A Measurement-based Traffic Profile of the eDonkey Filesharing Service

Kurt Tutschku

Institute of Computer Science, University of Würzburg,
Am Hubland, D-97074 Würzburg, Germany.

tutschku@informatik.uni-wuerzburg.de

Abstract. Peer-to-peer file sharing applications have evolved to one of the major traffic sources in the Internet. In particular, the eDonkey file sharing system and its derivatives are causing high amounts of traffic volume in today's networks. The eDonkey system is typically used for exchanging very large files like audio/video CDs or even DVD images. In this report we provide a measurement based traffic profile of the eDonkey service. Furthermore, we discuss how this type of service increases the "mice and elephants" phenomenon in the Internet traffic characteristics.

1 Introduction

Peer-to-peer (P2P) file sharing applications have evolved to the major traffic sources in the Internet. In particular, the eDonkey2000 P2P file sharing system [1] and its derivatives [2, 3] are causing high amounts of traffic volume [4]. The eDonkey[1] system is typically used for exchanging very large files like CDs or even complete DVDs images. The service is highly robust and obtains considerable short download times.

P2P file sharing traffic is considered to be hazardous for networks. This view is mainly due to the high traffic volume but also caused by the transfer of very large files. The latter feature might increase the "mice and elephants" phenomenon in Internet traffic [5, 6]. The phenomenon describes that the traffic consists of mainly short transfers (referred to as "mice") and long transfers (referred to "elephants"). Elephant streams are considered harmful for the network since they clog the system whereas mice may reduce the throughput if issued with high frequency [7].

The aim of this paper is to provide a traffic profile for the eDonkey service. The focus of the study is on the distinction of non-download traffic and download traffic. In addition, we discuss the "mice and elephants" characteristic in eDonkey and the origin and destination of eDonkey flows. The paper is organized as following. Section 2 outlines the eDonkey architecture and protocol. Section 3 describes at briefly the measurement setup and focuses on the measurements. Section 4 discusses related work on P2P behavior and traffic models. Section 5 summarizes the measurement results and provides a brief outlook.

[1] In this paper we subsume eDonkey2000 and all its derivatives by the single term eDonkey.

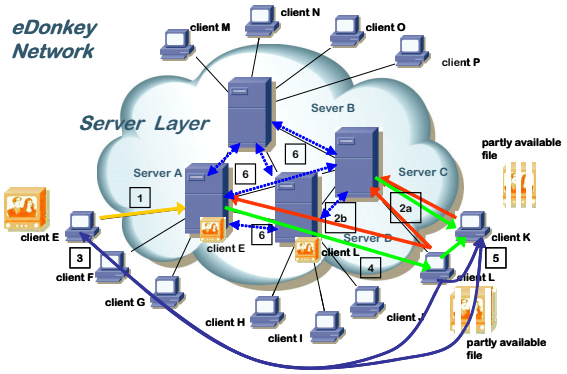

Fig. 1. eDonkey Communication

2 The eDonkey P2P File Sharing Service

The main features of eDonkey P2P file sharing application are: *a)* it doesn't rely on a single central server, *b)* a file can be downloaded from several different peers at once, and *c)* a file can be shared by a peer before it is completely obtained. The eDonkey protocol appears is not officially documented. A details have been obtained recently through reverse engineering [2, 3, 8].

Architecture and Operation: The eDonkey file sharing service belongs to the class of *hybrid P2P* architectures. Its architecture comprises two applications which form the eDonkey network: the eDonkey client[2] and the eDonkey server, cf. Figure 1. The eDonkey client is used to share and download files. The eDonkey server operates as an index server for file locations and distributes addresses of other servers to clients[3]. In the eDonkey network no files are transmitted through the server. Every eDonkey user is eligible to setup a server.

Searching and Sharing of Files: When a client connects to the eDonkey service, it logs on to one of the servers (using a TCP connection) and registers all files it is willing to sharing, cf. 1 in Figure 1. Each server keeps a list of all files shared by the clients connected to it.

