Praktische Untersuchung und Vergleich von Gnutella, KaZaA, eDonkey und SoulSeek

Studienarbeit am Institut für Telematik
Prof. Dr.rer.nat. Martina Zitterbart
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Christian Wagner

Betreuer:
Prof. Dr.rer.nat. Martina Zitterbart
Dr.rer.nat Thomas Fuhrmann
Dipl.-Ing. Kendy Kutzner

Tag der Anmeldung: 21. Juli 2003
Tag der Abgabe: 1. Dezember 2003

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 1. Dezember 2003

Inhaltsverzeichnis

1. Einleitung

Peer-to-Peer Netze werden immer populärer. Mit sogenannten File-Sharing Programmen tauschen Millionen von Benutzer untereinander Dateien. Da die meisten angebotenen Dateien jedoch rechtlich geschützt sind und somit illegal verbreitet werden, versuchen die Rechteinhaber, besonders die Musik- und Filmindustrie diese Tauschbörsen zu schließen. Waren die Netze der ersten Generation noch relativ einfach zu schließen, weil sie auf einen zentralen Server angewiesen waren, so sind die aktuellen Peer-to-Peer Tauschbörsen viel weiter entwickelt. Sie benutzen zum Teil verschiedene Netzwerkarchitekturen und bieten unterschiedliche Funktionalitäten an.

Ziel der Arbeit ist die praktische Untersuchung der derzeit populärsten File-Sharing Netzwerke: KaZaA, eDonkey und SoulSeek. Dazu wird die Struktur der Netze analysiert und die wichtigsten Funktionalitäten der Clients vorgestellt. Zur Analyse des SoulSeek Protokolls wurde ein Dissektor implementiert, mit dem man in Ethereal den Nachrichtenverkehr analysieren kann.

1.1 Gliederung der Arbeit

Die Arbeit beginnt in Kapitel 2 mit einem historischen Rückblick, mit dem Ziel das Phänomen File-Sharing zu erklären. Es folgt eine Definition des Begriffs Peer-to-Peer. In 2.3 werden dann die verschiedenen Modelle der Peer-to-Peer Netzwerken untersucht, sowie deren Vor- und Nachteile beschrieben. Anschließend werden die Anwendungsbereiche, in denen heute Peer-to-Peer Systeme eingesetzt werden vorgestellt.

In Kapitel 3 werden dann die drei populärsten File-Sharing Programme im Detail analysiert. Die Kapitel beginnen mit einem Überblick über die Betreiber der Netze und die aktuellen Clients. Danach folgt eine Untersuchung der Struktur der Netze, sowie der Unterschiede beim Login und der Suche nach Dateien. Die verschiedenen Systeme unterscheiden sich ebenfalls in der Funktionalität der Clients. Die Charakteristiken der

einzelnen Programme, verdeutlichen die Vorteile und die Einsatzmöglichkeiten und sind mitentscheidend für die Popularität der Clients. Anhang C enthält Statistiken über die drei beschriebenen Netze.

Zur Aufzeichnung des Nachrichtenverkehrs und Analyse der Protokolle kann ein Packet-Sniffer und ein Network-Analyser benutzt werden. Kapitel 4 beschreibt Ethereal, ein Programm, das beide Funktionen in sich vereint. Zur Analyse der Protokolle wird ein spezieller Dissektor benötigt. Ist dieser nicht in Ethereal vorhanden, kann er neu implementiert werden. Als Teil dieser Arbeit ist ein SoulSeek Dissektor für Ethereal geschrieben worden, dessen Funktionsweise in 4.2.3 beschrieben wird. Anhang B enthält Ausschnitte aus dem Quellcode des Dissektors mit den wichtigsten Funktionen.

Mit Ethereal werden die Nachrichten aufgezeichnet. Die so gewonnenen Nachrichtensequenzen sind in Kapitel 5 dargestellt. Die Bedeutung und der Aufbau der einzelnen Nachrichten, sowie der in ihnen enthaltenen Datenfelder sind mit Reverse-Engineering ermittelt worden. Diese Informationen wurden anschließend wieder im SoulSeek Dissektor implementiert, so dass man für jede übertragene Nachricht den Namen und die Bedeutung aller Datenfelder sehen kann. In Anhang A steht schließlich die komplette Protokollspezifikation, mit einer Auflistung aller Nachrichten, die im Rahmen dieser Arbeit ermittelt wurden.

Nach einer Zusammenfassung wird ein allgemeiner Ausblick über Peer-to-Peer Tauschbörsen gegeben und ihre Stellung in der heutigen Gesellschaft.

2. Peer-to-Peer Netze

2.1 Geschichte

In den letzten Jahren ist ein erfreut sich die Peer-to-Peer Technologie immer größerer Beliebtheit. Millionen von Internernutzer tauschen Milliarden Dateien untereinander (siehe Anhang C: Statistiken), so dass sogar ganze Industriezweige sich in ihrer Existenz bedroht fühlen.

Doch Peer-to-Peer ist keineswegs eine neue Technik. Das Konzept des Informationsaustauschs zwischen zwei oder mehreren gleichen Endpunkten ist so alt wie das Internet selbst. ARPANET, aus dem später das Internet wurde, wurde ursprünglich mit dem Ziel entwickelt unterschiedliche Arten der vorhandenen Netze sowie zukünftige Technologien in einer gemeinsamen Netzarchitektur zu verbinden. Jeder Knoten in diesem Netz konnte mit einem anderen Knoten Informationen austauschen.

Erst Anfang der 90er mit der Einführung des Web-Browsers und mit der Kommerzialisierung des Internets, verbreiteten sich die Client-Server Anwendungen die besser zu dem derzeitigen Verhalten der Internetsurfer passten. Die meisten waren nur für kurze Zeit online, um sich ein paar Webseiten anzusehen oder Daten herunterzuladen. Mit der Organisation der Netze mittels Firewalls und durch die Benutzung von dynamischen IP-Adressen oder Network Address Translation (NAT), wurde das Internet zu einer Client-Server Struktur. [32]

Die heutigen Peer-to-Peer Applikationen benutzen die ungenutzten Upload-Ressourcen der Internetnutzer und verbinden sich wieder zu einem gemeinsamen Netz in dem jeder Peer gleichberechtigt ist und mit jedem anderen Peer Informationen tauschen kann.

Das erste Peer-to-Peer Programm das sich explosionsartig verbreitete und die Welle der Tauschbörsen auslöste war Napster. Mit der Einführung der MPEG 1 Layer 3 Kompression kurz mp3 war ist es möglich geworden, Audio Dateien um Faktor 12 zu verkleinern, und somit die nur mehr 1-3 Megabyte grossen Dateien im Internet zu

tauschen. Die Lieder konnten von Webseiten heruntergeladen werden oder wurden auf speziellen FTP-Servern zum Tausch angeboten. Im Jahr 1999 wurde durch Napster das Tauschen von Musikdateien sehr vereinfacht. Mit Napster konnte jeder Benutzer selber Musik zum Download anbieten und unter allen angebotenen Liedern nach einem bestimmten Titel suchen und diesen dann direkt von einem anderen Napster-Benutzer herunterladen. Napster wurde schnell zur „Killer-Applikation“ und machte Peer-to-Peer Programme wieder populär.

Der Musikindustrie war diese Tauschbörse schnell ein Dorn im Auge und sie versuchte das Tauschen von lizenzierten Liedern zu unterbinden. Eine erste Maßnahme war das Verhindern von Suchanfragen nach lizenzierten Liedern. Dies war möglich weil Napster die Liste der angebotenen Lieder in einem zentralen Index-Server katalogisierte. Somit konnten ungewollte Suchanfragen durch ein Filtersystem verhindert werden. Doch eine simple Umbenennung des Künstlers oder Liedes ermöglichte wieder eine erfolgreiche Suche. Letztendlich wurde Napster durch Gerichtsbeschluss geschlossen, indem man den zentralen Index-Server ausschaltete. Diese erste Tauschbörse verdeutlichte die Schwachstellen in der Netz-Struktur, nämlich des einzigen zentralen Servers, ohne den das ganze Netz lahmgelegt war. Selbst 2 Jahre nach der Schließung von Napster steht der Name immer noch für den Erfolg von File-Sharing Programmen so dass man jetzt versucht mit Napster 2 eine legale Alternative bereitzustellen.

