P2P File-sharing Traffic Identification Method Validation and Verification

A Semester Thesis by
Roger Kaspar

Tutor:
Arno Wagner

Co-Tutor:
Thomas Dübendorfer

Supervisor:
Prof. Dr. Bernhard Plattner

9th March 2005

Student Thesis SA-2005-11

Abstract

P2P networks have become more and more important. Even technical unsophisticated users begin to use the possibilities of P2P file-sharing. Recently the P2P clients have started to use random ports what makes the detection with only port based methods nearly impossible. Therefore an algorithm has been developed during a master's thesis by Lukas Hämmerle [4] that implements a population tracking method of P2P hosts. This application is called PeerTracker.
This semester thesis verifies the results found by that population tracking algorithm. To verify the identified P2P hosts, an online verification algorithm has been developed and implemented. One of the main problems with that approach is, that many P2P hosts are located behind NAT boxes or restrictive firewalls and therefore cannot be reached by a polling algorithm. Nevertheless a lower limit for the reliability of the identification algorithm has been found by making some assumptions about those hosts which could not be reached by the verification. Additionally a traffic analysis has been done on the P2P traffic caused by the identified P2P hosts.
In this thesis 75-88% of the hosts identified as P2P peers by the PeerTracker could be successfully verified. Additionally, 7% of the total P2P traffic in the SWITCH network could be verified as P2P traffic.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of this thesis is to verify the results of the master thesis "P2P Population Tracking and Traffic Characterization of Current P2P File-sharing Systems" [4] by Lukas Hämmerle. Chapter 1 provides basic information that is needed to comprehend the goals, the results and the problems of this task. Chapter 2 will show several methods for online P2P host verification. Additionally, the fundamentals for the interpretation of the results found are given. How the P2P host verification and the traffic analysis are implemented is described in Chapter 3. In Chapter 4 the results of the measurements made and their interpretation can be found. In Chapter 5 the findings of this thesis are summarized. Chapter 6 presents some suggestions for further work.
This introduction only covers those topics which are necessary to understand the statement made in this thesis. An exhaustive introduction to P2P networks and DDosVax can be found in [4].

1.1 Overview

In the last years, P2P file-sharing has become one of the most important applications on the Internet. In 2003 estimated 81.5 million Internet users were downloading music using a P2P file-sharing program according to [24]. This corresponds almost to 5% of the total Internet users worldwide. P2P file-sharing contributes a large amount of traffic that in many networks exceeds the mail/web traffic which has been dominating the Internet so far [4].

1.1.1 DDoSVax Project

This thesis is part of the DDoSVax project [29], which is a joint research project between the ETH Zürich and SWITCH [27]. The goal of DDoSVax is to find methods of detection and defense against distributed DoS attacks. The detection is based on NetFlow-data that is collected at the border routers of the SWITCH network (see Section 1.3).

1.1.2 Reasons for P2P Host Identification

The foregoing thesis [4] implemented algorithms for P2P host identification to, among other things, estimated the traffic caused by P2P hosts in the SWITCH network. To know the amount of bandwidth consumed by P2P hosts is not only important for application specific traffic engineering and shaping, but also to detect anomalies on P2P traffic. Due to the increasing use of P2P file-sharing applications it is only a matter of time until some serious Internet worms will appear using P2P networks. If such worms can be detected in appropriate time serious damage can be avoided.

1.1.3 Reason for P2P Host Verification

The results of the foregoing thesis [4] are rather astonishing. The bandwidth consumption of the examined P2P networks was found to be a large fraction of the total bandwidth consumption and the number of identified P2P hosts was fairly high. This thesis was done to verify these results and check if the algorithm used provides reliable information. By verifying the identified P2P hosts a lower limit for the accuracy of the identification algorithm can be found.

1.2 Goals of this Thesis

1.2.1 Online Verification of Active P2P Hosts

The following goals have to be reached to find a lower limit of the reliability of the identification algorithm used by [4]:

1. Find and implement a polling mechanism for every P2P network whose peers we want to verify.

2. Find an appropriate subset of the identified hosts on which the verification can be done.

3. Analyze the data found and find the lower limit. Verify the choice of the subset for this P2P host verification.

1.2.2 Analysis of P2P Traffic in the SWITCH Network

In order to verify the P2P traffic measurements made by the foregoing thesis, the following goals were defined:

1. Find out how accurate the P2P traffic estimations by [4] were.

2. Verify the choice of the subset of identified P2P hosts for the host verification with the found P2P traffic insights.

3. Find answers to the following questions:

    - Which P2P network consumes the most bandwidth?
    - Do any major differences between P2P usage of different hosts exit?
    - How important is the use of non P2P default ports regarding the gained bandwidth for P2P users?

1.3 Network Environment

P2P identification and traffic analysis in this thesis are done by processing flow-level data gathered from the SWITCH network. SWITCH (Swiss Education and Research Network) connects almost all universities, research facilities and other educational institutions in Switzerland. There exist four border gateways that connect the SWITCH network to peering partners.

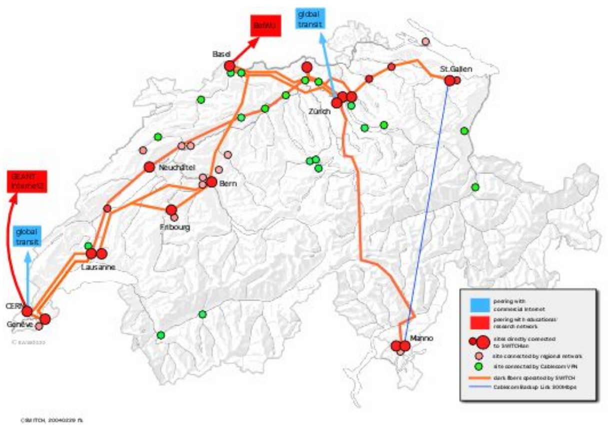
BelWü
global transit
Basel
St.Gallen
Zürich
GEANT Internet2
Neuchâtel
Bern
Fribourg
global transit
Lausanne
CERN
Genève
Manno
peering with commercial Internet
peering with educational/ research network
sites directly connected to SWITCHlan
site connected by regional network
site connected by Cablecom VPN
dark fibers operated by SWITCH
Cablecom Backup Link 300Mbps
©SWITCH, 20040229 fk

Figure 1.1: *SWITCH network topology [27]*

1.3.1 Flow-level Data

The NetFlow format by Cisco Systems is a data format for IP traffic information. Cisco routers generate NetFlow records, that are exported from the router in UDP packets and collected using a NetFlow collector. A flow is defined as series of IP packets with the same source IP, destination IP, source port and destination port [28]. NetFlows are unidirectional and for this reason a TCP connection results in two flows, one in each direction. Flows are expired from the router buffer and emitted to a collector if the maximum lifetime is exceeded (default is 15 minutes) or the maximum idle time is reached (default is 30 seconds).

1.4 P2P Host Identification by the Peertracker

In the foregoing master thesis "P2P Population Tracking and Traffic Characterization of Current P2P File-sharing Systems" an application was created that identifies P2P hosts out of NetFlow data - the PeerTracker. The P2P networks Kademlia, Overnet, eDonkey, BitTorrent, Gnutella and FastTrack have been considered.

1.4.1 Host Identification Method

The developed algorithm for P2P host identification uses a MURP[1] approach that uses the neighborhood relations to determine a peer. The criteria for a host to be considered as an active peer is, that it either has enough connections to previously identified active peers or that the number of suspicious flows (defined by the P2P default ports) is higher than a threshold. The PeerTracker uses five different states in its host pool (see Figure 1.2). The most important hosts for this thesis are the active ones. (Detailed information about the used identification method can be found in [4].)

1.4.2 Implementation

The PeerTracker was developed under Debian Linux using C. The program works as an UPFrame[2] [6] plug-in or in standalone mode. Therefore it either reads from UPFrame or a named fifo pipe. The PeerTracker reads the NetFlows continuously and updates the peer statistics accordingly. The host pool is evaluated periodically and the hosts are assigned

[1]Most Used Remote Port
[2]An application that provides a flexible plug-in framework to do online measurements of NetFlow data.

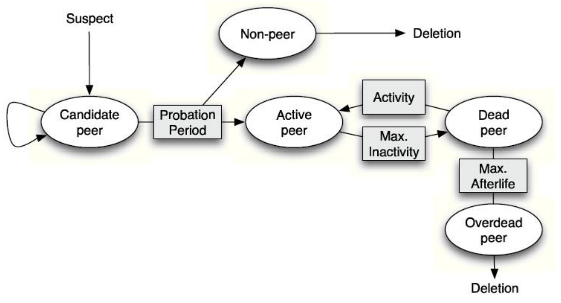

Figure 1.2: *States for identified hosts*

to the according states (see figure 1.2). When this is done, the active peers are written into a file. The verification algorithm developed in this thesis makes use of that dump file.

1.5 Privacy Issues

NetFlow data of the SWITCH network and its content have only been used for the identification and verification of P2P hosts. The identified hosts IP addresses were only stored temporarily. Information which relate hosts to users has not been analyzed neither has it been stored. The only purpose of this work was to analyze P2P usage in the SWITCH network. NetFlow data has only been transferred and stored in secure ways and has not been given to third persons.