When a client searches a file, cf. 2a in Figure 1, it sends the query to its main server. The server returns a list of matching files and their locations. The client may resubmit the query to another server, cf. 2b, if none or an insufficient number of matches have been returned. The major communication between client and server is typically implemented by TCP connections on port '4661'. Additional communication between clients and servers, e.g. further queries and their results, are transmitted via UDP on port '4665'.

[2] The terms "client" and "peer" are exchangeable in the context of eDonkey.
[3] In addition, eDonkey clients may also distribute server addresses among each other.

Downloading of Files: When an eDonkey client decides to download a file, it first gathers a list of all potential file providers and then asks the providing peers for an upload slot, see 3 in Figure 1. Upon reception of a download request, the providing client places the request in its *upload queue*. A download request is served as soon as it obtains an upload slot. eDonkey clients may restrict their total upload bandwidth to a given limit. An upload slot comes available when a minimum fair share of the upload limit is possible. When an upload slot is available, the providing client initiates a TCP connection to the requesting client, negotiates which chunk of the file is exchanged, and transmits the data.

The eDonkey protocols splits the file into separate pieces, denoted as *chunks*. A chunk has typically a size of 10MBytes. The consuming client can reassemble the file using the chunks or parts of chunks. A client can share a file as soon as it a has received a complete chunk, see 4 in Figure 1. A major feature of eDonkey is that the consuming client may operate in the *multiple source download* mode, cf. 5 in Figure 1. In this mode, the downloading client issues in parallel two or more requests to different providing clients and retrieves data in parallel from the providers.

Since an eDonkey client may leave the eDonkey service at any time, the requesting client has to renew its download request periodically otherwise the requests are dropped. In order to reliably check the availability of a client, the eDonkey protocol uses TCP connections on port '4662' for the communication between the clients. A client-to-client connection is terminated by the eDonkey application after an idle period of 40sec. It is worth to be mentioned here, that other P2P file sharing applications like Bearshare [9] or KaZaA [10] have implemented similar *multiple source download* schemes.

Server-to-Server Communications: The communication between eDonkey servers is very limited, cf. 6 in Figure 1. The servers contact each other periodically but with small frequency in order to announce themselves and to send back a list of other servers. In this way the servers maintain an updated list of working servers and affirm the search efficiency of the eDonkey service.

3 eDonkey Traffic Profile

3.1 Measurement Setup

The measurements in this paper have been carried out in Aug. 2003 over a duration of 296h on a 100Mbps, half duplex FE link connecting the department with the university's campus LAN. The Internet connection of the university is a 155Mbps link to the German Research Network (DFN). The measurements were performed on flow level using TCPdump which was configured to record all TCP flows on the eDonkey client-to-client port '4662'. The flows were classified in an semi-off-line procedure into non-download streams and download flows, which contain at least one of the eDonkey / eMule protocol opcodes '`OP_SENDINGPART`' or '`OP_COMPRESSEDPART`'.

Table 1. General Data on the Investigated eDonkey Data Set

number of observed TCP connections on port '4662'	3431743	
number of local hosts	25	
number of foreign hosts	242067	
total transmitted volume in all flows	$2.95 \cdot 10^{11}$ bytes	
total transmitted volume in download connections	$2.08 \cdot 10^{11}$ bytes	(70.5%)
number of download connections	77111	(2.24%)
number of inbound download connections	21344	(27.7%)
number of outbound download connections	55767	(72.3%)

Since the eDonkey protocol is semi-proprietary, it can't be excluded that the observed non-download flows contain also download traffic. The analysis given below show that a misclassification is quite unlikely. For the rest of the paper we denote a TCP connection as *inbound* if it was initiated by a eDonkey client residing outside the department network. A TCP connection is said to be *outbound* if it was initiated by a client inside the department's LAN.