Der Erfolg von Peer-to-Peer Systemen ist heute ungebremst. Mit der zunehmend wachsenden Zahl an Internet-Nutzern und mit der weiten Verbreitung von schnelleren Internetanbindungen wie z.B. DSL (Digital Subscriber Line) sowie der Einführung von Zeit- und Volumenunabhängigen Verbindungen zu einem festen Monatspreis als Flatrate sind die Tauschbörsen populärer als je zuvor. Die bessere Komprimierung von Video-Material durch DivX führte zu einer ähnlichen Situation bei der Filmindustrie wie damals bei der Musikindustrie. Es werden ganze Filme in CD-gerechten Dateien von 700 Megabyte angeboten, ja sogar 1:1 DVD-Kopien mit bis zu 9 Gigabyte werden zum Tausch angeboten. Doch die File-Sharing Programme sind nicht nur auf Musik oder Filme beschränkt, jede Art von Datei kann in den Tauschbörsen gefunden werden, ganze Musikalben als Archiv gepackt, eBooks, Programme als CD-Image usw.…
Es gibt eine unüberschaubare Anzahl von Tauschbörsen, die zum Teil auf spezielle Inhalte spezialisiert sind und besondere Funktionen, wie das Chatten unter den Benutzer oder das Kommentieren der angebotenen Dateien bieten.

2.2 Definition

Clay Shirky definiert Peer-to-Peer folgendermaßen:

"P2P is a class of applications that takes advantage of resources -- storage, cycles, content, human presence -- available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant or total autonomy from central servers.

That's it. That's what makes P2P distinctive" [33]

P2P benutzt also meist wenig genutzte Ressourcen am Rande des Internets und bindet diese in einem eigenen Netz, unabhängig von den (dynamischen) IP-Adressen, zusammen. Diese Endpunkte operieren unabhängig vom Domain-Name-System (DNS) und funktionieren meist völlig autonom, ohne zentralen Server.

Peer bedeutet in Deutschen „der Ebenbürtige“, „der Gleichgestellte“. Peer-to-Peer kann wörtlich "von gleich zu gleich" übersetzt werden und unterstreicht, dass die Endknoten in einem P2P System gleichgestellt sind.

Im Gegensatz dazu wird in einem Client/Server-Modell strikt zwischen dem Server und den Clients unterschieden. Der Server bietet die Daten an, und die Clients, verbinden sich mit diesem um die Daten abzurufen oder wieder zurückzusenden. Die Clients bauen untereinander aber keine Verbindung auf, sondern kommunizieren immer nur mit dem Server. In einer Peer-to-Peer-Architektur können Rechner, die normalerweise als Client in einem Client/Server-Modell fungieren, zugleich die Rolle des Client und des Server einnehmen und werden zu so genannten Servents.

Diese Funktionsweise ermöglicht den direkten Austausch von Dateien. Es können aber auch andere Ressourcen bereitgestellt werden, wie z.B. die Rechenleistung der Maschinen oder der leere Festplattenspeicher. Ein derartiges System verteilt die Last auf die Servents und ermöglicht es, spezielle Dienste zweckmäßiger zur Verfügung zu stellen.

2.3 Modelle von P2P-Netzen

Auch wenn der Dateiaustausch selbst auf Peer-to-Peer-Ebene abläuft, brauchen manche dieser Systeme eine zentrale Komponente. Um Daten zu suchen und mit anderen Peers zu tauschen, muss man erst deren Adresse kennen. Die Kommunikation zwischen den Peers wird aber dadurch erschwert, dass diese keine feste IP-Adresse haben, sondern nur für eine gewisse Zeit online sind, und sich bei jeder Sitzung ihre dynamisch zugewiesene IP-Adresse ändert.

Die Peer-to-Peer Systeme müssen also ein Namenssystem benutzen, das unabhängig vom DNS System operiert, und außerdem prüfen kann ob der jeweilige Peer gerade online ist oder nicht. Dieses Problem kann leicht gelöst werden indem man einen zentralen Server einführt, dessen Adresse fest ist und der all diese Informationen über die einzelnen Clients verwaltet und so die Kommunikation zwischen den Peers vermitteln kann.

2.3.1 P2P-Systeme mit einem zentralen Server

P2P-Systeme mit einem zentralen Server haben die einfachste Netz-Struktur und wurden die ersten Peer-to-Peer Netzwerken verwendet. Dieses Modell wurde auch von Napster benutzt. Hauptbestandteil eines solchen Netzes ist ein zentraler Server. Um

sich im Netz anzumelden genügt es sich beim Server zu registrieren, dieser verwaltet die IP-Adressen aller Benutzer und kann somit eine Kommunikation zu einem anderen Benutzer vermitteln. [34]

Die Vorteile einer solchen Netzstruktur ist dass sich eine Suche nach Dateien sehr einfach realisieren lässt. Da der Server alle Benutzer kennt, kann er auch gleichzeitig zum Index der Nutzer einen Index aller vorhandenen Dateien anlegen und somit als Suchmaschine für das Netz dienen. Beim Anmelden schickt der Benutzer also gleich eine Liste aller von ihm zum Download freigegebenen Dateien mit. Will ein Benutzer nun nach einer bestimmten Datei suchen, schickt er die Suchanfrage an den Server. Dieser generiert eine Liste mit allen gefundenen Dateien und den Peers die sie anbieten und schickt diese zurück an den Benutzer, der anschließend die Datei herunterladen kann indem er eine direkte Verbindung zu einem Peer aus dieser Liste aufbaut. Zusätzlich zum Namen der Datei werden meistens noch weitere Informationen mitgespeichert, wie die Größe der Datei, der Name des Künstlers, die Dauer des Lieder, die Bitrate usw.
Die Suche erfolgt schnell und durchsucht alle Dateien, ohne dabei ein große Netzlast zu produzieren.

Ein anderer Vorteil dieser Art von Modell ist dass man das P2P-Netz besser verwalten und kontrollieren kann, da der Server alle Benutzer kennt und auch ihre angebotenen Dateien, ist es sehr einfach möglich z.B. bestimmte Benutzer zu privilegieren.
Doch diese totale Kontrolle der Suchanfragen bietet außerdem einen erheblichen Nachteil für die Benutzer. Das System kann von den Betreibern beliebig reguliert werden. Eine Zensur der Dateien wird ermöglicht indem man im Server ein Filtersystem einrichtet, wie es die amerikanisch Musikindustrie RIAA gefordert hat um das Tauschen von lizenzierten Liedern zu unterbinden. Die Nutzer sind auch nicht anonym. Sie können eindeutig vom Server identifiziert werden z.B. um Benutzerprofile anzulegen.

Der zentrale Server ist jedoch zugleich der größte Schwachpunkt in diesem Modell. Da die Anmeldung aller Peers über diesen Server laufen und alle Suchanfragen vom Server behandelt werden, kann dieser hoher Traffic den Server belasten, er ist der Flaschenhals im System, der eine Skalierung der Netzes einschränkt. Schlimmer noch wäre ein totaler Ausfall des Servers. In diesem Fall wäre eine weitere Suche oder Anmeldung im Netz unmöglich, der Server ist also der so genannte „single point of failure“.

Es ist umstritten ob es sich bei einem solchen System überhaupt um ein richtiges Peer-to-Peer System handelt, da es abhängig von einem zentralen Server ist und somit eher einer Client/Server Struktur entspricht. Der eigentliche Datentransfer wird aber realisiert indem die Peers sich direkt miteinander verbinden.