Chapter 2

P2P Host Verification

The first section of this chapter describes the P2P networks that have been considered. Section 2.2 shows the initial position of this thesis and Section 2.3 explains the online verification mechanisms that have been used on the different P2P networks. Section 2.4 and 2.5 deal with the problem that many P2P hosts cannot be polled directly and how it is handled. Besides P2P host verification this thesis also analyzes the traffic caused by P2P hosts. Section 2.6 describes what traffic can be identified as P2P traffic and Section 2.7 explains the limitations of the approach.

2.1 P2P Systems Considered

In this thesis, algorithms to poll clients of several P2P networks are presented. The considered P2P networks are: eDonkey, Kademlia, Overnet, Gnutella and FastTrack. BitTorrent hosts cannot be verified by polling and therefore they are not considered in this thesis, even though the PeerTracker is able to identify BitTorrent clients [4].

Network	Users
eDonkey	3'361'422
Fasttrack	2'879'413
Gnutella	1'380'738
Overnet	1'376'256
DirectConnect	436'647
MP2P	255'855
Filetopia	4'611

Table 2.1: *Usage of P2P networks, 21.2.05 15:00-15:15, [1]*

According to Slyck.com [1], which is one of the most used forum and newspage regarding P2P file-sharing, the most used network in February 2005 is eDonkey followed by FastTrack, Gnutella and Overnet. The proportions can be seen in Table 2.1. BitTorrent does not show up in the Slyck statistics since its user base is difficult to estimate [1]. The Kademlia protocol is closely related to eDonkey since is was created by the developers of eMule [9], the most popular client for the eDonkey network. Until now only eMule clients implement the Kademlia protocol and therefore every Kademlia user is most likely also a eDonkey user.

In the following sections the considered P2P networks are discussed.

eDonkey

eDonkey is a P2P network that relies on a centralized server structure, which holds the file index of shared files. Unlike Napster, the dedicated server software is freely available and therefore about 80 eDonkey server exist. Due to the fact that every user has to authenticate with one of these servers, the overall number of users is known at each time. On an average day in February 2005 there were between 2.8 and 3.8 million hosts connected to the eDonkey network, sharing around 350 million files.
There exist several clients for the eDonkey network. The most used ones are the official eDonkey2000 [10] client by eDonkey's creator MetaMachine and the even more popular eMule client [9]. One of the main reasons for the popularity of eMule is its open source policy which permits the addition of useful features to the clients.
One of the main vulnerability of the eDonkey network is its centralized server structure. Therefore the developers of eDonkey2000 and eMule started to integrate another, completely decentralized network into their client. For the eDonkey2000 client this is Overnet and for eMule Kademlia.

Overnet

Overnet is a decentralized successor of the eDonkey network developed by MetaMachine. Overnet was supposed to replace eDonkey but so far this has not happened. To push the use of this network MetaMachine began to modify their client eDonkey2000 that it connects also to the Overnet network. Overnet uses - like Kademlia - the XOR-algorithm [13] that distributes the file indexes to the participants of the P2P network. UDP is used for signaling traffic.
When this thesis was written there existed only one official Overnet client. MetaMachine took the standalone Overnet client from the net and provides now only the hybrid client eDonkey2000 V1.0. The only inofficial Overnet client is the multi-hybrid mlDonkey client, which is not used by many users due to its complicated handling. It is used merely by rather technical sophisticated people.

Kademlia

The creators of the eDonkey client eMule developed a new completely decentralized network called Kademlia. Even though it also uses the XOR-algorithm [13] and UDP for signaling traffic it is not compatible to the Overnet protocol. At the time this thesis was written the Kademlia networks was in open test phase. It is supported by the eMule client but is not enabled by default. Therefore it can be assumed that only few eMule users have it turned on. So far eMule is the only client connecting to the Kademlia network. Therefore every Kademlia peer is also part of the eDonkey network. Unfortunately eMule uses the same port for eDonkey and Kademlia traffic, that is why these two cannot easily be separated.

Gnutella

In early 2000 the first Gnutella client was developed by Nullsoft employees. Is is a completely decentralized network that is fully open source, which makes the network so popular and almost immortal. It survived several setbacks, for example the scaling problems at the beginning. The open source community debugged and improved the protocol. In 2004, the Gnutella network became very popular again. Among other things because of many P2P users abandoned FastTrack or other networks which are financed through spyware and other third hand applications. Due to its open source character Gnutella provides also some protection from being sued. A user can be sure that it has been tested and

improved by many sophisticated users. In this way the less "dangerous" functions exist in the protocol (e.g. listing of shared files in Kazaa).
A Gnutella client has 4-10 TCP connections to other peers. For signaling traffic UDP is used and to make use of the benefits of server based networks a "ultra-peer" state was created. "Ultra-peer" status is self assigned by powerful peers and provides some extra functionality compared to ordinary nodes. There exist many freely available Gnutella clients. Some of the most popular are Limewire, Bearshare and Morpheus. However, the one that has the most increasing number of users is the Shareaza [38] client. It has a very pleasant GUI and connects also to eDonkey and BitTorrent.

FastTrack

In 2003, FastTrack was the most used P2P network. Originally based on Gnutella it was developed and brought to market by Sharman Networks [32]. Since it became public knowledge that it has built in spyware and other third hand applications, many users abandoned the FastTrack network and changed to Gnutella, BitTorrent or eDonkey. Another reason was also the fact that FastTrack users were the first ones sued by the RIAA[1] and MPAA[2].
FastTrack is a decentralized network whose protocol is not publicly available and therefore not completely documented. It utilizes something called "super-peers" to create temporary indexing servers that would allow the network to scale better. Any client may become a "super-peer" if the according computer and Internet connection are powerful enough. Every FastTrack client needs a TCP connection to one "super-peer" to join the network. The signaling traffic in Fasttrack is encrypted and uses UDP. Download traffic is not encrypted yet.
Only three official clients exist, namely Kazaa, Grokster and iMesh. Since all three of them in the "free" version are packed with unwelcome third-party programs, some people have started to modify the official clients what resulted in Kazaa Lite K++ or DietK [39]. However Sharman Networks makes its money with this thirds-party add-ons and therefore it forced several websites that offered these clients to shut down.
In February 2005 one of the several court proceedings against Sharman Networks brought a severe setback for FastTrack. An Australian court revealed some documents that show that Sharman Networks has control over their network and could ban every unwanted file from the indexing sites. If they are able to do that they could be forced to install filters against copyrighted material. This could be the end for FastTrack [22].

BitTorrent

BitTorrent is a P2P file-sharing protocol that became very popular for larger files (e.g. CD-size) and is developed by Bram Cohen. In contradiction to other P2P networks it has no built in search algorithm. To download a file the according .torrent file has to be downloaded from a webpage[3]. Torrent files contain the hash values of a file and the address of a tracker. The tracker supplies peers with addresses of other peers that share the wanted files. There exists no single BitTorrent network, but thousands of temporary networks consisting of clients downloading the same file. During the download process the tracker periodically sends updated information about new download locations. Communication between two peers is done with TCP. There exist many different BitTorrent clients which are in general freely available for everyone. Very popular are the original BitTorrent client written by Bram Cohen himself and the java based client Azureus [40].

[1]Motion Picture Association of America [34]
[2]Recording Industry Association of America [33]
[3]E.g. www.torrentreactor.to

2.2 Initial Position

First the available output from the PeerTracker will be analyzed and then the environment and the original setup will be taken into focus.

2.2.1 Available Data

The PeerTracker takes NetFlows as an input and then periodically generates two output files. One file contains the statistic (e.g. traffic information) and configuration variables (such as the evaluation period), the other information about the identified peers. The input parameters can be adjusted in the config file[4]. Basically two things had to be adjusted: The evaluation period (the time after the identified peers are dumped) and settings about which part of the host pool has to be dumped. Only the active peers were needed - a verification of dead peer does not make much sense.
The statistic files have been mainly important for the traffic analysis in section 4.3 whereas the peerdump file was used to gather important data about the peers that had to be verified. The file with the dumped active peers was uses as the input for the online verification. Important for the verification was the IP address of the host, the assumed P2P network and the three most used local ports (UDP and TCP).

2.2.2 Setup

Before this thesis the PeerTracker had only been used in offline mode. The storage system of NetFlow data, used by the DDosVax team, was designed for offline processing of the flows: The collected flows are available from the DDoSVax archive three hours after the last flow was recorded by the border routers. The NetFlows are collected by a server in the SWITCH network and stored into two files. One contains the flows of three border routers, the other those of the fourth one. Every hour two new files are created. These files then are sent to a DDoSVax server where they are stored (see Figure 2.1). A latency of tree hours is way too much for an online verification of identified P2P hosts - too many clients would not be running anymore. Therefore one of the tasks of this thesis was to build a setup where the NetFlow data can be processed with an appropriate latency. How this was done can be found in Section 3.1.
The PeerTracker reads either from a named fifo pipe or from UPFrame. In previous measurements the whole two files with one hour of NetFlow data have been combined and streamed into a named pipe.

