media claims that the pervasive litigation undertaken by the RIAA is causing an overall decline in P2P file-sharing activity. Section 8 concludes our paper.

2. DATA DESCRIPTION

Part of the analyzed traces in this paper are included in CAIDA's Backbone Data Kit (BDK) [1], consisting of packet traces captured at an OC-48 link of a Tier 1 US ISP connecting POPs from San Jose, California to Seattle, Washington.

Table 1 lists general workload dimensions of our datasets: counts of distinct source and destination IP addresses and the numbers of flows, packets, and bytes observed. We processed traces with CAIDA's Coral Reef suite [19].

We analyze traces taken on May 5, 2003 (D09), January 22, 2004 (D10) February 25, 2004 (D11) and April 21,2004 (D13). We captured the traces with Dag 4 monitors [13] and packet capture software from the University of Waikato and Endace [11] that supports observation of one or both directions of the link.

For our older traces (D01-D10), our monitors captured 44 bytes of each packet, which includes IP and TCP/UDP headers and an initial 4 bytes of payload for some packets. However, approximately 60%-80% of the packets in these traces are encapsulated with an extra 4-byte MPLS label which leaves no space for payload bytes.

Fortunately we were able to capture the February and April 2004 traces (D11 and D13) with 16 bytes of TCP/UDP payload which allows us to evaluate our nonpayload methodology. To protect privacy, our monitoring system anonymized the IP addresses in these traces using the Cryptography-based Prefix-preserving Anonymization algorithm (Crypto-PAn) [33].

3. PREVIOUS WORK

Most P2P traffic research has thus far emphasized detailed characterization of a small subset of P2P protocols and/or networks [18] [14], often motivated by the dominance of that protocol in a particular provider's infrastructure or during a specific time period. Typical data sources range from academic network connections [27], [20] to Tier 2 ISPs [21].

Other P2P measurement studies have focused on topological characteristics of P2P networks based on flow level analysis [29], or investigating properties such as bottleneck bandwidths [27], the possibility of caching [21], or the availability and retrieval of content [3] [12].

Recently, Sen et al. developed a signature-based payload methodology [28] to identify P2P traffic. The authors focus on TCP signatures that characterize file downloads in five P2P protocols based on the examination of user payload. The methodology in [28] is similar to our payload analysis and it is further discussed in section 4.

A number of Sprint studies [8] report on P2P traffic as observed in a major Tier 1 provider backbone. However, their volume estimates taxonomize applications based on fixed port numbers from CoralReef's database [22], which captures a small and decreasing fraction of p2p traffic.

Our approach differs from previous work in three ways:

- *We analyze traffic sources of exceptionally high diversity, from major Tier 1 ISPs at the Internet core.*
- *We study all popular P2P applications available*: Neither of our methodologies (payload and nonpayload) are limited to a subset of P2P networks. On the contrary we study those P2P applications that currently contribute the vast majority of P2P traffic.
- *We combine and cross-validate identification methods that use fixed ports, payload, and transport layer dynamics.*

4. PAYLOAD ANALYSIS OF P2P TRAFFIC AND LIMITATIONS

Our payload analysis of P2P traffic is based on identifying characteristic bit strings in packet payload that potentially represent control traffic of P2P protocols. We monitor the nine most popular P2P protocols: *eDonkey* [9] (also includes the *Overnet* and *eMule* [10] networks), *Fasttrack* which is supported by the Kazaa client, *BitTorrent* [4], *OpenNap* and *WinMx* [32], *Gnutella*, *MP2P* [23], *Soulseek* [30], *Ares* [2] and *Direct Connect* [7].

Each of these P2P networks operate on top of nonstandard, usually custom-designed proprietary protocols. Hence, payload identification of P2P traffic requires separate analysis of the various P2P protocols to identify the specific packet format used in each case. This section describes limitations that inhibit accurate identification of P2P traffic at the link level. In addition, we present our methodology to identify P2P flows.

4.1 Limitations

We had to carefully consider several issues throughout our study. While some of these restrictions are data related, others originate from the nature of P2P protocols. Specifically, these limitations are the following:

Captured payload size: CAIDA monitors capture the first 16 bytes of user payload[1] of each packet (see section 2) for our February and April traces. While our payload heuristics would be capable of effectively identifying all P2P packets if the whole payload were available, this 16-byte payload restriction limits the number of heuristics that can reliably pinpoint P2P flows. Furthermore, our older traces (May 2003, January 2004) only contain 4 bytes of payload for a limited number of packets, since our monitors were used to capture 44 bytes for each packet (e.g., TCP options will push payload bytes out of the captured segment. Limitations for our older traces are described in detail in section 7).

HTTP requests: Several P2P protocols use HTTP requests and responses to transfer files, and it can be impossible to distinguish such P2P traffic from typical web traffic given only 16 bytes of payload, e.g., "HTTP/1.1 206 Partial Content" could represent either HTTP or P2P .

Encryption: An increasing number of P2P protocols rely on encryption and SSL to transmit packets and files. Payload string matching misses all P2P encrypted packets.

Other P2P protocols: The widespread use of file-sharing and P2P applications yields a broad variety of P2P protocols. Thus our analysis of the top nine P2P protocols cannot guarantee identification of all P2P flows, especially given the diversity of the OC48 backbone link. However, our experience with P2P applications and traffic analysis convinces us that these nine protocols represent the vast majority of current P2P traffic.

Unidirectional traces: Some of our traces reflect only one direction of the monitored link. In these cases we cannot identify flows that carry the TCP acknowledgment stream of a P2P download, since there is no payload. Even if we monitored both directions of the link, asymmetric routing renders it unlikely to find both streams (data and acknowledgment) of a TCP flow on the same link.

We can overcome these limitations with our nonpayload methodology described in section 5.

4.2 Methodology

Our analysis is based on identifying specific *bit strings* in the application-level user data. Since documentation for

Table 2: Strings at the beginning of the payload of P2P protocols. The character "0x" below implies Hex strings.

P2P Protocol	*String*	*Trans. prot.*	*Def. ports*
eDonkey2000	0xe319010000	TCP/UDP	4661-4665
	0xc53f010000		
Fasttrack	"Get /.hash"	TCP	1214
	0x270000002980	UDP	
BitTorrent	"0x13Bit"	TCP	6881-6889
Gnutella	"GNUT", "GIV"	TCP	6346-6347
	"GND"	UDP	
MP2P	GO!!, MD5, SIZ0x20	TCP	41170 UDP
Direct Connect	"$MyN","$Dir"	TCP	411-412
	"$SR"	UDP	
Ares	"GET hash:"	TCP	-
	"Get sha1:"		

P2P protocols is generally poor, we empirically derived a set of distinctive bit strings for each case by monitoring both TCP and UDP traffic using tcpdump[31] after installing various P2P clients. Table 2 lists a subset of these strings for some of the analyzed protocols for TCP and UDP. Table 2 also presents the well-known ports for these P2P protocols. The complete list of bit strings we used is in [17].

We classify packets into flows, defined by the 5-tuple source IP, destination IP, protocol, source port and destination port. We use the commonly accepted 64-second flow timeout [6], i.e., if no packet arrives in a specific flow for 64 seconds, the flow expires. To address the limitations described in the previous section, we apply three different methods to estimate P2P traffic, listed by increasing levels of aggressiveness as to which flows it classifies as P2P :

M1: If a source or destination port number of a flow matches one of the well-known port numbers (Table 2) the flow is flagged as P2P.

M2: We compare the payload (if any) of each packet in a flow against our table of strings. In case of a match between the 16-byte payload of a packet and one of our bit strings, we flag the flow as P2P with the corresponding protocol, e.g., Fasttrack, eDonkey, etc. If none of the packets match, we classify the flow as non-P2P.

M3: If a flow is flagged as P2P, both source and destination IP addresses of this flow are hashed into a table. All flows that contain an IP address in this table are flagged as "possible P2P" even if there is no payload match. To avoid recursive misclassification of non-P2P flows as P2P, we perform this type of IP tracking only for host IPs that *M2* identified as P2P .

In all P2P networks, P2P clients maintain a large number of connections open even if there are no active file transfers. There is thus increased probability that a host identified as P2P from *M2* will participate in other P2P flows. These flows will be flagged as "possible P2P" in *M3*. On the other hand, a P2P user may be browsing the web or sending email while connected to a P2P network. Thus, to minimize false positives we exclude from *M3* all flows whose source or destination port implies web, mail, FTP, SSL, DNS (i.e., ports 80, 8000, 8080, 25, 110, 21, 22, 443, 53) for TCP and online gaming and DNS (e.g., 27015-27050, 53) for UDP [2].

In general, we believe that *M3* will provide an estimate closer to the real intensity of P2P traffic, especially with lim-

[1]Privacy issues and agreement with the ISP prohibit the examination of more bytes of user payload.

