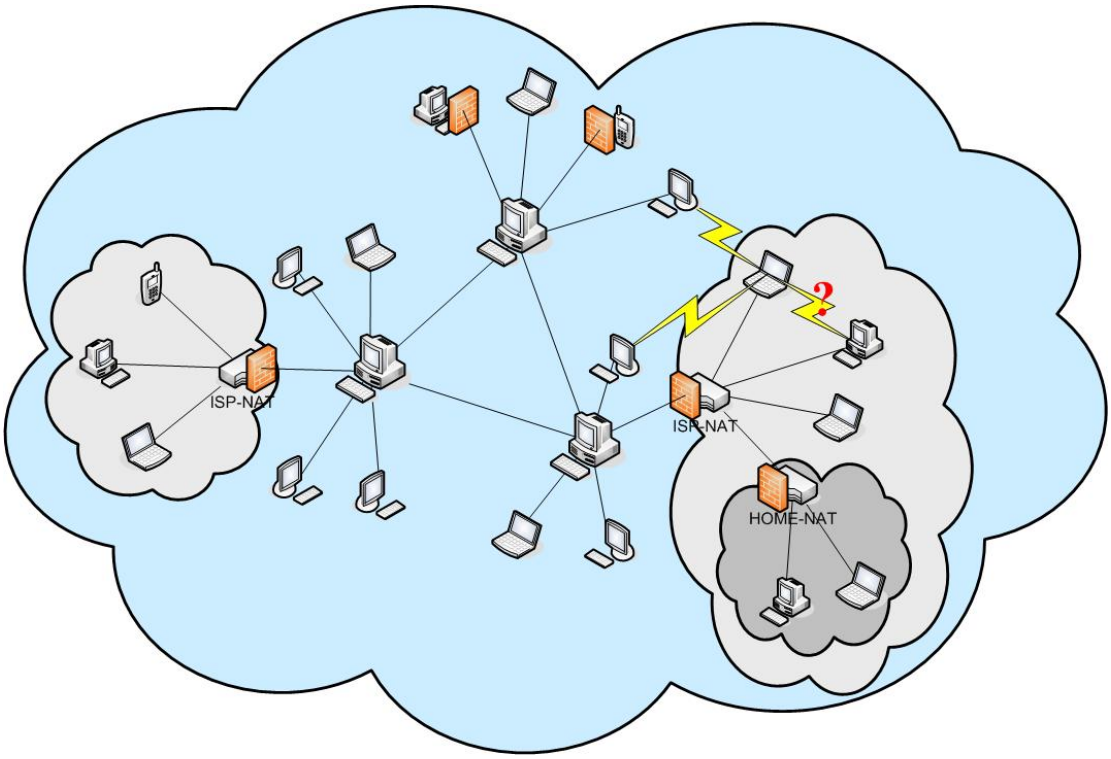
Peer-to-Peer Analysis

State-of-the-art



Tampere University of Technology
Alex Jantunen
Sami Peltotalo
Jani Peltotalo

27.2.2006
Version 1.0

Table of Contents

1 Introduction

There are lots of academic peer-to-peer (P2P) papers available. They often cover certain areas of the peer-to-peer technologies. Because peer-to-peer technologies are also developing very fast, the papers are often out-of-date. The goal of this paper is to provide up-to-date information in all interested aspects related to peer-to-peer delivery and our Delivery and Control Techniques in Content Networking (DelCo) project. We are focused on the most popular P2P networks: FastTrack, Gnutella, eDonkey2000, Overnet, and BitTorrent. Our research also covers some enhancements that some clients implement to improve its basic peer-to-peer protocol. We don't go too deep into details that are already discovered, we rather give a link for more detailed information about the subject when necessary.

2 Peer-to-Peer Networks

There are two classes of peer-to-peer overlay networks: Structured and Unstructured. The technical meaning of Structured is that the P2P overlay network topology is tightly controlled and content are placed not at random peers (nodes) but at specified locations that will make subsequent queries more efficient. Such Structured P2P systems use the Distributed Hash Table (DHT) as a substrate, in which data object (or value) location information is placed deterministically, at the peers with identifiers corresponding to the data object's unique key. [LCP+04]

In the unstructured category, the overlay networks organize peers in a random graph in flat or hierarchical manners (e.g. Super-Peers layer) and use flooding or random walks or expanding-ring Time-To-Live (TTL) search, etc. on the graph to query content stored by overlay peers. Each peers visited, will evaluate the query locally on its own content, and will support complex queries. [LCP+04]

Unstructured networks are more robust in the face of transients and support general search facilities, both important properties to P2P file-sharing. They are less adept than DHTs at finding needles, but this may not matter since most P2P queries are for hay. Thus for mass-market file-sharing applications, improving the scalability of unstructured P2P systems, rather than turning to DHT-based systems, may be the better approach. [CRB+03]

We have chosen five P2P networks that are under interest in DelCo project, they are FastTrack, Gnutella, eDonkey2000, Overnet, and BitTorrent. These are only fragmented set of P2P networks, but they are the most popular ones. All except the Overnet are unstructured networks. Overnet has been taken into account for the reason that it is a good example of a sophisticated structured network. Overnet implements Kademlia algorithm [MaMa]. Main task of these file-sharing overlay networks are to find peers having desired content. File transfer takes place out-of-band, directly between a peer wanting content and peer(s) having that.

2.1 FastTrack

FastTrack is a semi-centralized network, where nodes are classified as ordinary nodes and supernodes. Supernode is a higher level node with more responsibilities than an ordinary node. In Gnutella network the same higher level node is called ultrapeer. Supernodes act as temporary index servers. Any node with sufficient CPU and network connection can become a supernode, and election is done without centralized control. FastTrack is a proprietary protocol, but attempts at cracking the FastTrack protocol have been made but has failed to break the encryption between supernodes.

The most popular FastTrack client is KaZaA. Grokster and iMesh are two other clients that participate in the FastTrack overlay network. All the three clients are licensed by Sharman Networks, Inc and use the same protocol as KaZaA. Other clients are KLT K++ (Kazaa Lite Tools K++) and MLDonkey. KLT K++ is a modified program of KaZaA without extra software (spyware), but with some improvements and plug-ins. Other than licensed clients don't support supernode mechanism as long as communication between supernodes remains unresolved.

Some clients have modified participation level at the maximum value for better download performance. This makes the original incentive mechanism useless. Overall download performance is good with multi-source downloading, but system doesn't support sharing partial files. You have to download a file completely before you can upload it. Polluting the network has been quite easy because of UUHash algorithm, which does incomplete file hashes. Today new kzhash should solve the problem, but there is no reliable confirmation about that. [LKR04] [Slyck1]

The network has a low amount of legal content, and includes very much fake files. KaZaA software is also having legal challenges in the Australia. Future of the FastTrack network existence is not clear. Popularity of a network can be measured on the ground of user volume, but some of those don't have uniform overlay network and the user volume can't be calculated, which is the case with BitTorrent. User volume has been fluctuated quite much between 2 and 3 million users. It is second (excluding BitTorrent) in the popularity with 2.6 million users according to Slyck (27.2.2006) [Slyck]. With KaZaA you can download also high quality files bought to you from professional content creators via Altnet (http://www.altnet.com). These files are digitally rights managed and are typically offered for use either on a free basis, or on a free-trial basis.

