File-sharing in the Internet: A characterization of P2P traffic in the backbone.

Thomas Karagiannis
UC, Riverside

Andre Broido, Nevil Brownlee, k claffy
CAIDA, SDSC,UCSD

Michalis Faloutsos
UC Riverside

***Abstract*— Since the outbreak of peer-to-peer (P2P) networking with Napster during the late '90s, P2P applications have multiplied, become sophisticated and emerged as a significant fraction of Internet traffic. At first, P2P traffic was easily recognizable since P2P protocols used specific application TCP or UDP port numbers. However, current P2P applications have the ability to use arbitrary ports to "camouflage" their existence. Thus only a portion of P2P traffic is clearly identifiable. As a result, estimates and statistics regarding P2P traffic are unreliable. In this paper we present a characterization of P2P traffic in the Internet. We develop several heuristics that allow us to recognize P2P traffic at nonstandard ports. We find that depending on the protocol and metric used, approximately 30%-70% of traffic related to P2P applications cannot be identified using well-known ports. In addition we present several characteristics for various P2P networks, such as eDonkey2000, Fasttrack, Gnutella, BitTorrent, Napster and Direct Connect, as seen in traffic samples from two Tier1 commercial backbones in 2002 and 2003.**

I. Introduction

Over the last two years, peer-to-peer (P2P) activity has been a significant and growing component of Internet traffic. P2P applications are among the most popular applications; the top 5 downloads at SourceForge [39] (4 August 2003) were P2P applications. However, documentation of characteristics of P2P traffic has been limited.

P2P networking refers to virtual networks of computers that replace the distinct notions of server and client nodes with the the notion of *peers*. Despite huge differences among peers with respect to processing, connection speed, local network configuration or operating system, each member of the P2P network has the same functionality at the application layer. This peering functionality is in contrast to traditional network systems such as DNS where there is a clear distinction between the operations performed by each node. The absence of centralized authorities in P2P networks results in a totally distributed configuration of directly connected peers. Some P2P networks also have a small set of special nodes that usually handle queries. The main application of such networks is file sharing among users.

While P2P networks became popular only during the last few years, the concept of P2P networking was introduced early in the evolution of network communication systems. In fact both ARPANET in the late '60s and Usenet in the late 80's are in a sense early predecessors of today's P2P networks; they were distributed, decentralized networks intended for file transfer and sharing among equal peers. With the dramatic growth of the Internet in the early '90s, the popularity of the world-wide web somewhat displaced use and development of P2P networks. However, a series of technological developments lead to the explosion of P2P applications. First, the MPEG Audio Layer-3 (i.e., the popular mp3) encoding (1995 [20]) which facilitated huge data compression gains, accompanied by the release of free mp3 players, pervasively available by 1997 (e.g., winamp [43]). Encodings that offered considerable reduction for video data were also developed later (e.g., DivX [11] in 1999). Second, the increase of available bandwidth to end users with broadband technologies that provided inexpensive high-speed Internet access. Third, the pivotal Napster network [34] fielded in 1999 revolutionized file sharing, even though Napster was technically a hybrid-P2P rather than a pure P2P network since it retained the notion of a server for indexing content of the peers.

Since the advent and subsequent judicially imposed deterioration of Napster's service, dozens of different P2P networks have been developed, supporting millions of users and billions of file transfers. P2P applications have grown to be a considerable and on some links dominant fraction of Internet traffic; they contribute significantly to overall Internet traffic and performance characteristics. However, so far there have been only a few attempts to characterize P2P traffic [37]. So far the analysis and modeling community tends to neglect P2P traffic and/or assume that it follows general behavior of other traffic, although most P2P networks operate on top of nonstandard, usually custom-designed proprietary protocols. As a result, characterization of P2P traffic is problematic.

Furthermore, recent P2P networks tend to intentionally camouflage their generated traffic [26] to circumvent both

potentially filtering firewalls as well as legal issues most loudly articulated by the Recording Industry Association of America (RIAA). First-generation P2P traffic was relatively easily classified due to its use of well-defined port numbers specific to each network. However, an increasing number of P2P applications have the capability to utilize any port number, even port 80, traditionally used for Web traffic. There are also P2P applications that support encryption, and more that promise to do so. Although there are journalists reporting a drop in P2P application traffic in August 2003 [3], [14], it seems far more likely to simply reflect our (in)ability to recognize it (not to mention likely summer vacations for the largest population of P2P application users). For example, a decrease in P2P traffic could be a side-effect of P2P traffic appearing as Web or other types of traffic in increasing proportions.

In this paper, we characterize workload of P2P traffic and make a first attempt to estimate its real intensity beyond standard ports. We demonstrate P2P traffic patterns in the Internet backbone, using data collected at two different OC48 (2.5Gbps) links of Tier1 Internet Service Providers (ISPs) in 2002 and 2003. Our contributions include:

- We present a set of heuristics for identifying P2P traffic originating from several different P2P networks.
- We illustrate that P2P applications may use arbitrary ports to transfer data.
- We present an estimate of the percentage of P2P traffic using port 80.
- We demonstrate various P2P traffic characteristics, such as packet distributions and daily patterns.
- We discuss trends in P2P traffic by comparing traces from 2002 and 2003.

In general, we observe that not only has P2P activity not diminished, but on the contrary it corresponds to at least $1/5$th of the Internet traffic and is likely to continue to grow relentlessly in the future, RIAA behavior notwithstanding.

The rest of this paper is structured as follows. Section II presents previous work in P2P traffic analysis. Section III describes the analyzed backbone traces and statistics of their traffic composition. Section IV demonstrates a set of heuristics that enable us to identify P2P packets. Our analysis and results are presented in section V. In section VI we discuss the implications of the increase of P2P activity. Finally, section VII concludes our paper.

II. Previous Work

P2P measurement studies and analysis of P2P traffic characteristics has been limited. P2P research studies can be classified generally in three main categories: a) Information retrieval and content location, b) anonymity and privacy and c) characterization of P2P netowrks (topology, traffic etc.). Most studies have been concentrated to only a small number of P2P protocols, which usually include Fasttrack , Napster, and Gnutelladue to the popularity of the first and the easy access to the other two. While Fasttrack is still among the most popular networks, the popularity of Napster and Gnutella has been decreasing in favour of other networks such as eDonkey2000 or BitTorrent.

The first category consists of a series of studies stemming from the problem of the nonscalability of the original Gnutella network. Gnutella flooded the network to retrieve relevant information which resulted in poor performance. There are currenlty three different approaches to attack the problem of scalability. First, structured networks that distribute hash keys according to a predetermined protocol among the nodes, e.g. [41] [35]. Second, modifications to the Gnutella protocol, where usually knowledge from past search attempts is utilized to route subsequent queries. This is achieved by keeping track of different characteristics of past query responses from various peers in the network ([5], [22]). Finally, hybrid unstructured networks such as the existence of “Supernodes” in Fasttrack or the new Gnutella. These networks are not completely centralized (as in the case of Napster where there was one server/supernode), and also they are not completely unstructured. Generally, the latter design is believed to have the best performance to overhead ratio [45] and is widely used in todays P2P networks.