[2]Since nothing prevents P2P clients from using these ports also, excluding specific protocols by looking at port numbers may result in underestimating P2P flows.

ited 4-byte payload traces, while *M2* provides a loose lower bound on P2P volume. *M3* takes advantage of our ability to identify IPs participating in P2P flows as determined by *M2*, facilitating identification of flows for which payload analysis fails. M3 *is used only in section 7, where we examine the evolution of the volume of P2P traffic. In that section, we use* M3 *to overcome the problem of the limited 4-byte payload in our older traces. For all other analysis, payload P2P estimates are strictly based on payload string matching, namely* M2.

Recently, Sen et al. developed a similar signature-based payload methodology [28]. The authors concentrate on TCP signatures that characterize file downloads in five P2P protocols and identify P2P traffic based on the examination of *all* user payload bytes. [28] describes a subset of the signatures included in our methodology, since we also use UDP-based as well as protocol signaling signatures for a larger number of P2P protocols/networks (e.g., the WinMx/OpenNap network is not analyzed in [28], although it corresponds to a significant portion of P2P traffic [16]). On the other hand, [28] presents the advantage of examining all user payload bytes. While examining all bytes of the payload should increase the amount of identified P2P traffic, we expect only a minimum difference in the number of identified P2P flows between [28] and the methodology described in this section. First, characteristic signatures or bit strings of P2P packets appear at the beginning of user payload; thus, 16 bytes of payload should be sufficient to capture the majority of P2P flows. Second, we expect that missed flows due to the payload limitation will be identified by our *M3* method and/or by TCP and UDP control traffic originating from the specific IPs.

5. NONPAYLOAD IDENTIFICATION OF P2P TRAFFIC

We now describe our nonpayload methodology for P2P traffic profiling (**PTP**). Our method only examines the packet header to detect P2P flows, and does not in any way examine user payload. To our knowledge, this is a first attempt to identity P2P flows on arbitrary ports without any inspection of user payload.

Our heuristics are based on observing connection patterns of source and destination IPs. While some of these patterns are not unique to P2P hosts, examining the flow history of IPs can help eliminate false positives and reveal distinctive features.

We employ two main heuristics that examine the behavior of two different types of pairs of flow keys. The first examines source-destination IP pairs that use both TCP and UDP to transfer data (TCP/UDP heuristic, section 5.1). The second is based on how P2P peers connect to each other by studying connection characteristics of {IP, port} pairs (section 5.2). A high level description of our algorithm is as follows:

- *Data processing*: We build the flow table as we observe packets cross the link, based on 5-tuples, similar to the payload method. At the same time we collect information on various characteristics of {IP, port} pairs, including the sets of distinct IPs and ports that an {IP, port} pair is connected to, packet sizes used and transferred flow sizes.

Table 3: Excluded ports for TCP/UDP IP pairs heuristic.

Ports	Application
135,137,139,445	NETBIOS
53	DNS
123	NTP
500	ISAKMP
554,7070,1755,6970,5000,5001	streaming
7000, 7514, 6667	IRC
6112, 6868, 6899	gaming
3531	p2pnetworking.exe

- *Identification of potential P2P pairs*: We flag potential flows as P2P based on TCP/UDP usage and the {IP, port} connection characteristics.

- *False positives*: We eliminate false positives by comparing flagged P2P flows against our set of heuristics that identify mail servers, DNS flows, malware, etc.

5.1 TCP/UDP IP pairs

Our first heuristic identifies source-destination IP pairs that use both TCP and UDP transport protocols. Six out of nine analyzed P2P protocols use both TCP and UDP as layer-4 transport protocols. These protocols include eDonkey, Fasttrack, WinMx, Gnutella, MP2P and Direct Connect. Generally, control traffic, queries and query-replies use UDP, and actual data transfers use TCP. To identify P2P hosts we can thus look for pairs of source-destination hosts that use both transport protocols (TCP and UDP).

While concurrent usage of both TCP and UDP is definitely typical for the aforementioned P2P protocols, it is also used for other application layer protocols such as DNS or streaming media. To determine non-P2P applications in our traces that use both transport protocols, we examined all source-destination host pairs for which both TCP and UDP flows exist. We found that besides P2P protocols, only a few applications use both TCP and UDP transport protocols: DNS, NETBIOS, IRC, gaming and streaming, which collectively typically use a small set of port numbers such as 135, 137, 139, 445, 53, 3531, etc. Table 3 lists all such applications found, together with their well-known ports. Port 445 is related to the Microsoft NETBIOS service. Port 3531 is used by an application called *p2pnetworking.exe* which is automatically installed by Kazaa. Although *p2pnetworking.exe* is related to P2P traffic, we choose to exclude it from our analysis since it is not under user control[3] and specific only to the Kazaa client. Excluding flows using ports presented in Table 3, 98.5% of the remaining IP source-destination pairs that use both TCP and UDP in our traces are P2P, based on the payload analysis with *M2* described in Section 4. In summary, *if a source-destination IP pair concurrently uses both TCP and UDP as transport protocols, we consider flows between this pair P2P so long as the source or destination ports are not in the set in Table 3.*

5.2 {IP, port} pairs

Our second heuristic is based on monitoring connection patterns of {IP, port} pairs.

Since the lawsuit against Napster, the prevalence of centralized P2P networks has decreased dramatically, and distributed or hybrid P2P networks have emerged. To connect to these distributed networks, each P2P client maintains a

[3]The user cannot change the port number or control its functionality, and all flows of p2p.networking.exe use port 3531.

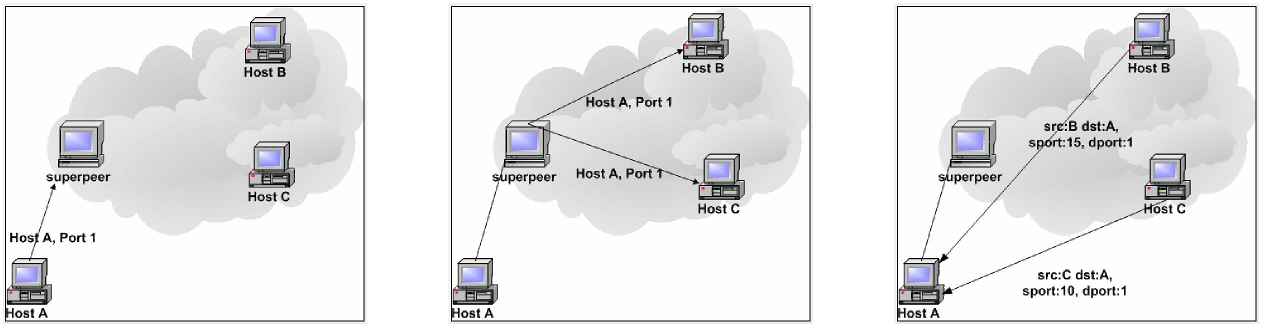


Figure 1: Initial connection from a new P2P host *A* to the P2P network. Host *A* connects to a superpeer picked from its host cache. Peer *A* informs the superpeer of its IP address and the port willing to accept connections from other peers. The superpeer propagates the {IP, port} pair to the rest of the P2P network. Peers willing to connect to host *A*, use the advertised {IP, port} pair. For the {IP, port} pair {A,1}, the number of distinct IPs (C,B) connected to it is equal to the number of distinct ports (10,15) used to connect to it. Our {IP, port} pair heuristic is based on such equality between the number of distinct ports and the number of distinct IPs affiliated with a pair in order to identify potential P2P pairs.

starting host cache. Depending on the network, the host cache may contain the IP addresses of other peers, servers or supernodes/superpeers.[4] This pool of hosts facilitates the initial connection of the new peer to the existing P2P network.

As soon as a connection exists to one of the IPs in the host cache (we will henceforth refer to these IPs as superpeers), the new host A informs that superpeer of its IP address and port number at which it will accept connections from peers. Host A also provides other information specific to each P2P protocol but not relevant here. While in first-generation P2P networks the listening port was well-defined and specific to each network, simplifying P2P traffic classification, newer versions of all P2P clients allow the user to configure a random port number (some clients even advise users to change the port number to disguise their traffic). The superpeer must propagate this information, mainly the {*IP, port*} pair of the new host A, to the rest of the network. This {IP, port} pair is essentially the new host's ID, which other peers need to use to connect to it. In summary, *when a P2P host initiates either a TCP or a UDP connection to peer* A*, the destination port will also be the advertised listening port of host* A*, and the source port will be an ephemeral random port chosen by the client.*

Normally, peers maintain at most one TCP connection to each other peer, but there may also be a UDP flow to the same peer, as described previously. Keeping in mind that multiple connections between peers is rare in our data sets, we consider what happens when twenty peers all connect to peer A. Each peer will select a temporary source port and connect to the advertised listening port of peer A. The advertised {IP, port} pair of host A would thus be affiliated with 20 distinct IPs and 20 distinct ports [5]. In other words, *for the advertised destination {IP, port} pair of host* A*, the number of distinct IPs connected to it will be equal to the number of distinct ports used to connect to it.* Figure 1 illustrates the procedure whereby a new host connects to the P2P network and advertises its {IP, port} pair.