2.3 Verification for Concrete P2P Systems

In this section the protocols of the analyzed P2P networks are discussed more closely and it is explained which part of each protocol can be used for an online verification. To verify whether a specific P2P client is really running on a host, one must either closely analyze the network traffic or the host must be contacted in some way. Due to the fact that within the DDoSVax project there is no possibility to examine the payloads of the packets that are sent or received by a host, in this thesis it was tried to verify the identified P2P hosts by contacting them. If parts of a P2P protocol are known that allow to communicate directly with active peers, a polling method can contact a potential P2P client and send it a typical message for the particular network. By analyzing the hosts answer a peer can be verified.

[4]PeerTracker.cfg

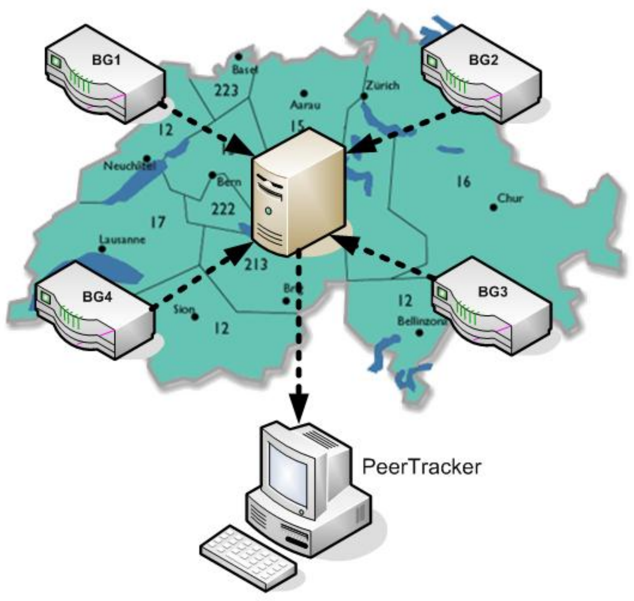

Figure 2.1: *Original setup*

2.3.1 eDonkey, Kademlia, Overnet

The most important thing for P2P clients is downloading and uploading files. Thus verification of a P2P client can be done at the point in the protocol where a peer contacts another to start a download process. This approach seems to be the most promising one.

With the eDonkey protocol, a client that wants to download a file contacts the target peer with an *eDonkey Hello* message. Even if the contacted peer does not share any files at all it has to answer to this request. This is specified in the protocol description [11]. That way every running eDonkey client (if it accepts connections from outside, see Section 2.4) can be successfully verified. As Kademlia and Overnet also makes use of this message, peers of these networks can be verified with the same method.

The eDonkey protocol is binary and therefore a bit more difficult to understand. Every packet comes with a header of five bytes. The first byte is the *eDonkey Protocol identifier*, which has the value 0xe3 for eDonkey, 0xc5 for eMule or 0xd4 for eMule with compression. The last one is not important for the verification process because it is not used with the *eDonkey Hello* message. The four following bytes indicate the length of the data encapsulated in the packet. The packet sent by the verification process then contains the two hex values 0x01 and 0x10, which denote an *eDonkey Hello* message, followed by the client's hash value, its ID and port. The important part of the *eDonkey Hello* message is summarized in Table 2.2.

The content of the peer's reply is not important to the verification process as long as the first byte is the eDonkey protocol marker hex value 0xe3 or 0xc5 (eMule extension). If such a message is received the peer is successfully verified. The verification algorithm does the explained procedure on the three most used local ports by connection count, assuming that the listening port is among them. In addition the default ports 4662, 4672 and 5662 are polled.

Bytes	Content
1	Protocol descriptor: 0xe3 or 0xc5
4	Packet data length
2	'Hello client' descriptor: 0x01 0x10
16	MD4 digest client hash
4	Client ID
2	Client Port

Table 2.2: eDonkey Hello *message*

2.3.2 Gnutella

Gnutella is an open source protocol and therefore the verification is somewhat more comprehensible. The counterpart of the *eDonkey Hello* message in the eDonkey handshake is the *Gnutella Connect* message in the Gnutella handshake. Every time a peers wants to connect to another one, that message is sent in plain[5] text via TCP. If the contacted peer is willing to accept the connection it replies with *Gnutella OK.* Any other response indicates the servant's unwillingness to accept the connection. A servant may reject an incoming connection request for a variety of reasons - a servant's pool of incoming connection slots may be exhausted, or it may not support the same version of the protocol as the requesting servant, for example [26]. In this case a peer answers either with *Gnutella maximum connections reached* or another error message. They all have one thing in common - they begin with the word "Gnutella".

The only problem with Gnutella is that in the protocol specifications it is only recommended to answer the *Gnutella Connect* request if a peer is unwilling to accept the connection. The most of the clients do it but there could exist some clients that do not reply at all. This could result in a little lower quota for successfully verified Gnutella hosts.

The explained *Gnutella Connect* message is used by the verification algorithm. It tries to establishes a TCP connection to an assumed Gnutella peer and sends a *Gnutella Connect* message. If then an answer containing the word "Gnutella" is received, the peer is successfully verified. Besides the three most used local ports the Gnutella default ports 6346-6349 are polled.

2.3.3 FastTrack

Due to the fact that the FastTrack protocol is encrypted the verification has to be made with a download request. This is the only known part that is in plain text. As the protocol is not freely available some reverse engineering attempts [8] were done, e.g. for the open source clients mlDonkey of giFT. However the published information is far from complete. Nevertheless it is known that a FastTrack download request starts with the HTTP command *GET /.files HTTP/1.1* that is sent using TCP. A FastTrack client then will reply with a pseudo HTTP response of the form *HTTP/1.0 403 Forbidden* followed by two decimal numbers that stand for the coming encryption. Some Kazaa clients also answer with *HTTP/1.0 404 Not Found/nX-Kazaa-Username.*

To verify a FastTrack peer the host is contacted with such a download request. If the reply contains either the string "Forbidden" or "Kazaa" the peer is successfully verified. Besides the three most used local ports the FastTrack default port 1214 is polled.

[5] Actually the message is "GNUTELLA CONNECT/x.y\n\n" where x.y stands for the protocol version.

2.3.4 BitTorrent

The polling algorithms used for the P2P host verification base on a characteristic handshake for the particular networks. BitTorrent immediately aborts this handshake if the request does not contain a distinguishing hash-value of the file that is requested. Unfortunately we have no information about the files offered by the hosts that we want to identify. Therefore BitTorrent hosts cannot be verified by polling without knowing anything about the shared files. If a contacted peer accepts incoming TCP connections and immediately drops them after BitTorrent requests it can be assumed that a BitTorrent client is running, but it is no proof for it. There is still a possibility that another service is running on the target host that uses the dedicated ports. There exist other possible methods for BitTorrent peer verification but this thesis does not cover or implement them due to time constraints. More about this techniques can be read in Chapter 6.1.

2.4 Drawbacks of Polling

As discussed in previous sections, in this thesis the approach of an online verification by polling is used. The drawback with this approach is the fact that every host that wants to be verified has to accept at least one TCP connection. As described in section 2.3 within this connection a request, which is typical for the assumed P2P network, is sent and the answer is interpreted accordingly. If a host does not accept any connections from outside at all the verification cannot take place. As explained in the following two sections, the main reasons for that to happen are restrictive firewalls and NAT devices. This problem is essential for this thesis because many P2P hosts are believed to be placed behind such a device. E.g. in [14] it is predicted that about 50% of Gnutella traffic is across firewalls and NAT boxes.

2.4.1 NAT

A NAT device[6] connects an inside network (mostly private networks) to an outside network. The NAT device represents all clients of the inside network and has one IP address assigned.

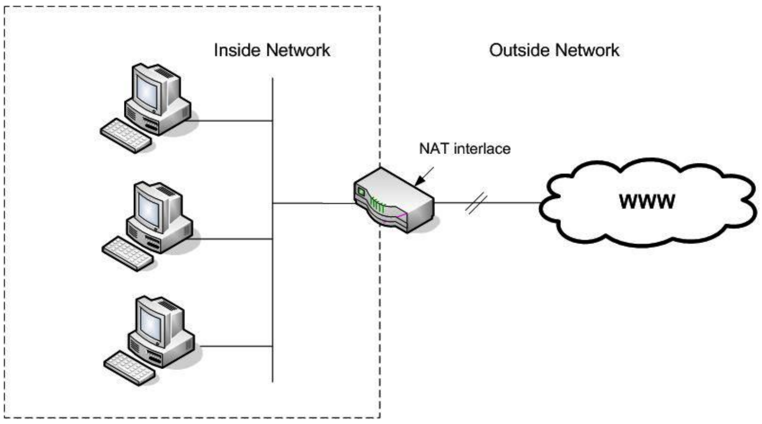

Figure 2.2: *Network address translation*

If a client tries to connect to a server in the outside network on a specific port, the NAT device replaces the sender IP address in the packet's header with its own and the source

[6]Network Address Translation

port with a unique port number. This information is then saved in a temporary table. When an answer is received, the NAT device compares the IP/port pair with the entries in its cache and forwards the packet to the according client. Normally a NAT box drops packets that come from an IP address which is not present in the temporary table of the device. Thus a peer that is behind a NAT device cannot be polled. The packets are dropped because the entry in the mapping table of the NAT is missing since the peer itself did not initiate a connection to the polling host. Some users, which run P2P-clients on a host behind a NAT device, have port mapping activated to map the P2P client's listening port directly to their host. In that case the peer should be verifiable because incoming connections on these ports are allowed. In the following, peers that are located behind a NAT device with port mapping activated are not considered "NAT users". By default, incoming ICMP requests are answered by the NAT device. However this can be turned off manually.