Privacy and ananomity have become significant issues in P2P networks following the RIAA threats. There is an increasing effort from P2P networks to hide file transfers and the identity of their users by various means (e.g., use of proxy servers). Most of these attempts only offer partial anonymity. True anonymous P2P networks include the pioneering Freenet network [7] and Mnet [31]. However, these networks are still at their early stages and intimidating for the nonexpert user.

In particular, P2P traffic was studied using *NetFlow* in [37]. The authors study three of the P2P networks, namely, Fasttrack, Direct Connectand Gnutella. They are able to show that P2P traffic contributed by different spatial aggregation levels (IP, prefix, AS) is stable over time. In addition, by comparing Web and P2P traffic, they indicate that prefixes contributing to P2P also contribute to Web traffic. The data from this study refer to fall 2001 measurements.

Usually, measurement studies examine topological properties of P2P networks [36] based on monitoring P2P

control packets. The study explore the tolopogy of the Napster and Gnutella networks. In addition, the Kazaa network has been studied in [26] and [25]. The authors present various statistics of the network by taking advantage of the fact that the Fasttrack network uses HTTP methods to transfer files and thus its activity can cached.

According to statistics from the IP Monitoring Project [40] which monitors the SprintLink IP backbone, P2P traffic is a significant fraction of the total workload. For example, for OC48 links sj-25 and sj-26 (utilization 48 Mbps and 64 Mbps respectively) approximately 20% of the packets, 25% of the bytes and 16% of the flows are related to P2P for July 1, 2003. For most of the rest of the links analyzed, P2P activity is only around 5%. Their analysis for the application breakdown is based on IANA [19] port assignments. For August 2002, the percentage of P2P traffic for the majority of the same links is approximately 20%. In addition, a large portion of the traffic is classifed as "other TCP" (approximately 15% for August 2002 and 25% for May 2003). Thus, according to these statistics P2P has declined since last year.

However, we believe that this is just an artifact of the fact that P2P is simply not visible using the standard port numbers. This may also be reflected in the increase of the "other TCP" percentage, which refers to TCP traffic that cannot be classified using known port numbers. As we will see in the following sections, a large portion of P2P traffic is characterized by arbitrary port numbers. In fact, we show that by adding the two percentages of P2P traffic at standard and arbitrary ports, P2P traffic is at least 20% of the total traffic in our backbone traces. This is in agreement with the August 2002 percentages from the Sprint network. In addition, we demonstrate that in August 2002, P2P applications did not have the capability to use arbitrary ports. Hence, we believe that the enhancement of P2P applications by utilizing arbitrary ports is reflected both in the increase of the "other TCP" category and in the decline of P2P activity in these statistics. That is, P2P traffic is shifting from known to arbitrary port numbers. These observations agree with similar comments in [16], where the authors observe an increase in unclassified TCP and web traffic when certain port numbers (Fasttrack ports) are rate-limited, implying the use of nonstandard port number by P2P applications.

III. Data description

We use packet traces (Table I) taken during 2002-2003 on OC48 links of US commercial Tier 1 backbones – specifically, the San Jose, CA to Seattle, WA SONET OC-48 (2.5 Gbps) links of two backbone networks. The backbone traces were collected by Linux-based monitors with Dag 4.11 network cards and packet capture software from the University of Waikato [29] and Endace.

We monitored traffic in both directions and captured 44-bytes of each packet, which includes IP and TCP/UDP headers (without options). Since most TCP/IP headers are 40 bytes long, we could examine the initial 4 bytes of payload after the header. This was mostly possible for Backbone 1 data where 66% of packets have 40-byte headers. In Backbone 2's network, most (over 75%) packets are encapsulated with an extra 4-byte MPLS label which leaves no space for payload bytes.

Most of our measurements were taken during business hours in the middle of the week to increase the likelihood of seeing a loaded link. This can potentially result in underestimation of P2P traffic, since file sharing is ostensibly more of a leisure activity. The 48-hour long trace D8 will have less such bias.

While we collected various datasets, we will only present analysis of the longest and most recent traces, D4 and D8 from Backbone 1 and D9 from Backbone 2.

A. Workload dimensions

The workload of forwarding equipment depends on bit and packet rates, which in turn depend on the number, durations and bitrates of flows on the link.[1] The number of flows depends in turn on the number of source-destination pairs communicating over the link.

Table I lists general dimensions of our datasets: counts of distinct source and destination IP addresses and the numbers of flows, packets, and bytes observed. Statistics that include destinations are often affected by scans, when a source sends a small number (1-3) packets to many destinations, most of which are autogenerated IP addresses. Such scans are prominent in D4S where the number of destinations exceeds the number of sources by a factor of more than 10. We leave identification of scans in these datasets as a subject for future study. Scans are unlikely to affect our analysis of P2P traffic since they mostly consist of SYN packets [24].

Bitrate averages in all our traces did not exceed 31% of link capacity even in the busiest hour of the day, which is typical for large backbone providers who overprovision capacity [15] to keep link utilizations below 50%. This approach allows for merging traffic from two links onto one in case of a link failure.

B. Load Variation

Variation in traffic load derives from diurnal patterns as well as extreme events. We measured load variation in

[1] We use a standard definition of a flow [6] as a sequence of packets with common 5-tuple of source and destination IP address, ports, and protocol, and a fixed timeout of 64 seconds.

TABLE I
BULK SIZES OF OC-48 AND OC-12 DATASETS

Set	Date	Day	Start	Dur	Dir	Src.IP	Dst.IP	Flows	Packets	Bytes
D4N	2002-08-14	Wed	09:00	8 h	Nbd (0)	2124 K	4074 K	106.6 M	2144 M	1269 G
D4S	2002-08-14	Wed	09:00	8 h	Sbd (1)	1122 K	12661 K	193.8 M	3308 M	2140 G
D8N	2003-05-07	Wed	00:00	48 h	Nbd (0)	3902 K	8035 K	275.5 M	4241 M	2295 G
D9VN	2003-05-07	Wed	10:00	2 h	Nbd (1)	904 K	2992 K	56.7 M	930.4 M	603.5 G
D9VS	2003-05-07	Wed	10:00	2 h	Sbd (0)	466 K	2527 K	47.3 M	624.2 M	340.0 G

the analyzed traces using CoralReef's [32], [23] `t2_rate` utility. In traces with medium utilization like D4 (8 hours) and D9 (2 hours), the load variation is small except for occasional floods and scans. In the lightly utilized traces such as the 48-hour backbone trace D8, link bitrate varies significantly, by a factor of 2-4 over the observation interval.

IV. HEURISTICS

This section describes our methodology to identify P2P traffic using nonstandard ports. We empirically derived our set of heuristics by studying various P2P protocols. Since documentation for these protocols is generally poor or based on disassembly and reverse-engineering of protocols by individual users, we have limited understanding of their specific characteristics. Thus to determine distinctive features for each protocol we monitored both TCP and UDP traffic using tcpdump[42] after installing various well-known P2P applications.

We identified two ways to differentiate among several P2P applications and distinguish them from other Internet traffic:

- *Packet format*: Control packets for each P2P protocol can usually be identified by examining a few bytes at the start of the TCP or UDP payload.
- *Block sizes*: Most P2P protocols use fixed block sizes for data transfers. Thus we can pinpoint P2P file transfers by spotting flows with specific block transfer sizes. Block sizes are associated with the TCP PUSH flag, so we can examine TCP packets with the PUSH flag set and use the TCP sequence numbers of two successive such PUSH packets to recognize P2P TCP flows.