[4]Superpeers/supernodes are P2P hosts that handle advanced functionality in the P2P network, such as routing and query propagation.

[5]The probability that two distinct hosts pick the same random source port at the same time is extremely low.

On the other hand, consider what happens in the case of web and HTTP. As in the P2P case, each host connects to a pre-specified {IP, port} pair, e.g., the IP address of a web server W and port 80. However, a host connecting to the web server will initiate usually more than one concurrent connection in order to download objects in parallel. In summary, *web traffic will have a higher ratio than P2P traffic of the number of distinct ports versus number of distinct IPs connected to the {IP, port} pair {W,80}.*

5.3 Methodology

Our nonpayload methodology builds on insights from previous sections 5.1 and 5.2. Specifically, for a time interval t we build the flow table for the link, based on the five-tuple key and 64-second flow timeout as with the payload methodology described in section 4. We then examine our two primary heuristics:

- We look for source-destination IP pairs that concurrently use both TCP and UDP during t. If such IP pairs exist and they do not use any ports from table 3, we consider them P2P.
- We examine all source {srcIP, srcport} and destination {dstIP, dstport} pairs during t (use of pairs will henceforth imply both source and destination {IP, port} pairs). We seek pairs for which the number of distinct connected IPs is equal to the number of distinct connected ports. All pairs for which this equality holds are considered P2P . In contrast, if the difference between connected IPs and ports for a certain pair is large (e.g., larger than 10), we regard this pair as non P2P.

These two simple heuristics efficiently classify most pairs as P2P or nonP2P. In particular the {IP, port} heuristic can effectively identify P2P and nonP2P pairs given a sufficiently large sample of connections for the specific pair. For example, with time interval t of 5 minutes there are no false positives for pairs with more than 20 connections in our February 2004 trace (D11 of Table 1.) That is, for this specific trace, if an IP pair has more than 20 IPs connect to it, we can classify it with high confidence as P2P or not P2P.

Whether a flow is considered P2P depends on the classification of its {IP, port} pairs. If one of the pairs in the 5-tuple flow key has been classified as P2P, this flow is deemed P2P. Similarly, if one of the pairs is classified as non P2P, so is the flow. Additionally, if one of the IPs in a flow has been found to match the TCP/UDP heuristic, the flow is also considered as P2P.

5.4 False positives

We now describe heuristics developed to decrease the risk of false positives. Considering the diversity of backbone links that feature a vast number of IPs and flows, we expect the previous methodology to yield false positives, i.e., classifying nonP2P pairs as P2P. False positives are most common in pairs with few connections, and also more frequent for specific applications/protocols whose connection behavior matches the P2P profile of our heuristics (e.g., one connection per {IP,port} pair), e.g., e-mail (SMTP, POP), DNS and gaming.

To decrease the rate of false positives we review the connection and flow history of all pairs where the probability of a misclassification is high, e.g., the source or destination port is equal to 25 and implies SMTP. Past flow history for these pairs enables accurate classification by investigating properties of specific IPs. In the following subsections, we describe heuristics that augment our basic methodology to limit the magnitude of false positives.

5.4.1 Mail

In our data sets, e-mail protocols such as Simple Mail Transfer Protocol (SMTP) or Post Office Protocol (POP) contribute most false positives. Mail false positives are not surprising since connection behavior resembles our {IP, port} heuristic. However, analysis of mail flows and connection patterns allows for identification of mail servers in our traces, forestalling misidentification of traffic to such IP addresses as P2P.

We examine all flows where one of the port numbers is equal to 25 (SMTP), 110 (POP) or 113 (authentication service commonly used by mail servers). In fact we treat these three port numbers as one (we consider ports 110 and 113 equal to 25), since for our purpose their behavior is the same. We identify mail servers based on their port usage history and whether they have different flows during the same time interval t that use port 25 for both source and destination port. The following observed flow pattern illustrates this characteristic behavior of mail servers by examining the usage of port 25 by IP *238.30.35.43*:

src IP	dst IP	proto	srcport	dstport
238.30.35.43	115.78.57.213	6	25	3267
238.30.35.43	238.45.242.104	6	22092	25
238.30.35.43	0.32.132.109	6	25	50827
238.30.35.43	71.199.74.68	6	22175	25
238.30.35.43	4.87.3.29	6	21961	25
238.30.35.43	4.87.3.29	6	22016	25
238.30.35.43	4.170.125.67	6	25	3301
238.30.35.43	5.173.60.126	6	22066	25
238.30.35.43	5.173.60.126	6	22067	25
238.30.35.43	227.186.155.214	6	22265	25
238.30.35.43	227.186.155.214	6	22266	25
238.30.35.43	5.170.237.207	6	25	3872

This case shows flows for IP *238.30.35.43* [6] with port 25 as source port for some flows and destination port for other flows. This behavior is characteristic of mail servers that initiate connections to other mail servers to propagate e-mail messages. To identify this pattern, we monitor the set of destination port numbers for each IP for which there exists a source pair {IP,25}. If this set of destination port numbers also contains port 25, we consider this IP a mail server and classify all its flows as nonP2P. Similarly for the set of source ports of an IP for which there exists a destination pair {IP,25}. In the above example, for the source pair {238.30.35.43,25}, the set of destination ports is [3267, 25, 50827, 3301, 3872]. Since port 25 appears in this set, we infer that IP 238.30.35.43 is a mail server and deem all of its flows nonP2P. We keep all IPs identified as mail servers in a *mailserver list* to avoid future application of our heuristics to them.

[6]Note that IP addresses are anonymized.

5.4.2 DNS

The Domain Name Server protocol runs on top of both TCP and UDP port 53 and is characterized mainly by numerous short flows, i.e, few packets/bytes and short duration. DNS connection patterns are analogous to our {IP, port} pair heuristic, although DNS pairs are easier to identify since most DNS source and destination ports are 53. For example, the following is a representative pattern of UDP DNS flows,

src IP	dst IP	proto	srcport	dstport
252.60.148.12	0.121.94.5	17	53	53
115.254.223.8	243.11.142.6	17	53	53

In this case the observed {IP, port} pairs are considered nonP2P , e.g., {252.60.148.12,53}, {0.121.94.5,53}, due to the use of port 53 as source and destination port in the flow 5-tuple. As with the mail server IPs, we maintain a list of rejected pairs to exclude from further analysis other possible flows for these pairs. For example, source pair {252.60.148.12,53} may have additional DNS flows to other IPs but with destination port other than 53. But since we identified the specific pair as a DNS false positive, we also rule out the possibility that any of these additional flows are P2P. The heuristic ensures that DNS flows will be effectively identified and removed for our P2P estimate even if a specific host is part of a P2P network. Thus, only true P2P flows of a host will be considered and not its DNS requests.

We do not restrict the use of this heuristic to DNS. On the contrary, we apply it to all flows and pairs where one of the ports is less than 501. This heuristic facilitates the removal of other false positives in commonly used ports (e.g., port numbers such as 25), especially those caused by a service that runs on port 500. In these flows both ports are equal to 500, similar to the pattern described here. Thus, *for all flows where the source port is equal to the destination port, and the port number is less than 501, both source and destination {IP, port} pairs are considered nonP2P, and they are inserted in a list of definitively nonP2P pairs.* This heuristic was inspired by DNS flow features and thus is called "DNS heuristic", although it is not necessarily specific to DNS.

5.4.3 Gaming and malware

On-line gaming runs mainly on top of UDP. Characteristic examples of on-line games with sufficient traffic in our traces are *Age of Empires*, *Half-life* and *Quake*. On the other hand, malware tends to run over TCP. By malware, we mean worm traffic (e.g. MyDoom on ports 3127,3128, or Beagle on port 2745) and port or address space scans, which appear often in backbone traces.

Gaming and malware bear a similar property: many flows to different IPs/ports, carrying the same number of packets/bytes and/or with same-sized packets. Consider the following UDP flow pattern from the game *Half-life*:

```
  src IP         dst IP      pr sprt dprt pkts bytes
3.195.130.255 145.46.189.100 17 1990 27015  4   160
13.15.101.255 145.46.189.100 17 2989 27015  5   200
115.254.14.42 145.46.189.100 17 3965 27015  1    40
```

For all flows of {IP, port} pair {145.46.189.100,27015}, all packets likely have the same size, or it is at least consistent with dividing the number of bytes by the number of packets in each flow (i.e., the mean packet size). In all flows, the average packet size is 40. On the contrary, if we consider the {IP, port} heuristic, we would accept all the pairs as P2P pairs, since the number of distinct IPs equals the number of distinct ports connecting to them, e.g., for pair {145.46.189.100,27015} there are 3 distinct IPs and 3 distinct ports, while for pair {3.195.130.255,1990} there is 1 distinct IP and 1 distinct port.