2.2 Gnutella

At first Gnutella was a decentralized protocol for distributed search on a flat topology of peers [Gnut04]. Because the search mechanisms were not scalable and generated unexpected loads on the network, Gnutella has developed to the semi-centralized network implementing FastTrack-like overlay network. Gnutella like FastTrack doesn't have any centralized control point. In Gnutella network nodes are classified as leaf nodes and higher level nodes as ultrapeers, which are high capacity nodes that act as proxies for lower capacity nodes. [Gnut06] [LCP+04]

Most popular Gnutella clients are LimeWire and Shareaza, both are open source software. LimeWire supports sharing partial files with the partial file sharing option checked, but Gnutella protocol itself doesn't support directly that feature. Like FastTrack there is no mechanism during download to prevent corrupted pieces of a file ruining the whole file. Complete file hashes with Gnutella (SHA-1) can be used for file verification, but only after the download is fully completed. Gnutella has grown to one of the biggest P2P networks and Gnutella is the third biggest with 2.1 million users (22.2.2006) [Slyck]. [LimeWire]

2.3 eDonkey2000

eDonkey2000 (ED2K) is a semi-centralized network developed by MetaMachine. There are loosely connected, separate index-servers, but there is no single centralized server. Unlike Napster who kept their central servers in Silicon Valley, MetaMachine released their server software into the wild. Users are encouraged to run the server and it's up to the community to maintain the network. With Multisource File Transmission Protocol (MFTP) eDonkey2000 supports simultaneous downloading from multiple sources and sharing partially downloaded files, so that a peer can upload a file while still downloading it. File hashes are used to identify files on the network. With ed2k links you can give file hash and optionally hash set (part hashes). The complete hash set ensures that blocks of a file are always correct and helps spreading new and rare files. Corrupted blocks of a file during download won't ruin the whole download. [eMule] [ED2K] [HeBo02]

ED2K has two popular client software, eDonkey2000 and eMule. There are also other clients like Shareaza and Mldonkey. eMule is an open source project and the most popular client (about 80% of the eDonkey2000 network users, although figures are changing all the time). However eDonkey2000 client is original, but it's not open source and comes with adware. There are only a few publicly available server programs, the most popular one is Lugdunum.

There are some misunderstandings between the eDonkey2000 and eMule developers. They use their own techniques to enhance the basic eDonkey2000 file-sharing protocol. In addition to the eDonkey2000 network, eDonkey2000 client can also connect to Overnet and eMule to Kad. Overnet and Kad both are based on Kademlia concept [MaMa], unfortunately they are not interoperable. Clients also have different incentive mechanisms in addition to the original score and queue system, eDonkey2000 uses proprietary Horde system and eMule its own credit system. eMule uses also enhanced corrupt handling. There are signs of separate islands on the eDonkey2000 network because of the differences in the clients; a peer favors same client software more. However, eDonkey2000 is the most popular P2P network with 3.5 million users (22.2.2006) [Slyck]. [KuBi05] [ED2K]

2.4 Overnet

Overnet is a fully decentralized network based on Kademlia, a general peer-to-peer routing algorithm that can be used to implement distributed hash tables (DHT). Each peer on Overnet gets a NodeID from the 128-bit key space. Key, value pairs are stored on peers with IDs close to the key, closeness is defined by the XOR-metric. A

NodeID-based routing algorithm will be used to locate peers near a destination key. [KuFu05] [MaMa] [LCP+04]

Original Overnet client is merged to the eDonkey2000 client, but it is possible to use only Overnet network. Overnet was created by MetaMachine to overcome eDonkey2000 scalability problems. As Overnet use MFTP and it is implemented into the same client as MetaMachine's ED2K. Overnet and ED2K have many similar characteristics regarding issues in downloading and sharing. However things related to network topology are very different. Overnet and Kad, both based on Kademlia, are not interoperable. Overnet has almost one million peers (31.1.2006) [Slyck].

2.5 BitTorrent

BitTorrent is a P2P system that uses a central location to manage users' downloads. The central location is a tracker that is contacted when you launch a torrent for file downloading. The tracker keeps track of all the users who have the file (both partially and completely) and connects users to each other for downloading and uploading. [Co03]

BitTorrent supports simultaneous downloading from multiple sources and sharing partially downloaded files, so that a peer can upload a file while still downloading it. "Info hashes" are used to identify files on the network. A torrent file includes a list of piece (block) hashes, which ensures that blocks of the file are always correct and corrupted blocks during download won't ruin the whole download. Corrupted blocks must be re-downloaded.

Today most BitTorrent clients support also trackerless torrents. There is no need for a central tracker with that approach. Trackerless support is done with the help of DHT, which is a layer added on the top of the BitTorrent network. The BitTorrent network and DHT portion operate independently. Each node in the DHT is responsible for indexing a certain percentage of hash files on the network. The first BitTorrent client to establish a DHT layer was Azureus, followed by the official BitTorrent client. Although both DHTs are based on Kademlia, the two DHT networks are not compatible. The DHT layer supported by the official client would be known as Mainline DHT network. Azureus client reports 500 000 – 700 000 users on its DHT network (15.12.2006). [Slyck3] [BitTo]

BitTorrent has many clients, for example the original BitTorrent client, which is open source and written in Python, and BitTornado, which is based on the original client. BitComet is another enhanced client, but is closed source. Azureus is a client with lots of features using Java language. It has also an embedded tracker with password protection, HTTPS and UDP communication support. The network is used for legal content delivery also, for example Linux distributions and game patches. BitTorrent is a download protocol where peers are connected together by the torrent bases (except DHT), so popularity of overlay network is irrelevant. But BitTorrent is clearly the most popular regarding download rates; ISPs report very high rates of bandwidth consumption because of BitTorrent usage. The biggest BitTorrent tracker, The Pirate Bay (http://thepiratebay.org), has almost two million connected peers (23.1.2006).

3 Characteristics of P2P Systems

A comparison of popular P2P systems is done in [LCP+04], which covers also the P2P networks presented in the previous chapter. Table 1 shows some characteristics of popular P2P networks. Scalability, availability, bootstrapping, download performance, flash crowd, pollution and content lifetime are covered more detailed in the following chapters. Values in the strong points category mean that P2P network is good within that area (e.g. pollution in the strong points mean P2P system is strong against pollution). Explanations for the selections are covered in the following chapters. There is also discussion about location awareness, which is currently more or less client software dependent and thus we were incapable to evaluate that.

P2P system	Strong points	Weak points
FastTrack	availability, scalability, content lifetime	pollution
Gnutella	availability, scalability, content lifetime	pollution
ED2K	content lifetime, pollution	scalability
Overnet	availability, scalability, content lifetime, pollution	bootstrapping
BitTorrent (without DHT)	download performance, flash crowd, pollution	availability, scalability, content lifetime

Table 1: A comparison of the five popular P2P systems by means of six characteristics.

Measurements study and analysis of the networks can be found here:

- [LKR04] presents a measurement study of KaZaA overlay network. [GDS+03] presents a measurement, modeling and analysis of KaZaA file-sharing traffic. [LRW03] presents an investigation (measurement) of KaZaA file-sharing traffic.

- [Tut04] presents a measurement-based traffic profile for the eDonkey2000 file-sharing service.

- A measurement-based study of Overnet is presented in [KuFu05]. A measurement study of Overnet availability is covered in [BSV03].

- [PGES04], [PGES05], [SGP04] and [IUB+04] present a measurement-based study of BitTorrent file-sharing system. Modeling and mathematical analysis of BitTorrent is done in [QiSr04].

3.1 Scalability

Here we are talking about load scalability and how well different P2P system can cope with it. When a peer joins to the network it shouldn’t pose unmanageable burden

to the overlay network and its operation. Rather it should provide more resources into the system e.g. in form of upload bandwidth, processing power or file capacity. Scalability is often achieved with decentralized distribution of the key components among participating peers on the network. However often there is a limit when system can't grow anymore and its functionality starts to drop. This can be because of limited network bandwidth or processing power of some important entity, or there is too many control messages generated on the network that can't be processed in time.

In general swarming (multi-source downloading) improves the scalability [SZR05]. Swarming supporting protocols are at least FastTrack, Gnutella, ED2K/Overnet and BitTorrent. BitTorrent, ED2K/Overnet and some Gnutella clients (e.g. LimeWire) support sharing partially downloaded files, which improves swarming and thus scalability. Location awareness also helps scalability in a way that links in the system are less stressed.

