Network Protocols and Algorithms
ISSN 1943-3581
2011, Vol. 3, No. 4

Controlling P2P File-Sharing Networks' Traffic

Miguel Garcia, Mohammed Hammoumi, Alejandro Canovas and Jaime Lloret

Integrated Management Coastal Research Institute, Polytechnic University of Valencia

C/ Paranimf nº 1, Grao de Gandía – Gandía, Valencia (Spain)

migarpi@posgrado.upv.es, simohammoumi@gmail.com, alcasol@posgrado.upv.es, jlloret@dcom.upv.es

Received: November 4, 2011 Accepted: December 5, 2011 Published: December 31, 2011

DOI: 10.5296/npa.v3i4.1365 URL: http://dx.doi.org/10.5296/npa.v3i4.1365

Abstract

Since the appearance of Peer-To-Peer (P2P) file-sharing networks some time ago, many Internet users have chosen this technology to share and search programs, videos, music, documents, etc. The total number of P2P file-sharing users has been increasing and decreasing in the last decade depending on the creation or end of some well known P2P file-sharing systems. P2P file-sharing networks traffic is currently overloading some data networks and it is a major headache for network administrators because it is difficult to control this kind of traffic (mainly because some P2P file-sharing networks encrypt their messages). This paper deals with the analysis, taxonomy and characterization of eight Public P2P file-sharing networks: Gnutella, Freeenet, Soulseek, BitTorrent, Opennap, eDonkey, MP2P and FastTrack. These eight most popular networks have been selected due to their different type of working architecture. Then, we will show the amount of users, files and the size of files inside these file-sharing networks. Finally, several network configurations are presented in order to control P2P file-sharing traffic in the network.

Keywords: P2P, file-sharing, control traffic.

1. Introduction

Since Internet became accessible to the world, the volume of connected users to the P2P networks has been growing up in spectacular fashion. Recently, the number of Internet users is estimated to be over 580 million with a steady growth rate of 4% every year [1]. In the last years, Internet has sufficiently matured to catch market's attention both at home and enterprise level. Besides, the recent transition to broadband access networks, with a current amount of 40 million of users worldwide [2], allows multimedia services. P2P technology is widely used in a great variety of purposes such as P2P file-sharing [3], P2P content delivery [4], P2P media streaming [5], etc.

The first distinction that has to be done is the difference between P2P networks and P2P desktop applications. P2P networks are a set of rules and interactions that allow P2P desktop applications to communicate. A P2P desktop application is a computer application that allows a user to interact with others in the same network. E.g. some P2P file-sharing networks have many P2P file-sharing desktop applications.

P2P network protocols are located at the application layer. These protocols can be programmed to run over TCP or UDP (see Figure 1). The communication between P2P clients, the transferred data and the routed data are performed independently of the lowest layers of the communication protocol stack.

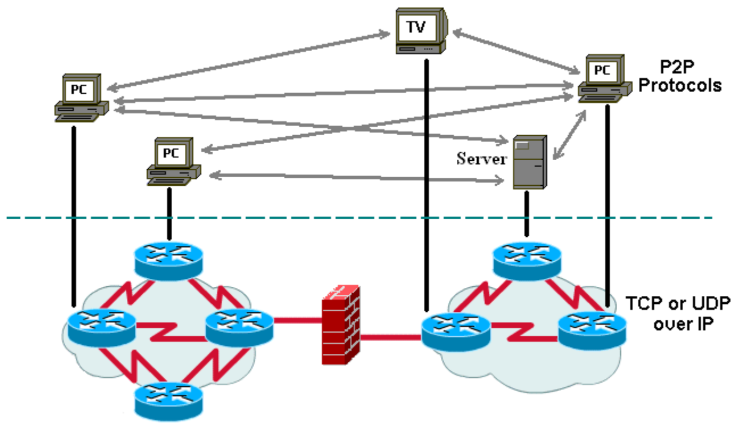

Figure 1. P2P network protocols.

P2P file-sharing is the P2P variant with a largest number of supporters. Many users merely interact with the P2P network in order to download files without providing any, but there are many others that share their files with the whole community without taking care who is downloading them. Those who share files often have a broadband always-on connection. The success of a P2P network inside a user community is determined by several factors:

- Simplicity: a P2P network with a graphical and easy-to-use desktop application is always welcome.
- Language: a P2P desktop application with multilanguage support allows a broader worldwide deployment.
- Download speed: some P2P file-sharing networks, due to its internal behavior, are optimal for downloading reduced size files. Others, however, use multisplitting mechanisms and permit to download a file from multiple sources, making the file-sharing network suitable for obtaining large files.

These parameters are mainly responsible for becoming a P2P network very popular or, on the other hand, unknown. Moreover, the above factors make a P2P network more attractive to users of a specific nation due to the utilization of a specific language or even social trends [6]. It is also interesting to distinguish between a P2P network and P2P desktop application; their development may not be necessarily evolving in parallel. If a P2P desktop application changes its P2P network, all its community users could remain using the P2P desktop application, not its P2P network. As an example, many users have remained 'loyal' to the Morpheus P2P desktop application client throughout its evolution. Sometimes the reason for changing to another newly offered P2P desktop application is its ability to simultaneously contact several P2P file-sharing networks, such as Shareaza [7] and MLDonkey [8].

Due to the significant social impact of P2P file-sharing networks, both industry and academia are spending time and money analyzing several aspects of these P2P networks. Probably the first considerations are the legality of the files that are being shared and the potential risks for both home users and industry (there are many workers using their workstations as P2P file-sharing clients [9]). Furthermore, most P2P file-sharing networks have not been designed to address security issues (authentication, file permissions and file integrity). Some P2P file-sharing clients even include adware or spyware. Moreover, as files are often downloaded from unknown users, they may include some viruses, worms or trojans. On the other hand, employees have to take care of which files are shared in their work-computers because they should to protect the company's intellectual property and their personal information. It is needed to mention that P2P file-sharing networks address any potential liability when employees download copyrighted material from the P2P network, simply the threat of legal action can be a negative impact on the corporation.

In this paper, we explain in detail and study the protocol and network traffic of two totally decentralized P2P networks (Gnutella and Freeenet), four partially decentralized P2P networks (FastTrack, OpenNap, eDonkey, and MP2P), a single centralized P2P network (SoulSeek) and a multiple centralized P2P network (BitTorrent). Finally several methods for controlling P2P file-sharing on networks are presented.

The remainder of the paper is organized as follows. Section 2 analyses the most common features of P2P architectures. The protocol procedure of Gnutella, Freeenet, Soulseek, BitTorrent, Opennap, eDonkey, MP2P and FastTrack are explaned and their main features are summarized and compared in Section 3. Section 4 provides the analysis of the traffic gathered for FastTrack, Gnutella, OpenNap, eDonkey and MP2P during several years. The methods used for controlling P2P file-sharing are described in Section 5. Finally, Section 6 concludes this paper.

2. P2P architecture analysis

In order to study and analyze P2P file-sharing networks properly, we should know which their common features. Moreover, it is also interesting to know what is different between them. A lot of P2P file-sharing P2P networks have the following common features [10]: user privacy, encryption, distribution, data redundancy, direct transfer and high availability.

However, there are several parameters that can be changed in these P2P networks: decentralization, routing algorithms and metrics, load balancing, traffic balancing, data search motor and file downloading system.

Based on their architecture, P2P networks can be decentralized, centralized or partially centralized. P2P software applications let the users communicate in order to exchange data. These applications allow a peer to become a server and a client at the same time, these peers are called servants. In decentralized networks, no peer is indispensable for the proper operation of the P2P network, otherwise, in partially centralized or centralized networks, some peers are needed. But in all cases the data transfer is performed between end clients.

In decentralized P2P networks, all computers have the same responsibility and role in the network (see Figure 2). A peer can make data requests to other peers and reply queries from other ones. Peers can play three roles: server, client and router. A peer can employ several search algorithms, e.g. using a list of known peers or sending a multicast or broadcast message to the P2P network. There are three basic actions: search active peers, query for resources and transfer content. When peer joins the decentralized P2P network, it broadcasts or multicasts ping messages to the network in order to know who is in the network and who can be its neighbor. Active peers will answer with pong messages and will become its neighbors. When the peer is searching for resources in the P2P file-sharing network, peers with contents matching that request will answer. Finally, the user selects the files to download. Examples of decentralized P2P file-sharing networks are Gnutella, and Freenet. We can also find other networks such as CAN [11], Chord [12], Pastry [13], Tapestry [14], and Salsa [15] not analyzed in this paper.

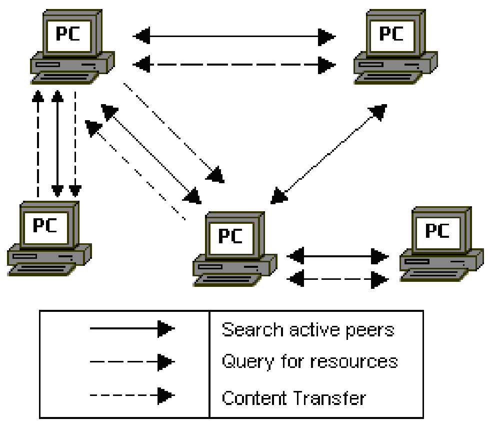

Figure 2. Decentralized P2P network.

Centralized P2P networks need a central server, although peers communicate directly (see Figure 3). Two subclasses of architectures can be differentiated:

- Peers consult services
- Peers and resources consult services.

In this paper, only the second one will be considered because it is the one used in P2P file-sharing networks. They use a central server that keeps track of active peers and indexes of the shared contents. There are three basic actions: registration, consultation and content transfer. During the registration process, the peer informs the server that it is active and which its shared content is. When the peer requests for a desired file, the server can process the query searching indexes or consulting connected peers. Then, the server replies to the original peer. Now, the peer can select the files to download them. An example of a centralized P2P network, with peers and resources query service, is Soulseek.

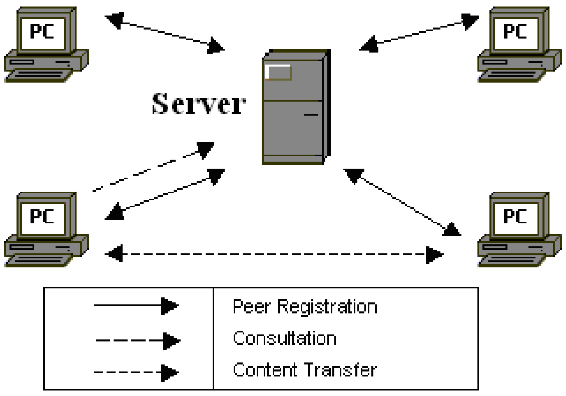

Figure 3. Centralized P2P network.

BitTorrent is a special case of centralized P2P file-sharing networks. It can be seen in Figure 4. It uses a Web server to store files with ".torrent" extension. The ".torrent" file has the hash of the file requested. There is always a peer with the entire file, and it is called seed. When a peer downloads the entire file, it becomes a seed. There is a central server called tracker to control connections between peers. There are a lot of seeds and trakers in existence, but, nowadays, no communication between them exists.

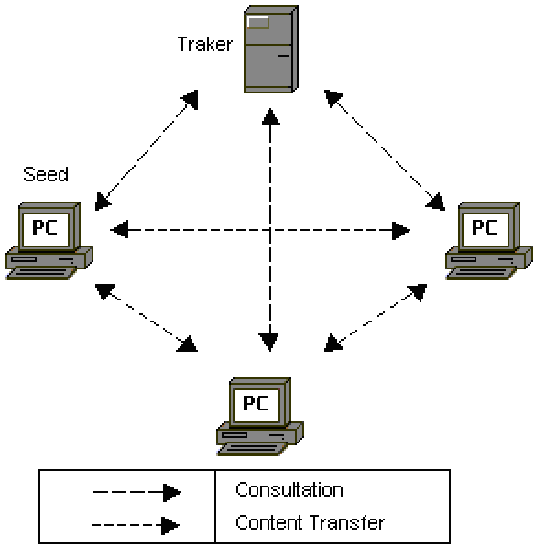

Figure 4. BitTorrent network.