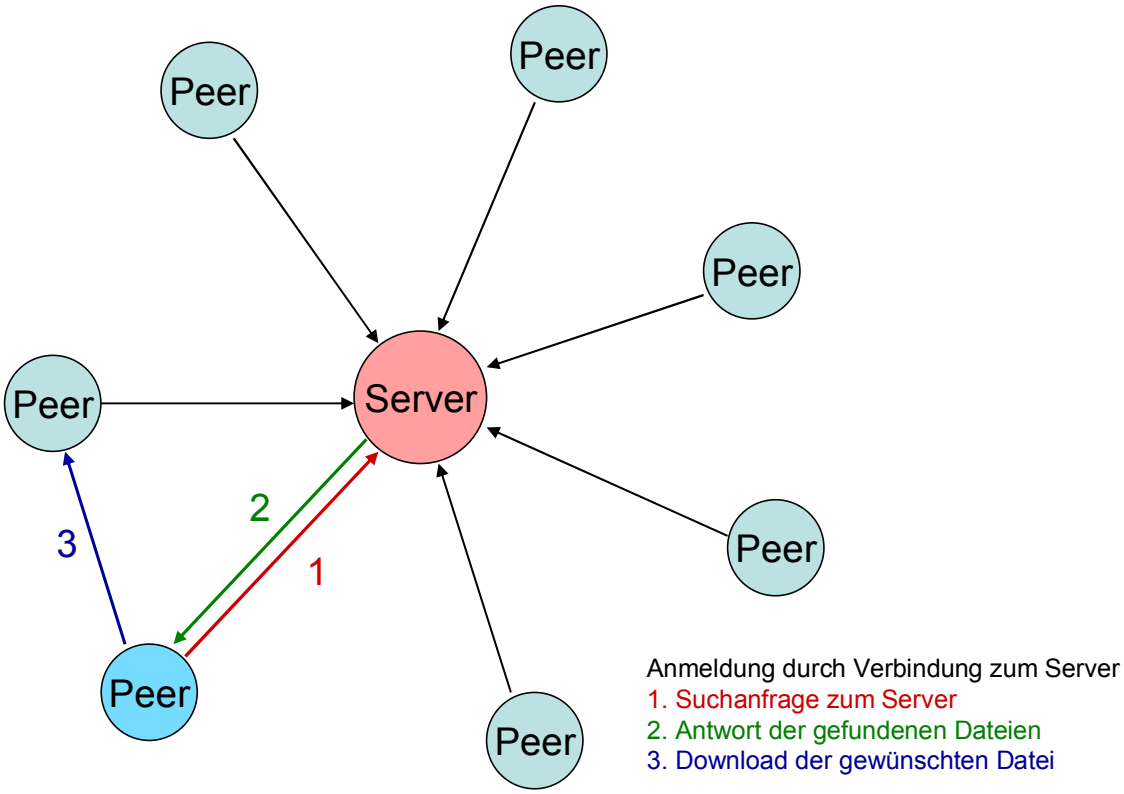


Abbildung 1: Suche in Netzwerken mit zentralem Server

Vorteile:

- Komplette Suche innerhalb aller angebotenen Dateien
- Sehr schnelle Antwortszeiten bei einer Suche
- Geringe Netzlast
- Einfache Verwaltung der Benutzer

Nachteile:

- Server beschränkt Skalierung des Netzes
- Server ist Single-point-of-failure
- Leicht zu zensieren durch Filtersystem im Server
- Leichte Erstellung von Benutzerprofilen

2.3.2 Dezentrale P2P-Systeme

Die dezentralen P2P-Systemem verwenden keinen zentralen Server. Gnutella war die erste Applikation die dieses Modell einsetzte. Das Netz besteht ausschließlich aus gleichwertigen Peers. Das bedeutet aber auch dass diese Peers die Funktionen übernehmen müssen, für die bei dem vorherigen Modell der Server verantwortlich war, also Anmeldung und Aufbau des Netzes, sowie die Suche nach Dateien. Jeder Peer ist also zum Teil Server und Client zugleich und wird daher auch als Servent bezeichnet. Diese Knoten sind mit einem oder mehreren anderen Knoten verbunden die insgesamt das Netz bilden. [34]

Bei der Anmeldung an einem dezentralen P2P-System baut der Servent eine Verbindung mit einem anderen Servent auf, der sich bereits im Netz befindet und dessen Adresse er aus einem Host-Cache, einer zufälligen Liste von Peeradressen, hat. Dieser Servent schickt anschließend die Informationen des neuen Benutzers an alle Knoten mit denen er verbunden ist. Diese Knoten verschicken die Information wiederum weiter. Somit propagiert sich die Nachricht im Netz und erreicht nach und nach alle Knoten, die den neuen Benutzer in ihre Liste auf, das Netz wird vergrößert. Ist der neue Benutzer erst mal im Netz „bekannt“, kann er auch nach Dateien suchen.

Die Suchanfrage wird dabei in gleicher Weise weitergeleitet wie bei der Anmeldung. Um zu verhindern dass die Suchanfrage zu lange im Netz herumwandert oder sich Schleifen bilden, muss die Suche durch einen Horizont begrenzt werden. Dies wird üblicherweise mit einem „Time to Live“ (TTL) Wert realisiert, der bei jeder Weiterleitung der Suchanfrage um 1 verringert wird. Besitzt ein Servent die gewünschte Datei, schickt er eine Antwortnachricht auf gleichem Weg zurück. Anschließend kann der Benutzer eine direkte Verbindung zu diesem Knoten aufbauen und die Datei herunterladen.

Ein großer Vorteil dieses P2P-Modells ist, dass die einzelnen Servents völlig autonom arbeiten, sie brauchen keine zentrale Anlaufstelle. Dies hat auch zur Folge dass das gesamte Netz viel robuster ist als solche mit einem zentralen Server. Wenn ein Servent ausfällt hat dies nur sehr geringe Folgen für System, da die Servents untereinander über mehrere Wege miteinander verbunden sind.

Diese Art von Tauschbörsen kann auch nicht zensieren, da jeder Peer gleich ist und die Suchanfragen über viele Wege weitergeleitet werden. Es ist ebenfalls viel schwieriger Benutzerprofile anzulegen, weil die Suchanfragen die IP-Adresse des Suchenden nicht beinhalten.

Das Hauptproblem bei dezentralen P2P-Systemen ist die hohe Netzlast, die jedes Mal bei Anmeldungen und Suchanfragen entsteht. Außerdem senden die Servents in regelmäßigen Abständen noch Ping-Nachrichten, zur Entdeckung weiterer Peers, die wie Suchanfragen im Netz weitergeleitet werden. Jeder Empfänger schickt dann eine Pong-Nachricht wieder zurück, die benutzt wird um einen lokalen Host-Cache beim Sender der Ping-Nachricht zu füllen.

Die Skalierbarkeit des Netzes ist somit stark von der verfügbaren Bandbreite abhängig. Benutzer von 56k Modem sind fast ausschließlich mit der Weiterleitung von Nachrichten beschäftigt. Im August 2000 brach das Gnutella-Netz sogar aufgrund dieser Überlastung völlig zusammen [35][36]. Die neueren Versionen des Gnutella-Protokolls haben dieses Problem insofern gelöst, indem sie Hierarchien einführten. Dadurch ist es aber nicht mehr ein reines dezentrales System.

Ein weiterer Nachteil ist dass es mitunter sehr lange Dauern kann bis eine Suche abgeschlossen ist, im Gegensatz zu den zentralisierten P2P-Systemen, bei denen der Server sofort die Antwort zurückschickt.