FastTrack and Gnutella (v06) networks have the concept of supernode or ultrapeer to improve scalability. Although this approach seems to offer good scalability, its design has not been analyzed. As long as there are enough higher level peers and edge peers are distributed fairly equally between higher level peers, there shouldn't be any major bottlenecks in sight. Of course at some point communication between higher level peers becomes a problem as the amount of them increases. On the other hand Gnutella and FastTrack clients can refuse to be a supernode or ultrapeer. This may hinder scalability if higher level peers don't exist enough.

eDonkey2000 can have many static servers to improve scalability, but they are loosely connected together and there is no mechanism to dynamically cope with large amount of clients. Servers, with more clients, find more peers for the queries. Because user can decide which server to connect to, some servers become overloaded. Normal user has also a much bigger step to run an eDonkey2000 index server than becoming a supernode/ultrapeer. Overnet was designed to solve eDonkey2000 scalability problems, but that hasn't yet proved. Good thing is that there are no issues of overloading central points as the network is fully decentralized. Still at some point queries and maintenance of the topology could become a problem with Overnet.

With BitTorrent (without DHT) simultaneous downloads, that a tracker can cope with is limited, unless decentralized with tracker farms. Network bandwidth and processing power pose big bottlenecks for the tracker. UDP trackers [UDPtrack] can serve more, but it doesn't remove the limit that one central tracker can serve. It is unknown how well new trackerless design scale. Each node in the DHT is responsible for indexing a certain percentage of hash files on the network. It would be interesting to know can the system adapt to the high interest of certain (trackerless) torrent.

[Zhe04] says that *"A major drawback of existing large scale content distribution systems is the directory service, which generally consists of an index server and a tracker server. The index server (e.g., a web server) hosting all the metadata of shared content. A user will have to contact the index server to search for specific content and retrieve the metadata of the interested file. After that the user launches the client download software to connect to a tracker server in order to get a list of peers who are downloading the same file. In effect, such a directory service does not*

scale well as it cannot accommodate a large number of requests when the population of the system increases rapidly."

To address the above mentioned problem the research community has developed the following ideas. Xeja [Zhe04]/SODON [ZhWa04] improve scalability of the system by eliminating the index server and tracker server from the system. Xeja/SODON is an effort to address the tracker server problem and piece-seeding problem in large-scale multi-source file download systems. Slurpie [SBB04] clients improve performance as the size of the network increases, and the server is completely insulated from large flash crowds entering the Slurpie network.

3.2 Availability

Availability and functionality of the key components, which are e.g. searching, downloading or indexing, is very important for the P2P networks. Responsibilities of the key components are often distributed among many peers to provide better availability. Availability of individual peer is also important, because they can provide more resources on the network. Users are often encouraged for being available on the network (e.g. giving highID/lowID when peer is reachable or better downloading speed when uploading more). NAT Traversal also has an effect on peer availability. Sometimes peer just can't be reachable because network doesn't have sufficient NAT Traversal support. Availability has a remarkable effect on the popularity of the P2P network.

FastTrack and Gnutella have good availability because of the distributed index and search mechanism with supernodes/ultrapeers. Gnutalla (with LimeWire) has better NAT-T support than FastTrack and provide better availability of the individual peers. Both networks have lots of peers that don't provide any resources on the network.

Availability of eDonkey2000 depends on the index servers. Each server can only serve certain amount of peers, so the amount of them is critical. There is no single point of failure as long as there are servers left. If a server becomes overloaded or goes down, a client can register to a new one. Overnet is fully decentralized so there is no problem with availability of central components. ED2K (and Overnet) clients have incentive mechanism to provide better download performance to the peers who upload more, that can help availability of the individual peers (e.g. peers open their firewalls).

In BitTorrent the availability of the key components are unpredictable and BitTorrent is quite vulnerable to potential failures. Search functionality is not part of the BitTorrent protocol. But often web servers with downloadable torrent files are thought to be part of the BitTorrent system. If there is no torrent file servers (web servers) available that will blocks all new downloads. This has bad influence on the availability together with the tracker. If a tracker goes down, the torrents depending on the tracker becomes very quickly non functional and no new peers can't join the swarm. Anyway there are many trackers in the whole BitTorrent world, which might remain operational. Some BitTorrent client software has a multiple tracker support, so if one goes down it can try to connect to a new one if the torrent file specifies multiple trackers.

BitTorrent download won't be completed also, if there is no seed available (peer with a complete file) in the swarm and some parts of the content is lost within that swarm. The availability of a complete file is largely determined by the popularity of the file. Xeja's / SODON's design goal is to provide a distributed directory service to improve content availability and system scalability.

3.3 Bootstrapping

P2P client has to know at least one operational node (e.g. central server, higher level peer or ordinary peer), which to connect to on the existing P2P network. A peer then becomes part of the overlay network. A list of operational peer(s) is often delivered with the client software and/or retrieved from a web page (or another static place). A peer in the list might become unreachable at any moment and bootstrapping client might not find any operational peer. This should be rare occasion, but it is possible if the list isn't retrieved and updated practically.

After a successful connection to the network client can receive an up-to-date list of other peers. Often more than one peer is connected to with the help of neighbor selection algorithm. These peers are called neighbors. If some neighbors become unreachable, it is good to have more than one connection as a backup. This prevents also bootstrapping from the beginning.

FastTrack and Gnutella clients have to find operational supernode/ultrapeer, which to connect to. There are different mechanisms to solve the situation and eventually one can be found with very high probability. After a successful connection, client gets an updated list of the other supernodes/ultrapeers. At least LimeWire Gnutella client connects to more than one ultrapeers, other clients are expected to do that as well. ED2K client software includes a list of index servers, but that can become insufficient during time and it is possible to update the list from some web sites (e.g., http://www.emule-help.com/servers.htm). User can manually select index server with the client. With the above protocols bootstrapping is fairly simple task.

Like others an Overnet peer has to solve at least one peer connected to the network. After that according to Kademlia peer starts iterative search for other peers in the 128-bit key space. To work effectively peer has to known several contacts from the key space and thus bootstrapping is somewhat slower than with other P2P protocols discussed.

In BitTorrent a torrent file defines a location of tracker/trackers. Bootstrapping can be considered a case when a peer retrieves a peer list from the tracker and connects to (some of) peers in that list. This set of connections together with a tracker can be considered "peer's local overlay network". Desired amount of connections are 20-50. In this contrast BitTorrent's bootstrapping is somewhat slow. However the process also includes a peer list with desired content and connections where to download the content from. With the other protocols you have to do additional search and form a swarm for that.

3.4 Download Performance

Download performance is better with multi-source downloading. A peer can more easily utilize its whole download capacity with that. Because download and upload rates match each other, there must be peers on the network to provide all that download bandwidth with their upload capacity. And that's why the whole system can have better download performance if other peers provide more resources to the system by uploading. Incentive mechanisms are used to courage for that and NAT Traversal helps to utilize resources of the NATed peers. Asymmetric broadband connections with lower upload capacity than download capacity make things more problematic. Upload capacity of the whole system might not be sufficient to satisfy full download rates of individual peers.

BitTorrent is the leader in download performance due to its advanced download distribution protocol. BitTorrent is focused to the download process and there is no search functionality through the overlay network. Examined P2P protocols still have very similar download techniques. Downloading/uploading is done out-of-band of the overlay network directly between end nodes (considering BitTorrent doesn't have real overlay network in that sense). All of the examined protocols support simultaneous downloading from multiple sources, but FastTrack and basic Gnutella (excluding e.g. LimeWire) don't support sharing of partially downloaded files. Peer selection algorithm (choosing among group of peers where to download pieces of the content from) has some effect on download performance. A good peer selection algorithm satisfies better peer's download capacity. Peer selection algorithms are often client software specific. With the BitTorrent uploading/sharing pieces of a file is done very effectively compared to the others. ED2K and Overnet queue-based scheduling of download with waiting times sometimes exceeds a few days for the lowID peers. As Overnet uses Multisource File Transmission Protocol (like ED2K), there shouldn't be big differences with Overnet and ED2K in download performance.

According to [PGES05], the average download speed of BitTorrent peers was 240 Kbit/s. According to [IUB+04], the download speed of single-session BitTorrent download was 1.3 Mbit/s and average for all sessions was 500 Kbit/s. According to [SGP04] the average download speed for all BitTorrent peers is quite high, being around 200 Kbps.

BitTorrent, ED2K and Overnet is especially well suited for large files. BitTorrent is not good for the small files, because of overhead. FastTrack and Gnutella suit well for transferring small files, because searching time is essential to the overall performance with small files.