There are two subclasses of partially centralized networks. The first subclass is similar to the centralized architecture, but instead of a single server, there is a farm of servers (see Figure 5).

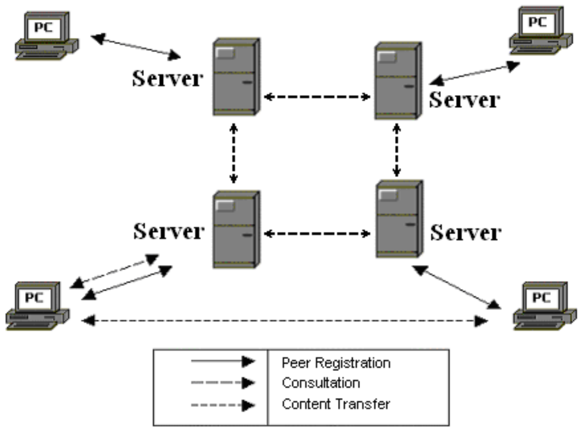

Figure 5. Partially centralized P2P network with a farm of servers.

The second subclass has some peers called superpeers which act as central peers (see Figure 6). These superpeers forward searches to other superpeers when a file is requested.

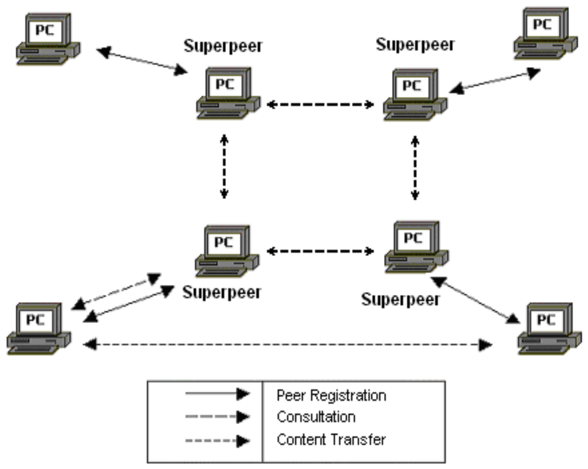

Figure 6. Partially centralized P2P network with superpeers.

Examples of the first subclass are OpenNap, eDonkey and MP2P, and examples of the second subclass are FastTrack and Gnutella 2 [16].

In order to find a file in a P2P file-sharing network, a search process is needed. The type of search algorithm that is used in every P2P network depends on the kind P2P network (centralized P2P, decentralized P2P, and so on). There are several types of algorithms [3] [17]. They are covered below:

- In Centralized indexes and Repositories Model (CIRM), peers are connected to a central server where they publish their shared files and some data such as name, size, etc. The central server keeps a database with the indexes of the clients and their contents. It is used when a peer searches for a file. After a search, the server will send back the name of the files that match with the search, together with reference data and index to the peer or peers that have the file. This model is used by the Soulseek and BitTorrent networks.
- In Distributed Indexes and Repositories Model (DIRM), there is a group of available

servers called "brokers". Each "broker" has the indexes of the local peers and in some cases the indexes of some files from neighbor "brokers". When a peer sends a query to a "broker", it searches in its local database and if it doesn't find a match, it uses the local index in order to find a neighbor "broker" to send the request. The indexes of the server are not static and can change according to the files that are available in the system. OpenNap, eDonkey and MP2P use this model. BitTorrent is a particular case of this model.

- In Flooded Queries Model (FQM), peers broadcast their queries to their directly connected neighbors. If a neighbor has the content, it replies, otherwise, it forwards the query to its neighbors. This model is used by Gnutella network.

- The Selective Queries Model (SQM) is based on the flooded queries model, but in this case the requests are sent to specific peers which may have greater probability of finding the request. Peers with a higher bandwidth and process capacity will be considered automatically superpeers. Peers with less bandwidth will be superpeers leaf. This type of system uses a flow control algorithm for sending queries and replies. It also has a diagram of priorities used to discard some messages. This model is used by FastTrack and Gnutella 2.

- The Documents Routing Model (DRM) is based on Distributed Hash Tables (DHT). In this type of model, the data is placed deterministically, not at random peers. Unique identifiers chosen from an identifier space, called keys, are assigned to data objects. Data object location information is placed at the peers with identifiers corresponding to the data object's unique key. Each peer maintains a small routing table consisting of its neighboring peers' NodeIDs and IP addresses. Lookup queries or message routing are forwarded across overlay paths to peers in a progressive manner (using the NodeIDs that are closer to the key in the identifier space [18]). This model is used by Freenet, as well as CAN, Chord, Pastry and Tapestry.

A very good extended survey of search and replication schemes in unstructured P2P networks is found in [19].

There are different file download systems:

- Single-source download system. The file is downloaded from one or several sources, but not simultaneously. This download system is used by Freenet, Soulseek and MP2P.

- Multi-source download system. The file (or parts of it) is downloaded from multiple sources allowing fast downloads. Normally a "hash" is assigned to the files that allow the desktop P2P file-sharing application to complete and verify the download. The downloading process is performed from the beginning of the file to its end. This downloading system is used by Gnutella, FastTrack, and OpenNap.

- Segmented multi-source download system. It is similar to the previous one, but it allows downloading parts of the file that are not sequential. In order to perform this downloading system, the desktop P2P file-sharing application segments the file in metadata. BitTorrent and eDonkey use this type of download.

But some P2P file-sharing networks can change their download system doing some changes in their protocols.

Each P2P network architecture, each search algorithm and each file downloading system may be the best according to the P2P Network purposes. For example, in centralized P2P architectures, the searches are faster than in other architectures, however, they have a single point of failure and a single index repository. Moreover, they suffer of scalability problems. On the other hand, decentralized P2P architectures perform slow searches, but there is no single point of failure and they don't have scalability problems. From the file download system point of view, multi-source download systems perform faster downloads than single-source download systems, and better than segmented multi-source download systems in case of small files. If the file to download is large, the fastest system is segmented multi-source download.

All these P2P file-sharing networks have their advantages and disadvantages. Moreover, each of them performs better than the other ones according to the working environment or according to a desirable parameter. More information about the parameters discussed in this paper can be found in [3].

3. P2P Protocols Procedure

In this section we describe the protocol procedure of eight Public P2P file-sharing networks: Gnutella, Freeenet, Soulseek, BitTorrent, Opennap, eDonkey, MP2P and FastTrack.

3.1 Gnutella architecture

Gnutella [20] was born as an independent project by a team of programmers: Justin Frankel and Tom Pepper. They were funders of Nullsoft (which was later acquired by AOL) at the end of 1999, which was very fast out of service. However, few time later, different programmers got the client system using reverse engineering and analyzing the protocol. Afterwards, they made public the code and several clients were developed based on the original protocol.

Gnutella network is based on a P2P decentralized architecture with flooding of requests. A simple desktop program is needed in order to connect to this P2P file-sharing network. When the desktop application is executed, it searches for other peers in the P2P network. Those ones that reply will be connected as neighbors (only one is enough to join the network). Once the peer is connected to the Gnutella network, it sends its IP address and the other peer replies in order to become its neighbor. After some time, a pyramidal system of connections begins to learn to which peers are the new peer able to reach. In order to do that, the application sends ping message to the other peer with a TTL number (Time to Live, is a field in the header of the IP packet) which gives the number of times that the message can be forwarded. This number is seven by default, but it can be settled by the user. It is decremented in one unit in each jump and is discarded when it reaches cero. The other peer will reply with a pong message, which has its IP address together with the information about the files that is sharing. Meantime, the IP address of our peer is being spread through the P2P file-sharing network so much times as we have set in the TTL. There is a limit in the number of connections, because a user that settles the TTL value higher than seven can generate in seconds a flow of trillions of pings in Internet. In the Gnutella architecture, the ping is 50% of the traffic and the requests are around the 25% of the total traffic. The architecture gives preference to the search of information versus the download of files.

Afterwards, the user can search the desired file. This request is sent to all its neighbor peers. Those peers will send the search to others. This forwarding process is repeated until the search gets a result or the TTL value decreases to zero. Each peer has a table with its previous searches in order to prevent sending twice and to advise routing loops inside the Gnutella network. This table has all the information needed so it will be used as a reply when another peer performs the same search. Once a user receives the list of the files found in the P2P file-sharing network, the user just has to select one of them and the desktop program will send a request to the peer that stores the file.

Because of the TTL value, there is the search limitation, and, moreover, Gnutella can't have scalability. One of the most meaningful disadvantages of this network is that a peer is very dependent of its neighbor peers, and can be exposed to random events such as node failures, loss of connection, etc. That is, when connection with other peer is lost, the connection with other users that were connected to it, are also lost. This makes Gnutella network very fragile.

There are many open code desktop applications that support Gnutella protocol. These are the most well-known ones: Limewire, Bearshare, Shareaza, Morpheus (first it belonged to FastTrack network, then it changed to Gnutella), Xolox, Gnucleus, Freewire, Atomwire, Newtella, Ares, NeoNapster, Gnotella, Gnutella, Ntella, Toadnode, Filetopia, Phex, Swapnut, etc.

In general, all these desktop applications have the following features:

- Use TCP 6346 port.
- They are able to search for, share or download all types of files: video, audio, applications, images, documents, etc.
- They generally use the same file search parameters (quality of download, name of file, kind of file, IP and kind of client, time, number of sources, speed, etc).
- The maximum number of files to serve and download at the same time can be defined.
- Allows all operating systems (multiplatform).

3.2 Gnutella2 architecture

In this open P2P file-sharing network [16], not all the nodes that participate in the system are equal. Gnutella 2 nodes have rules for sending, filtering and security. Gnutella 2 divides the nodes into two groups, leaf nodes and concentrator nodes.

The leaf nodes are the most common ones in the network. They don't have any special responsibility and are not part of the infrastructure of the network. They use to be nodes with limited resources (low bandwidth, little RAM or CPU, little capacity to accept many TCP or UDP connections, etc.). Leaf nodes are considered the end of the network. In practice, these nodes use to connect to 2 concentrators simultaneously, however from the point of view of the concentrator, each node will be considered as the end.

Concentrator nodes are an important and active part of the network infrastructure, because they organize leaf nodes and filter and redirect the traffic. Only the most capable nodes can act as concentrators. The selection criteria followed to become a concentrator node are: to be able of supporting more than 100 connections, to have enough bandwidth, to be able of accepting TCP and UDP connections, etc. The concentrator nodes must have many connections between them in order to form a network of concentrator nodes. The number of connections between concentrators must scale with the total size of the network. Each concentrator node must accept multiple leaf node connections (between 200 and 300), depending on the available resources.

The concentrator group that includes the local concentrator and its neighbors is named cluster of concentrators. Clusters of concentrators maintain constant communication between them by sharing information about the network load, statistics, exchanging cache entries and filtering tables. The cluster of concentrators is the smallest search unit of the network for a peer.

Concentrator node must contain updated information of the other concentrators in the cluster. It must maintain a routing table of the nodes in order to map the GUIDs of the nodes with the shortest route of TCP and UDP connections. Moreover, it must maintain hash tables of requests for each connection from leaf nodes as well as concentrator nodes. It also must have a hash table for the local content and the content of the leaf nodes that will be provided to the neighbor concentrators in case of a search. The concentrator node indexes the file of a leaf node by means of a routing table of requests with key words. Leaf node uploads this table to the concentrator. Subsequently these key words are sent to the neighbor concentrators in order to reduce the number of requests that are forwarded between peers. This reduces the used bandwidth because if a search is received, and this search is not found between the words of the routing tables, the search is not forwarded. This allows a user to find a popular file located in any part of the network easily without too much load over the P2P network. The success is to maximize the number of leaf nodes and minimize the number of concentrators; nevertheless, because of the limited nature of the resources, the maximum ratio of leaf nodes per concentrator is limited. The searches are performed over UDP. TCP is used to establish connections between nodes.

Gnutella 2 protocol uses HTTP/1.1 to serve services and upload, download and transfer files. The protocol messages have XLM format. It uses the protocol GNUTELLA CONNECT/0.6, of Gnutella v.0.6, for its connections. The port used by Gnutella 2 is 6346.

On the other hand, the system uses compression for the network connections in order to reduce the used bandwidth.