3.2 Traffic Profile

Table 1 provide general statistic values on the data set of the measurement. In total almost 3.5 million flows have been investigated which were carrying 295 Gbyte of data (non-download and download). Only 2.24% of all connections were download connections. However, they were carrying 705% of the total traffic.

eDonkey Flowsize: The average observed eDonkey flow size during the measurements was 86Kbytes, cf. Table 2. A more detailed inspection shows that the average size of download streams (2.48Mbytes) is two orders of magnitudes larger than the average size of non-download streams (16.7Kbytes). This feature doesn't change much when the direction of the flows is considered, i.e. it doesn't differ for inbound and outbound flows. Figure 2 depicts the complementary cumulative distribution function (CCDF) of the flow sizes. Part (a) and (c) of Figure 2 shows that the download flow size decreases stronger than linear in the log/log plot. That means that the flow sizes don't show a strong "heavy tailed" feature. An approximation of the observed data with a lognormal distribution achieves a good estimate. The reduced strength of the heavy tail feature is not expected, but can be explained: the download flows are limited due to the segmentation of files into chunks and due to the application of the multiple source download principle.

Part (b) and (d) of Figure 2 depicts the size of non-download flows. The probability that a flow is larger than a given value decreases almost exponentially until a limit of approx. 14Kbytes. Beyond this limit, the decrease is not regular. This is an expected behavior since non-download flows are typical signalling flows to renew requests. The above observed features in the flow sizes indicate that the "mice and elephants" phenomenon has not been worsen by eDonkey .

Table 2. eDonkey Flow Statistics

	average	std. deviation
TCP connection interarrival time (all directions)	0.310 sec	0.379 sec
download TCP connection interarrival time (inbound)	49.9 sec	61.4 sec
download TCP connection interarrival time (outbound)	19.1 sec	23.2 sec
non-download TCP connection interarrival time (inbound)	0.830 sec	1.04 sec
non-download TCP connection interarrival time (outbound)	0.515 sec	0.745 sec
flow size (all directions)	86.0 kbytes	5.79 Mbytes
download flow size (inbound)	3.28 Mbytes	15.8 Mbytes
download flow size (outbound)	2.48 Mbytes	5.32 Mbytes
non-download flow size (inbound)	42.3 kbytes	7.17 Mbytes
non-download flow size (outbound)	15.7 kbytes	4.49 Mbytes
TCP connection holding time (all directions)	67.9 sec	265 sec
download TCP connection holding time (inbound)	1010 sec	1460 sec
download TCP connection holding time (outbound)	851 sec	1500 sec
non-download TCP connection holding time (inbound)	47.7 sec	39.2 sec
non-download TCP connection holding time (outbound)	49.7 sec	78.4 sec
plain bandwidth (all directions)	109 bps	23.7 kbps
download plain bandwidth (inbound)	2.77 kbps	5.17 kbps
download plain bandwidth (outbound)	2.41 kbps	2.55 kbps
non-download plain bandwidth (inbound)	44.9 bps	4.61 kbps
non-download plain bandwidth (outbound)	59.9 bps	30.2 kbps
busy bandwidth (all directions)	716 bps	404 kbps
download busy bandwidth (inbound)	3.20 kbps	5.54 kbps
download busy bandwidth (outbound)	2.80 kbps	2.95 kbps
non-download busy bandwidth (inbound)	322 bps	4.75 kbps
non-download busy bandwidth (outbound)	878 bps	520 kbps

TCP Holding Time: The average eDonkey connection holding time on TCP level is 67.9 sec, cf. Table 2. As for the flow sizes, there is a significant difference between download and non-load flows. The mean duration of download connections is 851sec. This more than one orders of magnitudes longer than the duration of non-download streams, which is 47sec. However, the standard deviation of the flow duration is much larger for download flows than for non-download streams.