3.5 Flash Crowd Effect

Popular releases cause flash crowd effects i.e., content with sudden high demand. Protocols that use swarming can handle flash crowd effect better than simple client-server model in download performance. Swarming responds quickly to flash crowds, with only a slight increase in download time during the crowd and a rapid return to lower download times once the system returns to steady state [SZR05]. In the beginning of flash crowd there exist lots of peers with the incomplete file and only

few uploaders with the completed file. Uploading pieces of partially downloaded files greatly improves flash crowd tolerance as swarming is more efficient with more alternative peers to download from. At least BitTorrent, ED2K, Overnet and LimeWire Gnutella client support sharing partially downloaded files.

Peers' limited number of upload slots together with bad incentive mechanism hinders tolerance of the system against flash crowds as the content can't be spread fast enough. In many cases there are simply too few uploaders and users have to wait for free download slots. Often uploaders' slots are reserved for low demand content as well and not for the content, which has the high demand. Other than BitTorrent in our set of P2P networks seem to have above mention limitations. Besides their incentive mechanisms are often very bad for new and casual P2P users and their potential can't be used. This also prevents spreading the content effectively. BitTorrent determines number of connections on the torrent bases and therefore have better flash crowd tolerance in this contrast. Its incentive mechanism seems to work well also. It is shown in practice and researches that BitTorrent is capable of efficiently handling very large flash crowds [PGES05]. BitTorrent can sustain a high flash crowd since it quickly creates new seeds [IUB+04].

Flash crowds can also affect to overlay network and its functionality. FastTrack and Gnutella networks can dynamically distribute supernodes' tasks to new participating peers and thus be scalable in the face of large flash crowds. Overnet should be able to maintain its overlay network as new participating peers burden only fragmented set of peers and none of them are too critical. With ED2K and BitTorrent flash crowds can be a problem if scalability of the central servers is reached.

3.6 Pollution

Spreading spoofs or somehow harmful files on the network as they were correct is considered polluting. Polluting P2P networks are mostly done by anti-P2P companies like Overpeer and SafeNet to discourage users with illegitimate file-sharing. Polluting can be also done against legitimate usage as denial-of-service attacks so it is important that P2P system is strong against these threats. However [Slyck1] tells that it is not so easy to do the pollution anymore. There exist several pollution techniques and three of them are described here, which are used by SafeNet and other anti-P2P companies as well. More about pollution on the P2P networks and especially in FastTrack network can be found [LKX+].

1. *"Spoilers: Spoiling is a technique that directly interferes with the downloading of an illegal file by either partially delaying or entirely preventing download completion. Spoiling is effective in P2P networks that use multi-source downloading and also makes an impact when pirated files have already leaked onto a network."*

2. *"Spoofs: SafeNet's spoofing technique involves the distribution of non-downloadable, unavailable files on a P2P network. The file name and hash value are generated at random by SafeNet's proprietary virtual file system; however because the file contains no real content and has a hash value that does not match the title name, it will never actually download. With the*

Virtual File System, SafeNet can generate millions of spoofs for a client-specified media title without taxing bandwidth or network resources."

3. *"Decoys: SafeNet generates decoy files that mimic a pirate file in size, type and name but do not contain any real content. As a user searches for that particular illegal file, SafeNet's matching decoys are returned and then shared across the P2P network. Combining this technique with spoofing results in a powerful countermeasure solution that significantly minimizes the visibility of pirated software and frustrates users.*"

[Slyck2] (Original reference is not available anymore:
http://www.safenet-inc.com/Library/3/MSS_ISVOverview.pdf)

If sufficient file hash algorithm is not used, pollution can become a big problem on the network. Some estimates suggested that 50% of all files were either viruses or polluted in some way in FastTrack because of the incomplete UUHash algorithm [Slyck1]. Today FastTrack uses kzhash rather than UUHash. It is not known whether kzhash does a full-file hash. Actually incomplete hash algorithm is not the single cause for the pollution. It is incomplete file hash together with multi-source downloading. This would allow malicious uploader's injection of corrupted data specifically targeting the unhashed portion of the file to go unchecked. All P2P clients that multi-source should do a full-file hash check, ideally using the smallest segment size possible, since only the corrupt segment will be re-downloaded, and not the entire file. [Slyck2]

Pollution on the Overnet and ED2K networks are affected by the same threats as they both use the same multi-source downloading protocol MFTP. There used to be cases where users had to re-download 9500 KB chunks constantly on ED2K/Overnet because smaller pieces were corrupted within that chunk. Today things are much better with the smaller file segments with integrity checks. Unfortunately situation is not as good with the other P2P networks (except BitTorrent). Gnutella and FastTrack users may have to download whole files over and over again, because these protocols don't recognize corrupted chunks while downloading from multiple sources. File integrity check can be done only at the end of the download and one corrupted chunk ruins the whole file.

Some clients may use user based rating system which can help with decoy files, but it doesn't help with spoilers. Besides if there is no effective mechanism, ratings can be also faked.

BitTorrent hashes of the pieces are calculated beforehand and put into the torrent file. Size of pieces can be determined by a user, typical sizes are 128 KB and 256 KB. While multi-sourcing, peers can easily check correctness of received pieces against hash values in the torrent file. Pollution is not a problem with BitTorrent as long as the torrent file is not corrupted in the first place.

3.7 Content Lifetime

Because there is no directory-level file-sharing policy in the BitTorrent, average content lifetimes are relatively short. Many BitTorrent peers usually share/upload only

the content, which is currently being downloaded. Lifetime often depends of the popularity of the torrent and old torrents are often useless. This shouldn't be a problem if a sharing user (content provider) keeps initial seed available. Improvements in content lifetimes can however save user's sharing costs. A P2P system with directory-level sharing policy (FastTrack, Gnutella, ED2K, Overnet, etc.) allows data files to be located as long as the peer holding the content stays connected, which provides more resources and longer content lifetimes on the network.

3.8 Location Awareness

Location awareness improves scalability. P2P overlay network can be location aware, but also download process can take locality into account. Taking locality into account in download process, can overall traffic within links reduced significantly. This applies to the systems with multi-source downloading and we can choose among many peers with desired content where to download from.

"*What are the benefits of location aware P2P system? With Location aware Topology Matching (LTM) it is able to match the logical topology with the physical topology to significantly improve the search efficiency. LTM achieves about 75% reduction on traffic cost and about 65% reduction on query response time*." [LXL+05] There are several proposals for location aware P2P systems like [LXL+05], [YLZ05], [ArRo04] and [BMR04].

Older paper [GDS+03] claims that KaZaA is locality unaware, which is probably not true anymore. Newer paper [LKR04] says that FastTrack takes locality into account when dynamically creating the overlay network. Some Gnutella clients seem to do that also and use some kind of ultrapeer selection algorithms when creating the overlay network. eDonkey2000 overlay network is created with voluntary servers and peers can choose which servers to connect to, so there is only location awareness by users' behavior. We don't know what kind of peer selection algorithms different clients use in the download process if at all. We see that search results are quite random regarding of the location of desired content. But peers next to the same supernode, ultrapeer or ED2K server with the content are found most easily. Therefore if overlay network is location aware, then peer gets "location aware" search results.

BitTorrent is not location aware; the tracker returns a list of random seeds to the client when it starts file downloading [SZR05]. One nice option could be that the tracker returns a list, which consists of 50% random seeds and 50% of proximity seeds. A list of 100% proximity seeds might lead to an isolated network (swarm). Tracker is already bottleneck in the BitTorrent and location aware tracker could make things a lot worse.

4 Networking Issues

4.1 Firewall/NAT Traversal

Current firewall and Network Address Translation (NAT) solutions are suitable for client-server communication in the typical case when the client is on a private

network and the server is in the global address realm. For peer-to-peer communication they are not suitable. There is often a case when two peers on different private networks want to contact each other directly. The problem seems to be growing because lack of the IPv4 addresses and security threats make us to use more NATs and firewalls. In the future IPv6 won't completely solve the problem, because firewalls are here to stay, and we don't know how long the transition takes from IPv4 to IPv6.