Gnutella 2 uses the hash SHA1 for the identification of the files. Thus, only one file can be downloaded at the same time from several sources. It also allows uploading different parts of a file in parallel.

In order to create a robust searching system, Gnutella 2 has implemented a system of metadata to label, establish the ratio and the information quality in the file names that is displayed when it shows the result of the searches. Users can share this information in order to delete a file. For example, this allows adding labels informing that there is a virus or worm in a file without the need of keeping a copy of the file locally.

Given that Gnutella 2 is free and open for all the users, many developers have included this protocol in their desktop client applications.

3.3 Freenet architecture

Freenet [21] was created by Ian Clarke in 1999 while he was at the University of Edinburgh (Scotland). His intention was to create a free communication network over internet. The goals of this P2P file-sharing network are to create and keep a fully virtual and decentralized file system for file storage and recovery. Nowadays, its main features are that the storage and recovery is anonymous, efficient and robust. Nobody controls this architecture (even its own creator).

The design of Freenet is based on the assumption that the P2P network is not reliable and formal, because anyone can be a node in Internet. Its users act maliciously or can fail at any time without warning. It has good reliability even for malicious attack. The most of the research in Freenet has been based on how the requests are routed through the P2P network. Freenet computes the addressing (hashing) and other strong techniques of code to reach its goals. The information sent between peers travels encrypted. Moreover, it is routed between peers but without letting the intermediate peers know neither the sender nor the content.

In this P2P file-sharing network, a peer shares bandwidth with the rest of peers (because it forwards their searches), and space in the hard drive. But, Freenet doesn’t allow the user to manage this space, thus the user can’t know which content is there. Peers use passwords to save and recover data files. Each peer keeps its own local “warehouse” of data and keeps it available to the peers of the network in order to write and read them. Moreover, they maintain a dynamic routing table that keeps the address of the other peers and their passwords (see Figure 7). All peers in the Freenet network have the same privileges and are treated equally.

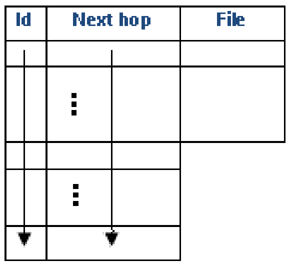

Figure 7.Routing table in each peer.

Data are identified by binary passwords that are obtained randomly by the application (SHA-1 of 160 bits). There is no matter in which position they are placed. Each file is stored, recovered and kept with its file password. There are three kinds of file passwords:

• Keyword-signed key (KSK). It is the simplest kind of file key. It comes from a short description of text selected by the user when a file is stored in the network.

• Signed-subspace key (SSK). It enables the personal namespaces.

• Content-hash key (CHK). It is useful for implement actualization and assignment.

The files shared in the architecture are deleted or kept in terms of its popularity. Therefore, the life time of the file is not controlled.

Freenet protocol operation is as follows. At the beginning, a peer must obtain or compute a key. When the peer sends a search message specifying the key and the TTL designed for this search message. When a peer receives a request, it checks its own data storage and gives it back if the key matches. If does not match, the peer will search the most close key to the key asked in its routing table, and forwards the request to the due peer (the most close key means the most close value in “hashing”). If this request has finally succeeded, the peer that has the objective’s data, gives it back through the path followed by the search (the file is duplicated in each peer in which it has passed through the search message). The final peer keeps the file and creates a new entry in its routing table. If the TTL finishes before reaching the goal, the search will be considered failed and a failure response will be send to the request peer.

In a N-peer network, Freenet needs Log(N) jumps to find the request. The dynamic routing of Freenet is duly adapted to network topology changes.

Figure 8 describes a typical request sequence. *a* peer sends a search request to *b*, which forwards to *c*. *c* peer has no connection with other peers and send back a “failure” message to *b*. Then, *b* peer takes its second election, *e*, and which sends the request to *f*. *f* makes the same process done by *b*, which detects a loop and sees that it is unable to connect with other additional peers, thus *f* peer sends back a message to *e*. Finally, *e* sends the request to its second election, *d*, and locates the file. *d* peer gives back the file by the same direct search path to *a* (that is *e*, *b*, and *a*).

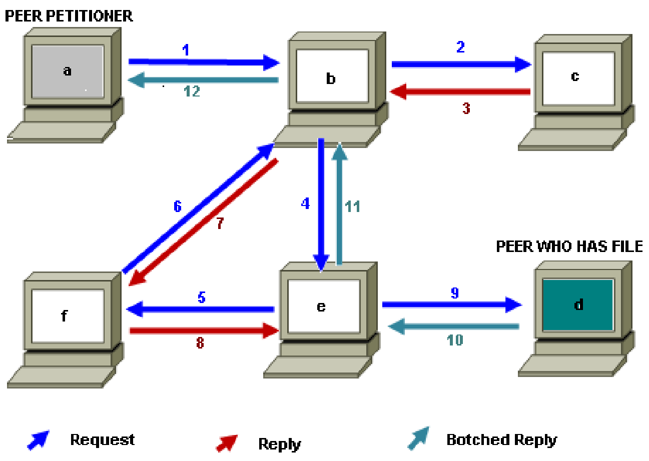

Figure 8. Freenet Search

New peers announce their presence in the network making a search operation. Before sending the message, which includes the peer IP address and its public key, the new peer must know the existence of other peer in the network at least.

Freenet network provides the storage and recovery of information anonymously. Given the cooperation of all peers in the network and a reliable and adaptable routing algorithm, the anonymous information is kept and ready to be used, while is highly scalable.

The most well-known desktop client application, Freenet, was developed by Ian Clarke. A lot of the developed software for this P2P file-sharing network are tools or complements for the original application. The desktop client application doesn't have an assigned port. It is based on the knowledge or discovery of other peers. Some desktop client applications are Freenet, Fra Zaa, Liberator, FreeWeb, fhc, etc.

In general, all the client applications of this P2P file-sharing network have the following features:

- The search inside the P2P network is complicated be performed.
- They spend quite CPU resources, because they are programmed in Java.
- They encrypt the data that are distributed inside the P2P file-sharing network.

3.4 Soulseek architecture

Soulseek P2P file-sharing network [22] works like Napster, with only one centralized server that manages all database of the P2P file-sharing network. Nowadays, it allows the transmission of many types of files (not only mp3 like in its beginning). The main problem is that when this server fails, the P2P file-sharing network stops working.

The desktop client application let the users choose which remote users can download from the local peer. In addition, a user can create a list of users with download privileges. The system allows knowing which peers don't have anybody in the download queue.

In order to make searches, the user can use jokers (like *) and operators (like & and -). The search is ignored when a word with less than 3 characters is searched. Moreover, the desktop client application has a tool named "wish list" that will keep searching in the P2P file-sharing network until a peer with the requested file joins the network. It also allows organizing downloads and downloading whole folders of other users.

Soulseek P2P file-sharing network has fast searches because of it is a centralized architecture. The protocol designed for this architecture does not allow downloading files from different sources simultaneously.

There are very few desktop client applications for this P2P file-sharing network. The most well-known ones are Soulseek, PySoulseek, Nicotine, SolarSeek and SoulFreek. All of them need very much processing capacity. Moreover, they have a closed code, but several development teams have been able to create open code desktop client applications by using reverse engineering.

In general, these desktop client applications:

- Use TCP 2234 and 5534 ports.
- Can search, exchange or download any kind of file (some time ago they only were able to download mp3).
- Use several types of search parameters (file name, size, time length, bit rate, frequency, user, speed, ping…).
- The maximum number of files to open and download simultaneously can be defined.
- Allow chat with other users.
- Allow seeing the shared files of other users.
- They do not have bothering banners or spyware.

3.5 BitTorrent architecture

BitTorrent protocol [23] is designed to transfer files. From the point of view of the content, the architecture is completely distributed. It doesn't need any central server. However, it is needed an initial file to begin to download the wished file. This file can be downloaded from any place, e.g. a web server. Almost all BitTorrent client applications are able to create this initial file with an extension ".torrent". It is formed by the information gathered from the shared file, which is virtually cut in very small pieces, and the information provided is the hash of the pieces of the file. The file .torrent has the information to download the file, not the file itself. It can be stored in the hard drive in order to start the download afterwards.

In BitTorrent file-sharing network, the peers are directly connected between them and they receive parts from the file. However, there is a central server (called Tracker) that coordinates the activities of the peers (see Figure 9). The Tracker only manages connections. It does not know the content of the files that are being distributed inside the P2P network. Therefore, it can maintain a great number of users. The peers of BitTorrent P2P file-sharing network must upload at the same time that they are downloading. Hence, the bandwidth is managed efficiently. Unlike other P2P protocols, BitTorrent works better when the number of users interested on a file is higher.

All peers in the P2P file-sharing network acts like a servant (Client/Server) for the rest of the peers. Let's suppose that all peers are interested in downloading the same file that is split in six parts. The first peer will inform that it has the parts A, C and D, the second peer will inform that it has A, B and D, and so on. Peers coordinate to exchange parts of the file until all of them download it completely.

There is also another peer called seed (*S*). It has a full copy of the shared file, so it doesn't need to get anything from the rest of the peers. This peer shares parts that no other peer in the P2P file-sharing network has. At the beginning, when a new peer joins the network it must communicate first with the seed in order to its piece of the file. However, peers don't get all parts of the file; rather each peer gets one different part. In few time, all peers have most part of the file, although no one has it completed. Thus, a peer can share a file with the rest of peers, without having the complete file.

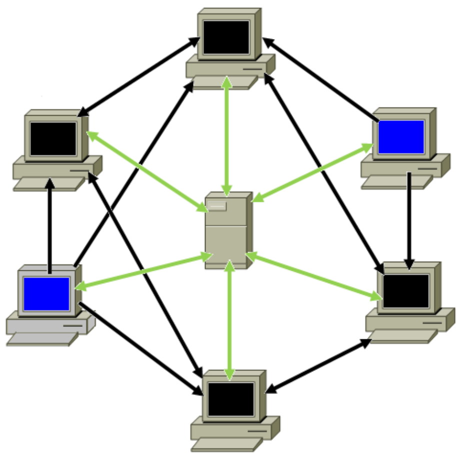
Figure 9. Interconnection between nodes and servers in the BitTorrent architecture.

The tracker coordinates the action of the BitTorrent peers. When the file ".torrent" is opened, the peer contacts with the tracker and asks for a list of peers to contact with in order to download the file. During the download, the peer contacts periodically with the tracker in order to inform how much it has downloaded and uploaded, how much time it has left the file shared (once it has been downloaded), and the status in which it actually is (beginning, finishing, downloading, stopped). The tracker divides the file in pieces of 512Kb. For example, a file of 700Mb will be divided into 1400 pieces.

When someone decides to upload a file to the BitTorrent P2P file-sharing network, first it should send the "nameoffile.torrent" to a tracker and the peer will become automatically a seed. Then, when a peer wants to download it, the peer will open this torrent file by using a desktop client application. The .torrent file has the place to download the file, so it starts to receive the pieces of the file, whether from another peer or from a seed. When it gets 100% of the file, automatically it becomes a seed. The more seeds and peers, faster the downloading of the file will be. Because of that, it is important to keep sharing the file although it has being downloaded completely. Files disappear when the Tracker discards them. It usually happens at 24 or 48 hours (it depends on the Tracker).

A peer does not always download from the same peers, but it changes. If the .torrent file is corrupted, the peer will not be able to contact the other peers.

The size of a BitTorrent network depends upon the capacity of the tracker. If the generated traffic is higher than the capacity of the tracker, a bottle neck will be formed.

The operative cost implied in the small file transfer (of some Kbytes) is very high. The total bandwidth spent by the protocol messages is meaningful high. We also must take in consideration that the protocol is not so robust, this causes that the nodes are exposed to possible attacks.

The official desktop client applications were developed by Bram Cohen, but there has appeared more clients for different operating systems (and platforms) because it is open code. These are some of them: BitTorrent Client, BitTornado, Azureus, ABC, burst!, Tomato Torrent, PTC, TorrenTopia, G3 Torrent,TorrentStorm, Turbo Torrent, BitSpirit, BitCometBTQueue, Flash! Torrent, TurbotBT, CTorrent, BitTorrent + +, etc. There are also plugins to support the BitTorrent protocol in programs like Shareaza or eDonkey.