2.4.2 Firewalls

A firewall is a piece of software or hardware that controls the traffic between two dedicated networks. Usually a firewall is in use to protect a personal network from unwanted access from outside and to prevent malicious programs from connecting to the Internet from inside. To do so, a firewall contains a packet-filter that checks the source, destination and protocol-type of each packet according to rules which can be defined manually. To protect on a higher level, firewalls nowadays also contain a content-filter which checks the content of a connection for malicious code (e.g. active-x controls or javascript). Often a firewall is configured that it rejects unknown connections by default or asks the user how the request should be handled. In most cases a P2P user configures its firewall in a way that makes connections from outside possible, at least for the client's listening ports. Peers behind such a firewall can be verified if a listening port is found. However, some users have firewalls running that block all incoming connection attempts, even those to the listening ports of the used P2P client. This can be due to security considerations or just because of lack of knowledge to reconfigure the firewall settings. In that case, the P2P clients cannot be polled because no connection can be established from outside.

2.5 Finding a Representative Subset for Peer Verification

The goal of this thesis is to find a lower limit of the accuracy of the P2P host identification algorithm presented by [4]. However, as seen in the previous section, some P2P hosts cannot be verified by polling because they have a restrictive firewall running or are located behind a NAT box. Therefore a representative subset of hosts that can be verified has to be found for the verification process. If it can be shown that an examined subset of the whole set of identified hosts is likely to have the same ratio of correct identified, then the lower limit found is valid for the whole set of identified peers by the PeerTracker. In this thesis the set of hosts that can be polled is taken as this representative subset. In the following it is explained what methods have been used to determine if a host is pollable. Then in Chapter 4.1.2, measurements are analyzed and the choice of this specific subset is justified.

2.5.1 Methods

First of all it is useful to know if a host or its NAT device is actually running or if it has been shut down in the short time between the identification by the PeerTracker and the online verification. This is done by sending an ICMP ping to the host. Then, it has to be checked if the host accepts incoming TCP connections. In order to discover this, TCP pings on several ports are done.

ICMP-"Echo Request"

The Internet Control Message Protocol (ICMP) is a required protocol tightly integrated with IP and is used among other things to exchange error, control, and informational messages [17]. ICMP pings are almost always successful because firewalls and NAT boxes, if not manually configured otherwise, answer these requests. For this reason not much information can be gained with this test. It only shows that the host or the according NAT device is alive. With a short time gap between identification and online verification reply very high reply quotas can be expected.

TCP-"Connection Attempt"

The transmission control protocol (TCP) is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent [18]. A TCP ping tries to establish a connection to a host on a certain port. Three different things can occur:

1. The pinged host accepts the connection and replies with an *ACK* message. This does not give any evidence about a P2P client running on the port, however it states that a service on the host accepts incoming connections on this particular port. In this case the TCP ping is successful.

2. The host immediately resets the connection and replies with a *RST* message. If no service is running that waits for incoming connections on a particular port, the connection is reset by default. This is stated in the specifications of the TCP protocol [16]. In this case the TCP ping is also successful.

3. Nothing is received from the pinged host, not even a reset message. In this case the TCP request had to be dropped somewhere on the host's side. Either a NAT dropped the incoming packet or a firewall or an intrusion detection system blocked the incoming connection. In this case the TCP ping is not successful.

In case 1 and 2, where the TCP ping is successful, the pinged host is reachable and thus possible P2P clients can be verified. If the TCP ping is not successful (case 3) it is not possible to poll P2P hosts because no TCP connection can be established to the according ports. A P2P client can only be connected to on its listening port, therefore the three most used ports of the according host and the default ports for the assumed P2P network are TCP pinged. If these pings are unsuccessful it can be assumed that either a firewall is running or that the host is located behind a NAT. This assumption is valid because the peers under examination transferred data on these ports only a few seconds before the verification process. This means the ports are used, but not accessible from outside. That makes these hosts unverifiable for the used algorithm.

2.6 Traffic Identification

One of the main goal of this thesis was to verify the traffic measurement made in the foregoing thesis [4]. To accomplish that, one has to find out what traffic can be counted as P2P. In the last years P2P clients and users started to use random ports for their traffic. It has to be assumed that a substantial amount of P2P traffic is transmitted on non-default ports. Straightforward traffic identification based on P2P default ports cannot generate good results[7]. Fortunately in this thesis only traffic calculations in relation with identified (and for the most part successfully verified) P2P hosts have to be made. This simplifies the problem because several assumptions about traffic usage of P2P users can be made:

- Hosts running P2P clients do most likely not run other applications with listening ports in the non-well-known[8] port range. Such applications include video/audio streaming or online games. These applications are very sensitive to network latency (QoS) and get full attention of the user while running. A running P2P client would noticeably decrease their performance because it is rather bandwidth consuming.
- Traffic that is more probable to occur simultaneous to P2P file-sharing is Web-traffic, remote login, FTP or SSH data transfer or mail traffic because these applications are not as sensitive to latency as those named above. All these applications have their listening ports in the well-known range [23].
- It can be assumed that very few P2P users manually set their clients listening port in the well-known port range. Only users with a certain level of technical knowledge know about the benefits of having a non-default listening port (e.g. some traffic shaping methods can be dodged). Mostly users with such a knowledge are also aware of the fact that ports in the well-known range are reserved for other applications and having abnormal traffic on such a port can cause problems with hardware, software and network administrators. Measurements show that for all of the considered networks less than 1% of the peers use listening ports in the well-known port range [4].

Concerning the given statements above, only traffic between two ports of the non-well-known range (that means bigger than 1024) are considered as P2P traffic. Furthermore the identified P2P traffic is added to the network the according host belongs to.

2.7 Limitations

Due to the fact that many host cannot be reached because of NAT boxes and firewalls, the online verification has to be made on a subset of the total amount of identified hosts by the PeerTracker (see Sections 2.4 and 2.5). Therefore the lower limit found in these measurements depend on the choice of the subset of verifiable hosts for the measurement's interpretation. To justify this choice of the subset several measurements have been made. The traffic estimation (see section 2.6) is based on the assumption that P2P hosts do not run other applications in the non-well-known port range. This seems to be true for the most of the hosts but definitely not for all. Therefore the estimated P2P traffic trends to be higher than the actual P2P traffic of the identified hosts.

[7]Even though the traffic shaping in the ETH Zürich network currently uses the default port approach. However, this might change soon.

[8]well-known port: 1-1024, non-well-known ports: 1025-65536

Chapter 3

Implementation

This chapter deals with the implementation of the online verification algorithm (Section 3.2) and the traffic validation (Section 3.3). To be able to read NetFlow data in realtime, the setup had to be changed in a way that the latency of the data processing becomes minimal.

3.1 Configuration Setup

To reach the goals of this thesis the PeerTracker had to run in online mode and the time between the identification by the PeerTracker and the online verification had to be kept very short. Before this thesis the needed NetFlow data was first on the DDosVax server three hours after the last flow had been captured. Therefore a script was written that reads the flows directly from the server that collects the flows from the border routers. This two streams[1] are then streamed into one named fifo pipe on a DDoSVax machine (see Figure 3.1).

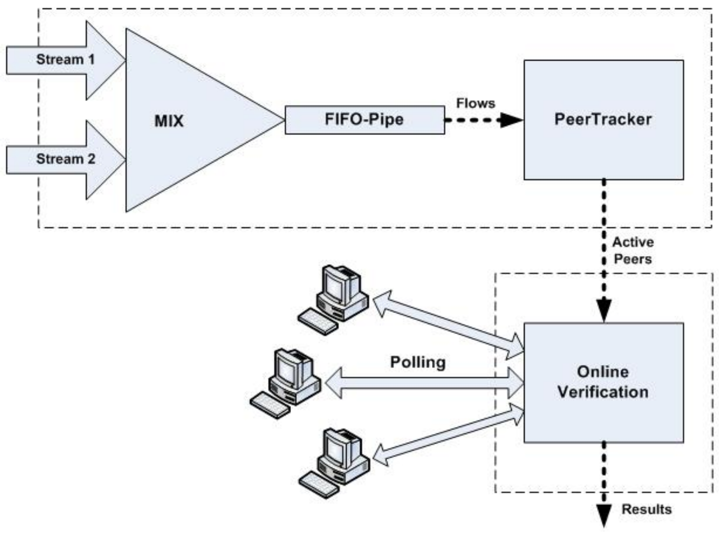

Figure 3.1: *Setup for the verification process*

The PeerTracker was configured to read from this named pipe. Due to the fact that the PeerTracker is able to process the delivered amount of NetFlow data in realtime, no

[1]One stream for the three smaller border routers, one for the biggest.

problem with its performance were encountered. After each evaluation period[2] the result of the P2P host verification were directly transferred to another machine where the online verification took place.
With this setup the results of the PeerTracker's host identification (namely the identified active peers) was available for the verification process 15 seconds after the last flow was captured by the border routers. A router dumps a flow not later than 30 seconds after the last matching packet has passed it. Thus the latency has been reduced from three hours to totally 45 seconds.