There are quite lot NAT Traversal techniques. The most reliable, but least efficient method is relaying, which makes communication between NATed peers act like in client/server architecture and benefits from peer-to-peer architecture are lost. Connection reversal technique can be used, when a peer with global address wants to connect to a peer with private address, by using an intermediate peer, which already has a TCP connection in place to the NATed peer. This is not useful between two NATed peers. More sophisticated methods are for example UDP hole punching and TCP hole punching (P2Pnat) [FSK05], STUN [RWH+03], STUNT [GuCo04], NatBlaster [BFW+05], NatTrav [Epp05], UPnP [UPnP] and Teredo [Hui05]. The problem always is that different NAT Traversal techniques don't solve the problem completely (except relaying). The problem exists in NATs itself as their behavior is not standardized. That is the reason why behavior of the NAT boxes is not always "P2P-friendly". Good thing is that NAT vendors recognize the demand for peer-to-peer protocols and NATs are becoming more "P2P-friendly". Properties of "P2P-friendly" NAT is discussed e.g. in [FSK05].

TCP NAT Traversal is the best solution for applications with the need of reliable transmission. Some of these techniques (NatBlaster, STUNT and P2Pnat) are compared in [GuFr05]. Even when NATs would be "P2P-friendly", there still exist some constraints among the different techniques e.g. NatBlaster requires administrator rights from the user, so it doesn't seem very attractive. In the current situation UDP NAT Traversal techniques has better success rate in hole punching than TCP, because it doesn't set so strict requirements for the NATs. Teredo has advantage of UDP hole punching as approaching the subject. Teredo allows IPv6 packets to traverse IPv4 NATs by tunneling IPv6 over UDP. Here, TCP runs natively in the sense that it is layered directly above IPv6. Teredo technology is built into Windows XP Service Pack 2 and the Advanced Networking Pack for Windows XP. More about Teredo in Windows can be found at [Microsoft]. Another solution is to use reliable UDP protocol, if that is wanted. UPnP allow applications to traverse a NAT through explicit cooperation with the NAT. This protocol is not yet widely supported by NAT vendors or applications. Besides it doesn't appear to address the increasingly important multi-level NAT scenarios. [GuFr05] [FSK05]

4.2 Firewall/NAT Traversal Support

FastTrack's two-tier hierarchy provides a mechanism to partially solve NAT problem. In KaZaA, when peer A sees that peer B has a private NAT address, instead of sending a request directly to peer B, it sends the request to peer B's parent supernode. The parent supernode then sends a message to peer B, indicating that it should initiate a connection directly back to peer A. With this connection in place, the file can be sent over the connection from peer B to peer A. This technique is called connection reversal. FastTrack or its clients doesn't support more sophisticated methods.

Connection reversal is very minimal today and should be expected in all seriously taken P2P clients or networks. More sophisticated NAT-T techniques could be implemented to the existing architecture with the help of supernode (LimeWire does that in Gnutella network). [LKR04]

Like FastTrack Gnutella supports connection reversal. LimeWire goes even further and make use of UDP hole punching technique. LimeWire has a concept of a push proxy for firewalled hosts, which is used to deliver a special PUSH message that tells the host to initiate a UDP connection to particular IP address and port pair. Both ends then start sending UDP messages at each other and shortly thereafter, they should both be able to receive those messages. For the NATed peers, behavior of the NATs involved must be "P2P-friendly". Our short test showed that it worked at some level. No extensive tests were run.

The complex scoring mechanism of eDonkey2000 decides which request is served next (eDonkey2000 and eMule clients use also their own systems). One important factor of the scoring system is the "highID/lowID" mechanism to ensure fairness among all peers. ED2K server assigns a lowID to clients behind a firewall or a NAT. Two lowID peers can't connect to each other normally, but a new eDonkey2000 client (1.4) claims to have NAT-T support. "*We just released a new version. It now has NAT Traversal support so in some cases if two people are behind firewalls then they can still make a connection*" [EK2K]. There is no more information about the technique used. We made a short download test and there was no successful lowID to lowID connection, even when there were over 30 possible lowID peers to download the file from. We can conclude that if there is a NAT-T support, it is currently very ineffective on the network (our test machine wasn't dependent on behavior of a NAT). One reason for the inefficiency is that there is lot's of other client software in the network and the NAT-T works only between eDonkey2000 clients. In the future if it's possible to connect between lowID peers successfully, then the LowID concept disappears. eDonkey2000 client and some eMule mods have UPnP support. As Overnet has been merged into eDonkey2000 client, same features are expected to stand in Overnet case as eDonkey2000 regarding Firewall/NAT Traversal.

Firewall/NAT Traversal is possible with BitTorrent using BitComet client using UDP hole punching technique. BitComet has also tracker software and NAT-T probably won't work with other tracker software in place. (Or is there another third party machine to help NATed peers to coordinate with NAT-T?) It is also quite obvious that BitComet's way to handle NAT-T works among BitComet clients only. BitComet, Azureus and official BitTorrent clients support UPnP. Azureus (2.3.0.6) supports also NAT Traversal with DHT; DHT NAT punching for firewalled peers. [Azur] [BitCo]

NAT Traversal between firewalled/NATed peers is relatively new subject in P2P and with most popular P2P protocols there is no support for that in the protocol level. Clients compete with each other adding their own techniques independently. Used techniques remain often unclear because those are not documented or just kept in secret. This makes the clients incompatible with each other regarding to NAT-T. That's why there should be a mechanism in the protocol level to make NAT-T. In this contrast until new protocol enhancements in place, UPnP seems the most practical way to handle NAT-T.

4.3 IPv6 Support

BitTornado supports IPv6. Another IPv6 capable BitTorrent client and tracker programs are available at http://reboot.animeirc.de/bittorrent/. You can test IPv6 tracker at http://6net.niif.hu/index.php?mn=3&sm=10&lg=en. According to our current knowledge there are no other IPv6 capable P2P clients for our interested P2P networks. It seems that IPv6 won't be supported widely in short time.

5 Downloading and Sharing

5.1 File Transfer Protocols

All studied P2P networks use TCP in their file transfer. NAT Traversal techniques in BitComet and LimeWire use UDP for file transmission. FastTrack and Gnutella use HTTP for file transfer and multi-source downloading is supported using Content-Range HTTP header. ED2K uses MFTP for file transfer, but HTTP and BitTorrent are also supported. In Overnet the file transfer protocol is practically the same as in ED2K [Ove], i.e., MFTP. BitTorrent uses its own BitTorrent protocol for file transfer, which supports multi-source downloading.

Some BitTorrent clients (e.g. uTorrent and Azureus) also have an end to end encryption support between participating peers in a swarm. All BitTorrent traffic between peers is encrypted. Encryption decision is done between individual peers. You can communicate encrypted with some set of peers and unencrypted with other. Encryption uses info-hash of a torrent as a shared secret in combination with Diffie-Hellman keys, which are generated when the connection is set up. The Diffie-Hellman helps minimizing the risk of passive listeners, and the shared secret helps avoiding man-in-the-middle attacks. The data stream is encrypted with RC4. [Slyck4]

5.2 Block Sizes

Swarming can scale to a wide range of block sizes [SZR05]. FastTrack uses 64 KB block sizes [KBB+03]. With Gnutella, a user can set maximum and minimum chunk sizes to be used. For example 500 KB for minimum and 5 MB for maximum could be used.

eDonkey2000 divides a file to so called chunks, which are 9500 KB (~9.28 MB), with the exception of the final chunk, which might be shorter. When a chunk is shared it is transmitted in units of blocks with size 180 KB. The downloading client becomes a seed after it has received at least one complete chunk. All chunks are downloaded in parallel. [HLP+04]

BitTorrent divides a file to so called pieces, which are for example 256 KB, with the exception of the final piece, which might be shorter. The piece size is told in a torrent file. The size of a torrent file grows if the number of pieces increases, because hashes of the pieces are listed into that torrent file. Creator of a torrent file can decide the piece size. BitTorrent clients download and upload pieces in smaller units called

slices. A downloading client becomes a seed when it has at least one complete piece. Clients may have possibility to set the size for download/upload slices, for example in BitTorrent client the default download slice size is 16 KB and the default maximum upload slice size is 128 KB.