Generally all desktop client applications in this P2P file-sharing network have the following features:

- Use 6881 to 6889 TCP ports.
- Many of them do not allow searching shared flies. The search of seeds is done with a different application or through Web browsers.
- Allow downloading multiple files simultaneously.
- Allow creating systems of priority queues.
- A download can be stopped and resumed.
- The maximum number of files to upload and download simultaneously from other users can be settled.
- Create the .torrent file.
- Display error statistics and downloads.
- Do not contain spyware, adware or any other unwanted hidden software.

3.6 OpenNap architecture

The first time that the desktop client program is executed, the user fills up some data such as the user name and password. This information is sent to the servers located in the company dependencies, where they are stored. This information will be sent each time a peer joins the P2P file-sharing network. Then, it sends the names of the shared files (chosen by the user with the desktop client application) to the central server. After those steps, the files are ready to be found in a search.

When a search is performed, the client sends a request to the server, which looks for those peers that are sharing that file in their data base. If the requested file exists, the central system provides the IP address of the peer in order to let the peer connect directly and start the transfer.

There is not only one OpenNap server, and not all OpenNap servers are similar. This P2P file-sharing network uses the same structure than Napster. It is also based on a central server, with the innovation that this one itself is subdivided into a network of servers named OpenNap [24]. It makes a less overloaded network and allows fast searches. The data base of is shared in the cluster of servers that support thousands of connected peers. In order to balance the number of connections to each server and simplify the QoS tasks, before connecting to the database, the peer consults the metaserver (it is a server with higher load capacity and bigger bandwidth).

The connection with the metaserver is performed by the TCP 8875 port, however the connection between peers and servers is using TCP 8888 and 7777 ports. The port used to listen new connections is, by default, TCP 6699. All this ports can be changed manually.

For the transmission of the files between peers there are some available options: download, upload, firewalled download (download through a firewall) and firewall upload (upload through a firewall). Last two modes invert the connection process, i.e, if a peer, where the file is placed, is located behind a firewall, this one will be who will establish the TCP connection.

There is a great quantity of OpenNap servers, thus it is usually difficult to lose the connection of the peer with the servers. The searches can give very geographically scattered results, and only provide links to those data that are reachable by the server which the user is connected to.

The protocol allows splitting big files into smaller pieces in order to increase the download speed. The architecture allows downloading any kind of file (video, audio, pictures, documents, applications, etc.).

Because it is an open code protocol, there are many desktop client applications that are able to connect to this P2P file-sharing network: Audiognome, Duckster, Gnapster, Gtknapster, Hackster, Jnapster, MacStar (for Macintosh), Lopster (for Unix), Xnap (for Linux), Swaptor, OpenNap, TekNap, CQ_EX, Sunshine UN, DM Napster, Napigator, etc. But there are also other closed code OpenNap clients: WinMX, FileNavigator, Rapigator and Spotlight. Some clients can use several network protocols (the OpenNap network and their own P2P file-sharing network), for example WinMx, FileNavigator y Swaptor.

Generally all these desktop client applications have the following features:

- Use 6699, 7777 and 8888 TCP ports.
- They are able to search for, share or download all types of files: video, audio, applications, images, documents, etc.
- Use the same search parameters file (file name, size, length (sec), bit rate, frequency, user, speed, ping, etc.).
- The user can set the maximum number of simultaneous uploading and downloading files.
- Show the number of users connected to the network.
- Some of these applications do not exceed 1 Mbyte.

3.7 eDonkey architecture

eDonkey [25] protocol operation is as follows. A peer must connect to a server (there are hundreds). The main features of the servers are in the nationality, the ping, their users (and thus their sharing files), the maximum number of users, number of files and the preference. It is a distributed and the servers are not centralized. Most of them are private, although any user can install a server and connect it to the P2P network (high Internet upload and download bandwidth is required). Therefore, a new peer needs an updated list of available servers in order to send its connection request. Moreover, servers must have also an updated list of known servers. When the peer joins the P2P network it also receives and updated list of servers, which dynamically updated by the peer (when the peer send a connection request and the server does not reply, it is removed from the list).

Once the connection has been established with a server, it will receive an ID that is estimated following equation 1.

$$ID = D\bullet(256)^3 + C\bullet(256)^2 + B\bullet(256) + A \tag{1}$$

Where the peer IP address is A.B.C.D. If the ID is higher than a million, it will be able to connect to any user, and the download queues will not be lost in case of disconnection from the server, and reconnect to another one (the ID doesn't change). If the ID is less than a million, it will be able to connect only to the users that have high ID, and therefore, if it connects to less peers (less connections), it may download with lower speed (but it does not happen always because it depends on the users that share the file). In this case, when it disconnects from the server, and begins to connect to another one, another ID will be assigned, so the download queues will be lost, which implies the need of more time to start or end the download of a file. The case of low ID happens when all the ports are blocked or not enabled.

After being connected, the peer sends to the server a list of its sharing files, so each server keeps a list of all the shared files of all peers connected to it. When a peer looks for a file (what the user really sends is a search for words that will be looked at the name of the files), it sends the search to its server. The server creates a list with all peers that have that file and it will forward the request to all servers it knows in order to see if there are more peers with that file. Once it has found all the peers with files with the searched words in the P2P file-sharing network, it sends the detailed list to the peer. When a peer selects to download one file in the list, the server adds the request to the queue of peers for that file. This request and the results are sent using UDP protocol in order to reduce the bandwidth and connections through the servers.

The architecture uses a system of credits, i.e., the more files download other peers from a peer, the faster is the download ratio of that peer in the P2P file-sharing network.

A user can download files directly by selecting the file from the received results or by downloading a "ed2k-quicklink", which is a link with the hash of the file that provides the information needed to download the file from other peers. Because the search is performed in each server separately, it is the faster option. A file can be downloaded from several peers simultaneously; this increases the effective download bandwidth. The protocol used to download files is the MFTP (Multisource File Transfer Protocol) that allows downloading different file pieces from multiple sources simultaneously. Besides, the files can be shared without being completed.

There are many servers in the eDonkey network, thus the search for files is fast. Furthermore, it allows to share any kind of file (video, audio, pictures, documents, application, etc.). Some of the most widespread desktop client applications are: eMule, eDonkey2000, Overnet, xMule, amule, Pruna, ed2k-gtk-gui, etc.

Generally all these desktop client applications have the following features:

- Use 4661 and 4662 TCP ports.
- They are able to search for, share or download all types of files: video, audio, applications, images, documents, etc.
- Use the same file search parameters (file name, size, availability, and file handle color).
- The user can set bandwidth and connection options.
- They remove failed servers automatically, reconnect in case of failure or loss, and even stay always connected.

3.8 MP2P architecture

MP2P [26], MANOLITO P2P protocol, was developed by Pablo Soto in 2001. he developed the In P2P file-sharing network, a peer joins the network by establishing a connection a server through the ports 80, 3128, 8080 or 8088. The architecture is formed by several servers, but in this case the IP address of the peers connected to the P2P file-sharing network is encrypted.

P2P uses a communication protocol based on UDP for both type of connections, between servers and between peers. MANOLITO is capable of testing up to 15,000 hosts per second, thus the search and connection speed with the servers is much faster than the traditional TCP/IP protocol. It neither provides the identification of the connected user nor what is sent through the P2P network. This provides anonymity of the connected users.

The connection between peers for data transmission is performed through UDP 41170 port. The communication between peers is faster than other P2P file-sharing networks thanks to the use of the UDP protocol.

This architecture uses a series of gateways. Moreover, it uses a compression technology to save thousands of IP addresses of the peers in an encrypted cache. It allows the peers to be connected to the P2P file-sharing network in case of failure of the servers. Its centralization degree and the existence of gateways-Web cache to support the P2P file-sharing network, it is very scalable. Moreover, the MP2P do not let watch the content of the frames through the network infrastructure. The protocol has a system to verify the integrity of the sent data, avoiding false or corrupt files.

The protocol was designed to only download music files in mp3, OGG and WMA format. The protocol is not designed to download files from multiple sources.

The MP2P file-sharing network has very few desktop client applications. All of them have closed code. These only work in Windows Operative System (neither Macintosh, nor Linux). The most well-known desktop client applications are Blubster, Piolet, and Rockitnet.

Generally, all these desktop client applications have the following features:

- Use UDP 41170 port.
- Use the same file search parameters (name, size, availability, and file handle color).
- The user can set the maximum number of simultaneous uploading and downloading files.
- Have a text chat and voice communications.
- They have advertising banners.

3.9 FastTrack architecture

FastTrack file-sharing network [27] was created by Niklas Zennstrom and Janus Friis in March 2001. It is based on the selective flooding queries model. It uses superpeers, which are peers with higher bandwidth than regular peers in order to stabilize the network. FastTrack do not belong to any company. All the peers (and superpeers) are connected directly without the need of a central server managing the network. The architecture assigns automatically the role of the superpeers to those peers with higher bandwidth and processing capacity. These superpeers act as regular peers and as servers when there are higher traffic needs. This property also can be also configured (or disabled) in the desktop client application. A superpeer connects to more peers than regular peers (also called leaf peers). The system can also indicate to other peers of the P2P file-sharing network that a superpeer will not be a superpeer anymore. Given the nature of the architecture, it is very difficult the failure of the system. Even disappearing many peers, the network keeps working.

The superpeers system used in FastTrack network has some differences compared with the one used in Gnutella II network. The main one is that in FastTrack network there is a need to connect to a registered server when it joins the network. This server carries out exclusively the authentication of the registered peers and registers active sessions. After the authentication of a peer, the server sends the IP address and the port of the superpeer to which it must connect with. This system forces all the peers of the P2P file-sharing network to be registered, like it happened in Napster.

The search is performed through the superpeers, because they have the fastest connections. Because the most powerful devices are assigned to perform the searches, results come faster, thus avoiding the main bottleneck given in decentralized P2P networks without superpeers. Connections between the leaf peers and the superpeers are performed by the TCP and UDP 1214 port. A superpeer receives small lists of the shared files by its leaf nodes. When a leaf node sends a search, it is sent to the nearest superpeer. First, the superpeer will reply the result of its search in its local table (formed by its leaf nodes). Then, the superpeer forwards the search to other superpeers and so on, until a predefined hop search limit is reached. This process makes faster searches faster, but a peer is not able to find the files in all superpeers of the network, only in the leaf peers of the reachable superpeers.

Once the search file is located, the downloading process is performed directly between the requesting peer and the peer that is sharing the file (not through superpeers). Big files are virtually divided in several pieces, to increase the downloading speed. The more files are downloaded from a peer by other peers, higher will be the level acquired by that peer. There are three levels of participation: Low (when the peer has downloaded more Megabits from other peers that the other peers from it), Medium (when the other peers have downloaded a considerable amount of Megabits), High (when other peers have downloaded much more Megabits from the peer than the peer from the others). Generally, a high participation level means that the peer has been connected for long periods of time and many users may download files from it. Some desktop client applications, such as KaZaa Lite, estimate the participation level numerically. In this case, peers reputation is reflected by their participation level. A peer starts at participation level 10 and can get a participation level between 0 and 1000. Peers with higher participation level are favored in queuing policies and should receive better quality of service (QOS). Moreover they receive more results in their searches (the desktop client application shows more results). In order to calculate the peer level, equation 2 is used:

$$Level = \frac{(upload(MB))}{(download(MB))} \times 100 \qquad (2)$$

On the other hand, a rate can be assigned to the downloaded or shared files, either by technical quality or by content. Peers that rate the files double the value of the Megabits that can download and have higher priority in the download queues. Its philosophy is to compensate the peers that share files and punish the peers that do not do it. This also allows a peer to promote its content.

The most well-know desktop client application has been KaZaa. But there have been others like KaZaa Lite, Mammoth, and iMesh.

Generally, all desktop client applications in this P2P file-sharing network have the following features:

- Use 1214TCP port.
- They are able to search for, share or download all types of files: video, audio, applications, images, documents, etc.
- Use the same file search parameters (title, artist, filename, size, availability,...).
- The user can set the maximum number of simultaneous uploading and downloading files.
- Sorts files more efficiently in order to perform an optimal search.
- Consume quite a lot of processing memory.