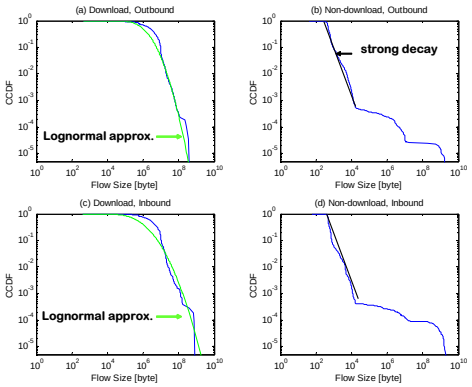

Fig. 2. CCDF of the observed eDonkey Flow Size

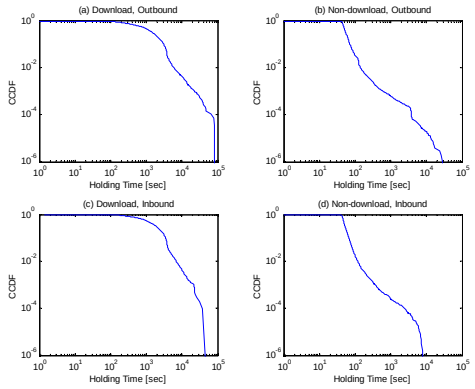

Fig. 3. CCDF of the observed eDonkey Flow Holding Time

This is an expected behavior since non-download connections are short and limited in their sensitivity on TCP flow control. Figure 3 depicts the CCDF of the flow holding times on TCP level. The download connection holding time CCDFs (part (a) and (c)) decrease moderately and reminds more to a linear decay in a log/log plot. The CCDFs of the holding time of non-download stream (part (b) and (d)) decrease rapidly as well as un-regularly. There is only little difference in the non-download case for in the different directions.

Correlation of eDonkey TCP Flow Holding Time and TCP Holding Time: The Figure 4 depicts a scatter plot describing graphically the correlation of the TCP holding time and the size of eDonkey flows. Each dot in the scat-

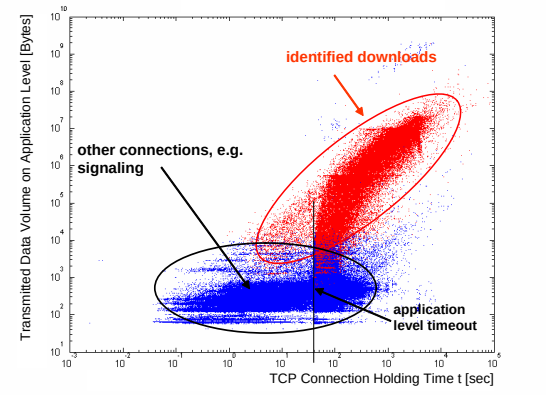

Fig. 4. Correlation of eDonkey TCP holding time and Flow Size

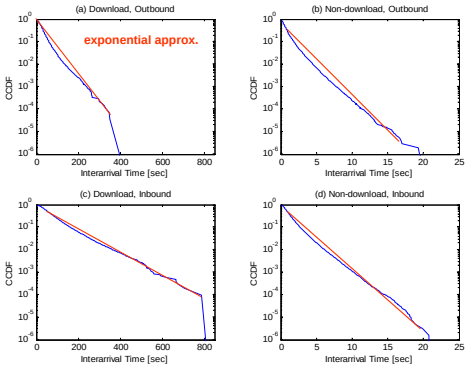

Fig. 5. eDonkey Flow Interarrival Time

ter plot represents an observed eDonkey flow. The brighter dots are identified download flows, the dark dots represent non-download connections.

The scatter plot shows that almost all identified download flows are within the same region. The overlap of both region is small and therefore the probability of a misclassification is low. This feature enables the possibility to classify download streams by their size and holding time instead of using computationally demanding pattern recognition of protocol opcodes. Furthermore, the application level timeout of 40sec is clearly visible.

eDonkey Flow Interarrival Time: The average flow interarrival time was 0.310sec, cf. Table 2. There is again a significant difference for download flows and non-download streams since download flow are more rarely. The average inter-arrival time of download flows is two orders of magnitudes higher than the one of non-download flows. The CCDFs of the eDonkey flow interarrival time reveals an exponential decay, cf. Figure 5. This is consistent with resent eDonkey measurements [4]. The high frequency of non-download flows, in general, reduces the throughput on links [7].

