The eDonkey File-Sharing Network

Oliver Heckmann, Axel Bock, Andreas Mauthe, Ralf Steinmetz

Multimedia Kommunikation (KOM)
Technische Universität Darmstadt
Merckstr. 25, 64293 Darmstadt
(heckmann, bock, mauthe, steinmetz)@kom.tu-darmstadt.de

Abstract:
The eDonkey 2000 file-sharing network is one of the most successful peer-to-peer file-sharing applications, especially in Germany. The network itself is a hybrid peer-to-peer network with client applications running on the end-system that are connected to a distributed network of dedicated servers. In this paper we describe the eDonkey protocol and measurement results on network/transport layer and application layer that were made with the client software and with an open-source eDonkey server we extended for these measurements.

1 Motivation and Introduction

Most of the traffic in the network of access and backbone Internet service providers (ISPs) is generated by peer-to-peer (P2P) file-sharing applications [San03]. These applications are typically bandwidth greedy and generate more long-lived TCP flows than the WWW traffic that was dominating the Internet traffic before the P2P applications. To understand the influence of these applications and the characteristics of the traffic they produce and their impact on network design, capacity expansion, traffic engineering and shaping, it is important to empirically analyse the dominant file-sharing applications.

The eDonkey file-sharing protocol is one of these file-sharing protocols. It is implemented by the original eDonkey2000 client [eDonkey] and additionally by some open-source clients like mldonkey [mlDonkey] and eMule [eMule]. According to [San03] it is with 52% of the generated file-sharing traffic the most successful P2P file-sharing network in Germany, even more successful than the FastTrack protocol used by the P2P client KaZaa [KaZaa] that comes to 44% of the traffic. Contrary to other P2P file-sharing applications like Gnutella that are widely discussed in literature the eDonkey protocol is not well analysed yet. In this paper, we shed light on this protocol with a measurement study.

In the next section we describe the protocol itself. In Section 3 our measurements are presented. They were made with the eMule client software and with an open-source eDonkey server we extended for the purpose of this paper. The results of the measurements are presented in Section 4 before we give a summary and draw the conclusions.

2 Protocol Description

The eDonkey network is a decentral hybrid peer-to-peer file-sharing network with client applications running on the end-system that are connected to a distributed network of dedicated servers.

Contrary to the original Gnutella protocol it is not completely decentral as it uses servers; contrary to the original Napster protocol it does not use a single server (farm) which is a single point of failure, instead it uses servers that are run by power users and offers mechanisms for inter-server communication. Unlike super-peer protocols like KaZaa, or the modern Gnutella protocol the eDonkey network has a dedicated client/server based structure. The servers are slightly similar to the KaZaa super-nodes, but they are a separate application and do not share any files, only manage the information distribution and work as several central dictionaries which hold the information about the shared files and their respective client locations.

In the eDonkey network the clients are the only nodes sharing data. Their files are indexed by the servers. If a client wants to download a file or a part of a file, it first has to connect via TCP to a server or send a short search request via UDP to one or more servers to get the necessary information about other clients offering that file.

The eDonkey network is using 16 byte MD4 hashes to (with very high probability) uniquely identify a file independent of its filename. The implication for searching is that two steps are necessary before a file can be downloaded in the eDonkey network. First, a full text search is made at a server for the filename, it is answered with those file hashes that have a filename associated which matches the full text search. In a second step, the client requests the sources from the server for a certain file-hash. Finally, the client connects to some of these sources to download the file.

A more detailed protocol analysis based on measurements with TCPDump can be found in our technical report [HB02].

3 Experiment Description

We ran two types of experiments to analyse the eDonkey protocol and to collect traffic samples:

In the first experiment we measured the traffic at a client connected to 768/128 kbps ADSL connection rsp. a 10 MBit at the TU Darmstadt. The most popular client of the network eMule [eMule] under Windows XP was used for this experiment. From the measurements we derive traffic characteristics like the protocol overhead, downloading behaviour and message size distributions.

For the second experiment we modified an open-source eDonkey server [Lus03] that was connected to the university network collecting application level information about the usage of the network, requested files, offered files and the overall size of the network[1].

4 Results

TCP/UDP summary. In the experiments with the client the share of the TCP traffic of the overall network traffic was 94% (number of packets) rsp. 99.8% (payload size). Those values do not differ significantly between ADSL and broadband connections. Note that all TCP ACK packets with zero payload were counted, too. On the ADSL line the packet loss was with 5.5% about 2.25 times higher than on the broadband line (2%).

The server-side measurements differ: TCP traffic only forms about 2,4% of the packets and 6% of the payload. The server is mainly busy with handling UDP requests from clients *not* directly connected: If a search of a client does not deliver enough results from the server it is connected to all known servers can be searched. This behaviour has implications on dial-up connections. If the dial-up IP address was assigned to someone running an eDonkey server before, many clients all over the world still reference to that IP address as a server. Server entries are typically kept 1 day or longer in most client implementations. These clients will still send UDP based search queries to that IP address consuming bandwidth. This even lead to a permanent entry in the FAQ section of the computer magazine c't [ct04].

The TCP packet sizes use the whole size spectrum from header only to MTU size. Some characteristic peaks can be identified, e.g., for TCP messages of payload size 24, they are typically used for the frequent *QUERY SOURCES* messages. In most cases a TCP packet carries only a single protocol message. Protocol messages can be identified quite reliably as the payload starts with "E3", for details see [HB02]. The UDP packets are much less variable in size, the size clearly indicating which kind of message is transported.

Protocol Messages. Looking at the client's UDP traffic the most observed protocol message by far is *QUERY SOURCES* with about 65%. Looking at TCP messages they are distributed far more evenly, all common messages have a percentual share between one and ten percent, while - surprisingly - the TCP *QUERY SOURCES* message has only about 0.008%.