3.11 Summary

A summary of the analyzed P2P file-sharing network protocols is shown in Table 1. In this table, on the one hand, we compare the architectures in terms of the grade of centralization, discovery and search algorithm, file download system, the type of file shared inside the network. On the other hand, we also compare the P2P file-sharing protocol in terms of transport layer protocol, protocol port, and string being delivered inside the messages. We can see that Gnutella and Freenet use a decentralized architecture. They are the pure peer to peer protocols. The rest of the protocols use partially centralized architectures except SoulSeek, which is based on a centralized architecture. The distributed indexes and repositories model (DIRM), is the most used discovery and search algorithm. It is used by BitTorrent, OpenNap, eDonkey and MP2P. All P2P networks except MP2P use TCP (with different ports) as transport protocol, but only Gnutella, eDonkey and MP2P use UDP.

The string used to start a P2P communication is different in all cases. In some of them we cannot know this parameter because it is not shown in the message. All P2P file-sharing networks, except MP2P, a peer can download any type of content. MP2P only shares audio files inside the P2P network.

Table 1. Comparison of analyzed P2P networks (* SQM in case of Gnutella 2, n/d: not defined port).

P2P Networks	**P2P Networks Characteristics**							
	P2P Architecture	***Discovery and Search Algorithm***	***File Download System***	***File Type***	***Protocol***	***Port***	***Protocol Name***	***String***
Gnutella	Decentralized	FQM/SQM	Multisource	All	TCP	6346, 6347	Gnutella	"GNUT", "GIV"
					UDP	6346, 6347	Gnutella	"GND"
FastTrack	Partially centralized	SQM	Multisource	All	TCP	1214	FastTrack	"Get /.hash"
Freenet	Decentralized	DRM	Single-source	All	TCP	n/d	Freenet	
BitTorrent	Partially centralized	DIRM	Segmented Multisource	All	TCP	6881-6889	BitTorrent	"0x13Bit"
OpenNap	Partially centralized	DIRM	Multisource	All	TCP	6699, 7777, 8888	OpenNap	-
SoulSeek	Centralized	CIRM	Single-source	All	TCP	2234, 5534	SoulSeek	-
eDonkey	Partially centralized	DIRM	Segmented Multisource	All	TCP & UDP	4661-4666	eDonkey	0xe319010000
MP2P	Partially centralized	DIRM	Single-source	Audio	UDP	41170	Manolito	GO!!, MD5, SIZ0x20

4. P2P network traffic analysis

Many new internet users, using broadband Internet access are also using P2P file-sharing networks. Most ISPs have found that P2P file-sharing usage creates high latency, congestion and performance problems in their broadband networks [28]. The network performance is significantly degraded when P2P file-sharing applications are used, especially when large files are being downloaded. This problem becomes higher when other users of the P2P network use the bandwidth of an enterprise to download files from the employee's computer. It can significantly slow other services such as e-mail and Web browsing, and therefore the e-commerce. Sometimes there are other network services, such as interactive games or remote terminal connections, which are very sensitive to network delay. A single P2P user is able to use all available bandwidth and cause large delays to other users. Thus, it may affect the Quality of Service (QoS) for all subscribers and often causes unplanned network expenditures [29]. The impact of P2P traffic in large scale networks is analyzed in [30].

There are some published works where authors study a specific P2P file-sharing network.

In 2006, M. Yang et al. after having measured many traffic, propose an analytical model for file diffusion in a peer-to-peer (P2P) file-sharing network based on biological epidemics [31].

Some ISPs observed a rapid congestion in their networks, and sometimes P2P traffic reached nearly 60% of the total traffic [1]. Although not so striking, Internet2 administrators also computed impressive results on 16 February 2004, where 10.46% of the total traffic was originated by P2P file-sharing [32]. CAIDA (Cooperative Association for Internet Data Analysis) also shows that Internet traffic is mainly dominated by P2P file-sharing protocols and HTTP [33]. In reference [34], some researchers collected the trace of all incoming and outgoing network traffic at University of Washington, USA. They quantified the rapidly increasing traffic in new content delivery systems, particularly P2P networks, and the derived caching implications in these systems. In [35], the study carried out in the campus of Southern California about the network traffic is based on connection ports techniques and patterns, i.e. they used detection techniques based on inherit methods, port-based methods or both. In [36] authors performed a study of P2P traffic in three European countries. They collected traffic for more than a year and a half through 5 strategic points for different access technologies in order to know users' habits. They saw that there was a P2P traffic decline but it stopped in all points and it stable since July 2010.

There are several measurement studies for Gnutella and Napster P2P file-sharing networks [37]. They show the percentage of downloads, uploads and shared files grouped by reported bandwidths. On the other hand, Napster reached 11 million connected users with 1.5 billion of shared files in July 2000 as it is shown in "The Pew Internet & American Life Project's Online Music Report" [38]. The report shows how many users download music online and what they think about it. Some documents study the economic cost of downloading a file using P2P networks. E.g. the work in reference [39] analyzes the required time to download movies, music albums, software and documents and shows some calculations about their average price. In [40], the authors presented a traffic profile for the eDonkey P2P file-sharing service. The measurement-based study revealed a strong distinction between download flows and non-download stream. Another interesting data is presented in [41], which says that in 2004 BitTorrent traffic made up 53% of all P2P traffic, and 30% of the total traffic. However between 2001 and 2002, P2P traffic was dominated by Kazaa [42]. In the 2007 BitTorrent traffic had for more than 60% of the total network traffic [43].

In order to know more information about the evolution of these P2P networks we performed several studies in [3], [44], [45] and [46]. In these papers we did several studies about P2P architectures showing their evolution, what kind of files (audio, video, software, documents) are more common, etc.

Now we present several measurements taken by us in 2006. They give us the amount of users, files and the size of the files for six P2P file-sharing networks. They show us the amount of users using these networks and data being shared in order to know their impact and show that, in many networks, there is a need to control it. Currently, P2P networks have still high usage, but their number of users has decreased because there are others file-sharing methods. In Figure 10 the number of users for different P2P networks is shown. The network with more users is FastTrack, with an average number of 3,467,918 users, then eDonkey with less than half of users and, finally, the network with less users is SoulSeek (about an average of 9000 users).

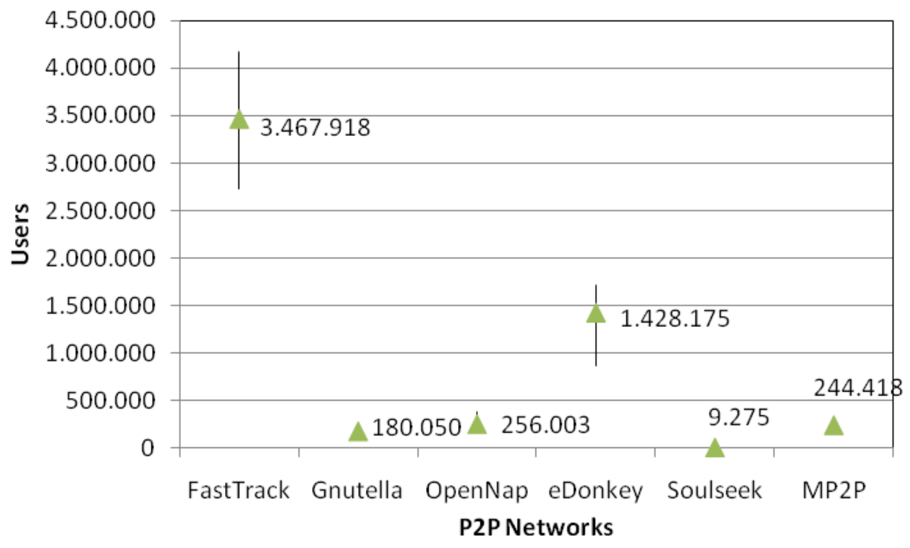

Figure 10.Number of users in several P2P file-sharing networks.

The number of files is shown in Figure 11. In this case, FastTrack P2P file-sharing network is the one with most files. This happens because it is the most used network. But, while eDonkey was the second P2P file-sharing network, according to the number of users, if we see the number of files, it is in the third position. On the other hand, OpenNap has very few users, but the number of files is high (it is the second P2P file-sharing network when we take into account the number of files).

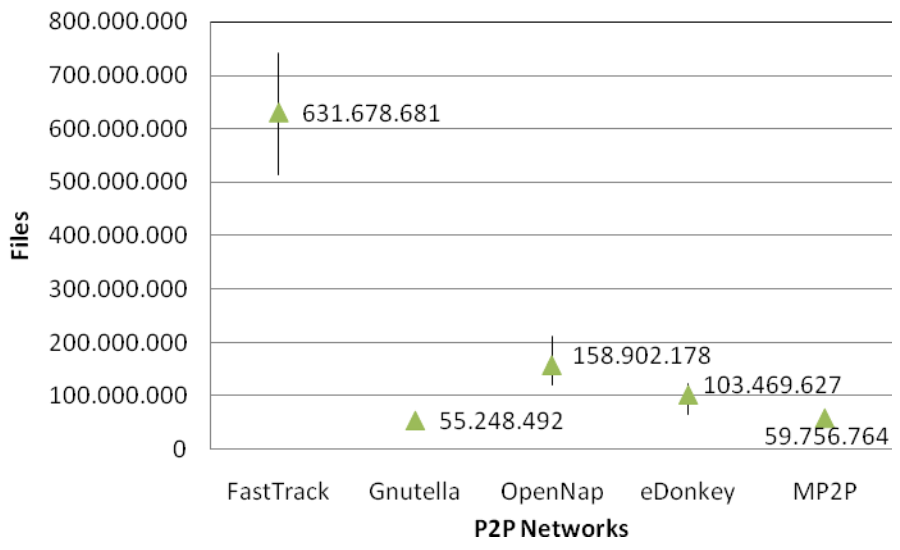

Figure 11.Number of files in several P2P file-sharing networks.

Finally, we measured the size of the files in those P2P file-sharing networks. It is shown in Figure 12. This information can only be obtained from FastTrack, OpenNap and MP2P

P2P file-sharing networks. OpenNap network is the network that has the biggest files. It is followed by FastTrack. Finally, MP2P network is the one with smallest files, this is because MP2P only shares music files.

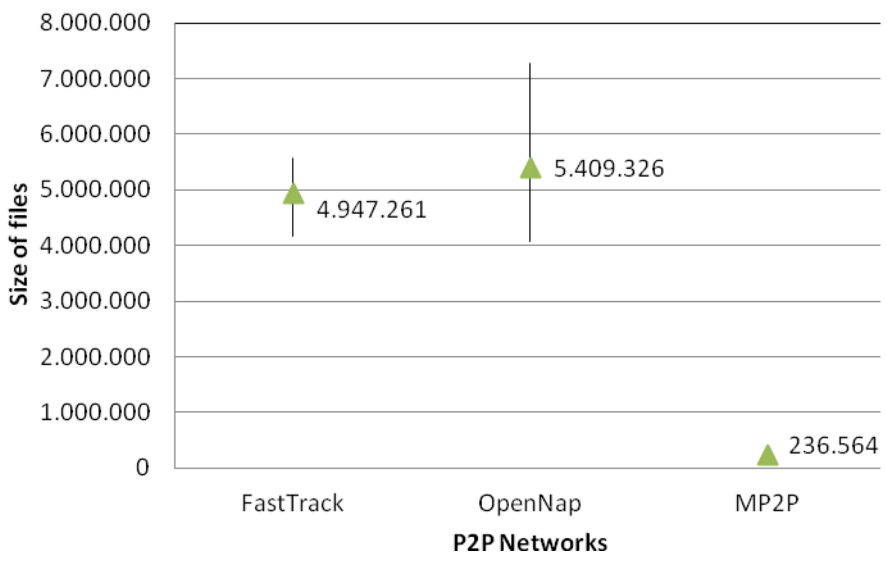

Figure 12. Size of files in several P2P file-sharing networks.