3.1.1 Modifications to the Original PeerTracker

In the original PeerTracker application the reading procedure from the fifo pipe had to be changed slightly. In the original version whole NetFlow packets with a NetFlow header and several flows are expected. When a file with NetFlow data of one hour is streamed into a pipe this causes no problem. However if those packets are streamed in realtime and the PeerTracker reads from the pipe simultaneously it can occur that incomplete packets are in the pipe. The IO-buffer of the operating system was either too small or too big. To avoid these errors the original PeerTracker was modified to only read if a whole packet is available in the pipe.

3.2 P2P Host Verification

The P2P host verification script was written in Perl under Debian Linux and used the Perl::NET module for networking issues.
The main thread reads the file with the active hosts identified by the PeerTracker and the verification results from the last verification. It then compares these two files to avoid that hosts, which could not be reached previously, are polled again[3]. For every host that has to be verified a thread is created and started. Multi-threading is used because sometimes the verification algorithm has to wait for several TCP timeouts. Processing all hosts in one thread would take much longer. Up to 200 threads are running simultaneously.
Each thread tries to poll its hosts with the methods described in Sections 2.3.1-2.3.3. Hosts that are identified as BitTorrent peers are not verified. Additionally the hosts are ICMP and TCP pinged to get information about their reachability (see Section 2.5.1).
When all threads are done, the main thread writes the results into a file. Interesting are the verification result (was polling successful and on which port) and the reachability of the hosts.
Information about the performance of the script can be found in table 3.1. More than 400 hosts can be verified in less than three minutes.

# Hosts	Verification Time
215	85s
444	177s

Table 3.1: *Time used by the verification script (two typical measurements)*

[2]Each *Evaluation Time* the PeerTracker evaluates its host pool. Mostly a period of 15 minutes was used.
[3]If measurement are made over a long period, ongoing TCP and ICMP requests can cause alerts in IDS (intrusion detection systems) and firewalls.

3.3 P2P Traffic Verification

To analyze the P2P traffic caused by the verified hosts another Perl script was written that reads NetFlow raw data and assigns traffic to the verified hosts. Traffic is considered as P2P if source and destination port are in the non-well known range (for explanation see Section 2.6). It then generates interesting relations like the results in Section 4.3. Among other things it generates a ranking of the most active peers.

Chapter 4

Findings

In this chapter out findings will be presented. Section 4.1 shows the results that justify the assumption, that reachable hosts are a representative subset for the P2P host verification. Sections 4.2 and 4.3 finally present the results of the P2P host verification respectively traffic validation.

4.1 Validating the Verification

In this thesis it is tried to find a lower limit for the accuracy of the P2P host identification algorithm used by the PeerTracker. As described and explained in Section 2.5 this lower limit is investigated by comparing those hosts which are successfully verifiable to those which are reachable in some way. The following two sections will show that this assumption is reasonable. In Section 4.1.1 findings about the reachability of the identified peers are presented and possible reasons are named. In Section 4.1.2 measurements are shown which legitimate the choice of the subset for the peer verification of Section 4.2.

4.1.1 NAT and Firewall Usage in the SWITCH Network

If an assumed peer is located behind a NAT device or a restrictive firewall, no connection from outside is possible. Therefore the used verification algorithm has no chance to verify those peers. Hence it is essential to know which hosts are located behind a NAT or a restrictive firewall. Section 2.5.1 explains how it is detected if this is the case.
As expected, almost all hosts (or the according NAT device) respond to ICMP pings. Table 4.1 shows the result of a measurement made on a Wednesday at 14:15 (CET). Those hosts which do not respond to ICMP requests have either been shut down in the very short time between the identification by the PeerTracker and the online verification or the according NAT or firewall is intentionally configured in a way that ICMP requests are blocked.
To find out which hosts are reachable, the identified peers are TCP pinged on the three most used TCP ports. The exact values of not responding hosts in a representative

Assumed P2P Network	ICMP: reply received	TCP: *ACK* or *RST* received
All	99.5%	52.0%
EDonkey, Kademlia, Overnet (EKO)	100%	53.6%
Gnutella (GT)	97.8%	47.7%
FastTrack (FT)	100%	50%

Table 4.1: *Reachable hosts, 9.2.2005 14:15, duration of measurement 172s (total 423 hosts)*

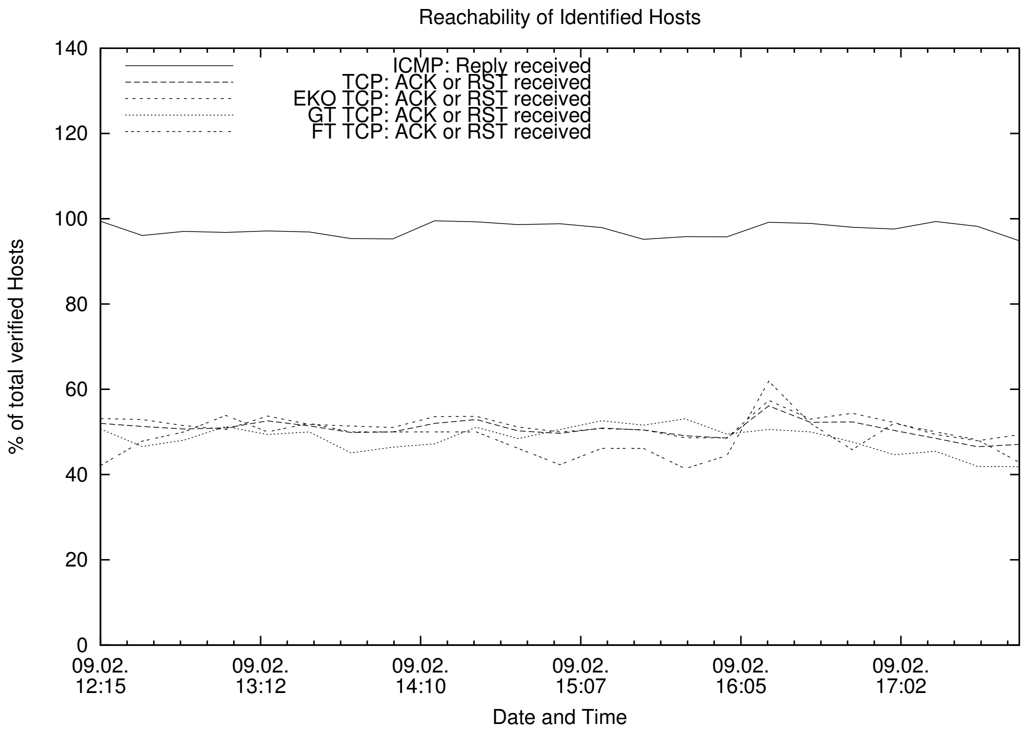

Figure 4.1: *Reachable hosts, 9.2.2005 (total 423 hosts)*

timeframe can be found in Table 4.1 and a record of the whole day is shown in Figure 4.1. Due to the reasoning in Section 2.5 and the results of the measurements it is save to say that around 50% of the verified hosts are located behind a NAT or have a restrictive firewall running. This assumption is reasonable because:

- The ICMP ping tests show that almost no host (or the according NAT) is down, so this is no reason for unreachability.

- Every host without a firewall running or behind a NAT would reply at least a *RST* to the connection attempt of a TCP ping. If the host is up (ICMP ping) and no answer at all is received, then it can be taken as a proof for a running firewall or a NAT.

- It can be seen that all three different network types have almost the same percentage of reachable peers. This matches with the presumption, that NAT devices and firewall usage are uniformly distributed in the SWITCH network.

- It matches with statements and measurements made in other projects. Among other things, that 50% of all Gnutella traffic is across a NAT or firewall [14].

	Reachable hosts	Unreachable hosts
P2P traffic in	127.2 kBit/s	112.0 kBit/s
P2P traffic out	167.2 kBit/s	140.8 kB/s

Table 4.2: *Average P2P traffic per host, 9.2.2005 15:00-15:15 (total 438 hosts)*

To strengthen this statements, the average P2P traffic of peers assumed to be behind a NAT or firewall is compared to those which are reachable by TCP. The results can be seen in Table 4.2.
It can be seen that these hosts, which are connectable from outside have slightly bigger download and upload rates. This can be explained with the fairness algorithms built into the P2P clients. E.g. the eDonkey network client eMule gives hosts which cannot be connected from outside lower priorities in the download queues (a so called LowID [9]). Such fairness mechanisms result in a lower download rate which corresponds with the results in Table 4.2.
As conclusion, it can be stated that the measurements done show that around 50% of all hosts are most likely located behind a NAT or a firewall that drops all incoming connections[1].

4.1.2 Choice of the Subset

As described in Chapter 2.5 and verified in the previous section, the subset for the peer verification will be those hosts which are reachable via TCP on the most used ports[2]. To find out how accurate this choice is, some more examination of the assumed P2P traffic in Table 4.2 has been made. The following arguments make the choice of the subset reasonable:

- It was found that the set of unreachable hosts has significant P2P download and upload rates. Thus many peers must be in the set of unreachable hosts.
- The amount of P2P data transferred per host is similar for reachable and unreachable hosts. This leads to the conclusion that about the same part of the unreachable hosts must be peers as of the reachable hosts. Otherwise the average down/upload rate of the unreachable hosts would not be as high as it is. The small difference can be explained with the fairness algorithms of the P2P-clients.