5.3 Incentives

P2P clients have often a variable of maximum number of upload slots that tells how many other peers can be uploaded simultaneously to. When upload limit is reached, new requesting peers can be put to the queue. This is pretty much what FastTrack and ED2K basically does. Which peers are uploaded first and which are pruned depends of the network and/or client software.

FastTrack has participation level, which can be 0-1000. Participation level describes sharing/upload activity of a peer. Peers with high participation level are served first by other peers. Some clients e.g. KLT K++ have modified participation level at the maximum for better download performance. This makes the original incentive mechanism ineffective. In Gnutella there is no incentive mechanism. Peers are served in order of arrival as long as there are upload slots available.

ED2K core has a basic queue system. ED2K clients can have different criteria how peers are served. With Shareaza users can even define their own criteria. In addition to basic queue, eDonkey2000 (and Overnet) uses proprietary Horde system and eMule credit system. Idea behind eMule's credit system is that the more a user uploads to a client the faster he advances in this client's queue. More detailed information about the credit system can be found at eMule's web site [eMule]. Downside with this approach is that new or casual user may not have anything interesting to share for others so that he could advance in others' queues. In this case grouping peers on the grounds of their interests is better approach, BitTorrent and Horde does that. Horde is the process of teaming multiple downloader of the same file together, to work towards the common goal of completing the file [ED2K].

The BitTorrent protocol is designed in a way, which discourages freeloaders, by having the nodes prefer peers from which data has been received; the download speed of a host will be reduced if the upload speed has been limited. This ensures that content will be spread among hosts and improves reliability. Each host informs new clients of the pieces it currently has, and sends notifications when new pieces are received. BitTorrent provides incentives only to the file-sharing peers, not to the seeding peers. More specifically, the seeds may not have any motivation to stay out there and upload to the network. This might be a significant limitation in the incentive mechanism on some environments. [SGP04] [Tam03]

BitTorrent uses upload and download rates as bases in its Tit-for-Tat incentive mechanism. [JuAh05] proposes a mechanism, which is based on absolute difference between upload and download amount. Another proposal could be a tracker with a "bonus" system, to provide incentive to keep the torrent open. Thus the more you upload the more "credit" you are given. If you think of it in subscriber terms, perhaps people that have a high "credit" (i.e. they leave their client open after being finished) would get earlier access to files. Or with a modified BitTorrent tracker it could be

possible to deny the clients, which are freeloaders, to contact to the tracker and thus they won't get any downloads.

5.4 Directory Sharing

With BitTorrent it is possible to share a complete directory with a single pointer. In that case the torrent file describes all files in the directory and their relative locations. Other P2P clients have directory based sharing, but there is no single pointer to the whole directory and its structure. Client has to ask each file in that directory separately.

6 Network Management and Control

6.1 Control Traffic

FastTrack's control traffic is carried over TCP, is encrypted and everything is in HTTP request and response messages. Gnutella has its own control protocol (ping, pong, bye, push, query, query hit), which is carried over TCP. In eDonkey2000 network, client-to-"associated server" and client-to-client communication is carried over TCP. Client-to-"other server" and server-to-server communication is carried over UDP. ED2K has its own control messages. Overnet control traffic is carried over UDP. BitTorrent uses HTTP tracker protocol (client-to-tracker communication), but UDP tracker protocol is also available [UDPtrack]. There are also HTTPS implementations of the tracker. The UDP tracker protocol is a high-performance low-overhead BitTorrent tracker protocol. The UDP tracker protocol uses (less than) 50% of the bandwidth the HTTP tracker protocol uses. And because UDP is stateless, limits to the number of (open) TCP connections a router or server can handle, do not apply. Client-to-client control traffic is carried over TCP using BitTorrent protocol.

6.2 Content Control Possibilities

FastTrack, Gnutella and Overnet don't have content control possibilities as they are completely semi-centralized or decentralized. At least controlling entity would be hard to implement into these environments without any central component.

BitTorrent and ED2K have some control possibilities which can be done with the central components. BitTorrent supports content controlling; it is possible to specify the tracker to offer and track only certain torrents. With ED2K it is possible to block certain files in the index server; the hash values for denied files are written into specific server configuration file. It is also possible to filter or allow certain types of files, for example mp3 files.

6.3 Transaction Tracking and User Information

With BitTorrent there is a possibility to see file-downloading info from the tracker's statistics. The info is available for each file (torrent). Example tracker info is available at http://cdimage.debian.org:6969.

With ED2K it is possible to see some general statistics from the index server. The statistics include: current amount of users, shared files, downloaded files, done searches and “get sources” requests.

With the decentralized networks (FastTrack, Gnutella, Overnet) it is possible to see some network statistics from the clients. These include for example number of users, shared files, and amount of shared data.

6.4 Group Limited Access

There is no flexible way to have user groups in FastTrack, Gnutella or Overnet, but it is possible to use IP filters and allow only users of the group to connect to each other. Maintenance of the group this way is very clumsy.

With ED2K Lugdunum server, it is possible to great private server. Server won’t announce its presence to other servers with a public=false parameter. Only clients that know the IP address of the server will be able to connect to. This isn’t very private or secure. It should be fairly easy to implement password protection to the server.

With BitTornado tracker it is possible to form groups by limiting the clients that can contact to the tracker by setting allowed IP addresses. One way is to control torrent distribution, but that doesn’t provide very secure or private user groups. With Azureus (embedded tracker) it is also possible to allow access only from certain IP addresses. You can also make password protected tracker in a way that you must give a password when you first time contact to the tracker. There is also HTTPS and end to end encryption support, so you can create quite secure and private user groups with Azureus.

7 Finding Content

7.1 User’s Content Search Possibilities

BitTorrent uses public web servers to announce the available content in the network. There are torrent files listed in some web sites (e.g., http://cdimage.debian.org/debian-cd/torrents), and users can search the sites or the torrent files using Google for example. There are currently also dedicated search engine at www.bittorrent.com. After the user has found the torrent file from the web he/she can start downloading the content. The torrent file contains the URL of the tracker, which knows the seeds for the file. Bad thing is that the user can’t know from the torrent file is the tracker up or not, or are there any seeds for the file.

In ED2K the user can make keyword queries to the index servers to find certain files. The replies from the servers contain the file name, the file hash (MD4) and possible the sources of the file. A second option is to put ed2k links (e.g., ed2k://|file|gentoo.linux.install-x86-minimal-2004.1.iso|85764096|F1819D1C731923327E140F09DB7400B6|/) to a publicly available web site. There are some web sites listing ed2k links, and those can be searched for example using Google. The link includes the file hash, which is the key point when searching the sources of the file

from the network. The “get sources” query is sent to the associated index server and possibly to other index servers also. The server returns then a list of available seeds.

In Overnet the user’s client makes keyword queries directly to other clients. A search query contains MD4-hash of the keyword (if more than one keyword are used, then second, third, etc. are used as a filter during the search phase). The reply includes the file name, the same MD4-hash than in the query and the MD4-hash of the source of the file. With Overnet it is also possible to use ed2k links and in that case the query contains MD4-hash from the ed2k link.

In FastTrack the user has the same possibilities to search content than in ED2K and Overnet. With FastTrack it is possible to use sig2dat links (e.g., sig2dat:///|File: gentoo.linux.install-x86-minimal-2004.1.iso|Length:85764096Bytes|UUHash:=axjBp 33O9JRlDcAcbWiuE3+lq2Q=|). The link includes the file hash, which is again used to search the sources for the file. It is also possible to use magnet links with FastTrack clients (e.g., magnet:?xt=urn:kzhash:1f5f0ab475ae44cce629c601facc05eef1a629603 ecb0dc974b387c3e57541691d6997d9&xl=85764096&dn=gentoo.linux.install-x86-minimal-2004.1.iso) to initiate search.

Gnutella utilizes also magnet links with SHA-1 hashes (e.g., magnet:?xt=urn:sha1:LOOQCDVHC6KRSQ5KNOAZZTTZJN5KATZO&xl=85764 096&dn=gentoo.linux.install-x86-minimal-2004.1.iso).