The previous analysis on the P2P network traffic shows that there is a huge amount of users and files inside these P2P file-sharing networks, so it is important to be managed by ISPs. The number of users connected to P2P file-sharing networks and the number of files placed in those networks are increasing daily. FastTrack, OpenNap and eDonkey have more than four Pentabytes shared inside them.

In [47], the authors review some important techniques for ISPs in order to manage P2P traffic. They show how to perform blocking, caching and localization, and, moreover, they compare their advantages and disadvantages. In general, blocking is used to limit P2P usage, while caching and localization are used to control P2P. The simplest way to reduce P2P traffic is to identify it and then block it. Obviously, ISPs' traffic blocking would degrade users' experience and result in battles between ISPs and Internet users. In [48], authors developed a systematic methodology to identify P2P flows at transport layer. It was based on connection patterns and some characteristics of P2P networks and their behavior. They didn't relying on the packet payload. Results were very effective (they identify 99% of P2P flows and more than 95% of P2P bytes, compared to payload analysis), while they limited false positives to under 10%. Another way to recognize P2P traffic was carried out in [49]. In this paper they developed several heuristics that recognize P2P traffic when they are not using standard ports. They identified two ways to differentiate among several P2P applications and distinguish them from other Internet traffic using two criterions: Packet format and Block sizes. The authors demonstrated that depending on the protocol and the used metric, approximately 30%-70% of the traffic related to P2P applications cannot be identified using well known ports. They concluded that P2P traffic during the time they were measuring continued increasing. In the 2009, the authors of [50] argue that the problem in the identification of P2P traffic is that P2P communications are continuously changing, from TCP layers using well known ports, in some first versions, to both TCP and UDP with arbitrary and/or jumping ports nowadays. In [50], the authors carried out a new approach to detect and control P2P traffic by WAN Optimization Technology. They claim that the most effective WAN optimization products use heuristics to detect P2P traffic. In [51], a series of techniques were proposed in order to control P2P traffic in the ISP networks with the purpose of reducing costs and improving customer satisfaction. In [52], the authors proposed a method based on discreteness of remote hosts (RHD) and discreteness of remote ports (RPD) to identify BT-like traffic.

The increasing popularity of P2P applications made platforms, like P-Cube Service Control Platform [53], look for remedies to address the problem of having a detailed usage accounting of Peer-to-Peer traffic. Their purpose is to facilitate monitoring and billing based on the obtained information in order to create smart control policies.

Nowadays, networks are overfull with ever-increasing volumes of recreational traffic generated by peer-to-peer (P2P) file downloads, access to broadband media via sites like YouTube, and repeated visits to popular social networking sites like Facebook, MySpace and LinkedIn. Recreational Internet traffic also increases congestion and competes with legitimate business applications for available bandwidth, creating delays, frustration and lost productivity when employees need to access business applications on the network. In [54], the authors propose as a solution an effective WAN optimization technology combined with the detection and suppression of recreational and encrypted P2P traffic. They also discuss strategies for controlling a broad range of recreational Internet traffic such as instant messaging, P2P file downloads and social networking activities.

In [55], the authors proposed a scheme for analyzing the peer to peer traffic that concentrates on the factors and characteristics of P2P communications with payload issues. The used method is based on a set of methods derived from the robust properties of P2P traffic. The system showed that packet-level statistics of P2P and non-P2P data flows are basically similar.

In [56] the authors evaluate the performance of P2P traffic in wireless networks. They analyze two ways for controlling the P2P bandwidth consumption in order to limit the impact on time-critical applications: network-controlled and user-controlled. They study the coexistence of TCP-based P2P traffic with delay sensitive audio and video applications in an asymmetric wireless access networks. Moreover, they analyze how the P2P traffic affects UDP-based audio and video traffic under various wireless access network configurations. Mobile P2P is growing fast, as it is indicated in [57], but current P2P applications are not designed to work under these conditions (mainly because the download bandwidth is quite lower than wired technologies).

Using the data collected in [58], we built the graph shown in figure 13. It shows the percentage of the P2P traffic in last decade, particularly in American and European networks. The line shows the logical trend of the P2P traffic. We can see that it has been increasing until 2006. After this date it has been decreasing and increasing along the years. We have also added some analytical values of the traffic. These values were obtained from the studies performed in different publications, which compare the P2P network traffic with the total network traffic.

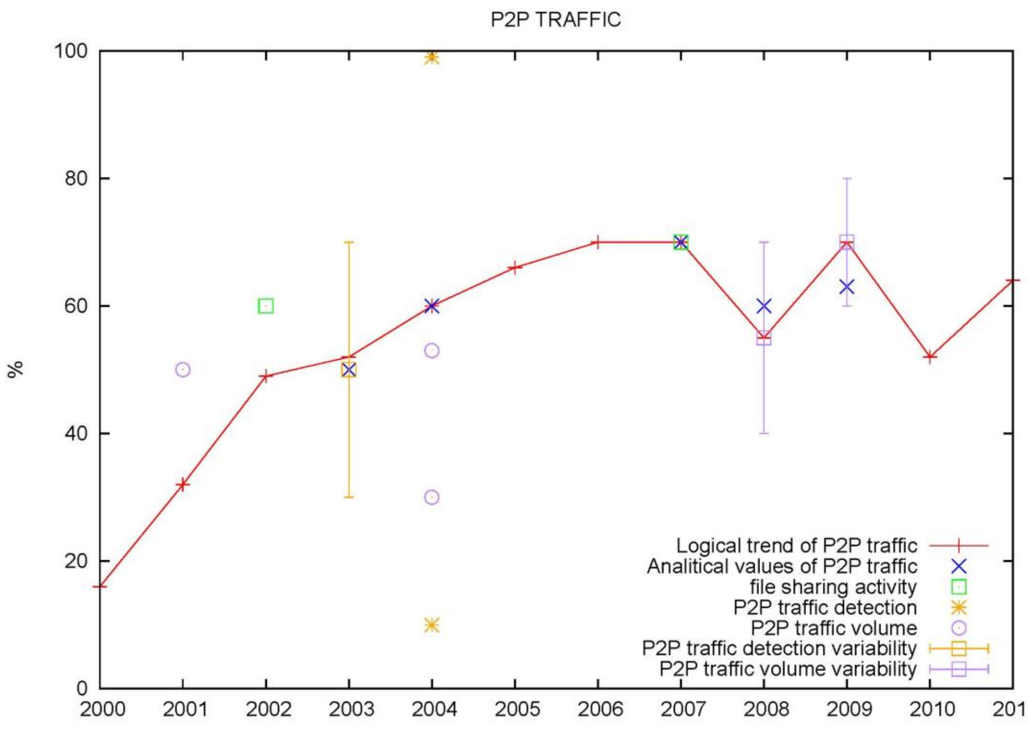

Figure 13. Different analysis related to P2P traffic.

A study of the P2P network traffic at Indiana University, performed in 2001, concludes that 50% of the University traffic is P2P traffic [20]. In 2003, it was found that 30% to 70% of Internet P2P traffic is not detected using the ports [59]. In 2006, 2007, 2008 and 2009, several studies concluded that P2P traffic volume is between 40% and 70% of the total Internet traffic [60][61] [62].

Some of the consequences of using public P2P file-sharing traffic in corporate networks are the bandwidth transit cost, the bandwidth utilization and the end station processing power. Therefore the network planning process is very complicated due to the unpredictability of this type of traffic.

We can highlight some main characteristics of the P2P file-sharing traffic:

- P2P desktop applications run all time. There is no standard peak time.

- Corporate networks are designed asymmetrically. Corporate networks with peers joined to P2P file-sharing networks may cause overload problems. Peers are servers, so upload bandwidth utilization could be greater than expected.

- P2P protocols use non-standard, non-registered application protocols, some of them are proprietary and some of them are open.

- P2P protocols use multiple parallel connections, consuming a lot of TCP and UDP ports to have connections with other peers. Moreover, in case of using NAT routers or firewalls, they will be overloaded.

- P2P file-sharing software applications can use non-registered fixed, random or dynamic ports.

Although a network administrator may add more bandwidth, P2P file-sharing networks will consume it. Adding more bandwidth is not an option because P2P networks are able to adapt to the network capacity. Network administrators have to limit or decrease the P2P file-sharing traffic.

5. Controlling P2P file-sharing networks traffic

In order to control P2P file-sharing traffic in the network, a traffic control cycle can be used. The traffic control cycle is a continuous process built around a network policy (see Figure 14).

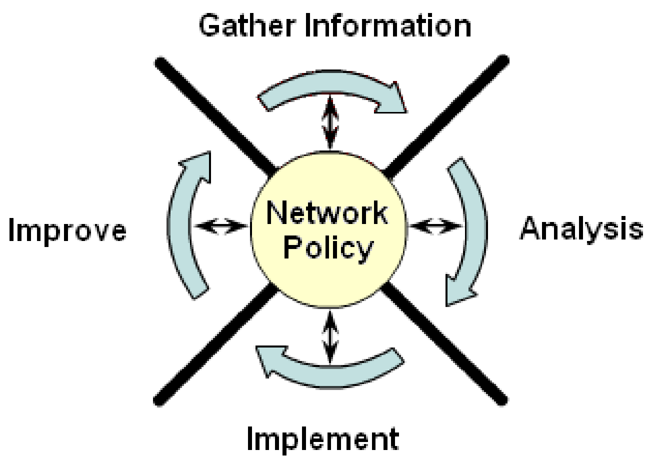

Figure 14. Traffic control cycle.

Step 1: Gather Information. This involves monitoring traffic over a period of time and gathering adequate baseline of data in order to make decisions. It is needed to find P2P traffic in basic and masquerading forms anywhere in the network. This includes emerging P2P network signatures.

Step 2: Analysis. This phase involves analyzing data collected and determines the key indicators of traffic behavior. Administrator has to determine how to control the traffic taking into account this data.

Step 3: Implement the Policy. In this phase a network traffic policy should be defined and implemented in order to control the P2P file-sharing traffic in the network. It can be done by using a traffic control device. The device should be positioned close to the host where the traffic either starts or ends.

Step 4: Improve the Policy. In this phase, the administrator has to verify if the implemented policy gives the expected results, that is, to verify that the countermeasure is in place and working properly. Administrator will go to step 1 if the countermeasure is not working properly. To keep P2P file-sharing networks as controlled as possible, the cycle of the traffic control must be continually repeated because new P2P file-sharing networks appear every day.

There are some options that can help network administrators to control or block this type of traffic without affecting network performance:

1) To set bit rate limits at MAC or IP layer for every host (see Figure 15). It will prevent using all available bandwidth. It can be accomplished by using the most appropriated switches or routers. But this option will affect to all packets forwarded from those computers (despite of the type of application that is sending information from that host).

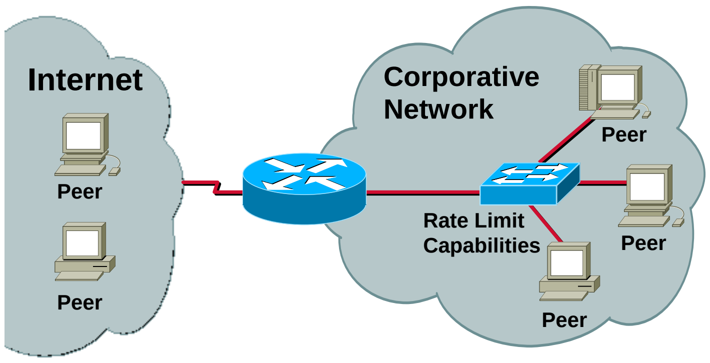

Figure 15. Setting host rate limits.

2) Setting byte caps or tags in the packets (see Figure 16). It permits to know how many packets are forwarded from every computer. It could be used to establish a threshold of packets forwarded per second, per minute or per hour from every computer. It can be accomplished by using a device which aggregates tags to the packets similar to the ones used in VLANS or by using a proxy. But if there are a large number of computers in the network, the administrator may need too many tagging devices or a proxy server with high processing capacity.