To remove such pairs, we maintain for each {IP, port} pair a set of distinct average packet sizes and a set of distinct total transfer sizes. We also have two different sets of port numbers:

```
KnownP2PPortsSet: [4661, 4662, 4665, 1214, 6346, 6347,
412, 411, 41170, 6881-6889, 6699, 6257, 2234]

MalwarePortsSet: [3127, 3128, 1433, 1434, 3531, 1080,
10080, 17300, 6129, 27015,  27016, 901, 2745]
```

The first contains the known P2P port numbers; the second, malware and gaming ports. We classify an {IP, port} pair as nonP2P if:

```
{ length(pair.transfer_sizesSet) == 1  or
length(pair.avg_pktssizesSet) < 3 }
```

AND

```
port not in KnownP2PPortsSet
```

AND

```
{ length(pair.IPSet) > 5  or  port < 501
or port in MalwarePortsSet }
```

where *transfer_sizesSet* is the set of distinct transfer sizes of all flows for this pair, *avg_pktssizesSet* is the set of distinct average packet sizes of all flows of this pair and *IPSet* is the set of distinct IPs of this pair. Note that these sets cannot contain duplicate entries since we only insert unique values into them once.

In summary, *we classify an {IP, port} pair as nonP2P if the following conditions all hold: the port is not in the known P2P ports set; the pair only has one transfer size or less than three distinct average packet sizes (average packet size is the total transfer size in the flow divided by the total number of packets); the pair has flows to more than five IPs or the port is in the malware ports set.* As with the DNS heuristic, we insert all {IP, port} pairs that agree with the above rule in a list of non P2P pairs.

5.4.4 Other heuristics

We apply a number of other rules that offer finer grained analysis of {IP, port} pairs.

Scans: In addition to the heuristic for malware and gaming, we count the number of {IP, port} pairs in which a specific IP appears, to rule out port scans as false positives. Specifically, we reject all IPs that appear in a large number of {IP, port} pairs and at the same time target a few IPs.

One-packet pairs: We remove all one packet flows whose IPs do not appear in any other flows in the trace. We have no way to consider these P2P traffic.

MSN messenger servers: We found and removed all flows to MSN messenger servers. We could identify these flows easily since they used port 1863 and three distinct destination IPs within the same prefix.

Port history: To further remove web, DNS and mail false positives, we examined the set of distinct ports used to connect to an {IP, port} pair. If all ports in the set reflect well-known services, e.g., mail, web and DNS, we rule out the pair as P2P if it appears in at least ten flows. While P2P applications may use port numbers that canonically map to well-known services, it is highly unlikely that P2P clients will connect only to such port numbers, since current versions of P2P clients randomize the port at which they accept connections. For such a case to exist, a large fraction of P2P users would have to change their client's listening port to 80 or 25.

5.5 Final Algorithm

Combining the techniques of all previous subsections yields our final nonpayload methodology for identification of P2P flows. Note that our algorithm is designed for analysis of passive traffic traces, allowing for multiple passes over the data if necessary. In addition, we have not optimized for memory consumption and performance. Adapting our algorithm for active real-time monitoring of P2P traffic at network speed is part of our ongoing work.

Algorithm 1 (*PTP*) presents in detail the procedure executed every time interval. At the beginning of section 5 we described three distinct phases, but during execution these phases overlap with one another. Across time intervals we maintain a set of different lists, since knowledge learned in one interval is likely to help in future intervals. The maintained lists are based on the flow table for the specific interval and correspond to our P2P identification heuristics as well as to our false positive handling methodologies. Specifically, the lists we maintain across time intervals include: the *P2PIP* list, which contains IPs already classified as P2P by the TCP/UDP IP pair heuristic (section 5.1); the *P2PPairs* list, which contains {IP,Port} pairs already classified as P2P by the {IP,Port} pair heuristic; the *Rejected* and *MailServers* lists, which contain rejected pairs and IPs (false positive heuristics); and the *IPPort* list, which includes the {IP,Port} pairs of all flows that are not in MailServers or Rejected lists. Each {IP,Port} pair that is an item of the *IPPort* list is coupled with sets that include: a) the distinct IPs (*IPSet*) appearing in flows with the specific pair, b) the distinct ports used in flows for the specific pair (*PortSet*), c) distinct average packet sizes in flows for this pair(*avg_pktssizesSet*) and d) distinct transferred flow sizes for this pair (*transfer_sizesSet*).

At the end of each interval, we analyze all {IP,Port} pairs in the *IPPort* list against our false-positive heuristics. This analysis is based on the aforementioned lists and the description of the heuristics in all previous subsections. If all false-positive heuristics fail, the specific {IP,Port} pair and all flows matching this pair are deemed P2P. All flows of IPs in the *P2PIP* list are also considered P2P .

Despite the existence of various constants in the *PTP* algorithm such as the different port lists (e.g., *KnownP2PPortsSet*), we expect human intervention to be minimum and port lists to be stable over periods of months. Specifically, these constants depend on the link in question (e.g., which games contribute large numbers of flows) and should be updated

Algorithm 1 Nonpayload algorithm for P2P flow identification

```
 1: procedure PTP                                   ▷ P2P Traffic Profiliing
 2:     FT ← Flow Table
 3:     for every src-dst IP pair in FT do
 4:         if TCP/UDP pair then
 5:             P2PIP.insert(srcIP)                 ▷ TCP/UDP heuristic
 6:             P2PIP.insert(dstIP)
 7:     for all flows in FT do
 8:         if src IP or dst IP in P2PIP then
 9:             print flow                          ▷ found by TCP/UDP pairs
10:             P2PIP.insert(srcIP)                 ▷ put both IPs in P2P list
11:             P2PIP.insert(dstIP)
12:         else if DNS heuristic is true then
13:             RejectedPairs.insert(src Pair)      ▷ pair=={IP,port}
14:             RejectedPairs.insert(dst Pair)
15:         else if src and dst IP not in MailServers then
16:             for src and dst IP-port pair do
17:                 if pair in P2PPairs then
18:                     print flow ▷ found in previous interval
19:                     P2PPairs.insert(src pair)   ▷ put both pairs in P2PPairs list
20:                     P2PIP.insert(src pair)
21:                 else if pair not in Rejected then
22:                     Udpate sets for pair
23:                     IPPort.insert(pair)
24:                 else if pair in Rejected then
25:                     Rejected.insert(src pair)
26:                     Rejected.insert(dst pair)
27:     for pairs in IPPort do
            ▷ examine pairs that were added during
            ▷ previous intervals and have not been yet classified
28:         if IP not in MailServers and pair not in Rejected then
29:             if IP in P2PIP or pair in P2PPairs then
30:                 P2PPairs.insert(pair)
31:                 print all flows of pair
32:             else
33:                 diff ← |pair.IPSet.len−pair.PortSet.len|
34:                 if diff < 2 or (diff < 10 and port in KnownP2PPorts) then
35:                     if Check_if_Mailserver == true then
36:                         MailServer.insert(IP)
37:                     else if Check_if_Malware == true then
38:                         Rejected.insert(pair)
39:                     else if Check_if_scan == true then
40:                         Rejected.insert(pair)
41:                     else if Port_History heuristic=true then
42:                         Rejected.insert(pair)
43:                     else
44:                         P2PPairs.insert(pair)
45:                         print all flows of pair
46:                 else if diff> 10 then Rejected.insert(pair)
```

when significant changes occur in the specific traffic mix (e.g., sufficient amount of packets or flows by a new worm or a new P2P network).

Finally, note that for the {IP, port} pairs heuristic we indirectly separate flows into two different classes. The first class is flows where one of the ports is within the KnownP2PPortsSet. In this case we deem the absolute difference of the sizes of *IPSet* and *PortSet* to be acceptable if less than 10, since the probability that the specific pair is actually P2P is higher. (Note that the sizes of *IPSet* and *PortSet* indicate the number of distinct ports and distinct IPs affiliated with the specific {IP,Port} pair, see {IP,port} heuristic, section 5.2.) While new P2P clients randomize port numbers, there still exist P2P clients that use known P2P ports (most users do not immediately upgrade to newer versions that randomize the port). On the contrary, for all other flows, we allow a maximum difference of 1 between the sizes of *IPSet* and *PortSet*. While our {IP,port} heuristic, assumes equal sizes of PortSet and IPSet, we allow limited inequality to account for possible failed connections which is common in P2P behavior [7].