These two criteria do not prevent false positives since certain block sizes used by P2P applications are fairly common among various other applications. Moreover, P2P traffic using port 80 (web traffic) is hard to distinguish, since Fasttrack and Gnutella use regular HTTP request methods. Thus, we use additional empirically identified rules specific to each protocol. We next describe the exact methodology used for each P2P protocol.

A. eDonkey2000

The eDonkey2000 P2P protocol [12] is currently supported by two well-known Windows-based clients: eDonkey2000 and the open-source eMule [13] client. Various operating systems support other clients, e.g., MLDonkey [30] and Shareaza [38]. The eDonkey2000 protocol uses two known TCP ports (4661, 4662) and one UDP port (4665). Data are transfered via TCP, whereas control packets can be either TCP or UDP (search related packets).

Block sizes: The source code of the eMule client suggests that eDonkey2000 transfers data blocks of 10,240 bytes, usually in 8 successive packets. However, we were seldom able to recognize such block sizes in eDonkey2000 traffic.

Packet format: All eDonkey2000 control packets are wrapped in a special header that starts with the eight-bit character *0xe3*, followed by an unsigned long that specifies packet length. Thus the minimum packet size for a TCP eDonkey2000 packet is 45 bytes (TCP header + 5 bytes of eDonkey2000 payload). On the other hand, eMule control packets start with a different eight-bit character, *0xc5*. Consequently, we can identify eDonkey2000 flows by looking at the first byte of the TCP or UDP payload. Since the first byte of the payload can be *0xe3* or *0xc5* for non-eDonkey2000 packets, a TCP flow must have at least ten such packets to be considered an eDonkey2000 flow (data packets that account for most packets in the flow rarely begin with 0xe3 or 0xc5). On the other hand, we require that 95% of the packets of a UDP flow agree with the eDonkey2000 pattern (since no data packets are transfered with UDP, all packets should begin with the special characters).

B. Fasttrack

Fasttrack is probably the most popular P2P network. Fasttrack clients include the well-known Kazaa Media Desktop (KMD) [10], Grokster [18] and iMesh [21]. By default Fasttrack uses port 1214 for both TCP and UDP traffic, but it is a simple user option to change to another port, including to port 80. Fasttrack flows are harder to identify due to the similarity of packet format and block sizes with HTTP traffic.

Block sizes: Analysis of our tcpdump traces revealed that Fasttrack block sizes are 65,536 (64K) bytes. In particular, KMD client flows consistently used 64K block sizes. iMesh flows also mostly used of 64K blocks, although a few iMesh flows used 2K (2,048 bytes) block sizes. In general this is overwhelming evidence of 64K block sizes for Fasttrack flows.

Packet format: The Fasttrack protocol uses regular HTTP requests to transfer files. File requests use HTTP's *GET* method, but the Fasttrack GET refers to a hash key for a specific file. Thus, a download starts with a TCP packet beginning with the string "GET /.hash=". Since our traces contain only 4 bytes of TCP payload, we only see the "GET", which is similar to the HTTP GET. In order to differentiate Fasttrack from HTTP flows we look for HTTP characteristics in the flow, e.g., "POST", "HEAD", "PUT", or "Update", which would rule it out as Fasttrack traffic. We also observed that the string *GIVE* (not an HTTP method) is common in the first four bytes of a TCP Fasttrack packet. On the other hand, UDP packets are easily distinguishable, since they contain specific byte patterns at the beginning of the UDP payload. These patterns include *0xc028*, *0x290000002901*, *0x280000002900*, *0x270000002980*, *0xc1* for packets with 5 bytes of payload and *0x2a* for packets with 3 bytes of payload.

Thus, identification of UDP Fasttrack flows is straightforward. To minimize false positives due to HTTP similarities for TCP flows, we use a combination of the above heuristics. We look for flows that: a) contain the keyword "GIVE" in the first four bytes of the payload; or b) they transfer block sizes of 64K without any packets that contain HTTP specific methods (e.g., POST, HEAD) or other HTTP characteristics in the first four bytes of the payload (e.g., HTML or XML patterns such as "</", "><", "<FON", "<img", "<?xm", etc.).

C. Gnutella

The Gnutella [17] network has similar traffic characteristics to Fasttrack. It also uses HTTP requests for file transfers. Gnutella uses ports 6346 and 6347 for TCP and UDP traffic. Known Gnutella clients include Morpheus [33], Limewire [27], BearShare [1] and XoloX [44].

Block sizes: We found that each Gnutella client uses different transfer block sizes. For example, Morpheus seems to transfer blocks of 32K (32,768 bytes) and Limewire uses 100,010 bytes. A 2K transfer size is also commonly used.

Packet format: As in Fasttrack, file downloads are initiated by a GET packet, with format is "GET /uri-res/". As with the Fasttrack protocol, we cannot identify such packets with only four bytes of payload. However, many TCP Gnutella packets start with strings "GNUTELLA" or "GIV". All UDP Gnutella packets start with the string "GND".

We recognize Gnutella flows by a methodology similar to that for Fasttrack protocol. We identify Gnutella flows by looking for the strings "GNUT" and "GIV" for TCP or "GND" for UDP at the beginning of a packet. Furthermore, we look for transfers with specific block sizes, that do not contain HTTP characteristics.

D. Direct Connect

Similar to eDonkey2000, the Direct Connect protocol [8] is supported by a number of clients for various operating systems, e.g., the open-source DC++ client [9]. Direct Connect clients use 411 and 412 as default ports for both TCP and UDP. All Direct Connect control packets begin with the special character *'$'*. Furthermore, tcpdump traces allowed us to identify specific control messages used by the protocol at the beginning of each packet, e.g., $Send, $Search, $Get, $MyInfo, $MyNick, $Direction, $Hello, $Quit, $Connect, $Lock and $Key for TCP packets, and $SR or $Pin for UDP packets. Thus Direct Connect flows can be effectively recognized by looking at the first few bytes of their payload.

E. Napster

The Napster network was the first to establish P2P networking. Almost every operating system has a free napster client. Napster uses a range of ports between 6699 and 6702 for TCP transfers, and 6257 for UDP.

Block sizes: We were able to identify that a number of Napster clients (e.g., WinMx) transfered blocks of 2,080 bytes in two successive packets. The first packet was usually a 1500-byte packet and the second 660 bytes (removing 40 bytes for both TCP headers yields 2,080 bytes).

We were not able to identify this specific block size at arbitrary ports besides the ones mentioned above. There were no specific packet formats recognizable by the use of tcpdump at the first bytes of the payload.

F. BitTorrent

BitTorrent [2] is a newer P2P protocol whose popularity is growing rapidly. BitTorrentuses 6881 as the default port; if that port is unreachable BitTorrent tries to connect to a number of successive ports up to 6889. If the client cannot connect to port 6889, it gives up. Thus, currently BitTorrent does not transfer files on arbitrary ports.