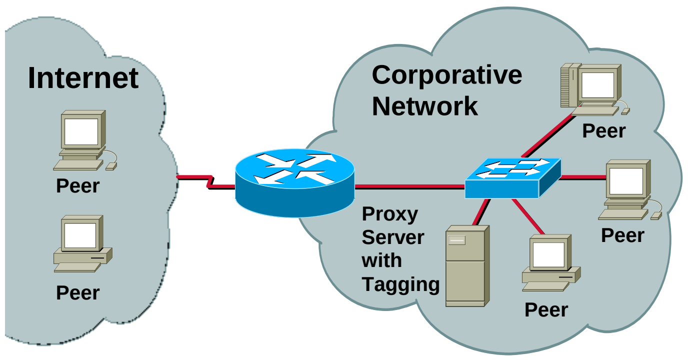

Figure 16. Setting byte caps or tags in packets.

3) Applying a maximum bit rate limit to top individual applications (see Figure 17). Due to the possibility of organizing VLANS by the protocols and applications used, one or several VLANs could be created only for this purpose. But it needs a software application with the signatures of the P2P file-sharing protocols that are going to be controlled. The VLANs containing this P2P file-sharing network traffic could be treated separately, with a maximum bit rate limit which would be different than the rest of the VLANs.

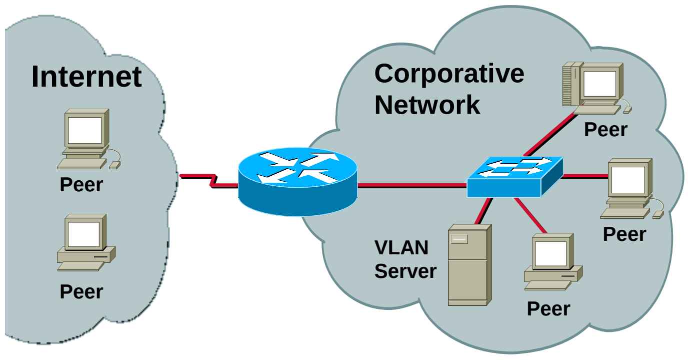

Figure 17. Applying maximum rate limit to top individual applications.

4) Controlling ports (see Figure 18). This option can be used when the P2P file-sharing protocols are using fixed ports. Port numbers used by the P2P file-sharing protocols are not always known beforehand because there are many P2P protocols and there are appearing new ones. In some P2P applications, if the traffic is stopped completely, the application is able to sense it and change the port number. To avoid port hopping, the traffic should be limited to a small value instead of zero. A tutorial to configure a PIX firewall and a Cisco Router, in order to block P2P file-sharing protocols using access-lists and Network-Based Application Recognition (NBAR) can be found at [63].

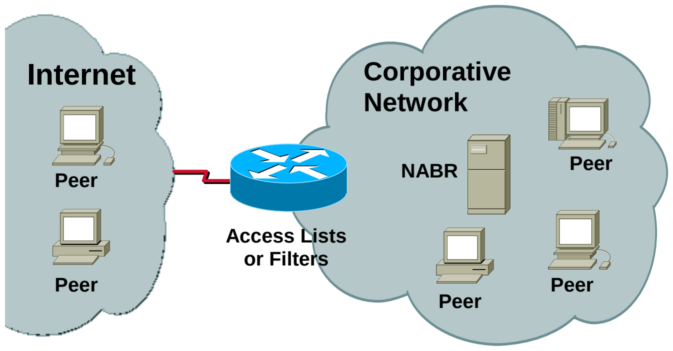

Figure 18. Controlling ports using Access Lists and Network-Based Application Recognition.

5) Controlling signatures at the application layer (see Figure 19). Some Protocols like FastTrack use signatures (FrastTrack uses the signature "KazaaClient"). This option cannot be applied if the data at the application layer is encrypted. In order to apply it, a signature for every P2P protocol is needed. It requires a positive identification of P2P file-sharing traffic. Controlling signatures can be accomplished by using an identification software, which compares the received packet with a known P2P application. Then, it is used a firewall that discards the traffic above a threshold. There are some devices that are able to accomplish this purpose [64]. This option could be used to discard packets when it is desired to block P2P connections, or to apply a maximum bit rate limit for the P2P desktop application.

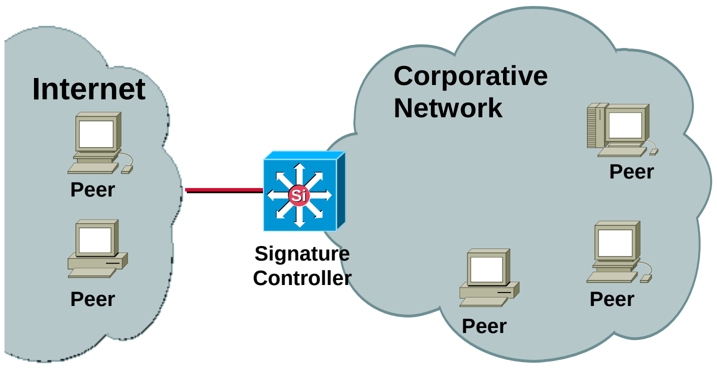

Figure 19.Controlling signatures at application layer.

6) Controlling unidirectional and bidirectional communications (see Figure 20). Packets belonging to a conversation between two hosts can be detected at the session layer. One of the ways it can be done is using a system like Context-based Access Control (CBACS) in Cisco firewalls [65]. CBAC can inspect all TCP sessions and UDP sessions, regardless of the application-layer protocol. CBAC uses timeout and threshold values to manage session state information. Thus, the administrator only will have to define a maximum allowed bit rate to pass through the device. All traffic above a this threshold will be discarded.

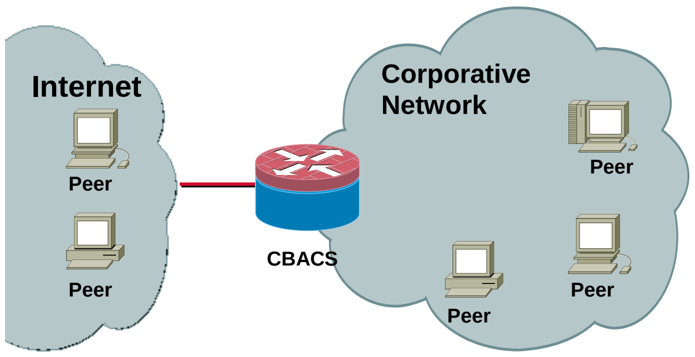

Figure 20. Controlling communications.

7) Using a single QoS appliance (see Figure 21). Perhaps one of the best technical approaches is to drive control of P2P traffic by using policies based on knowledge of the user and current network capacity and availability. This approach allows P2P traffic but controls the network overload. The volume of P2P traffic has to be limited. It keeps P2P traffic in check, so it cannot adversely affect to non-P2P users. The bandwidth allocation is accomplished by distributing bandwidth over users. This option can be applied by using a single QoS appliance, which should be placed on the traffic bottleneck [66]. It can be applicable to every user and with the ability to guarantee their bandwidth. Since the system limits all P2P traffic including the search, upload and download of content, the network will appear to be running slower for P2P traffic.

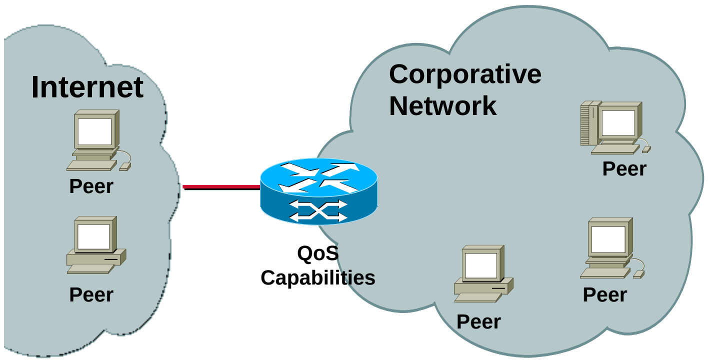

Figure 21.Using QoS Appliance.

8) QBone Scavenger Service (QBSS) [67] (see Figure 22). It is a network mechanism that divides the traffic in different classes: the default class (with best-effort traffic) and the scavenger class. This service creates a parallel virtual network with very scarce capacity. The default class capacity can be expanded when it has low traffic. Everything not used by default class is available for the scavenger class. This system let P2P applications use unused network capacity without affecting performance of the default best-effort class of service. P2P desktop applications have to be marked for scavenger treatment by in the IP packet headers. Routers put this traffic into a special queue with very small allocated capacity using a queuing discipline. When best-effort class needs more bandwidth, the system drops the P2P traffic. QBSS can be used with Cisco and Juniper devices. But this option requires testing all P2P desktop applications to mark all P2P file-sharing networks traffic.

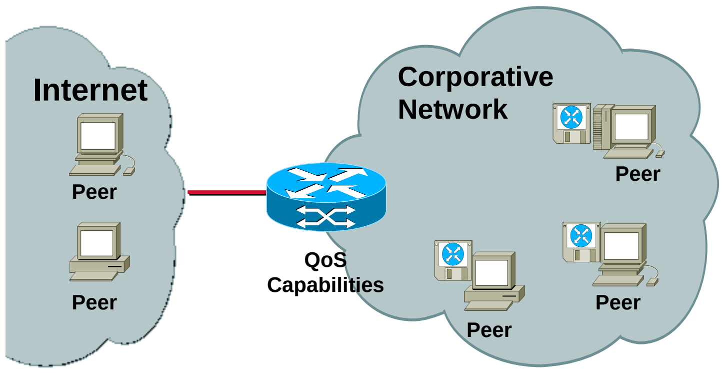

Figure 22. QBone Scavenger Service.

6. Conclusion

Currently there are a lot of public P2P file-sharing networks in existence, some of them with millions of on-line users. Furthermore, the number of users connected to public P2P file-sharing networks has been increasing along the years and new P2P file sharing networks have appeared. All of the above makes very difficult to detect and control all P2P file-sharing traffic.

This paper has discussed eight possible options to control the traffic of P2P file-sharing networks. In order to apply some of them, the administrator must know the signature of the P2P file-sharing protocols to be controlled. In others, the administrator must know the ports used by the P2P file-sharing protocols. Other options use a threshold to limit the amount of traffic forwarded by a computer with a P2P desktop application. However, the best options are those which classify the type of traffic, thus controlling which kind of traffic is allowed. These options let the available bandwidth be used for P2P file-sharing traffic.

References

[1] Nielsen-Netratings, Global Internet Population grows an average of four percent year-over-year, February 20, 2003. Available at http://www.nielsen-online.com/pr/pr_030220_hk.pdf (Last Access December 2011)

[2] Nielsen-Netratings, Nearly 40 million Internet connect via Broadband, growing 49 percent, June 17, 2003. Available at http://www.nielsen-online.com/pr/pr_030618_us.pdf (Last Access December 2011)

[3] J. Lloret Mauri, G. Fuster, J. R. Diaz Santos, M. Esteve Domingo. Analysis and characterization of Peer-to-Peer Filesharing Networks. WSEAS TRANSACTIONS on SYSTEMS. Issue 7, Volume 3.Pages 2574-2579. September 2004.

[4] B. Molina, C. E. Palau, M. Esteve, J. Lloret, On Content Delivery Network Protocols and Applications, WSEAS Transactions on Computers. Issue 6, Volume 3, Pp. 1981-1984. December 2004.

[5] Julius Müller, Thomas Magedanz, Jens Fiedler, NNodeTree: A Scalable Peer-to-Peer

Live Streaming Overlay Architecture for Next-Generation-Networks, Network Protocols and Algorithms, Vol. 1, No. 2, 2009.