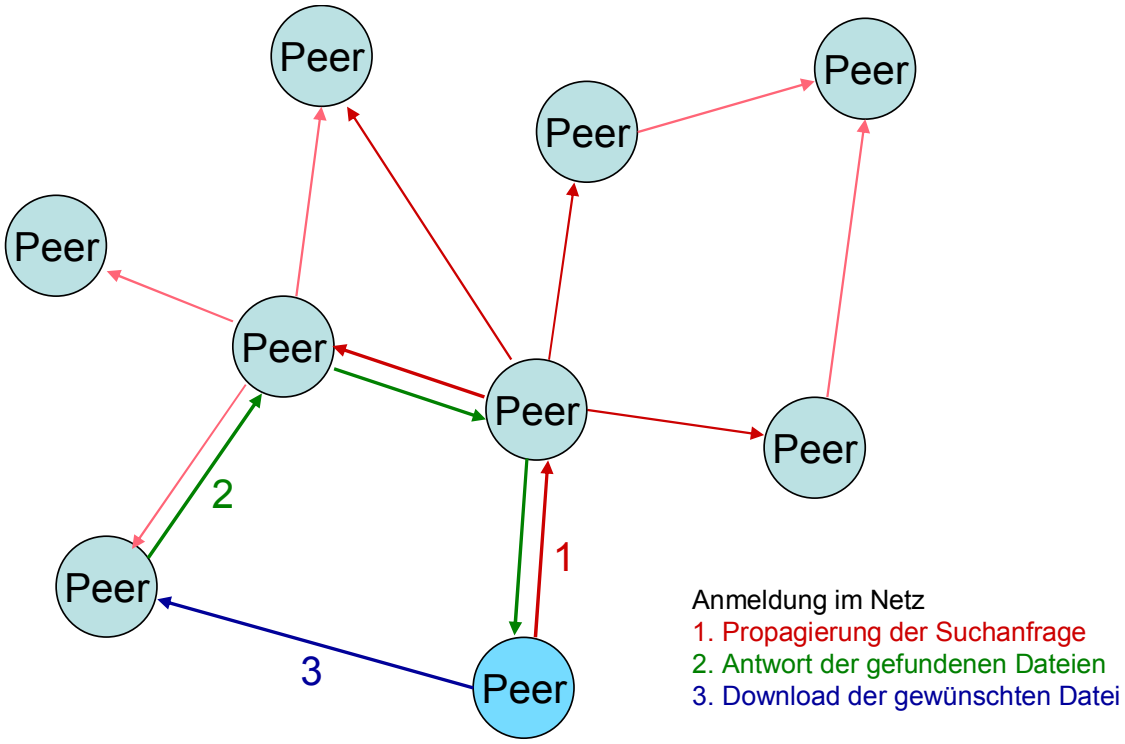


Abbildung 2: Suche in dezentralen Netzwerken

Vorteile :

- Kein zentraler Server nötig
- Große Robustheit beim Ausfall von Servents
- Keine Zensur

Nachteile :

- Hohe Netzlast
- Schlechte Skalierbarkeit
- Lange Suche

2.3.3 Hybride P2P-Systeme

Unter Hybriden P2P-Systemen versteht man Netze die weder einen zentralen Server besitzen noch dezentral aufgebaut sind mit gleichwertigen Peers. Solche Systeme verbinden meistens mehrere Topologien, so dass es Knoten gibt die verschiedene Rollen haben je nachdem welcher Topologie sie angehören. So kann ein Knoten eine zentrale Interaktion mit einem Teil des Netzes haben, gleichzeitig aber auch Teil einer anderen Hierarchie im Netz sein.

Es gibt übergeordnete Knoten, die ähnlich wie im Client/Server Modell mit mehreren Peers verbunden sind. Diese Super-Peers verwalten die Daten der Benutzer mit denen sie verbunden sind und bearbeiten deren Anfragen. Die Super-Peers sind wiederum untereinander in einer bestimmten Topologie verbunden. Um die Suche dennoch auf das gesamte Netz auszuweiten, müssen entweder die Suchanfragen an andere Super-Peers weitergeleitet werden, oder der Client muss mehrere Super-Peers kontaktieren.

Auf diese Weise versucht man die Vorteile von beiden Modellen miteinander zu verbinden. Der Single-point-of-failure und Flaschenhals den ein zentraler Server darstellt wird umgangen, dennoch skaliert das System sehr gut, da die Netzlast auf mehrere Server verteilt wird und bei Bedarf ist es immer möglich zusätzliche Server bereit zu stellen.

Die Zensur in einem Netz mit mehreren Servern ist schwieriger, weil man den Nachrichtenverkehr nicht von einer zentralen Instanz kontrollieren kann, die Erstellung von Benutzerprofilen ist aber dennoch möglich: Wenn man selber die Rolle einer Super-Peers einnehmen kann, kennt man die IP-Adressen der Nutzer, die sich bei einem einloggen. Da diese dann auch die Liste ihrer angebotenen Dateien und ihre Suchanfragen dann an den eigenen Super-Peer senden, kann man so zurückverfolgen wer was anbietet oder herunterlädt. Mit diesem Verfahren ist es aber nicht möglich das gesammte Netz zu erfassen, sondern man erhält nur die Informationen eines kleinen Anteils der Benutzer.

Heute benutzen die meisten aktuellen Peer-to-Peer Systemen hybride Netzwerke. Man kann unterschiedliche Arten von Topologien miteinander kombinieren, wie z.B. Ring-zentralisiert, dezentral-zentralisiert oder eine Hierarchie in Baumstruktur. [37]

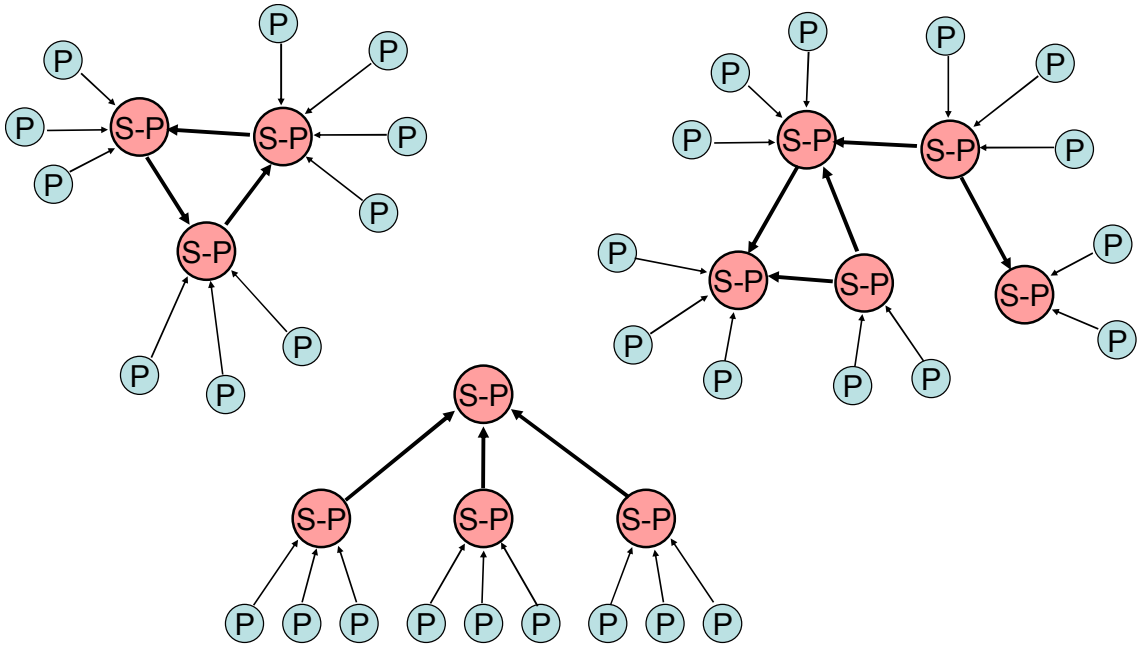


Abbildung 3: Hybride Peer-to-Peer Modelle

Vorteile:

- Gute Skalierbarkeit ohne Erhöhung der Netzlast
- Gute Robustheit beim Ausfall eines Super-Peers
- Keine Zensur

Nachteil:

- Erstellung von Benutzerprofilen möglich

2.4 Anwendungsbereiche von P2P-Systemen

Wenn man heute von Peer-to-Peer Systemen spricht meint man meistens Tauschbörsen wie Napster. Doch nach der Definition von P2P:

„…P2P is a class of applications that takes advantage of resources -- storage, cycles, content, human presence -- available at the edges of the Internet….“

beschränken sich die Ressourcen die von den Peers in Anspruch genommen werden nicht nur auf Dateien, File-Sharing Programme sind nur eine Facette der P2P-Systeme. [38][39]