TABLE II
CHARACTERISTIC BISTRING OF P2P PACKET FORMAT

P2P Protocol	*String*	*Transfer protocol*	*Default ports*
eDonkey2000	0xe3, 0xc5	TCP/UDP	4661-4665
Fasttrack	“Get /.hash”, “GIVE” 0x270000002980, 0x280000002900, 0x29000000, 0xc028 0xc1 (5 bytes), 0x2a(3 bytes)	TCP UDP UDP	1214
BitTorrent	“GET /announce?info_hash”, “GET /torrents/” “GET TrackPak”, “0x13BitTorrent” 0x00000005, 0x0000000d, 0x00004009	TCP TCP TCP	6881-6889
OpenNap/Napster (starting 3rd byte first 2 bytes is packet length)	0x31 (1 byte), 0x0000ca00 (4 bytes) 0xc800, 0xc900, 0xcb00, 0xcc00, 0x0000, 0xd600 0x0200, 0x0300, 0x0600, “SEND”, “GET” (3 bytes)	TCP TCP TCP	6699
Gnutella	“GNUTELLA”, “GIV”, “GET /uri-res/”, “GET /get/” “X-Versio”, “X-Dynami”, ”X-Query”, ”X-Ultrap”, “X-Max”, “X-Quess”, ”X-Try”, ”X-Ext”, ”X-Degree”, “X-Gnutel” “GND”	TCP TCP TCP UDP	6346-6347
MP2P	GO!!, MD5, SIZ0x20 , STR0x20 0x000000 (starting at 3rd byte)	TCP UDP	10240-20480 41170, 22321
Direct Connect	“$Send”, “$Search”,”$Connect”,”$Get”,”$MyInfo” “$MyNick”,”$Direction”,”$Hello”,”$Quit,”$Lock”,$Key “$SR”,”$Pin”	TCP TCP UDP	411-412
Soulseek (starting at 3rd byte first 2 bytes is packet length	0x00000100, 0x00000300, 0x00000700, 0x00001200, 0x00001a00 0x00000900, 0x00002800, 0x00002900, 0x00002a00, 0x000003310000, 0x0000(4 bytes)	TCP TCP TCP	2234, 5534
Ares	“GET hash:”, “PUSH ”	TCP	
EarthStation 5	“GET /$$$$$$$$$/” 0xcf64 (starting at 3rd byte)	TCP UDP	

V. ANALYSIS OF P2P TRAFFIC

We now present P2P traffic characteristics for the traces described in section III. First, we describe packet and bit rates for each protocol for August 2002 and May 2003. Second, we compare their packet size and transfer size distributions. Finally, we analyze the magnitude of P2P activity in arbitrary ports as found by our methodology described in section IV.

A. Packet and bit rate of P2P protocols

We demonstrate that the percentage of P2P traffic on the observed links increased by approximately 2% from August 2002 to May 2003. We compare packet and bit rate of each protocol between August 2002 and May 2003 for Backbone 1. Note that because of the different utilization of the link for these two months, we can only make a relative comparison of the applications to total traffic. Also, the duration of the two traces is different (8 hours, 09:00-17:00, for the August 2002 trace and 48 hours for the May 2003 trace beginning at midnight). The August 2002 dataset corresponds only to daytime, where P2P activity may be lower than at night. However, during overlapping times of day, both traces follow some general trends. Finally, observations in this section refer only to our compiled list of known protocol ports.We discuss analysis of P2P traffic on arbitrary ports later in this section.

Fig. 1 shows the bitrate for both August 2002 and May 2003. The top six strips in each case plot the rate for the corresponding protocol; the bottom strip plots aggregate P2P traffic (sum of top 6 strips) compared to the total traffic on the link (log scale; upper line of the bottom plot corresponds to total traffic on the link, lower line is aggregate P2P traffic). Note that these bitrates refer only to known protocol ports. Furthermore, as we will see in section V-C we observed no P2P traffic on arbitrary ports in August (with the exception of eDonkey2000).

From this figure we can observe the following:

- **eDonkey2000, Fasttrack and Napster are the most popular P2P networks:** These three protocols contribute the vast majority of P2P traffic. Although Fasttrack is considered the most popular network, Napster and eDonkey2000 generate comparable levels of traffic to Fasttrack.
- **eDonkey2000 seems more popular in Europe and**

Asia: The May dataset of Fig. 1 shows that eDonkey2000 is "out of phase" with other P2P protocols in its daily variation. In fact, eDonkey2000 traffic daily patterns correspond to Europe and Asia time-zones. We confirmed this hypothesis with initial observations of source and destination IPs of eDonkey2000 flows.

- **Manifestation of new P2P protocols:** BitTorrent and Direct Connect seem to gain an increasing portion of P2P traffic. BitTorrent's popularity growth is particularly dramatic, with considerable magnitude in May 2003 while literally nonexistent in August 2002. On the other hand, usage of Direct Connect is also expanding as is visible in the rate plots. While the rate of Direct Connect is similar both in August 2002 and May 2003, recall that the utilization of the link in May 2003 is six times lower, see section III.
- **These data sets show an increase in P2P traffic of approximately 2.5% for packets and 1.5% for bytes relative to the total traffic from August 2002 to May 2003:** Even by only considering the standard ports, the percentage of aggregate P2P traffic increased relative to the total traffic. The total number of bytes increased from 12.5% of the total traffic in August 2002, to 14.2% in May 2003, on average. Further, the total number of P2P packets transferred increased from 13.9% to 16.6%, on average. In addition, the P2P load on the same Backbone 1 link over the comparable time interval (09:00-17:00, PDT) increased from approximately 12% to 20% relative to total traffic. While we do not know how much of P2P relative traffic growth is attributed to changes in customer cross-section or routing engineering, our traces are definitely inconsistent with a decrease in P2P traffic.

Fig. 2 shows similar stripcharts for our other backbone trace for both directions of the link. Surprisingly, P2P traffic in this link is only a minor portion of total traffic (less than 5%). This contrasts both with results from Backbone 1's link shown above. Despite this fact we can still confirm our previous observations from Backbone 1's link regarding the popularity of each application. In addition, recent measurements from the Backbone 2 link (1-7 August 2003), show that traffic on this link consists of 15% P2P. This change in the link's traffic composition since May 2003 could be a result of changes in routing, but regardless it provides further evidence that P2P traffic has not declined.

B. Packet and Transfer Size Distributions

We present packet size distributions for all analyzed P2P protocols. Due to control message behavior, packet size patterns can help with heuristic classification of P2P traffic. The cumulative distribution function (CDF) of transfer sizes and especially of one-packet flows lends further credence to this methodology. Combining these distributions and our heuristics to recognize P2P flows is part of our ongoing work.

Fig. 3 (left) shows the packet size distribution of each P2P protocol. We plot all histograms using log-squared normalization of the frequency (y-axis), but Y-axis labels correspond to the true frequencies of each packet size (i.e., they are not log-squared normalized). This specific normalization allows us to see all spikes of the histograms while preserving their relative magnitude. We computed histograms over one hour of data (23:00 to Midnight) from Backbone 1's link.

