# 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*IMC'04*, October 25–27, 2004, Taormina, Sicily, Italy.
Copyright 2004 ACM 1-58113-821-0/04/0010 ...$5.00.

## 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 [25], 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-



Cloudflare - Exhibit 1047, page 121

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

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 [20].

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 [14] and packet capture software from the University of Waikato and Endace [12] 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 (CryptoPAn) [33].

## 3. PREVIOUS WORK

Most P2P traffic research has thus far emphasized detailed characterization of a small subset of P2P protocols and/or networks [19] [15], 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], [21] to Tier 2 ISPs [22].

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 [22], or the availability and retrieval of content [3] [13].

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 [23], 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* [10]



Cloudflare - Exhibit 1047, page 122

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

(also includes the *Overnet* and *eMule* [11] networks), *Fasttrack* which is supported by the Kazaa client, *BitTorrent* [4], *OpenNap* and *WinMx* [32], *Gnutella*, *MP2P* [24], *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 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 [18].

**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:" | | |

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.



Cloudflare - Exhibit 1047, page 123

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

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 [17]). 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.



Cloudflare - Exhibit 1047, page 124

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

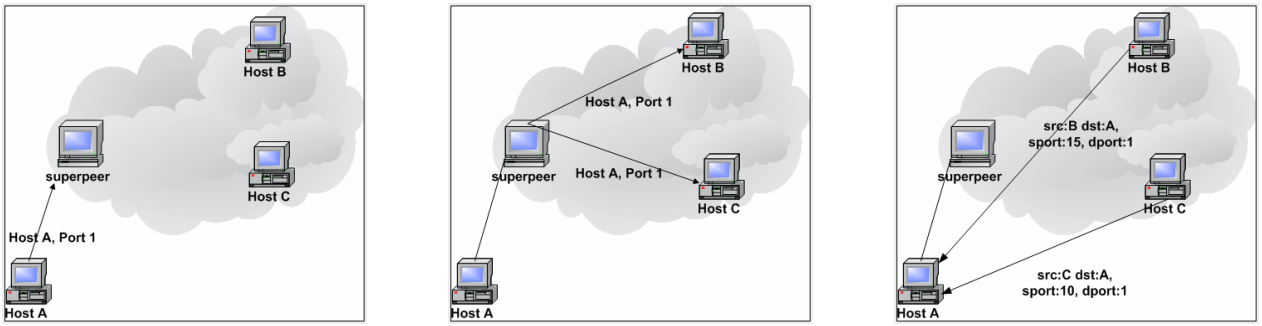


**Figure 1:** **Initial connection from a new P2P host $A$ to the P2P network. Host $A$ connects to a superpeer picked from its host cache. Peer $A$ informs the superpeer of its IP address and the port willing to accept connections from other peers. The superpeer propagates the {IP, port} pair to the 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.



Cloudflare - Exhibit 1047, page 125

DOCKET ALARM

Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.