Je nachdem welche Ressourcen die P2P-Anwendungen benutzen, kann man sie in folgende Anwendungsbereiche einteilen:

- File-Sharing
- Instant Messaging
- Collaboration / P2P-Groupware
- Grid Computing

2.4.1 File-Sharing

File-Sharing ist natürlich die bekannteste und auch eine der meist genutzten Anwendungsmöglichkeit von P2P-Systemen. Die Applikationen ermöglichen eine Suche nach Dateien, die die Ressourcen am Rande des Netzes bilden und so einen dezentralen verteilten Speicher bilden. Anschließend können die Dateien über eine direkte Verbindung zu einem anderen Peer sofort heruntergeladen werden.
Diese Technik bietet somit die Möglichkeit z.B. im betrieblichen Umfeld einen zentralen kostenintensiven Datenserver zu ersetzen, indem die Daten auf mehrere Rechner verteilt werden, die freien Speicherplatz zur Verfügung haben. Dadurch wird auch der Single-point-of-failure, den ein Zentraler Server darstellt, eliminiert.
Im kommerziellen Umfeld werden meistens Peer-to-Peer Programme mit Adware oder Spyware finanziert. Bei der Adware blendet das Programm während der Benutzung Werbebanner ein. Spyware wird meistens vom Benutzer unbemerkt installiert und spioniert das Surfverhalten der Benutzer aus, um dadurch Werbung einblenden zu können, die den Interessen der Benutzer besser entsprechen oder dessen Benutzerprofile weiterverkaufen.

2.4.2 Instant Messaging

Ein anderer, sehr verbreiteter Anwebdungsbereich von P2P-Systemen ist Instant Messaging. Man versteht darunter Programme die einen direkten und schnellen Nachrichtenaustausch unter mehreren Benutzer ermöglichen.
Der Hauptvorteil von Instant Messaging gegenüber einem System wie Email ist, dass man gleich sieht ob der Kommunikationspartner gerade online ist oder nicht. Die Mitglieder eines IM-Systems benutzen einen Client um sich in einem Netz einzuloggen,

ähnlich wie auch bei den File-Sharing Systemen. Ist der gewünschte Kommunikationspartner im gleichen IM-Netz angemeldet, kann er sich direkt mit diesem verbinden und ihm ohne Umweg über ein anderes Vermittlungssystem eine Nachricht senden. Die meisten IM-Programme ermöglichen aber auch das versenden von ganzen Dateien, sind also eine Art Kombination aus dezentralen Chatten und File-Sharing. Andererseits sind in den meisten File-Sharing Programmen auch IM-Funktionalitäten integriert.

Beispiele für IM-Programme sind: ICQ [40], AIM (AOL Instant Messenger) [41], MSN Messenger [42], Jabber [43], Yahoo Messenger [44]. Diese basieren meistens auf dem Modell mit zentralem Server.

Eine Weiterentwicklung des Instant Messaging mittels Text-Nachrichten ist das Telefonieren über Internet, das nach ähnlichem Prinzip funktioniert wie z.B. das Programm Skype [45] von den Kazaa-Entwicklern.

2.4.3 Collaboration / P2P-Groupware

Unter Groupware versteht man Programme der Verwaltung von Arbeits- bzw. Personengruppen dienen. Diese Software wird eingesetzt zur Kommunikation, Kooperation und Koordination von Mitarbeitern eines Unternehmens, die an einem gemeinsamen Projekte arbeiten. Typischerweise bieten solche Programme den Nachrichtenaustausch mit anderen Mitarbeitern, das gemeinsame Bearbeiten von Dateien und einen gemeinsamen Terminkalender. Von dem Hersteller des erfolgreichsten Groupware-Programms Lotus-Notes, wurde auch die P2P-Lösung Groove Networks [46] entwickelt.

Groove kombiniert Kommunikationstechnologien wie Voice-Chat, Instant-Messaging und Video-Konferenzen mit File-Sharing. Weitere Funktionen sind das gemeinsame Editieren von Dateien, Ansehen von Presentationen und Terminplanung. Die Benutzer können für jedes Projekt einen gemeinsamen Raum anlegen, in dem die Mitarbeiter die gemeinsamen Dateien ablegen, dieser Raum wird auf den Rechner aller Mitarbeiter verschlüsselt gespeichert. Wird nun eine Datei verändert kümmert sich Groove um die synchronisation dieser verteilten Räume und überträgt die Updates an alle Nutzer, die so über die Grenzen der Büros oder Firmen hinweg zusammenarbeiten können. [47][48]

2.4.4 Grid Computing

Die Idee beim Grid Computing oder Distributed Computing ist es, die ungenutzten Rechenkapazitäten der Desktop PCs zu verwenden um an einem gemeinsamen Projekt zu arbeiten. In der alltäglichen Benutzung sind die meisten Desktop PCs völlig unterfordert. Beim Schreiben von Texten oder beim Surfen im Internet wird nur ein Bruchteil der vorhandenen Rechenkapazität verwendet. Lässt man nun diese Rechner gemeinsam an einem Projekt arbeiten summiert sich eine Rechenleistung, die man sonst nur mit teueren Supercomputern zur Verfügung hätte. Somit eignet sich Grid Computing für Projekte denen die finanziellen Mittel für die benötigten

Rechenkapazitäten fehlen. Der Benutzer wird zum „Spender“ von Taktzyklen. Häufig werden die Clients als eine Art Bildschirmschoner implementiert, oder als Programm mit der niedrigsten Priorität, so dass es nur dann rechnet, wenn der Benutzer keine CPU-lastigen Anwendungen betreibt.

Vorreiter der Grid Computing Applikationen ist Seti@Home [49]. SETI steht für „Search for Extraterrestial Intelligence“ und ist ein Projekt der Universität Berkley das sich mit der Suche nach außerirdischer Intelligenz im Universum beschäftigt. Mit einem Teleskop in Puerto Rico werden rund 35 Gigabyte Daten pro Tag aufgezeichnet die auf mögliche Signale in den Radiofrequenzen untersucht werden. Um die Daten zu analysieren benötigt man Grosscomputer, da das SETI-Programm nicht über die nötigen finanziellen Mittel verfügt um sich solche Computer anzuschaffen fand man eine Alternative: Wer sich an dem Projekt beteiligen will installiert einen Client, der den Bildschirmschoner ersetzt. Die gesammelten Daten werden in 0,25 Megabyte große Pakete aufgeteilt, sogenannte „work-units“, diese werden dann vom SETI@home-Server über das Internet an Menschen in aller Welt zur Analyse gesandt, vom Client offline analysiert und die eruierten Resultate werden danach wieder zurücksendet. SETI@home baut nur dann eine Internet-Verbindung auf, wenn Daten transferiert werden müssen, also sobald der Bildschirmschoner die Analyse einer work-unit abgeschlossen hat und er die Resultate zurücksendet, sowie ein neues Paket anfordern möchte.

Mit über 4,7 Millionen Benutzern, die insgesamt mehr als 1 Milliarde Pakete analysiert haben, verfügt das System über eine Rechenleistung von 57.64 TeraFLOPs/sec. Zum Vergleich: der derzeit schnellste Supercomputer, Earth-Simulator hat 35,86 TFLOPs.

Andere Beispiele für Grid-Computing sind

- GIMPS - die Suche nach neuen Primzahlen [50]
- FightAIDS@home - die Suche nach neuen Medikamenten gegen AIDS [51]
- Internet Movie Project - Projekt das Ressourcen aus dem Internet benutzt um einen Film aus Computeranimationen mittels POV-Ray zu rendern. [52]
- Money Bee - Ein Bilschirmschoner der die Aktienkurse analysiert und dann zukünftige Trends vorausberechnet. [53]
- DALiWorld - Ein virtuelles Salzwasseraquarium. Wenn man mit dem Internet verbunden ist werden einige der Fische in die Aquarien anderer Nutzer auswandern, später kann man sehen wo der Fisch schon überall war und wer ihn erschaffen hat (nur zur Unterhaltung). [54]

3. Beispiele aktueller File-Sharing P2P Netze

3.1 Das FastTrack Netzwerk

KaZaa [2] ist heute die populärste aller Tauschbörsen. Beim Download-Dienst Download.com ist der Kazaa-Media-Desktop das am häufigsten heruntergeladene Programm, mit über 2 Millionen Downloads pro Woche. Insgesamt wurde das Programm schon über 300 Millionen mal nur von diesem Dienst heruntergeladen [55]. Ein anderer Client des FastTrack Netzwerks iMesh [4] wurde ebenfalls mehr als 60 Millionen Mal heruntergeladen.