Observation of packet size histograms reveals important differences among the protocols. Each histogram has at least one spike at packet sizes specific to the corresponding protocol (e.g., Direct Connect at 932 bytes, BitTorrent at 377 bytes). Each protocol also has the typical spikes at 40, 576 and 1500 bytes characteristic of most Internet TCP traffic. Most P2P protocols carry a large proportion of small packets, presumably control packets, while triggered P2P data transfers use full-sized (1500) packets.

In addition, Fig. 3 (right) presents the CDF of transfer sizes for eDonkey2000, Fasttrack, Gnutella and Direct Connect. The CDF corresponds to flow sizes for one hour of Backbone 1's trace. As expected the distributions are long-tailed indicating large transfers. But for small flow sizes (less than 1500 bytes which imply one-packet flows), the CDF exhibits discontinuities for all P2P protocols. For instance, the Fasttrack CDF sharply increases from approximately 17% to 27% at 120 bytes. As with the histograms, these plots imply preferred packet sizes specific to each protocol. Moreover, certain block sizes are also visible in the distributions, which confirm our heuristics in section IV. We indicate two such blocks in Fig. 3; the 64K block observed in Fasttrack and also the 2K block size for Gnutella.

C. P2P traffic at arbitrary ports

We have observed that most P2P applications may use arbitrary ports both for control and data packets. Flows are uniformly distributed across a range of ports with the exception of few preferred port numbers. As a result, any statistic regarding P2P traffic using only the standard,

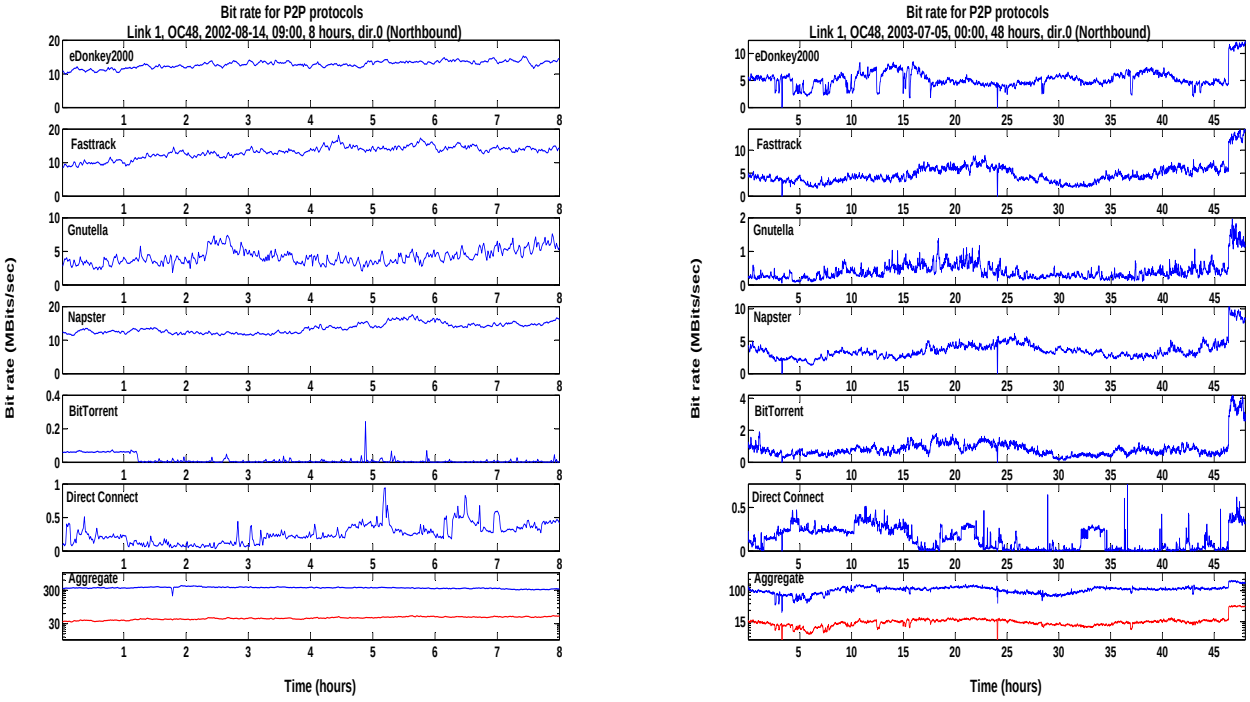

Fig. 1. Backbone 1 link's bitrate for August 2002 and May 2003 for each of the protocols and for the aggregate traffic. Considering the drop in utilization, the percentage of P2P traffic has remain constant relative to the total traffic on the link.

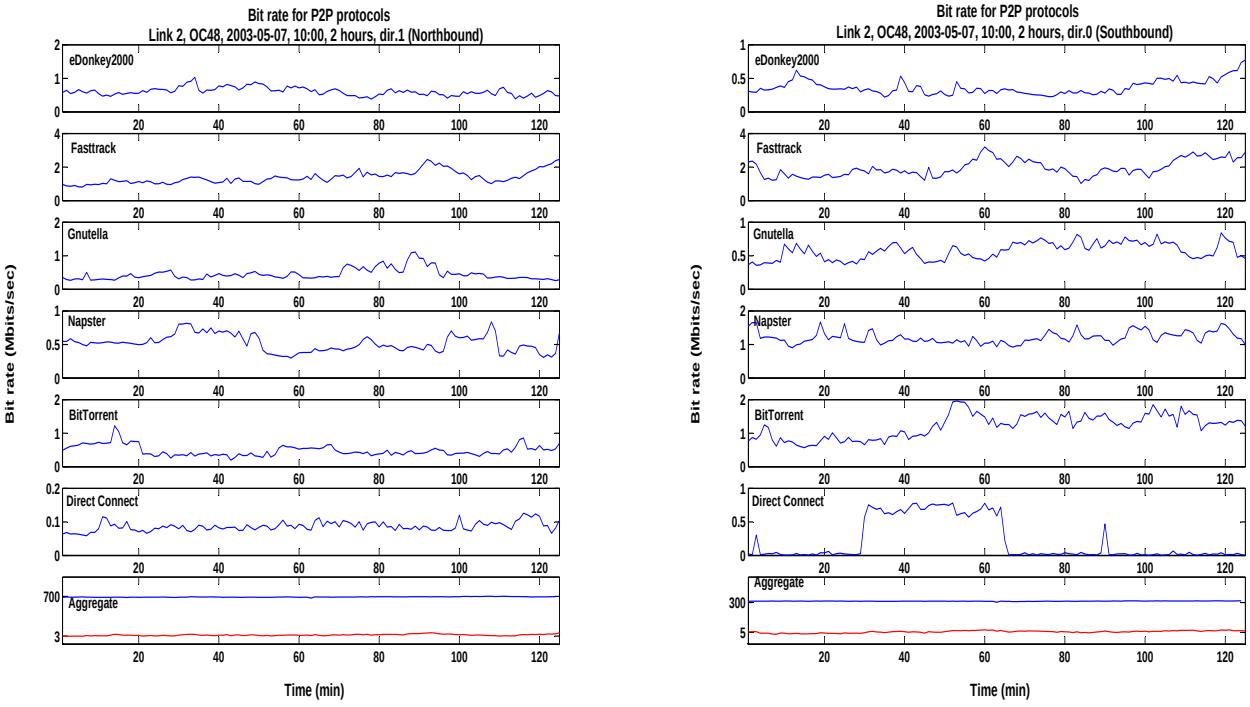

Fig. 2. Backbone 2 link's bitrate for August 2002 and May 2003 for both directions of the link and for each of the protocols and for the aggregate traffic. Surprisingly, P2P traffic corresponds to only a small portion of the total traffic.