As seen above, it seems to be reasonable to take the reachable hosts as the subset for the peer verification process. Not only the considered traffic measurements but also the distribution of the reachable hosts on the different networks and the comparison with other projects results speaks in favor of an uniformly distribution of the identified peers into the categories reachable and unreachable. This means if a lower limit of the accuracy of the P2P host identification algorithm is found on the set of reachable hosts, it can be assumed with a high probability that this limit is also valid for the total set of identified hosts.

4.2 P2P Host Verification

One of the main goal of this thesis was to find a lower limit of the accuracy of the P2P host identifying algorithm used by the PeerTracker. As discussed in Section 2.5 this lower limit can be found by comparing the number of successfully verified hosts to those which are reachable from outside by TCP. This assumption was justified in Section 4.1.2. Several measurements have been made to find out how reliable the PeerTracker is. The results of one particular day can be found in Figure 4.2.
By considering only those hosts that are verifiable for the used online polling algorithm (those that are reachable by TCP connections from outside), we find a lower limit of 75%

[1]Some ports could also be unblocked by the user to get better download performance. However if the PeerTracker does not find this port the host stays unreachable.
[2]As defined earlier a host is reachable even if only a RST message is received.

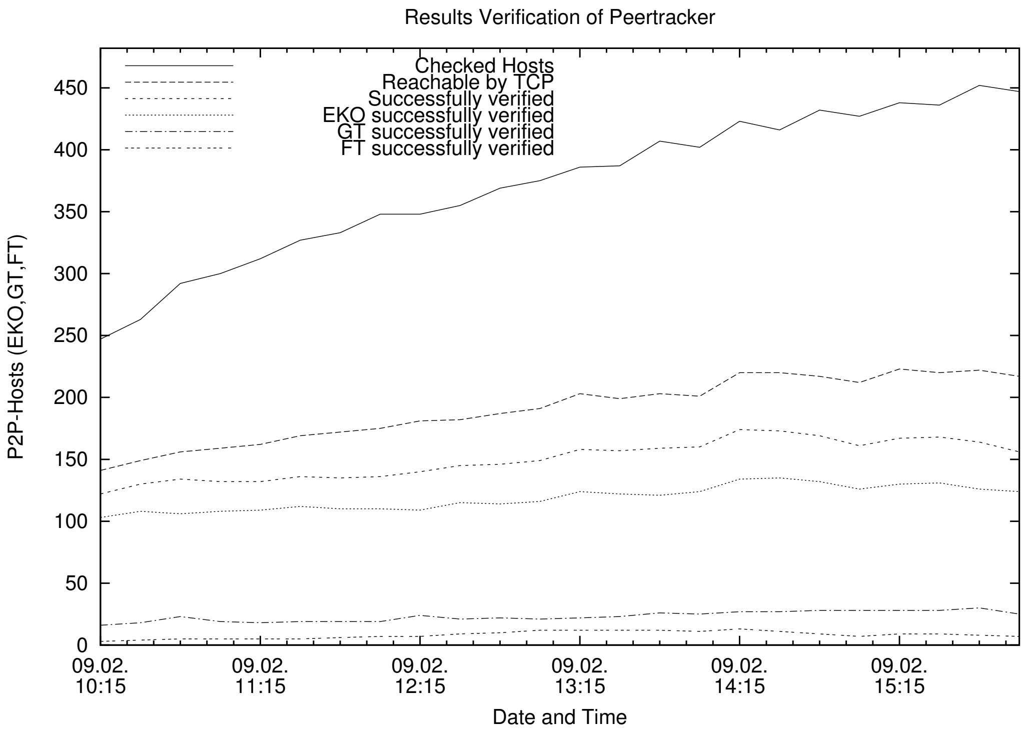

Figure 4.2: *Verified hosts, 9.2.2005*

successfully verified peers (see Table 4.3). That means with respect to the findings in Section 4.1 the PeerTracker works with an accuracy of at least 75%. This was the lowest value found in several measurements. However values up to 88% have been encountered. It seems to depend on the probability that P2P clients are shut down in the short time between the identification by the PeerTracker and the online verification. At night this is very improbable and therefore higher results are found then. Also the number of identified peers seems to have an influence. With many peers that have to be polled the verifying process has to handle much more threads in the same time and the verification takes a longer time. To longer the verification takes the higher the probability of clients to shut down in the meantime.

	Total hosts	Reachable hosts (TCP)	Successfully verified hosts
All networks	438	223 (50.9%)	167 (38.1%, 74.9% of reachable hosts)
EKO	317	161 (50.7%)	130 (41.0%, 80.7% of reachable hosts)
GT	95	50 (52.6%)	28 (29.5%, 56.0% of reachable hosts)
FT	26	12 (46.2%)	9 (34.6%, 75% of reachable hosts)

Table 4.3: *Reachable hosts vs. successfully verified hosts, 9.2.2005 15:15, duration of measurement 172s*

In Table 4.3, it can be seen that only about 38%[3] of the verified hosts are successfully identified as P2P clients. This rather lower number is caused by the fact that about 50% of the verified hosts could not even be reached on the tested ports because of NAT boxes, firewalls and IDS.
Kademlia, Overnet and eDonkey (the three networks that have been verified with the

[3]The upper limit of this value in the measurements made was 49%

same method) have the largest percentage of successfully verified peers (80,7%). In the same range lies the factor of successfully verified FastTrack peers. Only verified Gnutella peers have a little lower value, as predicted in Section 2.3.2.

4.2.1 Interpretation of Unsuccessfully Verified Hosts

Some identified hosts by the PeerTracker could be reached by the verification algorithm but not verified successfully. These hosts are analyzed more carefully in this section. In the following the different cases that occur are stated and interpretations are given.

TCP Connectable (*SYS ACK* Reply)

Hosts that accept a TCP connection on a particular port but do not reply to a handshake attempt of the verification algorithm do not occur often. In a measurement made in February 2005 to 179 hosts a TCP connection could be established whereof only 12 (6.7%) hosts could not be effectually verified. This can be seen in table 4.4. The different reasons for that are:

- The host is a peer but has been tested for the wrong P2P network. Today there exist many hybrid clients which make the PeerTracker's decision for a particular network even more difficult.
- The host is actually a peer of the assumed P2P network but the client does not reply to the tried handshake as expected. As shown in Section 2.3.2 it most likely occurs with the Gnutella protocol because its handshake is not defined strictly. In Table 4.4 it can be seen that there exist more Gnutella hosts that are TCP connectable but not successfully verified. In contradiction to the 6.7% found above, 20% of the Gnutella hosts that accepts TCP connections could not be verified.
- The host is not a peer and was wrongly identified by the PeerTracker. Another non-P2P service is running on the particular port. These hosts are false positives.

	Total hosts	TCP connectable	Successfully verified
All networks	438	179	167
EKO	317	134	130
GT	95	35	28
FT	26	10	9

Table 4.4: *TCP connectable hosts vs. successfully verified hosts, 9.2.2005 15:15, duration of measurement 172s*

Reachable by TCP but not TCP Connectable (*RST* Reply)

By comparing Table 4.3 with Table 4.4 it can be found that about 20% of the hosts that could be reached by TCP accepted no TCP connection. That means that 20% immediately replied with a TCP packet with set *RST* flag.

- The P2P client could have been shut down and the PeerTracker did not realize it. It is quite difficult to decide how long a host should be considered as active. If this interval is too short the status of the hosts are changed too often what adulterates the measurement. To get an impression about how many peers can be considered as down those with very little P2P traffic in the last evaluation period and a default P2P port among their three most used local ports are shown in Table 4.5. Due to the fact that a P2P default port is among the three most used local ports, the host's listening port was most likely found by the PeerTracker. Considering the little traffic and the fact the listening port has been found we find that a respectable amount (16 out of 44 in this measurement) of reachable but no TCP connection accepting host must have shut down their P2P client during the evaluation period.

	# Hosts	Hosts with ∅ P2P traffic <8kBit/s
# Hosts	32	16

Table 4.5: *Reachable hosts that immediately reset TCP connection with P2P default port among their most used local ports, average over 15 minutes, 9.2.2005 15:00-15:15*

- Another reason that a host does not accept any TCP connections on the tested ports could be that the listening port has not been found by the PeerTracker. Many incoming flows are needed to accurately determine a peer's listening port. In table 4.6 it can be seen that unsuccessfully verified hosts have much less average P2P traffic and suspicious flows than successfully verified peers. 47% of all unsuccessfully verified peers have less than 8 kBit/s average P2P traffic (only 0.6% of the successfully verified hosts have that less P2P traffic). The same thing can be observed with the amount of suspicious flows.

	Successfully verified	Unsuccessfully verified
# Hosts	167	56
∅ P2P traffic	328 kBit/s	198 kBit/s
∅ suspicious flows	26318	16064
Hosts with ∅ P2P traffic < 8kBit/s	1	18
Hosts with ∅ P2P traffic > 8kBit/s	166	38
Hosts with ∅ suspicious flows < 1500	5	15
Hosts with ∅ suspicious flows > 1500	162	42

Table 4.6: *Successfully verified hosts vs. unsuccessfully verified hosts, average over 15 minutes, 9.2.2005 15:00-15:15*

- The according host could have been identified as a P2P host by the PeerTracker even though there is no P2P client running on the host. This is another reason of unsuccessfully verified hosts with little traffic and suspicious flows (Table 4.6). These hosts are false positives.