6. EVALUATION

In this section we evaluate the accuracy of our methodology, by comparing nonpayload versus payload estimates of P2P traffic. We will use our recent 16-byte payload traces (February and April 2004) and strict payload matching (method *M2* in section 4.) We avoid using *M3* for comparison purposes in this section; first, 16-byte payload traces offer sufficient number of payload bytes to identify the vast majority of P2P flows. Second, *M3* could introduce ambiguity in evaluating the *PTP* Algorithm, since we would compare against flows that we cannot determine with absolute certainty if they are P2P or not (*M3* introduces false positives and targets 4-byte payload traces to mitigate the disadvantage of only 4-bytes of user payload). Thus, we only use *M2* for payload analysis in this section. We first compare the number of P2P flows and bytes as identified by the two methodologies. In addition, we study the extent and nature of false positives. Finally, we show how our methodology overcomes disadvantages of payload analysis and present the volumes of additional P2P flows that we were able to identify.

6.1 Fraction of identified P2P traffic

We now demonstrate how *PTP* Algorithm performs compared to payload analysis, in particular what fraction of P2P traffic found by payload analysis (*M2*, Sec.4) can be found by *PTP* Algorithm.

Fig. 2 and 3 summarize our findings. Fig. 2 presents the bitrate of P2P traffic determined by payload analysis (upper line) and the fraction identified by *PTP* Algorithm (bottom line). For all P2P flows that were previously discovered by payload inspection, we examine whether *PTP* Algorithm also classified them as P2P. As shown in Fig. 2, the two lines fall almost on top of each other in all three traces,

[7]The values of 1 and 10 in the difference between the sizes of the IPSet and PortSet have a minimal effect in our algorithm. Allowing larger differences will only impact slightly the number of false positives. The two classes of flows reflect the fact that pairs with port in the knownP2PPortSet are more likely to be P2P.

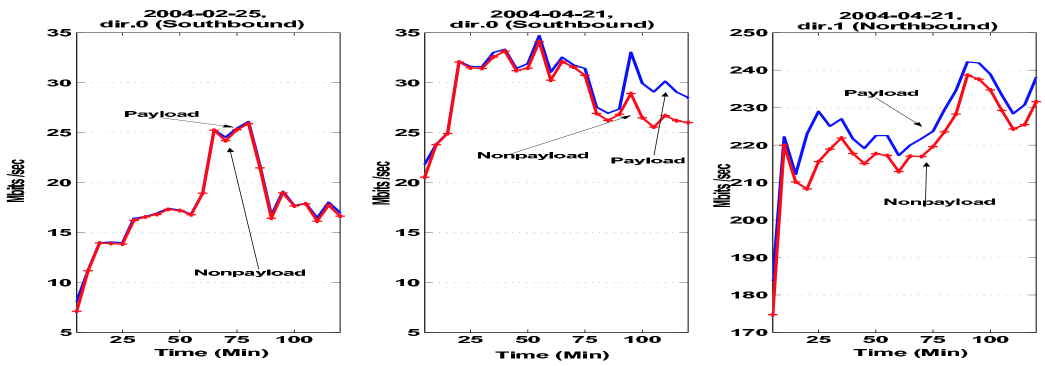


Figure 2: The bitrate of P2P traffic as determined by our payload methodology (upper line) and the portion that was identified by our nonpayload algorithm (bottom line). In all three traces our nonpayload methodology successfully identifies more than 90% of P2P bytes. Even with increasing P2P bitrate (approx. 220 Mbps, right plot), the algorithm identifies more than 95% of P2P bytes.

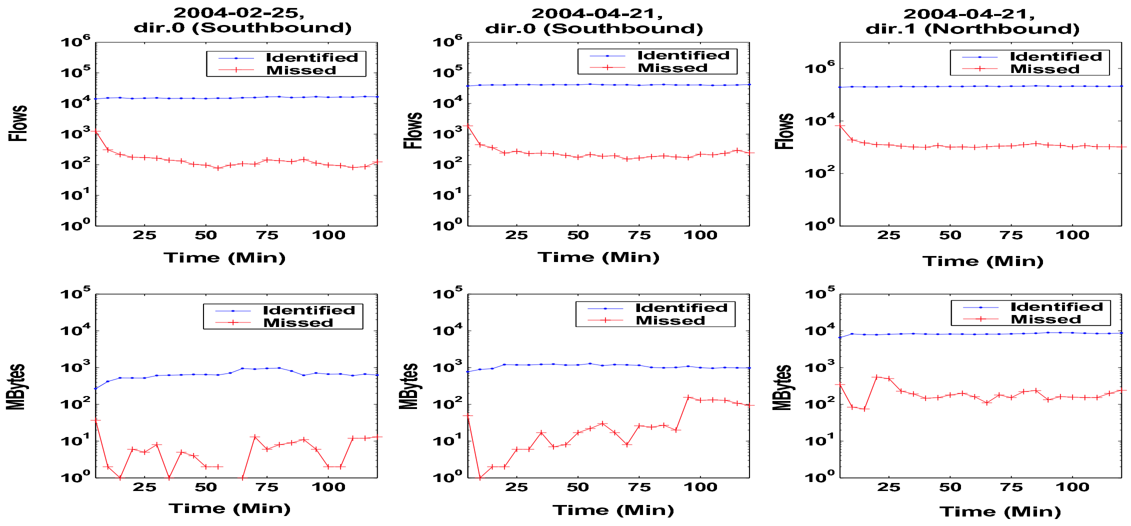


Figure 3: Identified (upper line) vs. missed (bottom line) P2P flows and bytes of our nonpayload methodology as compared to payload examination (logarithmic scale.) Flows and bytes are shown in total volumes every five minutes. Only 0.5% of P2P flows are not identified. Note that despite large difference in utilization across our traces, the fraction of missed flows remains almost constant.

indicating that our approach is able to accurately identify the vast majority of P2P traffic.

Fig. 3 better depicts the success of the nonpayload methodology. Specifically, we examine what portions of flows and bytes found by payload inspection are also identified by *PTP* Algorithm as P2P . The bottom line plots the total number of P2P flows and bytes missed by the nonpayload algorithm in five-minute intervals. The upper line plots the total volumes of identified flows and bytes. The Y axis is plotted on logarithmic scale to facilitate comparison. The top row of plots presents the number of flows while the bottom row the volumes in bytes. Finally, each column of plots in Fig. 3 reflects a different trace.

Our nonpayload based methodology discovers more than 90% of total P2P bytes and 99% of P2P flows. These percentages appear to be independent of the total traffic on the link. Despite large variation in traffic volumes across our traces (approximately one order of magnitude difference between northbound and southbound direction in our monitored link), *PTP* Algorithm performs sufficiently in all cases. Note that the number of missed P2P flows is declining with time, illustrating increasing knowledge of {IP,port} pairs and their connection behavior for *PTP* Algorithm. On the other hand, the fraction of unidentified P2P bytes depends on the fluctuation of the volume of specific flows, which manifests itself in the time-varying lines of missed bytes. That is, the shape of the plot of missed bytes vs. time is affected by the sizes of missed P2P flows. While the number of missed flows using the nonpayload methodology may be decreasing, a large missed P2P flow may cause the volume of missed P2P bytes to noticeably increase.

6.2 False positives

As described earlier in the paper (section 5), several of our heuristics aim at minimizing false positives, i.e., flows misclassified as P2P. Minimizing false positives is a challenging task considering the dynamic nature, diversity of sources and sheer volumes of traffic in the Internet core. We strived to develop heuristics to account for and characterize the behavior of major classes (e.g. web, mail) of traffic in order to discriminate them from P2P behavior. Thus, false positives may originate from types of traffic that are

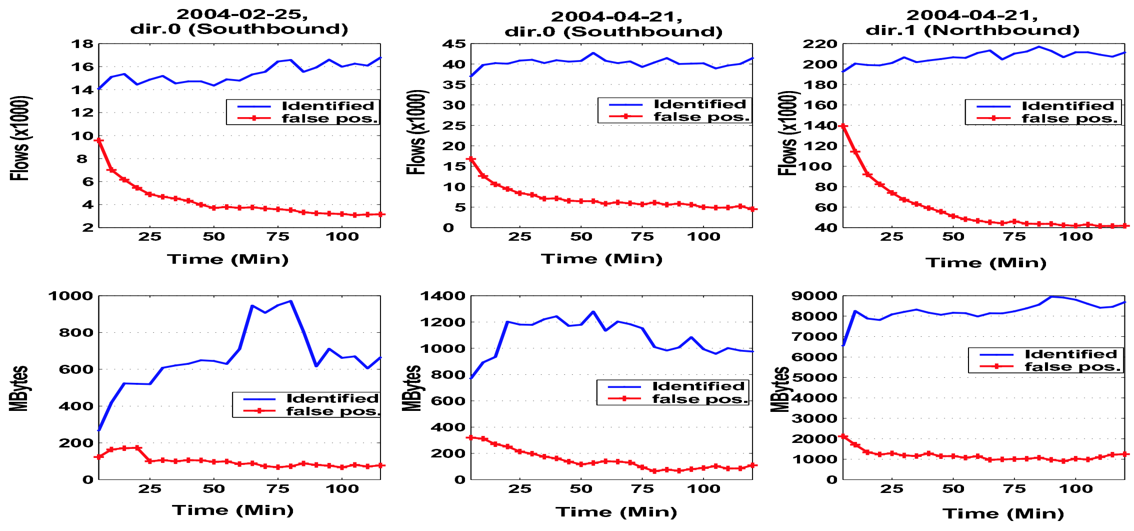


Figure 4: False positives vs correctly identified P2P flows and bytes. Flows and bytes are shown in total volumes every five minutes. False positives account for 8%-12% of the total estimate (false positives plus correctly recognized P2P traffic).