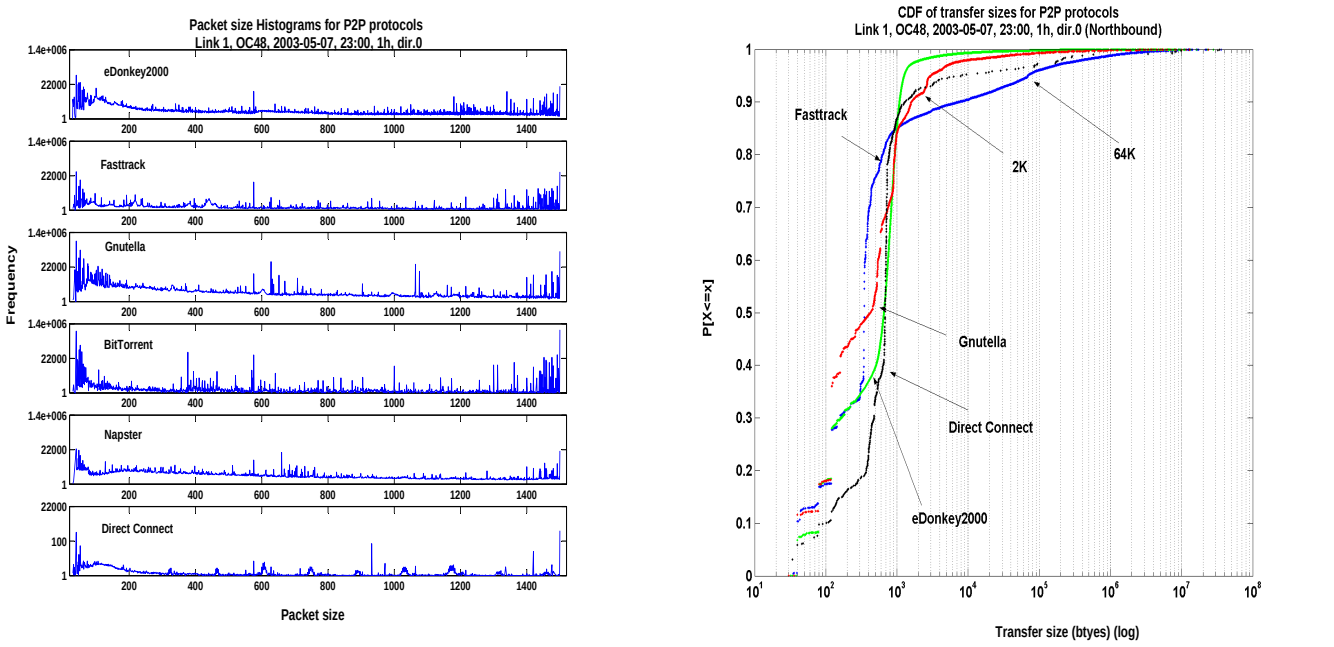

Fig. 3. (LEFT) Packet size distributions for all analyzed P2P protocols. Each protocol has different preferred packet sizes that correspond to control packets, shown by spikes in each histogram. (RIGHT) Cumulative distribution function (CDF) of transfer sizes of flows within an hour for eDonkey2000, Fasttrack, Gnutella and Direct Connect. Discontinuities in the CDFs (for sizes less than 1500 bytes corresponding to one-packet flows) also imply preferred packet sizes for each protocol.

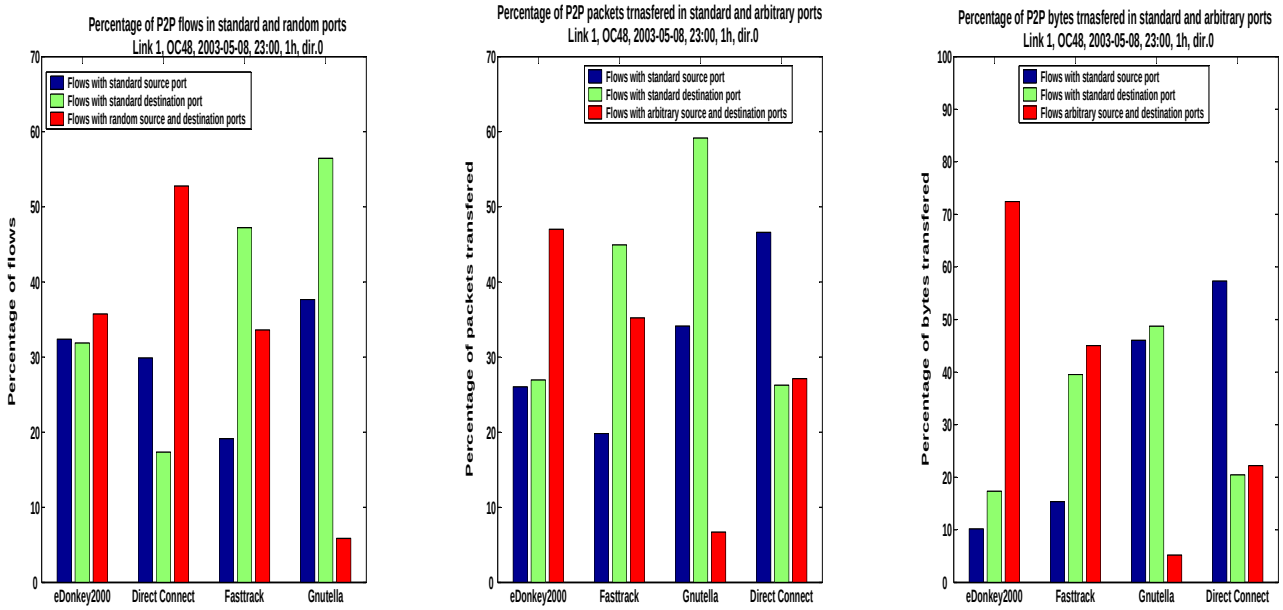

Fig. 4. Percentages of flows and transferred packets and bytes using standard and arbitrary port numbers. The most popular protocols, Fasttrack and eDonkey2000, use mostly arbitrary port numbers. Except for Gnutella more than 30% of the flows have nonstandard source and destination ports. Almost 50% of eDonkey2000 packets , 70% of eDonkey2000 bytes and more than 40% of Fasttrack bytes are transferred in arbitrary ports for both source and destination. Increasing percentages in byte transfers for these protocols imply large packets for data transfers.

well-known port numbers dangerously underestimates the magnitude of P2P activity. Here we attempt for the first time to quantify how much P2P traffic uses arbitrary TCP or UDP ports. To our knowledge, there have not been any

TABLE III

TOP 5 DESTINATION PORTS AS INDICATED BY THE PERCENTAGE OF FLOWS IN EACH PORT IF FLOW HAS ARBITRARY PORTS FOR BOTH SOURCE AND DESTINATION.