To conclude with, it can be said that the accuracy of the identification process used by the PeerTracker must be quite higher than the found 75% in Section 4.2. In this section several reasons for an unsuccessful verification are shown and only two of them point to a situation where hosts were wrongly accused of P2P file-sharing.

4.3 P2P Traffic in SWITCH Network

This section deals with the estimated P2P border traffic of the SWITCH network. Several measurements have been made to verify the statements made in the foregoing thesis [4]. For all the measurements, the assumption from Section 2.6 regarding the decision, which

traffic should be considered as P2P, has been used. Furthermore an important fraction of the P2P traffic could not been taken into account because BitTorrent clients could not be considered in the verification process. Thus P2P traffic caused by BitTorrent clients has not been included into the measurements.
As already shown in Section 4.1.1 it does not matter if a host has been successfully verified or not when the caused P2P traffic is examined. Table 4.2 displays that hosts that are reachable for the verification process have about the same average P2P traffic than unreachable hosts. In the following with "verified traffic" the traffic of the verified hosts is meant, regardless if they were successfully or unsuccessfully verified.

4.3.1 General Traffic Consumption

Table 4.7 shows the traffic of 438 verified hosts. These are all hosts identified by the PeerTracker except the BitTorrent peers. As the table shows, the SWITCH network has more outgoing P2P traffic than incoming. SWITCH is more sender than a receiver - there is about 30% more outgoing P2P traffic than incoming. The incoming non-P2P traffic is four times higher than the outgoing. This is most likely caused by the WWW traffic which is certainly more incoming than outgoing.

	P2P traffic	Non-P2P traffic	Total
In	52.5 MBit/s (81.5%)	11.9 MBit/s	64.4 MBit/s
Out	67.8 MBit/s (95.8%)	3.0 MBit/s	70.8 MBit/s

Table 4.7: *P2P traffic vs. non-P2P traffic, average over 15 minutes, 9.2.2005 15:00-15:15*

81.5% of the total incoming traffic of the verified hosts is P2P traffic. On the outgoing traffic the fraction is even higher (96%). The most of the verified hosts, successfully or unsuccessfully, must be P2P host otherwise there would not be so much P2P traffic caused by this hosts. 89% of the total traffic of the verified hosts is P2P traffic.

	Traffic (in&out)
Total	1703 MBit/s
Total verified	135 MBit/s
Verified P2P	120 MBit/s

Table 4.8: *Total traffic analyzed and verified, average over 15 minutes, 9.2.2005 15:00-15:15*

Comparing the total traffic in the SWITCH network to the verified P2P traffic, we find a value of only 7%. The numbers can be found in Table 4.8. This result does not confirm the findings from the foregoing thesis [4]. There 28-36% of the total traffic was identified as P2P traffic. However the following things have to be taken into account:

- The measurements of this thesis have been made during the semester holidays. Most students are home and do not use the network anymore at university. As SWITCH connects almost all universities and other educational institutions of Switzerland this has to result in an considerable decrease of P2P traffic measured. It can be assumed that students use much more often P2P software than the staff at the universities.

- BitTorrent traffic has not been considered since those hosts could not be verified. However BitTorrent is considered as the most bandwidth consuming network of all. Some sources even say that it is responsible for more than the half of the total P2P traffic worldwide [31].

If we compare the different P2P networks regarding the P2P traffic we see that all the verified networks have similar average P2P traffic per host (Table 4.9). eDonkey, Kademlia and Overnet (EKO) have a significant difference between outgoing and incoming P2P traffic. This is most likely caused by the priority algorithm used in these networks. That means those peers, which do not accept TCP connections from outside, have a lower download priority than others. As found in Section 4.1.1 many hosts in the SWITCH network are not reachable from outside and have therefore lower download rates.
There is no priority for peers that allow incoming TCP connections in Gnutella what explains the similar average upload and download rates. Even though the values in Table 4.9 show that FastTrack peers have higher download than upload rates, this particular result is not very accurate. Only 26 FastTrack hosts had been verified.

	P2P traffic in	P2P traffic out	Total P2P traffic
Total	52.5 (0.120) MBit/s	67.7 (0.155) MBit/s	120.2 (0.275) MBit/s
EKO	35.3 (0.111) MBit/s	51.6 (0.163) MBit/s	86.9 (0.274) MBit/s
GT	13.3 (0.140) MBit/s	13.8 (0.145) MBit/s	27.1 (0.285) MBit/s
FT	3.9 (0.150) MBit/s	2.3 (0.088) MBit/s	6.2 (0.238) MBit/s

Table 4.9: *Traffic consumption by the different networks (in brackets average traffic per host over 15 minutes), 9.2.2005 15:00-15:15*

4.3.2 Other Findings

It is interesting to see that only a few hosts are responsible for a considerable amount of the P2P traffic in the SWITCH network. In Table 4.10 the first five of these so called "heavy hitters" are listed. It is surprising that 5 hosts out of 438 (1.1%) are responsible for 11.8% of the total P2P traffic. 4% of the verified hosts are accountable for 30% of the total verified P2P traffic. To get such a high down- and upload rate one has to change its P2P client's listening ports to a non-default port to get around traffic shaping restrictions often in use[4]. As expected, none of the 10 most active verified hosts use a default listening port.

Rank	∅ P2P traffic (in&out)	% of total P2P traffic (438 hosts)
1.	4.89 MBit/s	4.1%
2.	3.13 MBit/s	2.6%
3.	2.89 MBit/s	2.4%
4.	2.67 MBit/s	2.2%
5.	2.51 MBit/s	2.1%
All first 5 together	14.19 MBit/s	11.8%

Table 4.10: *"Heavy hitters" in SWITCH network, average traffic per host over 15 minutes, 9.2.2005 15:00-15:15*

The usage of P2P default ports is an important topic for traffic shaping mechanisms. In Table 4.11 the traffic of eDonkey, Kademlia and Overnet (EKO) peers using a P2P default port is compared to those, which do not use a default port. The other networks are not considered here because the number of according hosts was not enough for a representative conclusion. EKO peers on default ports have less P2P traffic than those on non default ports. This can have the following reasons:

[4]The ETH limits the incoming traffic on default P2P ports to 10 MBit/s

- Peers with non default listening ports are not as affected by traffic shaping mechanisms as other peers. If they download a file from another peer with a non-default listening port traffic shaping based on default ports is useless.
- The hosts that do not use a P2P default port are harder to identify by the PeerTracker, only those with enough suspicious traffic and flows are detected. This results in a higher average traffic for these hosts.

In Table 4.11, it can also be seen that hosts that do not use a P2P default listening port have a higher upload/download quota. This can also be explained with traffic shaping mechanisms. Incoming traffic is restricted because the source port sometimes is a P2P default port (listening port of the other peer). However outgoing traffic always has non-default source and destination ports since the source is the non-default port of the inside peer and the destination port is a freely chosen one. Therefore the upload rate of such a peer is higher.

	∅ P2P traffic in	∅ P2P traffic out	Total ∅ P2P traffic	# Hosts
EKO default	116.0 kBit/s	157.0 kBit/s	273.0 kBit/s	94
EKO non-default	168.6 kBit/s	237.1 kBit/s	405.7 kBit/s	36

Table 4.11: *Clients on default port vs. on non-default port (only successfully verified hosts), average traffic per host over 15 minutes, 9.2.2005 15:00-15:15*

Chapter 5

Conclusion

The goal of this thesis was to verify the P2P host identification method developed and implemented by the foregoing master thesis ”P2P Population Tracking and Traffic Characterization of Current P2P File-sharing Systems” [4]. Furthermore the found traffic usage of P2P systems in the SWITCH network should be validated.

5.1 P2P Host Verification

In order to verify identified P2P host, polling algorithms for eDonkey, Gnutella, Overnet, Kademlia and FastTrack networks have been developed and implemented. The verifying mechanisms base on the particular handshakes of the different networks. Polling P2P hosts is a difficult task because many hosts cannot be reached by a verification application - restrictive firewalls, NAT boxes and intrusion detection systems make this often impossible. However, in this thesis it could be shown, that a lower limit of the accuracy of the identification algorithm can also be found on the subset of reachable hosts (by TCP). With this supposition it can be stated that the identification algorithm by [4] works with an accuracy of at least 75% for the examined P2P networks[1]. BitTorrent peers were not taken into account because direct online polling is not possible.
Additionally some facts about the usage of NAT boxes and firewalls in the SWITCH network could be found. Around 50% of all identified P2P hosts were found to accept no TCP connections from outside.

5.2 P2P Traffic Verification

In this thesis the P2P traffic usage could not be verified completely. The verified (successfully and unsuccessfully) P2P hosts are only responsible for around 7% of the total border traffic of the SWITCH network. However it has to be considered that the most traffic consuming P2P network (BitTorrent) could not been taken into account. Additionally measurements have been taken during semester holidays when the most intensive P2P users (the students) do not use the SWITCH network.
Secondary many interesting findings about the P2P traffic in the SWITCH network have been made. The SWITCH network sends more P2P traffic than it receives - there is about 30% more outgoing P2P traffic than incoming.
Furthermore only a few peers are responsible for a large amount of P2P traffic. Less than 4% of the verified hosts are accountable for more than 30% of the total verified P2P traffic in the SWITCH network.