On the server side the things the most seen messages are *QUERY SOURCES* ones, with 36% (TCP) and 95% (UDP) occurrence.

Throughput. On the eMule/ADSL and eMule/broadband clients we observed an average overall throughput of 30 respectively 45 kB/s while downloading, which decreased to the preset maximum upload bandwidth (10 kB/s in our case) after downloads were finished. UDP traffic is not very important in this scenario.

[1]The server proved to be quite popular and due to the sheer amount of flows was ranked on the second place of the traffic statistic of the whole TU Darmstadt network for the duration of the experiment.

On the server, the UDP in/out ratio was 16/2 kB/s after roughly a day (so we have 18 kB/s overall throughput), while TCP is about 0.75/0.25 kB/s after the same time. In the beginning though TCP throughput is 1.5/1 kB/s and decreases significantly after the server gets more known and UDP throughput increases massively thus suppressing the TCP communication.

Connection Statistics. We observed about 100 (eMule broadband), 85 (eMule ADSL) and 150 (server) connection requests per minute. The share of connections actually used for data exchange is 77%, 74% and 72%. The number of simultaneous connections is 30 to 50 for eMule/broadband, 30 to 45 for eMule/ADSL and quite exactly 700 for the server (gathered by TCP trace file analysis).

For getting the average bandwidth use per connection we looked at all connections carrying more than 0.5 MB incoming payload. Of those the absolute majority (several hundred) utilises roughly two to four kB/s, with some more (about 40) ranging from 5 to 10 kB/s. About ten more connections used bandwidths up to 55 KB/sec. Those values are connection independent.

Our measurements show that an average of roughly four megabytes of data is transferred per (TCP) connection. The maximum size transferred we encountered was 150 MBytes. The average TCP connection time is 30 minutes, the average idle time 875 seconds (ADSL) and 2500 seconds (broadband).

User Statistics. We concentrated on getting information about how much the users share in size and file numbers, and what they search most (indicated by search terms and identified files). Our research showed that the average eDonkey user shares 57.8 (max. 470) files with an average size of 217 MB (max 1.2 TByte).

Most searched-for keywords were media-related words like ”MP3”, ”AVI” and ”DVD” and a very high share of clearly recognisable words from current blockbuster movie titles.

Because all files are identified by their unique hash values we were able to identify the most wanted files by analysing *QUERY SOURCES* messages. The corresponding filenames were extracted from the *PUBLISH FILES* messages. This showed that the vast majority of the identified files were movies just played in the cinemas at the time of the experiments.

Looking at these results we can state that eDonkey is mainly a movie network.

Geographical Analysis and Network Size. From all clients connected to the broadband test client the IP addresses were reverse resolved to lookup their top-level domains for a rough geographical overview. `.de` dominates easily with 66.21%, followed by the ”dot-net”-Domains (10%), then `.fr` (6%) and `.at` (1%), while clients from Taiwan and Guatemala were both seen more times than clients with `.uk`. While it is probable that clients from a certain region tend towards higher interconnectivity among themselves, e.g. because they exchange movies in their language this is also an indication that the eDonkey network is more popular in Germany than in other countries. The latter is supported by the study in [San03].

To estimate the network size we monitored the number of servers in a server-list provided by ”ocbmaurice” [Ocb04] . This showed that the average size shrinked down from roughly 220 servers in beginning of 2002, over 100 servers in 2003 down to 47 servers in 2004. This indicates that the eDonkey network as a whole is shrinking. One reason is probably that since the end of 2002 some people operating an eDonkey server had trouble with the police[2].

5 Summary and Conclusion

In this paper we presented results from two measurement experiments in the eDonkey network. They show that the eDonkey network is primarily used for downloading current movie blockbusters. Most of the traffic is caused by long-lived TCP connections. A P2P traffic model will therefore be fundamentally different to a web traffic traffic model.

References

[San03] Sandvine Incorporated. *Regional Characteristics of P2P - File Sharing as a Multi-Application, Multinational Phenomenon*. White Paper, 2003.

[eDonkey] eDonkey2000 Original Client Homepage. http://www.edonkey2000.com.

[eMule] eMule Project Homepage. http://www.emule-project.net.

[mlDonkey] mlDonkey Homepage. http://savannah.nongnu.org/projects/mldonkey/.

[KaZaa] Sharman Networks. *The KaZaa application for the FastTrack network*. Official Homepage. http://www.kazaa.com.

[Kli02] Alexey Klimkin. *pDonkey, an eDonkey Protocol Library*. http://pdonkey.sourceforge.net.

[Lug04] ”Lugdunum”. *Building an Efficient eDonkey Server on Linux/FreeBSD/Win32*. http://lugdunum2k.free.fr/kiten.html 2004.

[Lus03] Thomas Lussnig. *cDonkey, an Open-Source eDonkey Server with Overnet and eMule Extensions*. http://cdonkey.suche.org. 2003.

[Mul02] Tim-Philip Muller. *eDonkey2000 tools*. http://ed2k-tools.sourceforge.net.

[Ocb04] ”Ocbmaurice”. *eDonkey2000 Server-lists and Statistics*. http://ocbmaurice.dyndns.org 2004.

[HB02] Oliver Heckmann, Axel Bock. *The eDonkey2000 protocol*. KOM technical report. http://www.kom.e-technik.tu-darmstadt.de/publications/abstracts/HB02-1.html

[ct04] c’t - Magazin für computer technik. *FAQ Section Entry - ”Nervige Port Scans” (”Annoying Port Scans”)*. Heise Verlag. http://www.heise.de/ct/faq/qna/nervige-port-scans.shtml

[2]For example the operator of the Danish site http://www.siffan.dk