eDonkey2000		*Fasttrack*		*Gnutella*		*Direct Connect*	
Port	% Flows	Port	% Flows	Port	% Flows	Port	% Flows
4672	19%	3088	2%	34931	41%	1412	15%
103	13%	2230	1%	2541	7%	23	5%
4246	11%	3120	1%	6348	4%	20695	4%
7658	10%	1882	1%	12998	2%	21885	4%
80	2%	1642	1%	6349	2%	13727	4%

TABLE IV

TOP 5 DESTINATION PORTS AS INDICATED BY THE PERCENTAGE OF FLOWS IN EACH PORT IF THE WELL-KNOWN SOURCE PORT IS USED

eDonkey2000		*Fasttrack*		*Gnutella*		*Direct Connect*	
Port	% Flows	Port	% Flows	Port	% Flows	Port	% Flows
4672	0.3%	80	2%	6346	2%	1412	13%
4665	0.08%	3088	1%	34931	1%	412	4%
80	0.05%	2230	1%	4672	0.2%	23367	1%
4246	0.05%	1214	1%	4246	0.1%	5182	1%
1031	0.04%	3896	1%	103	0.1%	2071	1%

studies in the literature presenting similar estimates.

Using the heuristics described in section IV, we identified numerous flows with arbitrary source and destination ports. Since only the identification of UDP flows is straightforward, and we are able to identify only a portion of the total number of TCP P2P flows, statistics presented here will likely underestimate the magnitude of P2P traffic.

Our major observations can be summarized in the following points:

- **With the exception of eDonkey2000, we saw no P2P traffic on arbitrary ports in August 2002.** In the August 2002 trace, either the destination or source port was well-known. Moreover, both ports for Fasttrack UDP traffic were always 1214. There was no UDP traffic on ports 6346 or 6347 (Gnutella).
- **With the exception of Napster and BitTorrent, all other P2P protocols can transfer packets on arbitrary ports.** We were not able to recognize flows on arbitrary ports for Napster and BitTorrent. For other P2P applications, approximately 30%-40% of the flows use arbitrary ports for both source and destination.
- **Port 80 usually appears as the destination port for P2P transfers.** P2P applications camouflage their traffic under port 80 by taking advantage of the fact that firewalls do not filter web traffic. In fact, websites describing P2P applications suggest to their users to change the application port to port 80 in case of firewall issues.

Tables III and IV show the top 5 preferred destination port numbers. The first table refers to flows with arbitrary source port; the second describes flows with the standard source port. These tables show that: a) Most percentages are low, implying no preferred port numbers; rather, port numbers are uniformly distributed across the supported range. (Note that the major portion of port numbers corresponds to ephemeral ports used by the Windows OS.) b) Similar port numbers are in the top five list across different protocols (e.g., 4246, 103 for eDonkey2000 and Gnutella), perhaps due to P2P clients that can connect to more than one of the networks (e.g., Shareazaa).

Fig. 4 shows the percentage of flows, packets and bytes transferred on standard and arbitrary ports. Percentages refer to the total number of flows for the specific application. With the exception of Gnutella, more than 30% of flows use nonstandard source and destination ports. Fasttrack and Gnutella packet percentages are similar to flow percentages. On the other hand, approximately 50% of eDonkey2000 packets are transferred using arbitrary ports. Also, while most Direct Connect flows use arbitrary ports, approximately 60% of Direct Connect packets are transferred in flows with the standard source port, consistent with the use of arbitrary port numbers for control packets but not large data transfers.

Most eDonkey2000 bytes (70%) are transferred on ar-

bitrary ports. Since only 50% of packets belong to flows with nonstandard ports, eDonkey2000 most likely uses large packets for data transfers. Further, most Fasttrack bytes are transferred in arbitrary ports (in contrast to flow and packet percentages). For Direct Connect and Gnutella , percentages are similar to those for packets (with the exception of the increased percentage of transfers using the standard source port for Gnutella[2].)

To confirm our results, we modified *NeTraMet* [4] to integrate our heuristics into its functionality. Fig 5 plots NeTraMet-gathered data for two different bitrates of eDonkey2000 for both directions of Backbone 2's link: (1) the bitrate identified using the standard port numbers; and (2) the bitrate if our heuristics are used. The trace was collected 7 August 2003 at 20:00 for two hours continuously. Due to CPU strain required for string matching every packet of a 20% utilized OC48 link (over 100,000 packets per second), the hardware dropped around 20%-25% packets every second. Assuming random loss, the curves in Fig. 5 confirm our previous results. The P2P bitrate increases 30%-60% with our heuristics relative to just counting P2P traffic using standard ports. Note also that eDonkey2000 bitrate on standard ports agrees with the August 2002 Backbone 1 bitrate. (These two traces, the Backbone 1 August 2002 and the two-hour NetTraMet Backbone 2 trace, have similar utilizations.) We also plan to perform similar analysis using NeTraMet for the rest of the protocols .

VI. Discussion

The importance of P2P traffic for the future of the Internet stems from the different requirements that it places on bitrate provisioning in edge and access networks, as well as on peering relations among ISPs.

Bit rates of many access links, in particular for DSL and cable modems, are currently provisioned asymmetrically in anticipation of users downloading much more data than they send upstream. This assumption is framed by the client-server model that has historically predominated the Internet in the last decade. The client-server model was satisfactory while multimedia content development was more commonly a professional activity using studio-class equipment and distributed from adequately provisioned servers.

But this decade brings a clear trend toward audiovisual content as a household commodity produced and exchanged among end users for leisure. Factors driving this trend include the increasing availability of: relatively inexpensive bandwidth; new technologies that facilitate creation of multimedia content without training or background; proliferating free or inexpensive professionally created content; and affordable and powerful hardware/software platforms; (Today's laptop is 1000 times faster, has 1000 times more storage and memory and is 1000 times cheaper than a mainframe of the 80's, an improvement of 10^6.)

Wide adoption of the P2P paradigm will bring dramatic changes in Internet supply and demand. Technologies such DSL and cable modem were quite sufficient when downstream throughput was the main concern. Their attractiveness will fade and their market share dwindle if alternative broadband technologies are deployed that offer comparable upstream and downstream performance, Increased P2P traffic levels may also result in more predictable (if higher) traffic patterns due to larger file transfers.

Evolution toward more symmetric load on access networks will potentially result in increased peering among ISPs. Current practices require balanced bidirectional load among peers[3], a stipulation much easier to achieve with symmetric link utiliations as the norm. There is no doubt that the P2P paradigm will change Internet engineering as we know it today. The only remaining question is when, not if.

VII. Conclusions

Our primary goal of this study is to accurately estimate the levels and growth of P2P traffic . P2P clients are now among the most popular applications, and P2P networks serve millions of users and perform hundreds of millions of file transfers every week, while P2P protocols grow more sophisticated with each new software release. As a traffic category that threatens a dramatic change in how we must approach traffic engineering, it has become too important to ignore.

Traditionally, P2P traffic was identified by well-known port numbers unique to each protocol. Using these standard ports we were able to show an increase in aggregate P2P traffic relative to total network traffic by studying backbone traces for two tier 1 backbones for 2002 and 2003. In particular, in contrast to recent claims suggesting a decline in the intensity of P2P traffic [3], [14], our detailed analysis shows that the percentage of P2P traffic

[2]Note that in all cases, use of well-known source or destination ports does not imply in any way that standard ports are used for *both* the source and destination ports. On the contrary, table IV indicates that at least one of the ports is arbitrary.