Jederzeit sind mehr als 3,5 Millionen Benutzer im FastTrack Netz miteinander verbunden und tauschen mehr als 600 Millionen Dateien, was insgesamt einem Datenvolumen von mehr als 5 Petabyte entspricht.

Das FastTrack Netzwerk wurde von dem Schweden Niklas Zennstrom entworfen und zusammen mit dem ersten Client KaZaA von der holländischen Firma Kazaa BV im März 2001 implementiert. Die Schließung von Napster, hatte zur Folge, dass MusicCity, das bisher auf OpenNap Server basierte, das Protokoll wechselte und ihr Client Morpheus von nun an ebenfalls auf der Technologie von FastTrack basierte. Somit erbte das FastTrack Netz von Anfang an eine große Benutzerzahl. Im November 2001 verloren die Betreiber von Kazaa eine Gerichtsverhandlung in Holland und übergaben die Rechte des FastTracks an die Firma Sharman Networks [1], die ihren Geschäftssitz in der Inselnation Vanuatu hat. [59]

Im Februar 2002 wurde Morpheus durch eine Änderung des Protokolls aus dem Netz ausgeschlossen, weil die Lizenzgebühren für die Benutzung des Netzwerks nicht bezahlt wurden. Später wurde auch der open-source Client GiFT [6] durch eine Serie von Protokolländerungen ausgeschlossen. Dies war möglich, weil die Protokollspezifikationen nicht offen liegen, und die offiziellen Clients so programmiert sind, dass sie automatisch Updates installieren.

Heute gibt es mit KaZaA, iMesh und Grokster [5] drei Programme, die auf dem FastTrack Netzwerk aufbauen. Es gibt nur Versionen für Windows, und da es sich um ein proprietäres Netz handelt, versuchen alle Anbieter, durch Adware oder Spyware Geld zu verdienen. Im April 2002 begann der Kazaa Client auch den Altnet Service zu verwenden um Medien zu verteilen, die durch Digital Rights Management (DRM) geschützt wurden und erst nach einer Bezahlung freigeschaltet werden [56]. Kurz darauf wurde der Client gehackt und Kazaa Lite [3] entstand, das von Ad- und Spyware befreit war, sowie einige zusätzliche Funktionalitäten bietet.

Die Zukunft von FastTrack ist noch ungewiss. Ein Gericht entschied zwar dass man Grokster nicht verantwortlich machen könne für die Copyright-Verstöße der Benutzer, weil das FastTrack Netz im gegensatz zu Napster keinen zentralen Server besitzt der alle Dateien verwaltet. 2002 wurde aber gegen Sharman in den USA Klage eingereicht, im September 2003 auch gegen iMesh. [57][58]

3.1.1 Struktur des Netzes

Das FastTrack Netz beruht auf einem hybriden Netzwerk Modell wie in 2.3.3 vorgestellt. Das Netz besteht aus mehreren Servern hier Supernodes genannt, die untereinander dezentral verbunden sind. Die Benutzer verbinden sich mit einem dieser Supernodes. Die Supernodes indizieren alle Dateien die die Peers, die mit ihnen verbunden sind, anbieten.

Eine der Besonderheiten dieses Netzes ist die seine Selbstorganisation. Die Zahl der Supernodes kann sich dynamisch ändern, bei Bedarf kann jeder Peer die Rolle eines Supernodes einnehmen, die Serverfunktionalität der Supernodes sind schon in jedem Client mit integriert. Dabei bestimmt das Programm selbst ob die Rechenleistung und die Bandbreite der Maschine ausreichen um die Funktionalität eines Supernodes zu übernehmen. Die Anzahl der Supernodes steht in Relation 1:100 zur Anzahl der Peers.

Ein Vorteil dieses Modells ist, dass es sehr einfach skaliert. Ohne Überwachung der Betreiber richtet das Netz neue Supernodes ein, gleichzeitig wird aber auch berücksichtigt, dass nicht jeder Peer den Anforderungen gerecht werden und verhindert dass weder die Supernodes noch die Peers überlastet werden. Das daraus resultierende Netz bleibt also robust beim Ausfall oder Abmelden eines Supernodes, da es viele alternative Supernodes gibt.

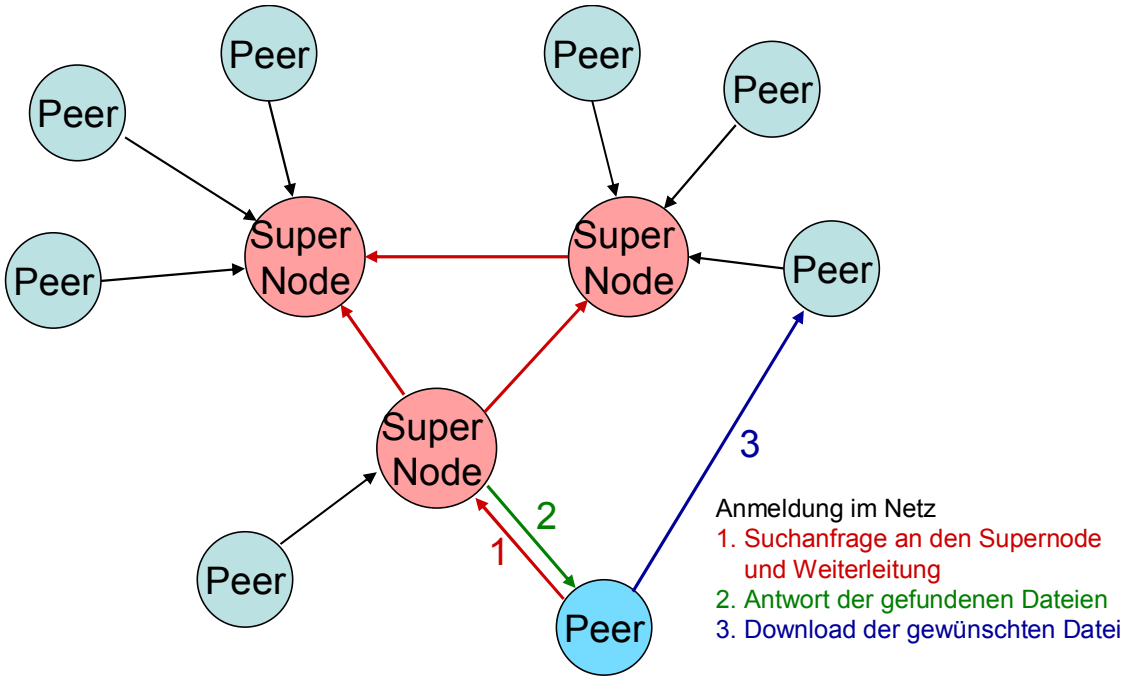


Abbildung 4: Struktur des FastTrack Netzwerks

Login:

Um sich das erste Mal mit dem FastTrack Netz zu verbinden, hat der Client eine Liste von Supernodes mit fester IP Adresse integriert. Er versucht eine Verbindung zu einem dieser Supernodes aufzubauen und fordert anschließend eine Liste der Adressen aktueller Supernodes an, zu denen er sich in Zukunft verbinden kann. Der Client ist jeweils nur mit einem Supernode verbunden, zu dem sendet er nach dem Verbindungsaufbau eine Liste mit den Dateien die er zum Upload anbietet. Alle Suchanfragen werden ebenfalls von diesem Supernode behandelt.