7.2 RSS Feeds

Really Simple Syndication (RSS) feeds can be used to automate the downloading process. There is an RSS Feed Scanner plug-in available for Azureus (a BitTorrent client). The RSS Feed Scanner is an automatic RSS feed parser, which is highly configurable and allows unattended operation via its advanced filtering capabilities. It supports multiple feed URLs, each of which can be individually or globally configured. The filter strings can be easily edited via the graphical configuration tool, and each filter can also support targeting of specific episodes within a series of the same titles (for example in downloading episodes of shows). The download history of torrents is maintained so that the specific torrents are only downloaded once, even if you shutdown and restart. Plug-in also has full help included in the interface, so you can start using the plug-in without any prior knowledge needed. [RSSfs] There is also another RSS plug-in available for Azureus. An RSS Import plug-in reads an RSS feed and imports the torrent files into Azureus, according to the filters you set. [RSSIm] Other P2P clients don’t have RSS feed capabilities.

Videora 1.0 is a new personal video downloading program. Utilizing BitTorrent peer-to-peer technology and RSS feeds, Videora automatically and intelligently finds and downloads video you want to watch. With easy to use features like Want Lists and Season Tickets you will be able to watch your favourite video, no matter where you are in the world. All you need to get started is a broadband Internet connection and Windows operating system. [Vid]

8 Security Threats

Some security issues are covered in pollution chapter. Files downloaded from P2P networks may contain viruses; Worms, Trojans, Backdoors. If reliable and trusted content pointers are used (e.g. torrents from debian.org), viruses shouldn't pose a big problem. Pollution can be used as denial-of-service attacks as discovered earlier. P2P systems could be used also as a weapon. It could be used as a distributed denial-of-service (DDoS) engine for attacks against a targeted host, which is covered more detailed in [NaRo]. Some of the P2P clients may contain spyware and/or adware. With P2P software a user has a big risk to violate copyright issues even unintentionally. There is also risk to share sensitive private data accidentally.

9 Conclusion

Researched P2P networks have quite different characteristics and none is strong in all areas. FastTrack, Gnutella and Overnet protocols are suitable for very wide content sharing because of the strong scalability and availability properties. Lack of any kind of control component makes them unsuitable for the scenarios where content of the network must be strictly controlled. That kind of feature wouldn't be trivial task to implement into these environments preserving other properties. BitTorrent and eDonkey2000 has strong points here with the control in central servers, but scalability and availability becomes more problematic. Although there are solutions for that (e.g. with server farms), but they are more rigid. Used protocol should be selected on the grounds of application. There isn't currently any solution that suits well for all tasks.

P2P networks that support multi-source downloading and sharing partial files with proper integrity checks for file chunks provide the most complete system today regarding download performance, security and strength against pollution. Only ED2K, Overnet and BitTorrent do that. Location awareness could be great improvement to the download performance and overlay topology; improving scalability as well. Content lifetime shouldn't be a problem with legitimate uses as content provider can always keep one seed available.

Current networks are bad with Firewall/NAT Traversal. Peers behind Firewall/NAT are often unavailable to other Firewalled/NATed peers and therefore don't provide any potential resources to them. As number of Firewalled/NATed peers is growing the whole system becomes more ineffective, peers are only using resources and not providing any. Used mechanism for Firewalled/NATed peers is often lower priority service, that they can't consume resources that much. We think most development is needed with Firewall/NAT Traversal and location awareness as they can greatly improve the performance of current systems.

10 References

[ArRo04] F. Araújo, and L. Rodrigues, "GeoPeer: A Location-Aware Peer-to-Peer System", Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications (IEEE NCA04), Cambridge,

MA, USA, August 2004. Available at http://www.di.fc.ul.pt/~ler/reports/nca04.pdf (17.2.2006)

[Azur] Azureus http://azureus.sourceforge.net (17.2.2006)

[BFW+05] A. Biggadike, D. Ferullo, G. Wilson, A. Perrig "NATBLASTER: Establishing TCP Connections Between Hosts Behind NATs*", SIGCOMM Asia Workshop 2005 Beijing, China. Available at http://www.andrew.cmu.edu/user/ggw/natblaster.pdf (17.2.2006)

[BitCo] BitComet http://www.bitcomet.com (17.2.2006)

[BitTo] BitTorrent http://www.bittorrent.com (17.2.2006)

[BMR04] D. Bickson, D. Malkhi, and D. Rabinowitz, "Locality-Aware Content Distribution", Leibnitz Center TR 2004-52, School of Computer Science and Engineering, The Hebrew University, 2004. Available at http://research.microsoft.com/~dalia/pubs/julia-tr.pdf (17.2.2006)

[BSV03] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding Availability", IPTPS, 2003. Available at http://www.cs.ucsd.edu/~voelker/pubs/avail-iptps03.pdf (17.2.2006)

[Co03] Bram Cohen, "Incentives Build Robustness in BitTorrent", May 22 2003. Available at http://www.bittorrent.com/bittorrentecon.pdf (17.2.2006)

[CRB+03] Y. Chawathe, S. Ratnasamy, L. Breslay, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable", In ACM SIGCOMM, Germany, August 2003. Available at http://berkeley.intel-research.net/sylvia/1103-chawathe.pdf (17.2.2006)

[ED2K] eDonkey2000 http://www.edonkey2000.com (17.2.2006)

[eMule] eMule http://www.emule-project.net (17.2.2006)

[Epp05] J. L. Eppinger, "TCP Connections for P2P Apps: A Software Approach to Solving the NAT Problem", Carnegie Mellon Tech Report CMU-ISRI-05-104, January 2005. Available at http://reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-104.pdf (17.2.2006)

[FSK05] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators", USENIX Annual Technical

Conference, April 2005. Available at http://www.brynosaurus.com/pub/net/P2Pnat.pdf (17.2.2006)

[GDS+03] K. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan, “Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload”, Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), New York, USA, October 2003. Available at http://www.cs.rochester.edu/sosp2003/papers/p118-gummadi.pdf (17.2.2006)

[GuFr05] S. Guha and P. Francis. "Characterization and Measurement of TCP Traversal through NATs and Firewalls" in Proceedings of Internet Measurement Conference (IMC), Berkeley, CA, Oct 2005. Available at http://nutss.gforge.cis.cornell.edu/pub/imc05-tcpnat.pdf (17.2.2006)

[GuCo04] S. Guha and U. Cornell, Internet-Draft: “STUNT - Simple Traversal of UDP Through NATs and TCP too”, December 11, 2004. Work in progress. Available at http://nutss.gforge.cis.cornell.edu/pub/draft-guha-STUNT-00.txt (17.2.2006)

[Gnut04] The Annotated Gnutella Protocol Specification v0.4. http://rfc-gnutella.sourceforge.net/developer/stable/index.html (17.2.2006)

[Gnut06] RFC-Gnutella 0.6. http://rfc-gnutella.sourceforge.net/developer/testing/index.html (17.2.2006)

[HeBo02] O. Heckmann, A. Bock ”The eDonkey 2000 Protocol”, KOM Technical Report 08/2002, December 2002, updated 27.06.03. Available at ftp://ftp.kom.e-technik.tu-darmstadt.de/pub/papers/HB02-1-paper.pdf (17.2.2006)

[HLP+04] T. Hoßfeld, K. Leibnitz, R. Pries, K. Tutschku, P. Tran-Gia, and K. Pawlikowski, “Information Diffusion in eDonkey Filesharing Networks”, Technical Report No. 341, September 2004. Available at http://www-info3.informatik.uni-wuerzburg.de/TR/tr341.pdf (17.2.2006)

[Hui05] C. Huitema, Internet-Draft: “Teredo: Tunneling ipv6 over udp through nats”, Apr. 2005. Work in progress. Available at http://ietfreport.isoc.org/idref/draft-huitema-v6ops-teredo (17.2.2006)

[IUB+04] M. Izal, G. Urvoy-Keller, E.W. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, “Dissecting BitTorrent: Five Months in a Torrent's Lifetime”, In Passive and Active Measurements, Antibes Juan-les-Pins, France, April 2004. Available at http://www.pam2004.org/papers/148.pdf (17.2.2006)