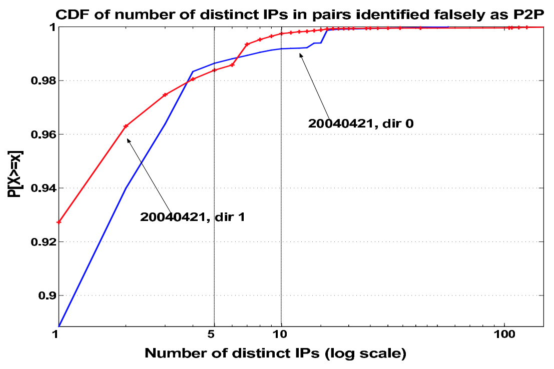


Figure 5: Cumulative distribution function of number of distinct IPs in {IP-port} pairs that lead to false identification of flows as P2P. 98% of misclassifications were based on pairs with fewer than five distinct IPs in the IPSet. These false positives are due to an insufficient sample for the specific pairs.

not profiled by our heuristics. However, attempting to eliminate false positives by profiling all types of traffic will only increase the computational burden without sufficiently improving the outcome. It is also unrealistic, especially in the Internet core where we need to pinpoint tens of thousands of P2P flows among the millions of flows crossing a backbone link (Tab.1).

Fig. 4 indicates the number of false positives produced by *PTP* Algorithm. False positives represent flows classified as P2P by *PTP* Algorithm but not identified as such by payload analysis. On the other hand, *PTP* Algorithm detects true P2P flows that were missed by payload analysis due to the limitations described in section 4. These flows are not included in the false positives in Fig. 4; we will describe them in the next section.

Fig. 4 presents the volume of flows and bytes correctly classified as P2P, compared to the corresponding amounts of false positives. The figure is structured similarly to Fig. 3; the top and bottom rows show flows and bytes respectively, while columns refer to our three different traces, and volumes plotted in five-minute intervals.

False positives correspond to approximately 8% to 12% of the total estimate of P2P traffic. The percentage of misclassified flows depends on the trace and the time within the trace, but drops in all cases below 15% after the first few time intervals. Similar to missed flows in the previous section, false positives decrease and stabilize with time as knowledge about the characteristics of specific IPs or pairs increases.

The majority of false positives originate from the limited number of samples for the specific {IP,port} pair. That is, the sizes of both IPSet and PortSet in *PTP* Algorithm are sufficiently small to allow specious inference of connection patterns. Fig. 5 illustrates the cumulative distribution function of the IPSet size of all false positives in our April trace. In both directions of link (southbound and northbound), the size of IPSet for 98% of all false positives is less than five (an IPSet size of 5 implies that the specific {IP-port} pair had connections with only five other distinct IPs). In fact, approximately 90% of false positives represent {IP,port} pairs that communicate with one IP only.

6.3 Robustness of *PTP* Algorithm

We examine the robustness of our algorithm with respect to the effect of the time interval t between successive executions of *PTP* Algorithm.

The analysis of identified and missed P2P flows and false positives in the previous subsections is based on 5-minute time intervals ($t = 5$ min in *PTP* Algorithm). We now examine how t affects the percentages of missed flows and false positives. Note that variable t designates the period during which the flow table is formed and the time interval between successive executions of the algorithm.

Intuitively, short time intervals should be more sensitive to transient phenomena and to IPs appearing in the link for the first time. Larger t allows for more efficient profiling of IPs and pairs. However, as t increases, memory requirements, e.g., the flow table size, increase.

Fig. 6 illustrates how missed and false positive flows vary for three different time intervals (t) during our April southbound trace. To facilitate comparison we present the average number of missed and false positive flows per minute.

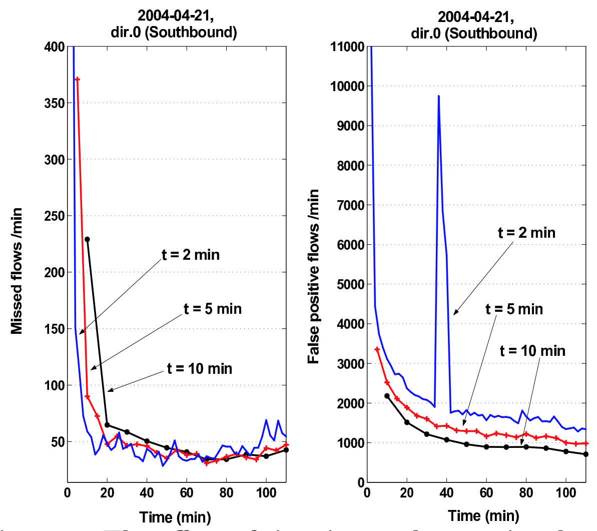


Figure 6: The effect of time interval t on missed and false positive flows. As t increases the numbers of missed flows and false positive flows decreases.

As expected, the 2-minute line appears noisier between successive intervals (observe the large transient spike in the number of false positives), in contrast to the smoother lines for 5- and 10-minute intervals. While for missed flows the three lines fall on top of each other after the first 30 minutes in the trace, the number of false positives drops continually as t increases. However, the difference in the volume of false positives for five and ten minute intervals is trivial relative to the number of flows in the link, especially when comparing byte volumes. Since memory requirements for 5-minute intervals are only a fraction of those for 10-minute intervals, we settled on 5-minute intervals for our analysis.

The large spike in the 2-minute line of false positives is caused by an address space scan in our trace. At this specific time interval a distinct source IP scanned the address space at destination port 4899 creating approximately 7,000 flows per minute. Our false positive heuristics effectively recognized that this IP is not P2P after two intervals when $t = 2$ min, indicated by the false positive line dropping sharply after the sudden increase. For larger time intervals our heuristics effectively recognize all non P2P scanning flows, and thus no spike appears in the 5 or 10 minute lines in Fig. 6.

6.4 Not so false “false positives"

An advantage of nonpayload identification of P2P traffic is the possibility to overcome limitations of payload analysis (see also section 4). Two inescapable limitations of payload analysis are the following:

Payload methodologies cannot identify the invisible. If no payload exists, P2P flows cannot be identified. Such flows might be actual nonpayload flows (e.g., TCP acknowledgment streams of file transfers), or flows with encrypted payload.

Payload methodologies can only verify and not discover. Inherently, payload methodologies require a priori knowledge of the anatomy of P2P protocols, and as such they can only be applied to previously reverse-engineered, known protocols.

In contrast, our methodology is not affected by these constraints. Instead we are able to discover numerous flows that were missed by payload analysis. To identify such flows we separately examined flows that were identified as P2P by *PTP* Algorithm but were missed by payload inspection, and had at least one port number from our known P2P ports list or one of the IPs consistently using P2P source or destination ports. History and connection patterns of IPs participating in such flows reveal their P2P nature.

Additionally and most important, *PTP* Algorithm can effectively discover unknown P2P protocols. We encountered this powerful capability in the process of minimizing the number of false positives. Comparing nonpayload with payload classification, we observed numerous false positives in five specific port numbers, namely 22321, 7674, 7675, 5335 and 9493. Inspection of payload for traffic under these port numbers revealed that all of the aforementioned ports represent traffic of three distinct P2P protocols/networks unknown to us. Two of the networks originate in Asia and use both TCP and UDP: *Soribada* (ports 7674,7675, 22321) and *GoBoogy* (port 5335). To date we have not been able to identify the P2P protocol responsible for the traffic under port 9493. However, a large number of packets contain the string “*GET /?p2pmethod=*” in the 16-byte packet payload available to us. Since access to the full packet payload is not possible, we have no way of knowing what fraction of the rest of the false positives mask yet more P2P protocols that remain unidentified.

In total we were able to discover approximately 18,000 additional P2P flows (350 additional Mbytes) over those discovered with the payload methodology every five minutes on the average for the April northbound trace, 3,000 additional flows (15 additional MBytes) for the April southbound trace and 1,900 additional flows (20 additional Mbytes) for the February southbound trace.