Suche:

Zur Suche sendet der Client eine Suchanfrage an den Server mit dem er verbunden ist, dieser erstellt aus seinem Index eine Liste mit den gefundenen Dateien und sendet diese zurück. Um das Suchergebnis noch weiter zu steigern, können die Suchanfragen an andere Supernodes weitergeleitet werden, die dann wiederum eine Liste der bei ihnen gefundenen Ergebnisse zurückliefert.

Im Gegensatz zum Gnutella Netzwerk, wo jeder Peer sowohl Client als auch Server ist und Suchanfragen über das gesamte Netz geflutet wurden, was gerade Rechner mit einer 56k Modemanbindung völlig überlastete, bleibt beim FastTrack Netz eine gewisse Hierarchie enthalten. Die Suchanfragen werden nur unter den Supernodes geflutet, somit bleibt die Netzlast auch bei wachsenden Benutzerzahlen gering. Steigt die Netzlast dennoch an, können dynamisch neue Supernodes gebildet werden. Dabei werden nur die Clients zum Supernode, deren Rechenkapazität und Internetzugang den höheren Anforderungen gewachsen ist.

Download:

Zum Herunterladen der Datei kontaktiert der Suchende dann einen der Clients aus der Liste der Anbieter direkt.

3.1.2 Funktionalität

Alle FastTrack Clients haben ein ähnliches Layout: Das GUI ist in mehrere Fenster unterteilt, die die verschiedenen Funktionen anbieten. Im 'Web'-Fenster ist ein Browser untergebracht. In 'My Kazaa' werden alle Dateien, die zum Tauschen freigegeben sind übersichtlich in einer Baumstruktur nach ihrem Typ klassifiziert. Ein eingebauter Media-Player, mit dem man heruntergeladene Audio oder Video Dateien wiedergeben kann, befindet sich im 'Theater'-Fenster. Im 'Search'-Fenster kann man nach Dateien suchen und die Suchergebnisse in einer Liste betrachten. Das 'Traffic'-Fenster schliesslich informiert den Benutzer über den Fortschritt der Down- und Uploads.

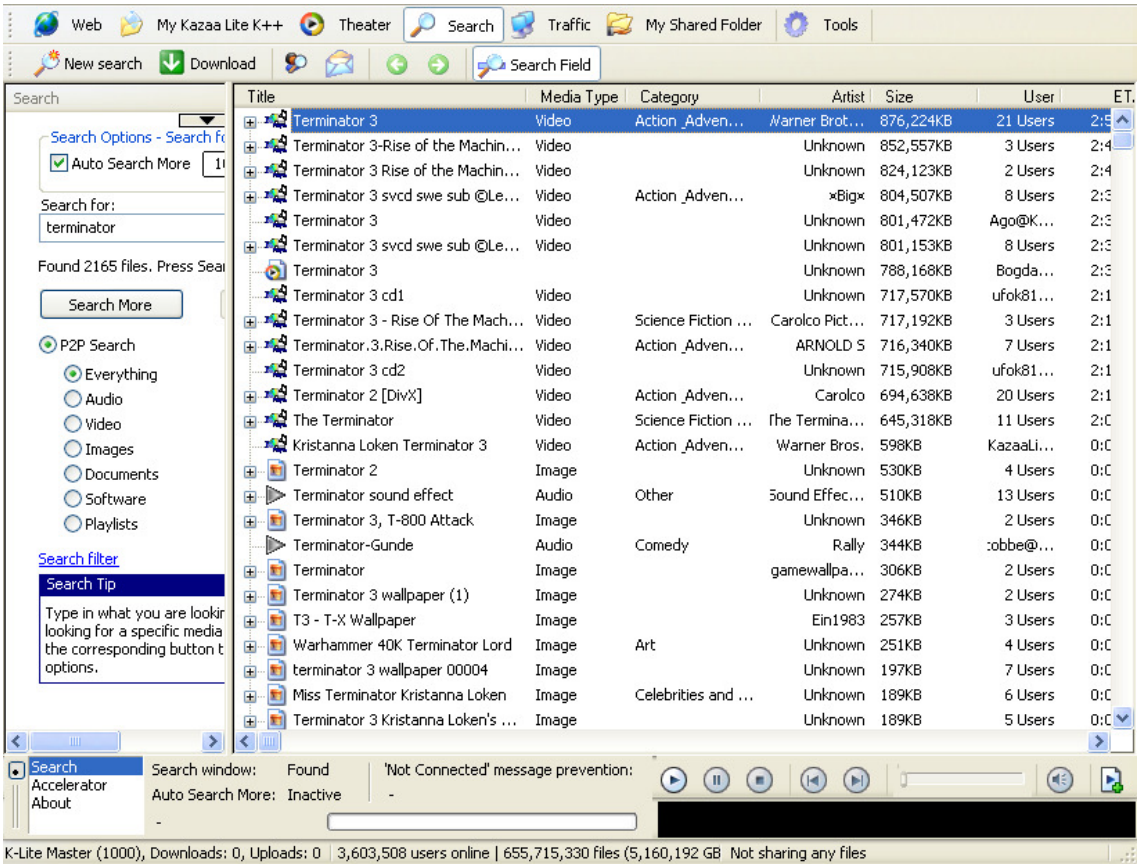


Abbildung 5: Suche in KaZaA Lite K++

Die Suche ist sehr komfortabel, da die Supernodes viele Metadaten über die angebotenen Dateien speichern, kann man sehr präzise Suchanfragen stellen. Bei der Suche nach Audio- oder Video-Dateien kann gezielt nach dem Titel, Autor, Sprache, Bitrate, Auflösung, Kategorie oder Größe gesucht werden. Die Benutzer können die von ihnen angebotenen Dateien selbst mit diversen Metadaten versehen, wie z.B. die Musikrichtung in der Kategorie angeben. Die Rechteinhaber von Musikstücken haben

versucht die Suche nach bestimmten Liedern zu sabotieren, indem sie eine Fülle von Dateien anboten, die nicht den gesuchten Inhalt boten, oder Störungen oder Werbung enthielten. Daraufhin wurde das 'Integrity Rating' Tag eingeführt, mit dem die Benutzer die Qualität der angebotenen Dateien beschreiben können.

Das Suchergebnis wird in einer Liste aufgeführt, die die Metadaten anzeiget, sowie die Bandbreite des Anbieters und die geschätzte Dauer des Downloads. Werden Dateien von mehreren Benutzer angeboten, werden diese in einer Baumstruktur zusammengeführt. Man kann ebenfalls nach allen Dateien suchen, die ein bestimmter Benutzer anbietet.

Beim Download einer Datei versucht der Client diese von mehreren Benutzern gleichzeitig herunterzuladen, und somit die Downloadgeschwindigkeit zu steigern. Hat ein Anbieter seine maximale Anzahl von Uploads bereits ausgeschöpft landet man in dessen Warteschlange. In regelmäßigen Abständen sucht der Client nach neuen Anbietern, man kann dies aber auch manuell einleiten. Die maximale Anzahl der gleichzeitigen Down- und Uploads kann in den Einstellungen des Clients festgelegt werden.

Die Dateien sind mit einem Hash-Wert versehen, der eine Datei unabhängig von ihrem Dateinamen eindeutig identifiziert, somit kann man auch unterbrochene Downloads zu einem späteren Zeitpunkt wieder aufnehmen.