[JuAh05] S. Jun, and M. Ahamad, “Incentives in BitTorrent Induce Free Riding”, 2005. Available at http://www.cs.duke.edu/courses/fall05/cps296.1/papers/jun-P2P-econ.pdf (17.2.2006)

[KBB+03] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, “File-sharing in the Internet: A characterization of P2P traffic in the backbone”, Technical Report, UC Riverside, 2003. Available at http://www.cs.ucr.edu/~tkarag/papers/tech.pdf (17.2.2006)

[KuBi05] Y. Kulbak and D. Bickson “The eMule Protocol Specification”, January 17, 2005. Available at http://www.cs.huji.ac.il/labs/danss/presentations/emule.pdf (17.2.2006)

[KuFu05] K. Kutzner, and T. Fuhrmann, “Measuring Large Overlay Networks - The Overnet Example”, Konferenzband der 14. Fachtagung Kommunikation in Verteilten Systemen (KiVS 2005), February 28 - March 3, 2005. Available at http://i30www.ira.uka.de/research/documents/P2P/2005/kutzner05overnet.pdf (17.2.2006)

[LCP+04] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, “A Survey and Comparison of Peer-to-Peer Overlay Network Schemes”, IEEE Communications Survey and Tutorial, March 2004. Available at http://www.cl.cam.ac.uk/Teaching/2004/AdvSysTop/ieee-P2P-survey.pdf (17.2.2006)

[LimeWire] LimeWire www.limewire.com (17.2.2006)

[LKR04] J. Liang, R. Kumar, and K.W. Ross, “The KaZaA Overlay: A Measurement Study”, September 15 2004. Available at http://cis.poly.edu/~ross/papers/KazaaOverlay.pdf (17.2.2006)

[LKX+] J. Liang, R. Kumar, Y.Xi and K.W. Ross, “Pollution in P2P File Sharing Systems”. Available at http://cis.poly.edu/~ross/papers/pollution.pdf (17.2.2006)

[LRW03] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, “Deconstructing the Kazaa Network”, 3rd IEEE Workshop on Internet Applications (WIAPP'03), San Jose, CA, 2003. Available at http://people.cs.uchicago.edu/~matei/PAPERS/kazaa.pdf (17.2.2006)

[LXL+05] Y. Liu, L. Xiao, X. Liu, L.M. Ni, and X. Zhang, “Location Awareness in Unstructured Peer-to-Peer Systems”, IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol. 16, No. 2, February 2005. Available at http://www.cs.wm.edu/hpcs/WWW/HTML/publications/papers/TR-05-4.pdf (17.2.2006)

[MaMa] P. Maymounkov and D. Mazi`eres, “Kademlia: A Peer-to-peer Information System Based on the XOR Metric”. Available at http://www.cs.rice.edu/Conferences/IPTPS02/109.pdf (17.2.2006)

[Microsoft] Microsoft Technet: “Teredo Overview”, Published January 1, 2003 Updated October 20, 2005. Available at http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/teredo.mspx (17.2.2006)

[NaRo] N. Naoumov and K. Ross, “Exploiting P2P Systems for DDoS Attacks”. Available at http://cis.poly.edu/~ross/papers/p2pddos.pdf (22.2.2006)

[Ove] The Overnet Protocol https://opensvn.csie.org/mlnet/trunk/docs/overnet.txt (17.2.2006)

[PGES04] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips, “A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System”, Technical Report PDS-2004-003, Delft University of Technology, The Netherlands, April 2004. Available at http://www.isa.its.tudelft.nl/~pouwelse/bittorrent_measurements.pdf (17.2.2006)

[PGES05] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips, “The BitTorrent P2P File-Sharing System: Measurements and Analysis”, 4th International Workshop on Peer-to-Peer Systems (IPTPS'05), Feb 2005. Available at http://www.isa.its.tudelft.nl/~pouwelse/Bittorrent_Measurements_6pages.pdf (17.2.2006)

[QiSr04] D. Qiu, and R. Srikant, “Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks”, In ACM SIGCOMM, Portland, OR, USA, September 2004. Available at http://tesla.csl.uiuc.edu/~srikant/Papers/sigcomm04_revised.pdf

[RSSfs] RSS Feed Scanner http://azureus.sourceforge.net/plugin_details.php?plugin=rssfeed (17.2.2006)

[RSSim] RSS Import http://azureus.sourceforge.net/plugin_details.php?plugin=RSSImport (17.2.2006)

[RWH+03] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, RFC: 3489, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, March 2003. Available at http://www.ietf.org/rfc/rfc3489.txt (17.2.2006)

[SBB04] R. Sherwood, R. Braud, and B. Bhattacharjee, “Slurpie: A Cooperative Bulk Data Transfer Protocol”, In IEEE Infocom, Honk Kong, China,

March 2004. http://www.ieee-infocom.org/2004/Papers/19_3.PDF (17.2.2006)

[SGP04] K.-A. Skevik, V. Goebel, and T. Plagemann, “Analysis of BitTorrent and its use for the Design of a P2P based Streaming Protocol for a Hybrid CDN”, Delft University of Technology Parallel and Distributed Systems, Report Series, Technical Report, June 2004. http://www.ifi.uio.no/english/research/groups/dmms/papers/129.pdf (17.2.2006)

[SZR05] D. Stutzbach, D. Zappala, and R. Rejaie, “The Scalability of Swarming Peer-to-Peer Content Delivery”, IFIP Networking, May 2005. Available at http://www.cs.uoregon.edu/~reza/PUB/networking05.pdf (17.2.2006)

[Slyck] Slyck P2P forum www.slyck.com (17.2.2006)

[Slyck1] “End of the Road for Overpeer” http://www.slyck.com/news.php?story=1019 (17.2.2006)

[Slyck2] Discussion forum: “End of the Road for Overpeer” http://www.slyck.com/forums/viewtopic.php?t=17155 (17.2.2006)

[Slyck3] “Azureus Introduces DHT Layer” http://www.slyck.com/news.php?story=772 (17.2.2006)

[Slyck4] “BitTorrent End to End Encryption and Bandwidth Throttling - Part I” http://www.slyck.com/news.php?story=1083 (20.2.2006)

[Tam03] K. Tamilmani, ”Robustness of the BitTorrent Protocol”, Project proposal, June 2003. Available at http://mnl.cs.sunysb.edu/home/karthik/BitTorrent/Robustness_of_BT.doc (17.2.2006)

[Tut04] K. Tutschku, “A Measurement-based Traffic Profile of the eDonkey Filesharing Service”, In Proc. of the 5th Passive and Active Measurement Workshop (PAM 2004), Antibes Juan-les-Pins, France. April 2004. Available at http://www.pam2004.org/papers/159.pdf (17.2.2006)

[UDPtrack] UDP tracker protocol http://xbtt.sourceforge.net/udp_tracker_protocol.html (17.2.2006)

[UPnP] Microsoft Corporation. UPnP — Universal Plug and Play Internet Gateway Device http://www.upnp.org (17.2.2006)

[Vid] Videora - The First Personal Video Downloading Program http://www.videora.com/en-us (17.2.2006)

[YLZ05] Y. Yu, S. Lee, Z. Zhang, “Leopard: A Location-Aware Peer-To-Peer System With No Hot Spot”, Technical Report 04-031, Computer Science and Engineering Department, University of Minnesota, Jul 2004. Available at https://wwws.cs.umn.edu/tech_reports_upload/tr2004/04-031.pdf (17.2.2006)

[Zhe04] P. Zheng, “Xeja: A Scalable Channel-Based Multi-Source Content Distribution System”, In Proceedings of the Ninth International Workshop on Web Content Caching and Distribution (IWCW'04), Beijing, China, October 2004. Available at http://gargoyle.arcadia.edu/mathcs/zhengpei/publications/zheng_iwcw04.pdf (17.2.2006)

[ZhWa04] P. Zheng, and C. Wang, “SODON: A High Availability Multi-Source Content Distribution Overlay”, In Proceedings of 13th International Conference on Computer Communications and Networks (ICCCN'04), 2004. Available at http://gargoyle.arcadia.edu/mathcs/zhengpei/publications/zheng_sodon.pdf (17.2.2006)