6.5 Payload vs. nonpayload identification of P2P traffic

The previous section suggests the flexibility of nonpayload methodologies. Here we provide an overall assessment of advantages and disadvantages of payload versus nonpayload methodologies based on our experience. Overall, nonpayload methodologies provide diverse benefits over payload analysis, specifically with regard to:

Privacy issues: Nonpayload methodologies offer an ideal solution to the many perceived and real privacy and legal alarms triggered by even the idea of inspecting of user payload. Indeed, RIAA litigation has inspired among end users as well as ISPs increased concern over privacy, which will make providers even more reluctant to allow payload analysis. One could possibly obviate privacy issues in the payload analysis by reporting only aggregate information of P2P traffic at the monitoring site.

Anonymization of IP addresses: Nonpayload methodologies do not require anonymization of IP addresses, which if performed inhibits further analysis of topological characteristics of traffic (e.g., IPs cannot be aggregated to Autonomous Systems). If payload examination is permitted, ISPs require anonymization of IP addresses so that individual users cannot be linked to packet payload.

Storage overhead: The storage needed to support passive analysis of payload traces significantly grows with increasing bytes of captured payload. Our approach requires only up to layer-4 header information. Alternatively, packet or flow sampling could reduce storage overhead.

Processing overhead: Both in passive and active monitoring, payload processing at network speed of an OC-48 link is far beyond trivial due to the system memory bus bottleneck. The bus is used by network monitoring cards to transfer the whole packet header plus the examined payload bytes to memory. Increasing the volume of captured payload risks packet loss at high utilizations of a monitored link.

Reverse-engineering of protocols vs. P2P behavior analysis: As noted previously in the paper (section 6.4), payload methodologies have the ability only to verify and pinpoint the existence of protocols that have been dissected in advance. On the contrary, monitoring nonpayload P2P behavior bypasses the requirement of previous knowledge and facilitates detection of unfamiliar P2P networks.

Encryption: Payload methodologies fail for encrypted payload which is bound to eventually become common, especially for newer versions of P2P protocols.

On the other hand, our methodology, at least in its current form, is inferior to payload analysis regarding detailed analysis of specific P2P protocols. Since we model the general behavior of distributed (or semi-distributed) P2P networks, our algorithm currently cannot monitor individual protocols.

7. P2P /FILE-SHARING TRAFFIC TRENDS

Recently, popular media sources have reported a sharp decline in peer-to-peer (P2P) traffic during the last year [5] [26], with P2P user populations reportedly dropping as much as 50%. This assertion, if true, would indicate a reversal in the trend of the constant increase of P2P activity over the last years (five out of the top six downloads from sourceforge.net were P2P clients on July 27 2004).

In this section, we discuss these alleged P2P claims and contrast them to our own results. Notwithstanding the inherently challenging nature of P2P traffic classification, as we have definitively illustrated in this study, media reports are rarely based on measuring, much less classifying, any traffic on the Internet. Indeed, these reports base their conclusions on telephone surveys or periodic samples of log files for a limited number of P2P networks/clients (specifically for Kazaa, WinMx and a small number of Gnutella clients, such as Morpheus, Grokster and Bearshare) that might have been waning in popularity relative to newer, more advanced P2P networks (e.g., eDonkey or BitTorrent).

However, using both payload and nonpayload methodologies, our OC-48 traces indicate that, if measured accurately, P2P traffic has never declined; indeed we have never seen the proportion of overall P2P traffic decrease over time (any change is an increase) in any of our data sources. While bitrate trends do not necessarily reflect trends in user population counts, we believe that these statistics show that P2P networks are largely unaffected by RIAA litigious practices.

In addition to the limitations of payload analysis methodologies described in section 4, we list here further complications that may affect comparison of P2P traffic volumes. Specifically:

44-byte packets: In our older traces (D09 of May 2003, D10 of January 2004), CAIDA monitors captured 44 bytes of each packet (see section 2), leaving 4 bytes of TCP packets for examination (TCP headers are typically 40 bytes for packets that have no options). To facilitate fair payload comparison, we only use 4-byte payload heuristics throughout this section for all traces. On the other hand, a UDP header is only 8 bytes, which leaves enough payload bytes for effective string matching on those packets.

MPLS: 60%-80% of the packets in our traces are encapsulated with 4-byte MPLS (Multiprotocol Label Switching) headers. MPLS is used by this Tier1 ISP for routing and traffic engineering purposes. MPLS decreases the number of packets that can be matched against our string table since for a significant amount of traffic there is no payload (4-byte MPLS header + 40-byte TCP header).

ISP caching: To alleviate the effect of P2P traffic, ISPs sometimes employ caching of P2P content [15]. P2P caching, similar to web caching, is capable of reducing upstream traffic yielding large savings for ISPs.[8] Naturally, P2P requests that are served by these caches do not reach the backbone, resulting in a limited view of P2P usage especially when comparing with past years before P2P caching became common.

P2P versus copyrighted traffic: Typically, the majority of P2P traffic is related to copyrighted material. Although we cannot necessarily equate P2P with copyrighted traffic, the dominance of copyrighted material in most P2P networks is largely accepted to be true. Our study cannot identify the trends in the use of P2P networks for exchanging copyrighted material.

Link utilization and time of the day: Two traces can present drastically different characteristics, even if taken on the same link at different times of day. While most of our traces are collected during business hours, we compare traces with varying utilizations and captured at different time within the day. We thus compare P2P traffic relative to the total volume of traffic in the link rather than focusing on absolute values.

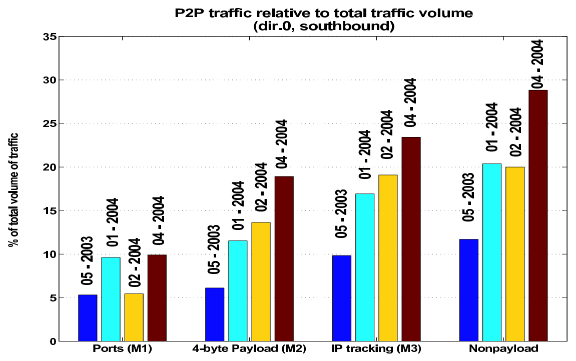
Conflicting traffic engineering goals: Because of its large volume, ISPs are tempted to manipulate P2P traffic according to their economic objectives. Networks that pay for transit have an incentive to keep traffic within their boundaries or those of their non-charging peers [25]. Networks that charge for transit can try to attract traffic by adjusting routing and/or performance metrics (some P2P clients prefer peers with lower RTTs; some, like BitTorrent, choose ones with highest bitrate). An increase in peering among cable companies was recently attributed to the rise in P2P traffic [24]. Pricing of international vs. domestic traffic can also play a role. Competitive peering behavior can cause unpredictable link workload changes even when other conditions are equal.

Finally, we note that many limitations of this analysis (e.g., varying utilization across traces), as with virtually all Internet measurement studies, are neither new nor unique to Internet science.

7.1 P2P traffic is growing

We compare traces D09 from May 2003 (southbound and northbound), D10 of January 2004 (southbound and northbound), D11 of February 2004 (southbound) and D13 of April 2004 (southbound and northbound). (Recall that Ta-

[8]ISPs are usually charged based on the traffic they send upstream to their own providers.



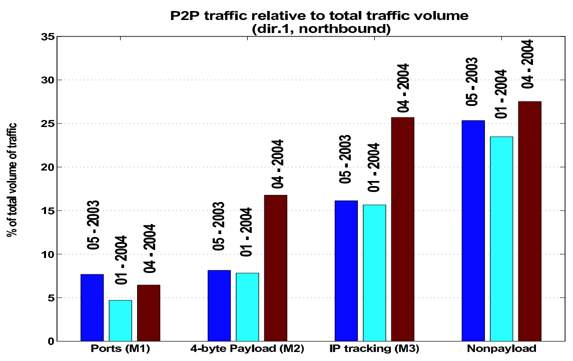


Figure 7: Estimation of the volume of P2P traffic using four different payload and nonpayload methodologies. In all cases we observe a growing trend of P2P traffic, validated by all methodologies. Note also that estimates of P2P traffic volume increase by 20%-100% when comparing port numbers with payload. Especially in our northbound April 2004 trace, the payload estimate is more than double the estimate produced by port numbers.

ble 1 lists bulk volumes and utilizations.) We use both payload and nonpayload analysis. We present payload findings analytically for each step of the methodology described in section 4 (*M1-M3*).

Fig. 7 demonstrates the average bitrate of P2P traffic as detected by each method for both directions of all traces. To facilitate comparison, we present P2P volume as a percentage relative to total traffic volume across each trace. Despite the aforementioned limitations, we make the following general observations from Fig. 7:

Significant decline of P2P traffic is not corroborated: On the contrary, P2P traffic in our recent traces is, if not growing, at least comparable to older traces from 2003 and January 2004. This trend is supported by *all* methodologies examined for the southbound direction. For the northbound direction, May 2003 and January 2004 volumes are comparable for all methods beyond *M1* (P2P rate in "known" ports); P2P traffic in our April trace (even with significantly higher total traffic volume on the link) surpasses all other traces.