Average Bandwidth on Application-Level: The average bandwidth of the eDonkey connections was also investigated. In the context of the herein presented measurements, the average bandwidth is defined as the ratio between the amount of transmitted data on application-level and the TCP connection holding time. This bandwidth is widely used and denoted in this paper as the *average plain bandwidth*. The analysis of the eDonkey flows, however, showed that a large number of connections have a significant idle period before the TCP connection is terminated by a FIN or a RST packet. From network point of view, the bandwidth is only experienced during data transmission. Therefore, the idle time is subtracted from the flow duration. The shorter times are used

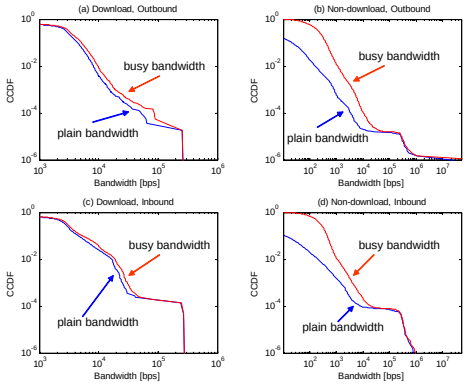

Fig. 6. eDonkey Average Bandwidth

for the calculation of the *average busy bandwidth*. We now compare the observed statistical values and distributions for both bandwidth definitions.

The average plain bandwidth for all eDonkey flows is 109bps, cf. Table 2. This value is very low and strongly influenced by idle period at the end of the TCP flows as the average net bandwidth of 716bps shows. The average plain bandwidth of download flows is typically two orders of magnitude higher than the plain bandwidth of non-download streams. The CCDF of the average plain bandwidth, lower curve in Figure 6, shows strong decay for download flows (part (a) and (c)) and a moderate decay for non-download flows (part (b) and (d)). The CCDF for the average busy bandwidth has a similar behavior however the decay is delayed and even stronger (part (b) and (d)). The comparisons shows that the influence of the idle time is less stronger for the average bandwidth of download flows as for the bandwidth of non-download streams. All features suggest to use the average busy bandwidth instead of the simple approach of the average plain bandwidth.

Origin and Destination of eDonkey Traffic: Finally, the origin and the destination of the observed eDonkey flows were investigated. The IP addresses of the foreign hosts were mapped to the Autonomous Systems (AS) which take care of these addresses. All traffic and connections for an AS were combined in Top7 lists. The Top7 list for the traffic amount, cf. Table 3, is dominated by the traffic originating or terminating in the AS of the German Telecom (DTAG). This characteristic of eDonkey indicates that majority of the observed traffic is locally and not world-wide distributed. Table 4 underlines this result for the number of established connections. A large number of established connections, however, does not necessarily mean a high traffic volume. This feature is caused by the eDonkey protocol requirement to renew download requests.

Table 3. eDonkey TOP 7 Autonomous Systems in Traffic Volume

Owner	Country	AS num.	total bytes	bytes download		bytes non-download	
				outbound	inbound	outbound	inbound
DTAG	.de	AS3320	50258 MB	15890 MB	4798 MB	9231 MB	20331 MB
Polish Tel.	.pl	AS5617	22703 MB	1761 MB	574 MB	9590 MB	10777 MB
France Tel.	.fr	AS3215	10527 MB	2353 MB	811 MB	2910 MB	4452 MB
BTnet UK	.uk	AS2856	8992 MB	1299 MB	720 MB	3197 MB	3776 MB
Verizon	.us	AS19262	6395 MB	0.877 MB	0.001 MB	3196 MB	3197 MB
Arcor IP	.de	AS3209	5579 MB	1133 MB	415 MB	1656 MB	2373 MB
NTL Grp. Ltd	.uk	AS5089	5224 MB	1055 MB	322 MB	1557 MB	2289 MB

4 Related Work

The traffic profile for the eDonkey service presented in this paper is a first step towards a more detailed behavior model and traffic model for the service. In general behavior model and traffic model of P2P services can classified into three main categories: models for multiple peer characteristics, models for the content or the shared resources, and models for the individual peer behavior.

The models for multiple peer characteristics comprise characterizations for the P2P overlay network topology, e.g. the Power-Law feature in degree distribution of unstructured P2P networks [11], for the P2P overlay variability [12], and for wide-area network traffic pattern of P2P services [13]. The characterization of the content comprises models for the popularity of files and providing peers [14] and file size [15]. The individual peer behavior can be characterized by state models describing the idle, down, active, searching, or responding state of the peer [16]. A comprehensive traffic model for the Kazaa P2P file sharing service was investigated in [17].