[6] Sandvine Incorporated, Regional Characteristics of P2P: File sharing as a multi-application, multi-national phenomenon, White Paper, October 2003. available at http://www.slyck.com/misc/Euro_Filesharing_DiffUnique.pdf (Last Access December 2011)

[7] Shareaza client. Available at http://www.shareaza.com (Last Access December 2011)

[8] MLDonkey client. Available at http://mldonkey.berlios.de (Last Access December 2011)

[9] United States House of Representatives Committee on Government Reform – Staff Report, File-sharing programs and peer-to-peer networks privacy and security risks, May 2003, available at http://democrats.oversight.house.gov/images/stories/documents/20070620172554.pdf (Last Access December 2011)

[10] James Walkerdine, Lee Melville, Ian Sommerville, Dependability Properties of P2P Architectures, Second International Conference on Peer-to-Peer Computing, 2002. (P2P 2002), Linköping, Sweden, 5-7 September 2002. Pp. 173 – 174. 10.1109/PTP.2002.1046331

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A Scalable Content-Addressable Network, 2001 conference on Applications, technologies, architectures, and protocols for computer communications (ACM Sigcomm 2001), San Diego, CA, USA . August 27-31, 2001. http://dx.doi.org/10.1145/964723.383072

[12] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications, 2001 conference on Applications, technologies, architectures, and protocols for computer communications (ACM Sigcomm 2001), San Diego, CA, USA . August 27-31, 2001. http://dx.doi.org/10.1145/964723.383071

[13] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[14] B. Zhou, D.A. Joseph, J. Kubiatowicz, Tapestry: a fault tolerant wide area network infrastructure, ACM SIGCOMM Computer Communication Review, Volume 32, Issue 1, January 2002.

[15] Safwan Mahmud Khan, Nayantara Mallesh, Arjun Nambiar, Matthew Wright, The Dynamics of Salsa: A Robust Structured P2P System, Network Protocols and Algorithms, Vol. 2, No. 4, 2010, http://dx.doi.org/10.5296/npa.v2i4.474

[16] Gnutella Protocol Development Website, http://rfc-gnutella.sourceforge.net/ (Last Access December 2011)

[17] Ed So, Michael Collins, Richard Lee, Rob Lawless, Sean Reilly, TCD 4BA2 Project. Available at: http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p8.html (Last Access December 2011)

[18] C. Plaxton, R. Rajaraman, and A. Richa, “Accessing nearby copies of replicated objects in a distributed environment”, 9th Annual ACM Symposium on Parallel Algorithms and Architectures, (SPAA 1997). Newport, RI, USA. June 23-25, 1997, http://dx.doi.org/10.1145/258492.258523

[19] Sabu M. Thampi and Chandra Sekaran. K, Survey of Search and Replication Schemes in Unstructured P2P Networks, Network Protocols and Algorithms, Vol. 2, Issue 1, 2010.

http://dx.doi.org/10.5296/npa.v2i1.263

[20] Eytan Adar and Bernardo Huberman. Free riding on gnutella. First Monday, 5(10), October 2000. Available at http://www.hpl.hp.com/research/idl/papers/gnutella/gnutella.pdf (Last Access December 2011)

[21] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong, Freenet: A distributed anonymous information storage and retrieval system, ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.

[22] Soulseek website, Available at http://www.slsknet.org/ (Last Access December 2011)

[23] Bram Cohen, Incentives Build Robustness in BitTorrent, Workshop on Economics of Peer-To-Peer Systems (NetEcon'03), Berkeley CA, USA, June 2003.

[24] Open Source Napster Server (OpenNap), Available at http://opennap.sourceforge.net/ (Last Access December 2011)

[25] Oliver Heckmann, Axel Bock, Andreas Mauthe, Ralf Steinmetz. The eDonkey File-Sharing Network. Lecture Notes in Informatics, Vol P-51, Pp. 224-228, September 2004.

[26] MP2P Protocol. Available P2P, http://www.mp2p.net/mp2p.html (Last Access December 2011)

[27] Nathaniel Leibowitz, MateiRipeanu, and Adam Wierzbicki, Deconstructing the Kazaa Network, 3rd IEEE Workshop on Internet Applications (WIAPP'03), June 2003, San José, CA, USA.

[28] Subhabrata Sen and Jia Wang. 2004. Analyzing peer-to-peer traffic across large networks. IEEE/ACM Transactions on Networking, Volume 12, Issue 2. Pp. 219-232. April http://dx.doi.org/2004. 10.1109/TNET.2004.826277

[29] Sandvine Incorporated, Peer-to-Peer File Sharing: The impact of file sharing on service provider networks, December 2002. Available at: http://downloads.lightreading.com/wplib/sandvine/P2P.pdf (Last Access December 2011)

[30] M. Yang, Y. Dai, and J. Tian, Analyzing Peer-to-Peer Traffic's Impact on Large Scale Networks, in Proc. International Conference on Computational Science 4, Lecture Notes on Computer Science, Vol. 3994, Springer, 2006, pp. 412-419. http://dx.doi.org/10.1007/11758549_59

[31] K. Leibnitz, T. Hoßfeld, N. Wakamiya, and M. Murata, On pollution in eDonkey-like peer-to-peer file-sharing networks, in Proc. 13th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 27-29 March 2006, Nuremberg, Germany, pp.285-302.

[32] Internet2 NetFlow, Weekly reports, Available at http://netflow.internet2.edu/weekly/20040216/ (Last Access December 2011)

[33] Wolfgang John, Characterization and classification of internet backbone traffic, PhD. Thesis, Department of Computer Science and Engineering, Clarmers University of Technology, Göteborg, Sweden, 2010.

[34] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002, http://doi.acm.org/10.1145/844128.844158

[35] Genevieve Bartlett, John Heidemann, Christos Papadopoulos, and James Pepin.

Estimating P2P traffic volume at USC. Technical Report ISI-TR-2007-645, USC/Information Sciences Institute, June 2007

[36] Jose Garcia Dorado, Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M. Munafò, Characterization of ISP Traffic: Trends, User Habits and Access Technology Impact, TR-091811-polito, No.TR-091811, September 2011.

[37] Stefan Saroiu, P. Krishna Gummadi and Steven D. Gribble, A Measurement Study of Peer-to-Peer File Sharing Systems, Proceedings of the Multimedia Computing and Networking (MMCN), San Jose, USA. January 18-25, 2002.

[38] A. Lenhart and S. Fox, The Pew Internet & American Life Project's Online Music, September 28, 2000, info at www.pewinternet.org/~/media/Files/Reports/2000/PIP_Online_Music_Report2.pdf.pdf (Last Access December 2011)

[39] Artur Marques, 2003. Available at http://arturmarques.com/docs/economics/arturmarques_dot_com_freeloading.pdf (Last Access December 2011)

[40] Kurt Tutschku. A measurement-based traffic profile of the edonkey file-sharing service. In Passive and Active Network Measurement, Lecture Notes on Computer Science. Vol. 3015. Pp. 12–21, 2004.

[41] J. Pouwelse, P. Garbacki, D. Epema, and H. Sip, The Bittorrent P2P File-sharing System: Measurements and Analysis, 4th International Workshop on Peer-to-Peer System (IPTPS 2005), Ithaca, NY, USA, February 24-25, 2005.

[42] A. Puig-Centelles, O. Ripolles and M. Chover, P2P Traffic Mesurements on the Emule System, Networks, 51-58, 2007.

[43] Chen Liang and Gong Jian, Analsis of BitTorrent Flow Behavior on Large-scale Networks, Journal of Southeast University (Natural Science Edition), Vol.38 No.3, p.390-395, 2008.

[44] J. Lloret Mauri, B. Molina Moreno, C. Palau Salvador and M. Esteve Domingo. Public Peer-To-Peer Filesharing Networks Evaluation. The 2nd IASTED International Conference on Communication and Computer Networks (CCN 2004). MIT Cambridge, MA, USA. November 2004.

[45] Jaime Lloret, Juan R. Diaz, Jose M. Jimenez and Fernando Boronat, Public Domain P2P File-sharing Networks Measurements and Modeling, International Conference on Internet Surveillance and Protection. ICISP'06, Cap Esterel (France), August 26-31, 2006. http://dx.doi.org/10.1109/ICISP.2006.28

[46] J. Lloret Mauri, J. R. Diaz Santos, J. M. Jiménez, M. Esteve Domingo, The Popularity Parameter in Unstructured P2P File Sharing Networks, 4th WSEAS International Conference on Applied Informatics and Communications, Puerto de la Cruz, Tenerife (Spain), de December 17-19, 2004.

[47] Jessie Wang, Chungang Wang, Jiahai Yang, Changqing An, A study on key strategies in P2P file sharing systems and ISPs' P2P traffic management, Peer-to-Peer Networking and Applications, Vol. 4, No. 4, Pp. 410-419 (2011). http://dx.doi.org/10.1007/s12083-010-0098-7

[48] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and K. C. Claffy. Transport

layer identification of P2P traffic. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC '04). ACM, New York, NY, USA, 121-134. 2004.
[49] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos. File-sharing in the Internet: A characterization of P2P traffic in the backbone. University of California, Riverside, USA, Tech. Rep. November, 2003. http://www.cs.ucr.edu/~tkarag/papers/tech.pdf (Last Access December 2011)
[50] Exinda Inc, Controlling Recreational Peer-to-Peer Traffic on Corporate Networks, White paper, February 2009. http://www.nle.com/literature/Exinda_Controlling_P2P_traffic_on_corporate_LANs.pdf (Last Access December 2011)
[51] Marc Morin, Managing P2P Traffic on DOCSISTM Networks, Sandvine Incorporated, November 18, 2003. http://www.scte.org/documents/western/sessiond/ppt/Managing%20P2P%20Traffic%20on%20DOCSIS%20Networks.pdf (Last Access December 2011)
[52] Weiqing Cheng, Jian Gong and Wei Ding, Identifying file-sharing P2P traffic based on traffic characteristics, The Journal of China Universities of Posts and Telecommunications, vol.15, pp.112-120, Dec. 2008. http://dx.doi.org/10.1016/S1005-8885(08)60414-8
[53] P-Cube Technology, Controlling Peer to Peer Bandwidth Consumption, White paper, 2003. http://downloads.lightreading.com/wplib/pcube/controlling_peer_to_peer.pdf (Last Access December 2011)
[54] Exinda Inc, Controlling Peer-to-Peer and Recreational Internet Traffic, White paper, June 2009. http://www.boll.ch/exinda/assets/Controlling_Peer-to-Peer_and_Recreational_Internet_Traffic.pdf (Last Access December 2011)
[55] M. Sadish Sendi, N. Nagarajan, An Optimized Method for Analyzing the Peer to Peer Traffic, European Journal of Scientific Research, Vol.34, No.4 (2009), pp.535-541.
[56] D. Astuti, M. kojo, Evaluating the Behaviour of Peer-to Peer IP Traffic in Wireless Access Networks, 5th International Network Conference, Samos, Greece, July 5-7, 2005.
[57] P. M. Eittenberger, Seungbae Kim, U.R. Krieger, Damming the Torrent: Adjusting BitTorrent-like Peer-to-Peer Networks to Mobile and Wireless Environments, Advances in Electronics and Telecommunications, Vol. 2, No. 3, September 2011.
[58] Sandvine. Global Internet Phenomena Report, Europe, 39, 2011. Available at http://www.sandvine.com/downloads/documents/10-26-2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Report%20-%20Fall%202011.pdf (Last Access December 2011)
[59] A. Madhukar, C. Williamson, A Longitudinal Study of P2P Traffic Classification, 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006), Monterey, California, USA, 11-14 Sept. 2006. http://dx.doi.org/10.1109/MASCOTS.2006.6
[60] M. Perenyi, T. D. Dang, A. Gefferth, and S. Molnar, "Identification and Analysis of Peer-to-Peer Traffic," Journal of Communications (JCM), vol. 1, no. 7, pp. 36–46, November/December 2006.
[61] Ipoque Internet Study Report 2007. Available at:

http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2007.pdf (Last Access December 2011)

[62] Ipoque Internet Study Report 2008/2009. Available at: http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf (Last Access December 2011)

[63] Cisco Systems, Network-Based Application Recognition. Available at http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/ps6616/prod_case_study09186a00800ad0ca.html (Last Access December 2011)

[64] Nortel Networks Alteon Application Switch, available at http://sysdoc.doors.ch/NORTEL/nn102341-102302.pdf (Last Access December 2011)

[65] Context-based Access Control (CBAC), available at http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_tech_note09186a0080094e8b.shtml (Last Access December 2011)

[66] Jarkko Niittylahti. Statelog White Paper. March 2004, available at http://www.staselog.com/files/Staselog-whitepaper-p2p.pdf (Last Access December 2011)

[67] Internet2 QoS Working Group, QBone Scavenger Service (QBSS). Available at, http://qbone.internet2.edu/qbss/ (Last Access December 2011)

Copyright Disclaimer