Failure of conventional estimation methodologies: As depicted in Fig. 7, using port numbers for traffic classification is misleading. P2P traffic measurements based on port numbers results in underestimating P2P traffic by more than 50%, especially in recent traces. Fig. 7 also illustrates the migration to random port numbers when comparing 2003 with 2004 traces. While for May 2003 the difference in the estimates of *M1* and *M2* is minimal, it explodes in our 2004 traces. The change is the effect of newer P2P clients automatically randomizing the port number.

Sufficient payload size: Comparing payload estimations of P2P traffic with 4-byte and 16-byte payload for the February and April 2004 traces, demonstrates that the payload limitation is also significant for robust identification. While P2P traffic with 16-bytes of payload is estimated at approximately 17% and 25% for February and April respectively (shown in Fig. 2), estimates using 4-bytes of payload are considerably lower. Increasing the captured payload size beyond 16 bytes will potentially result in a further increase in the estimates of P2P traffic. As noted in section 4, even with 16 bytes of payload there are still conflicting bit strings between web and P2P protocols. However, many factors, among them a hardware bottleneck of our monitoring system, limit the size of payload we can capture.

M3 *vs. nonpayload methodology*: Since the previous analysis suggests that our nonpayload algorithm overestimates (compared to the payload estimates, section 6.2) approximately 10%, *M3* and nonpayload estimates in most cases are comparable. Note that nonpayload estimates also include traffic from the three protocols discovered during our nonpayload analysis, which supports our conjecture that *M3* in our payload methodology more accurately estimates P2P traffic.

In general, P2P traffic has grown to constitute a considerable portion of traffic in our monitored backbone link, confirming our assumption that estimations of P2P traffic intensity based on a limited number of P2P networks or characteristic port numbers is unrealistic. Our findings also illustrate the expanding software alternatives for P2P users; three previously unknown protocols in our traces constitute a characteristic example of this increasing diversity of protocols and networks.

8. CONCLUSIONS

This paper focused on the non payload identification and monitoring of a significant and growing component of Internet traffic, namely P2P applications. Traditionally, P2P traffic has been classified by well-known port numbers unique to each protocol. However, growing concerns due to legal and other complications have pushed P2P networks to challenge network "standards" by randomizing their port numbers and in general making some effort to disguise their activity. As a result conventional measurement analyses are bound to underestimate P2P traffic, and indeed, reliable identification of P2P traffic requires examination of user payload data.

We presented a method that relies on network and transport layer behavior to identify P2P traffic. Specifically our algorithm is based on profiling flow patterns of IP addresses. In addition, to validate our methodology, we developed a payload scheme to identify P2P flows by reverse-engineering and analyzing the nine most popular P2P protocols/networks.

A key feature of our algorithm is its ability to identify "unknown" P2P protocols. Since the methodology is based on the general behavior of P2P networks, prior knowledge or analysis of protocols is not required. Indeed, our algorithm detected three distinct P2P protocols previously unknown to us.

We show that our methodology is able to effectively pinpoint among million of flows more than 95% of P2P flows and bytes in traces from an operational OC48 backbone link. The number of false positives ranges approximately from 8% to 12% of the total payload-based estimate. Furthermore, we demonstrated that our algorithm has the ability to identify P2P flows missed by payload analysis.

Using estimates from both methodologies, we also challenged claims of a sharp decline in P2P activity. All of our estimates of real Internet traffic, even based on simplistic port number analysis, confirm our hypothesis that P2P traffic is growing in volume and will continue to grow unabatedly in the future.

We consider a number of future extensions to strengthen our algorithm. First, we wish to exploit the availability of bidirectional traces by merging IP pairs that appear in both directions of the link. We also want to consider additional heuristics that use knowledge of specific packet sizes that may reflect control traffic of P2P protocols. Additionally, the IP ID field may facilitate the identification of many existing connections by observing gaps in sequence numbers.

We are in the process of extending and generalizing our methodology for use in more general traffic profiling. During this study we have illustrated connection characteristics and patterns of various popular applications. Since simple port-based application breakdown has become problematic for most workload characterization tasks, extending our methodology to general traffic profiling may offer a higher integrity alternative.

Accurate monitoring of P2P traffic has become an important aspect of Internet traffic modeling. P2P traffic has already risen to a significant percentage of the total traffic, 15%-20% in our monitored links. On the other hand, its idiosyncrasies (e.g., bandwidth symmetry) portend a dramatic change in our approach to network provisioning and traffic engineering. In our previous work [16] we predicted that the P2P paradigm threatens the asymmetrical bandwidth assumption inherent in many broadband infrastructures, e.g., DSL and cable modems, and may even result in further increases in local peering among ISPs [24]. These changes in business choices among providers affect the global Internet topology and routing system, not to mention competitive market dynamics, in ways that we have only begun to consider. But with efficient, accurate methods of workload characterization in the P2P realm, we can at least head in to the future with better vision.

Acknowledgments

We are thankful to the Coral Reef suite team that made this study possible. Thanks are due to our colleagues Ken Keys, Colleen Shannon, Nevil Brownlee, Daniel Andersen and Khushboo Shah for their support throughout this work. We would also like to thank Carey Williamson and our IMC anonymous referees for their helpful comments.

9. REFERENCES

[1] A.Broido, Y.Hyun, R.Gao, and kc claffy. Their share: diversity and disparity in IP traffic. In *PAM*, 2004.
[2] Ares. http://www.softgap.com/.
[3] R. Bhagwan, S. Savage, and G. Voelker. Understanding Availability. In *IPTPS 03*, 2003.
[4] BitTorrent. http://bitconjurer.org/BitTorrent/.
[5] John Borland. RIAA threat may be slowing file swapping. "http://news.com.com/2100-1027-1025684.html".
[6] K. Claffy, H.-W. Braun, and G. Polyzos. A Parametrizable methodology for Internet traffic flow profiling. In *IEEE JSAC*, 1995.
[7] Direct Connect. http://www.neo-modus.com/.
[8] C.Fraleigh e.a. Packet-Level Traffic Measurements from the Sprint IP Backbone. In *IEEE Network*, 2003.
[9] eDonkey2000. http://www.edonkey2000.com/.
[10] eMule. http://www.emule-project.net/.
[11] Endace, 2004. www.endace.com.
[12] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-Peer Networks. In *INFOCOM*, 2004.
[13] I.Graham, M.Pearson, J.Martens, and S.Donnelly. Dag - a cell capture board for ATM measurement systems, 1997. wand.cs.waikato.ac.nz.
[14] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garc'es-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM*, 2004.
[15] Joltid. http://www.joltid.com.
[16] T. Karagiannis, A.Broido, N.Brownlee, kc claffy, and M.Faloutsos. Is P2P dying or just hiding? In *IEEE Globecom 2004 - Global Internet and Next Generation Networks*, 2004.
[17] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos. File-sharing in the Internet: A characterization of P2P traffic in the backbone. Technical report., 2004. http://www.cs.ucr.edu/∼tkarag.
[18] P. Karbhari, M. Ammar, A. Dhamdhere, H.Raj, G. Riley, and E. Zegura. Bootstrapping in Gnutella: A Measurement Study. In *PAM*, 2004.
[19] 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.
[20] K.Tutschku. A Measurement-based Traffic Profile of the eDonkey Filesharing Service. In *PAM*, 2004.
[21] N. Leibowitz, A. Bergman, Roy Ben-Shaul, and Aviv Shavit. Are File Swapping Networks Cacheable? Characterizing P2P Traffic. In *7th IWCW*, 2002.
[22] D. Moore, K. Keys, R. Koga, E. Lagache, and kc claffy. Coralreef software suite as a tool for system and network administrators. In *Usenix LISA*, 2001.
[23] MP2P. http://www.slyck.com/mp2p.php.
[24] W. B. Norton. The evolution of the u.s. internet peering ecosystem, 2003. http://www.equinix.com/pdf/whitepapers/PeeringEcosystem.pdf.
[25] M.L. Garcia Osma, F.J. Ramon Salguero, G. Garcia de Blas, J. Andres Colas, J. Enriques Gabeiras, S. Perez Sanches, and R. Trueba Fernandez. Enabling local preference in peer-to-peer traffic. COST 279 TD(04)017 technical document, 2004.
[26] Pew Internet & American Life Project. Sharp decline in music file swappers: Data memo from PIP and comScore Media Metrix, January, 2004. http://www.pewinternet.org/reports/.
[27] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *MMCN*, 2002.
[28] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In *WWW*, 2004.
[29] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. In *IMW*, 2002.
[30] Soulseek. http://www.slsknet.org/.
[31] tcpdump. http://www.tcpdump.org/.
[32] WinMx. http://www.winmx.com/.
[33] J. Xu, J. Fan, and M. H. Ammar. Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. In *IEEE ICNP*, 2002.