5 Conclusion

In this paper we have presented a traffic profile for the eDonkey P2P filesharing service. The measurement-based study revealed a strong distinction between download flows and non-download stream. Both types of flows have to be considered differently. A single model for P2P flows, as provided in a first analysis in [13], would lead to a significant mischaracterization of the P2P traffic. It turned

Table 4. eDonkey Top 7 Autonomous Systems in Connections

Owner	Country	AS num.	total conn.	num. conn. download		num. conn. non-download	
				outbound	inbound	outbound	inbound
DTAG	.de	AS3320	2114910	8518	12775	1048937	1044680
TDC	.dk	AS3292	207390	860	801	102835	102894
Arcor IP	.de	AS3209	178412	587	850	88619	88356
AOL Transit	.us	AS1668	176404	845	1342	87357	86860
France Tel.	.fr	AS3215	153900	1224	2415	75726	74535
TDE	.es	AS3352	140402	1145	1417	69056	68784
Polish Tel.	.pl	AS5617	131750	1076	828	64799	65047

out that the traffic caused by eDonkey doesn't seem to worsen the "mice and elephants" phenomenon. However, a more detailed investigation is still necessary. In a next step we will define a detailed traffic model for eDonkey flows. The observed origins and destinations of eDonkey flows indicates that it is favorable for network operators trying to keep the traffic within their AS.

Acknowledgement: The author wants to thank M. Brotzeller for carrying out the measurements and P. Tran-Gia for supporting this research.

References

1. Meta Search Inc.- eDonkey2000 Home Page: (http://www.edonkey2000.com/)
2. eMule Project Team Web Site: (http://www.emule-project.net/)
3. mlDonkey Web Site: (http://mldonkey.org/)
4. Azzouna, N., Guillemin, F.: Analysis of ADSL traffic on an IP backbone link. (In: GLOBECOM 2003, San Francisco, California, Dec. 2003.)
5. Paxson, V., Floyd, S.: The failure of the Poisson assumption. IEEE/ACM Trans. on Networking (1995) 226–244
6. Bhattacharyya, S., Diot, C., Jetcheva, J., Taft, N.: Pop-level and access-link-level traffic dynamics in a tier-1 pop. (In: 1nd Internet Measurement Workshop, San Francisco, USA, Nov. 2001.)
7. Boyer, J., Guillemin, F., Robert, P., Zwart, B.: Heavy tailed M/G/1-PS queues with impatience and admission control in packet networks. (In: Proceedings of INFOCOM 2003, San Francisco, USA, April/March 2003.)
8. Lohoff, F.: Lowlevel documentation of the eDonkey protocol: (http://silicon-verl.de/home/flo/software/donkey/)
9. Free Peers Inc. - Bearshare: (http://www.bearshare.com/)
10. Sharman Networks - KaZaA Media Desktop: (http://www.kazaa.com/)
11. Ripeanu, M., Foster, I.: Mapping gnutella network. (In: 1st International Workshop on Peer-to-Peer Systems (IPTPS02), Cambridge, Massachusetts, March 2002.)
12. de Meer, H., Tuschku, K., Tran-Gia, P.: Dynamic Operation of Peer-to-Peer Overlay Networks. Praxis der Informationsverarbeitung und Kommunikation **26** (2003) 65–73
13. Sen, S., Wong, J.: Analyzing peer-to-peer traffic across large networks. (In: 2nd Internet Measurement Workshop, Marseille, France, Nov. 2002.)
14. Adar, E., Huberman, B.A.: Free riding on gnutella. Research report, Xerox Palo Alto Research Center (2000)
15. Saroiu, S., Gummadi, P., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of Multimedia Computing and Networking 2002 (MMCN '02), San Jose, CA, USA (2002)
16. Schlosser, M., Condie, T., Kamvar, S.: Simulating a p2p file-sharing network. (In: 1st Workshop on Semantics in Peer-to-Peer and Grid Computing, Budapest, Hungary, May 2003.)
17. Gummadi, K., Dunn, R., Saroiu, S., Gribble, S., Levy, H., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. (In: Proceedings of 19th ACM Symposium on Operating Systems Principles, Bolton Landing (Lake George), USA, Oct. 2003.)