[1]Depending on the number of verifiable hosts, this lower limit can go up to 88%.

It has also been found that many peers that reach high download an upload rates do not use default listening ports. It must be assumed that some host operators manually change the listening ports of their P2P clients on purpose to circumvent traffic shaping mechanisms (which for example ETH Zürich is using).

Chapter 6

Outlook and Further Work

6.1 Verifying BitTorrent Hosts

BitTorrent peers cannot be verified online in the way the other analyzed networks were, because the TCP handshake that initiates the connection does already contain information about the requested file (hash-value). In contrast to eDonkey, Overnet, Gnutella, FastTrack and Kademlia peers a BitTorrent peers does not authenticate itself as a participant of the network unless it has one of the requested files. If the requested file is not offered by the contacted peer the host immediately drops the connection.
Another approach to verify BitTorrent peers would be to gather information from trackers. A BitTorrent client downloading a file receives a list with other peers that share the according file periodically from the tracker. A BitTorrent peer would be verified if it appeared in one of those lists. This approach can only lead to partial success if the most downloaded files and the according trackers are known. This could be done by collecting the most downloaded files from websites [1] that offer torrents. With this information a list of a considerable part of the BitTorrent peers could be generated. However the problem is the huge amount of existing trackers and the almost endless variety of shared files.

6.2 Detection whether Hosts are behind NAT/Firewall

In this thesis it has been assumed when a host is not reachable via TCP (that means not even a *RST* packet is returned), it is most likely located behind a NAT box or a firewall that blocks incoming TCP connections. To verify this assumption an algorithm could be developed that finds those hosts which only have outgoing connections from NetFlow data. Several tests made showed that it is possible to find out which host initiated the connection by comparing the timestamps of the according two flows[2]. However this would require a buffer that contains several thousands TCP flows to find two related flows and then find out which host opened the connection.

6.3 Verification of Hosts behind NAT/Firewall

6.3.1 eDonkey, Kademlia, Overnet

A client behind a NAT device or a restrictive firewall cannot be polled directly because no incoming TCP connections are possible. eDonkey uses IDs to identify the different peers. There exist low IDs and high IDs. Peers that are not connectable on their listening ports

[1] E.g. http://www.btsites.tk/
[2] For one TCP connection there always exist two NetFlows, one in each direction.

get a low ID. If the ID of the host that wants to be verified as a peer can be found, one would know if the host was located behind a NAT or a firewall. There exists a mechanism in the eDonkey protocol to get the ID from the host's IP address from a server, but it could not be tested due to time constraints [11].

6.3.2 Gnutella

As eDonkey peers Gnutella clients behind a NAT or a firewall cannot be polled directly. The protocol does it with a *push* message which contains the ID of the client that offers the wanted file. One possible approach could be to send such a *push* message to a host that wants to be verified. However, two problems exit: First, we do not have the Gnutella-ID (GUID[3]) of the host we want to push, second the *push* message is only routed back the way it came [26]. If the message did not come at all, it will most likely not be forwarded. Therefore another approach has to be found.

6.3.3 FastTrack

As widely known, the FastTrack protocol is not an open source project. Some attempts to reverse engineer the protocol have been made and even some open source clients that can participate in the network are available. However there does not exist a fully verified description of the network specifications and therefore no significant statement about possibilities to verify peers behind NAT devices and restrictive firewalls could be made. One eventuality that could be found, is the implementation of an open source FastTrack Client (giFT) that uses a callback sequence to download from firewalled peers [8]. If this was a standard within FastTrack it could be used to verify also peers which cannot be polled directly.

[3]The GUID is free selectable and can only be obtained by a successful search result that is returned.

Bibliography

[1] *"Slyck.com - File Sharing News and Info"*
http://www.slyck.com/ (Ferbruary 2005)

[2] *T. Christiansen, N. Torkington: "Perl Cookbook"*
O'Reilly (2003)

[3] *L. L. Peterson, B. S. Davie: "Computer Networks, A System Approach"*
Morgan Kaufmann Publishers (2000)

[4] *L. Hämmerle: "P2P Population Tracking and Traffic Characterization of Current P2P File-sharing Systems"*
Master's Thesis (2004)

[5] *P. Jardas: "P2P File-sharing Systems Real World NetFlow Traffic Characterization"*
Bachelor's Thesis (2004)

[6] *E.D. Team and C.Schlegel, "UPFrame: UDP Processing Framework."*
http://www.tik.ee.ethz.ch/∼ddosvax/upframe/ (2004)

[7] *E. Siever, S. Figgins and A. Weber: "Linux in a Nutshell"*
O'Reilly (2003)

[8] *"The FastTrack Protocol"*
http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?rev=1.19&content-type=text/vnd.viewcvs-markup
(July 2004)

[9] *"EMule - A file-sharing client for eDonkey2000"*
http://www.emule-project.net (February 2005)

[10] *"EDonkey 2000 - An eDonkey Client"*
http://www.edonkey2000.com/ (February 2005)

[11] *"Unofficial eDonkey Protocol Specification v0.6.2"*
http://sf.gds.tuwien.ac.at/00-html/p/pdonkey/eDonkey-protocol-0.6.2.html (2004)

[12] *Y. Kulbak and D. Bickson: "The eMule Protocol Specification"*
http://www.cs.huji.ac.il/labs/danss/presentations/emule.pdf (January 2005)

[13] *P. Maymounkov and D. Mazières: "Kademlia: A Peer-to-peer Information System Based on the XOR Metric", tech. rep.*
http://www.sics.se/∼sameh/research/P2P/Kademlia/Kademlia-%20(SV%20newer)%20%20A%20Peer-to-peer%20Information%20System%20Based%20on%20the%20XOR%20Metric/kpos.pdf

[14] *J. Kuptz: "Gnutella: Unstoppable by Design"*
http://www.wired.com/wired/archive/8.10/architecture.html (2000)

[15] *ETH Informatikdienste: "Filter zwischen ETHZ und SWITCH"*
http://www.id.ethz.ch/services/list/netzwerk/filter/index (2005)

[16] *"RFC 793 - Transmission Control Protocol"*
http://www.faqs.org/rfcs/rfc793.html (1981)

[17] *"ICMP Protocol Overview"*
http://www.freesoft.org/CIE/Topics/81.htm

[18] *Webopedia: "What is TCP?"*
http://www.webopedia.com/TERM/T/TCP.html

[19] *"Using IP Network Address Translation (NAT)"*
http://www.support.nsgdata.com/RouterDocs/software_documentation/openroute_32/ipprot/natransa.htm

[20] *Wikipedia: "Firewall"*
http://de.wikipedia.org/wiki/Firewall

[21] *Slyck.com: "BitTorrent Remains Powerhouse Network"*
http://www.slyck.com/news.php?story=649 (January 2005)

[22] *Slyck.com: "Damning Kazaa Evidence Released"*
http://www.slyck.com/news.php?story=656 (February 2005)

[23] *"Port Numbers and Services Database"*
http://www.sockets.com/services.htm

[24] *IT Innovations & Concepts: "Digital Piracy - Definitive P2P piracy figures for Year 2003...", tech. rep.*
http://www.itic.ca/DIC/News/2004/08/11/P2P_piracy_figures_2003.html
(July 2004)

[25] *The official BitTorrent Homepage: "BitTorrent Protocol Description"*
http://bittorrent.com/protocol.html

[26] *"The Gnutella Protocol Specification v0.4"*
http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[27] *"SWITCH - The Swiss Education and Research Network"*
http://www.switch.ch/ (February 2005)

[28] *"Cisco NetFlow Services Solution Guide"*
http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm
(Ausgust 2004)

[29] *A. Wagner and T. Dübendorfer: "DDoSVax Project"*
http://www.tik.ee.ethz.ch/ ddosvax (January 2005)

[30] *B. Cohen: "Incentives build robustness in BitTorrent", tech. rep. (May 2003)*
http://bitconjurer.org/BitTorrent/bittorrentecon.pdf

[31] *News.com: "Survey: Movie-swapping up - Kazaa down"*
http://news.com.com/Survey+Movie-swapping+up+Kazaa+down/2100-1025_3-5267992.html?tag=nl

[32] *"Sharman Networks"*
http://www.sharmannetworks.com (February 2005)

[33] *"RIAA - Recording Industry Association of America"*
http://www.riaa.com/default.asp (February 2005)

[34] *"MPAA - Motion Picture Association of America"*
http://www.mpaa.org/ (February 2005)

[35] *"Tcpdump"*
http://www.tcpdump.org/ (July 2004)

[36] *"Ethereal - The world's most popular network protocol analyzer"*
http://www.ethereal.com/ (February 2005)

[37] *"Kerio"*
http://www.kerio.com/ (2005)

[38] *"Shareaza - The Ultimate P2P Client"*
http://www.shareaza.com/

[39] *"Diet K"*
http://www.dietk.com/

[40] *"Azureus - Java BitTorrent Client"*
http://azureus.sourceforge.net/