[3]"The ratio of the aggregate amount of traffic exchanged between the Requester and the WorldCom Internet Network with which it seeks to interconnect shall be roughly balanced and shall not exceed 1.5:1." [28].

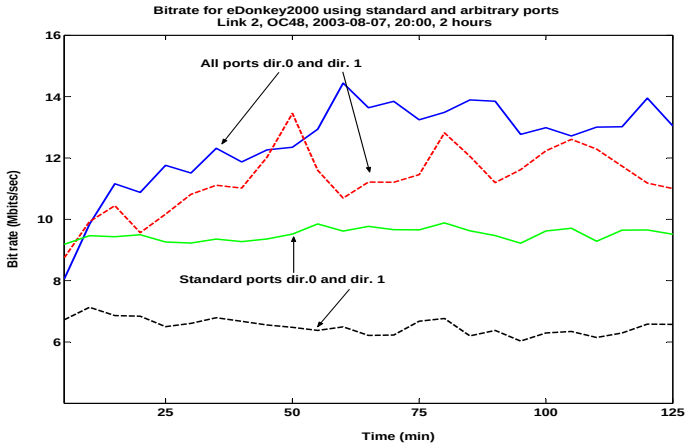

Fig. 5. eDonkey2000 bitrate using standard and all ports for a 2-hour trace captured in August 7, 2003. The bitrate is 30%-60% more depending on the direction if also nonstandard are considered.

on our monitored link increased by approximately 2%. In addition, we presented several different characteristics for six P2P protocols/networks, namely eDonkey2000, Fasttrack, Gnutella, Napster, BitTorrent and Direct Connect.

We emphasized that port numbers reveal only a fraction of total P2P traffic. Most P2P protocols after napster were improved to support the use of any possible port to transfer control and data packets. Hence, any statistic regarding P2P activity using only the standard ports underestimates its levels, potentially drastically.

To our knowledge this paper is a first attempt to characterize P2P traffic on both standard and arbitrary TCP or UDP ports. We developed a set of straightforward heuristics specific to each protocol, and demonstrated that, depending on the protocol and traffic metric (e.g., number of flows, bytes or packets transfered), 30%-70% of traffic generated by the most popular P2P applications (Fasttrack, eDonkey2000) cannot be identified by the traditional approach using standard port numbers. This result implies that aggregate P2P traffic constitutes more than 20% of total traffic (considering that P2P traffic for standard ports is around 15%) on the observed links. We also emphasize that this technique *still* underestimates P2P traffic since we cannot effectively recognize all P2P TCP flows using these heuristics. We are continuing work to develop non-port-based heuristics, e.g., using packet size patterns, for identifying P2P Internet traffic and believe we will eventually be able to identify virtually all P2P flows. CAIDA also plans to incorporate the knowledge gained from this study into the CoralReef software suite.

ACKNOWLEDGMENTS

We are thankful to our colleagues Ken Keys, David Moore, Marina Fomenkov and Srinivas Shakkottai for their feedback and encouragement.

REFERENCES

[1] BearShare. http://www.bearshare.com.
[2] BitTorrent. http://bitconjurer.org/BitTorrent/.
[3] John Borland. RIAA threat may be slowing file swapping. http://news.com.com/2100-1027_3-1025684.html?tag=lh.
[4] Nevil Brownlee. Using NeTraMet for Production Traffic Measurement . In *Integrated Management Strategies for the New Millenium*, 2001.
[5] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *ACM SIGCOMM*, 2003.
[6] K. Claffy, H.-W. Braun, and G. Polyzos. A parametrizable methodology for internet traffic flow profiling. In *IEEE Journal on Selected Areas in Communications*, 1995.
[7] I. Clarke, T. W. Hong, Scott G. Miller, O. Sandberg, and B. Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.
[8] Direct Connect. http://www.neo-modus.com/.
[9] DC++. http://dcplusplus.sourceforge.net/.
[10] Kazaa Media Desktop. http://www.kazaa.com/us/index.htm.
[11] DivXNetowks. http://www.divxnetworks.com/.
[12] eDonkey2000. http://www.edonkey2000.com/.
[13] eMule. http://www.emule-project.net/.
[14] Benny Evangelista. Kazaa use down after threats; crackdown's impact on file sharing unclear. http://news.com.com/2100-1027_3-1025684.html?tag=lh.
[15] C. Fraleigh, F. Tobagi, and C. Diot. Provisioning IP Backbone Networks to Support Latency Sensitive Traffic. In *IEEE Infocom*, San Francisco, March 2003.
[16] A. Gerber, J. Houle, H. Nguyen, M. Roughan, and S. Sen. P2P The Gorilla in the Cable. In *National Cable & Telecommunications Association(NCTA) 2003 National Show*, Chicago, IL, June 8-11, 2003.
[17] Gnutella. http://www.gnutella.com/.
[18] Grokster. http://www.grokster.com/.
[19] iana. Internet Assigned Numbers Authority. http://www.iana.org/assignments/port-numbers.
[20] Fraunhofer IIS. http://www.iis.fraunhofer.de/amm/techinf/layer3/index.html.
[21] iMesh. http://www.imesh.com/.

[22] K. Sripanidkulchai and B. Maggs and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.

[23] 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*, 2001.

[24] C. B. Lee, C. Roedel, and E. Silenok. Detection and Characterization of Port Scan Attacks. work in progress.

[25] N. Leibowitz, A. Bergman, Roy Ben-Shaul, and Aviv Shavit. Are File Swapping Networks Cacheable? Characterizing P2P Traffic. In *7th International Workshop on Web Content Caching and Distribution (WCW)*, 2002.

[26] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the Kazaa Network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, 2003.

[27] Limewire. http://www.limewire.com/.

[28] MCI. Worldcom policy for settlement-free interconnection with internet networks, 2003. http://global.mci.com/uunet/peering/.

[29] J. Micheel, S. Donnelly, and I. Graham. Precision timestamping of network packets. In *Proceedings of ACM Sigcomm Internet Measurement Workshop (IMW)*, Nov 2001.

[30] MLDonkey. http://www.nongnu.org/mldonkey/.

[31] Mnet. Internet Assigned Numbers Authority. http://mnet.sourceforge.net/.

[32] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and kc claffy. Coralreef software suite as a tool for system and network administrators. In *Usenix LISA*, 2001.

[33] Morpheus. http://www.morpheus.com/.

[34] Napster. http://www.napster.com/.

[35] S. Ratnasamy, . Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *ACM SIGCOMM*, 2001.

[36] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.

[37] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. In *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[38] Shareaza. http://www.shareaza.com/.

[39] SourceForge.net. http://sourceforge.net/.

[40] Sprint. Packet Trace Analysis, 2003. http://ipmon.sprintlabs.com/packstat/packetoverview.php.

[41] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[42] tcpdump. http://www.tcpdump.org/.

[43] WINAMP. http://www.winamp.com/.

[44] XoloX. http://www.xolox.nl/.

[45] B. Yang and H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In *The VLDB Journal*, pages 561–570, 2001.