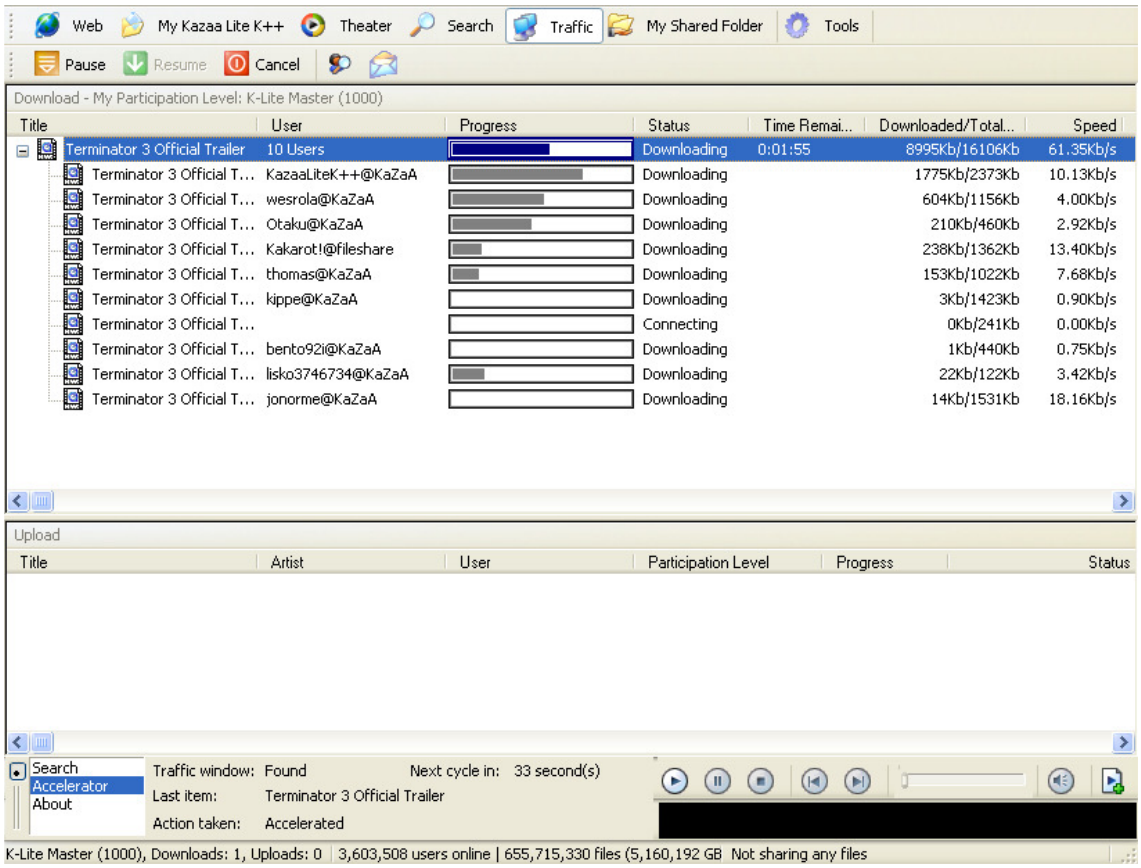


Abbildung 6: Download in KaZaA Lite K++

Um die Benutzer von Kazaa zu belohnen, die viele Dateien anbieten, wurde vor kurzem ein 'Participation Level' eingeführt. Je mehr Dateien ein Benutzer hochlädt, desto größer sein Participation Level. Benutzer mit einem hohen Wert, werden in einer Warteschlange bevorzugt behandelt.

Kazaa Lite K++ umgeht einige Limitierungen vom originalen Kazaa Media Desktop und bietet zusätzliche Funktionen. Bei Kazaa werden viele Adware und Spyware Programme mitinstalliert, wie New.net, Bonzi Buddy, SaveNow, b3e projector, Cydoor and Altnet, die Systemressourcen oder Bandbreite beanspruchen. K++ ist frei von solchen Programmen oder Werbebanner. K++ blockiert die Altnet Angebote, kommerzielle Suchergebnisse, die durch Digital Rights Management geschützt sind.

Bei K++ ist der Participation Level immer fest auf den Maximum (1000 Supreme Being) gesetzt und macht das System damit komplett nutzlos. Während man bei Kazaa eine Datei nur von maximal 8 Anbietern gleichzeitig herunterladen kann, ist dieser Wert bei K++ auf 40 erhöht worden. Eine Limitierung von Kazaa die das tauschen von Musikdateien mit einer Bitrate höher als 128kit/s verhindert wurde aufgehoben. Man kann die Suche nach zusätzlichen Quellen, die in Kazaa eingeschränkt ist besser automatisieren.

Die Suche bei Kazaa erfolg normalerweise innerhalb von ein paar Sekunden, doch da die Suchergebnisse von normalen PC's anderer Benutzer generiert werden, die die Rolle eines Supernodes eingenommen haben und nicht von dedizierten Servern, kann es vorkommen, dass die Suche länger dauert, wenn z.B. der Rechner wenig Ressourcen frei hat, oder aber nur wenige Treffer zurückgesandt werden, weil der Supernode nur mit wenigen Peers verbunden ist. In dem Fall bietet K++ die Möglichkeit auf einen anderen Supernode zu wechseln oder die Suche während einer längeren Zeit fortzusetzen. In K++ kann man auch verhindern selbst zum Supernode zu werden, um den Rechner oder die Bandbreite nicht zusätzlich zu belasten, oder aber manuell die Rolle eines Supernodes einnehmen, was besonders die Suche nach Dateien stark verkürzt, da der eigene Rechner in dem Fall einen Index der angebotenen Dateien aufbaut.

3.2 Das eDonkey Netzwerk

Das eDonkey Netzwerk ist eines der meist benutzten Netze zum Tauschen von Dateien. Bei dem open-source Interportal SourceForge.net steht der Client eMule an erster Stelle der populärsten Dateien, mit über 22 Millionen Downloads [60].

Im Netz sind über 1,5 Millionen Benutzer miteinander verbunden, die mehr als 100 Millionen Dateien tauschen.

Der offizielle Client eDonkey2000 [8] wurde von MetaMachine im September 2000 entworfen, und benutzt das Multisource File Transfer Protocol (MFTP). Das eDonkey Netz besteht aus mehreren zentralen Servern mit denen die Peers verbunden sind. Bei einer großen Benutzerzahl stellen diese Server somit einen Flaschenhals dar. MetaMachine hat daraufhin Overnet [9] entworfen, ein komplett dezentrales Netz, das

ohne zentrale Server auskommt. Doch es war von Anfang an schwierig die Benutzer zu überzeugen auf dieses neue System umzusteigen, zu groß war die Benutzerzahl und das Dateiangebot vom eDonkey-Netz im Vergleich zu Overnet. MetaMachine hatte die Weiterentwicklung des eDonkey2000 Clients eingestellt, doch ein neuer open-source Client, eMule [10] wurde immer populärer. eMule benutzt ebenfalls das MFTP Protokoll und hat es noch um zusätzliche Funktionalität erweitert. Es kann Dateien mit den Benutzern von eDonkey2000 tauschen und ist übersichtlicher und einfacher zu bedienen.

Heute gibt es eine Vielzahl von Clients, meist modifizierten Versionen von eMule, die sich mit dem eDonkey Netz verbinden können. [12][11] Einige Clients können außerdem im Overnet nach Dateien suchen und tauschen.

3.2.1 Struktur des Netzes

Im eDonkey Netz wird zwischen Server und Clients unterschieden, die auch unterschiedliche Programme benutzen. Jeder Client ist mit einem Server verbunden. Die Server akzeptieren nur eine gewisse Benutzerzahl, abhängig von ihrer Rechenkapazität und Internetverbindung. Die meisten eDonkey Server basieren heute auf Lugdunum [15], einer open-source Implementierung des eDonkey/eMule Servers für Linux.

Kleine Server können nur einige Hunderte Benutzer verwalten, die größten verbinden aber über 200.000 Benutzer. Dies sind spezielle Rechner mit viel Speicher, die über einen Internetzugang mit hoher Bandbreite verfügen müssen. Razorback [16], einer der derzeitigen größten Server verwaltet mehr als 300.000 Benutzer und verfügt über zwei Athlon MP 2400+ Prozessoren mit 3,25 Gigabyte Speicher und ist mit 100 Mbps an das Internet gebunden. Der Nachfolger Razorback 2 ist schon in Planung und soll über 2 64-Bit Opteron Prozessoren mit insgesamt 8 Gigabyte Speicher verfügen. Razorback wird derzeit ausschließlich durch Spenden und Werbebanner auf der Homepage finanziert.

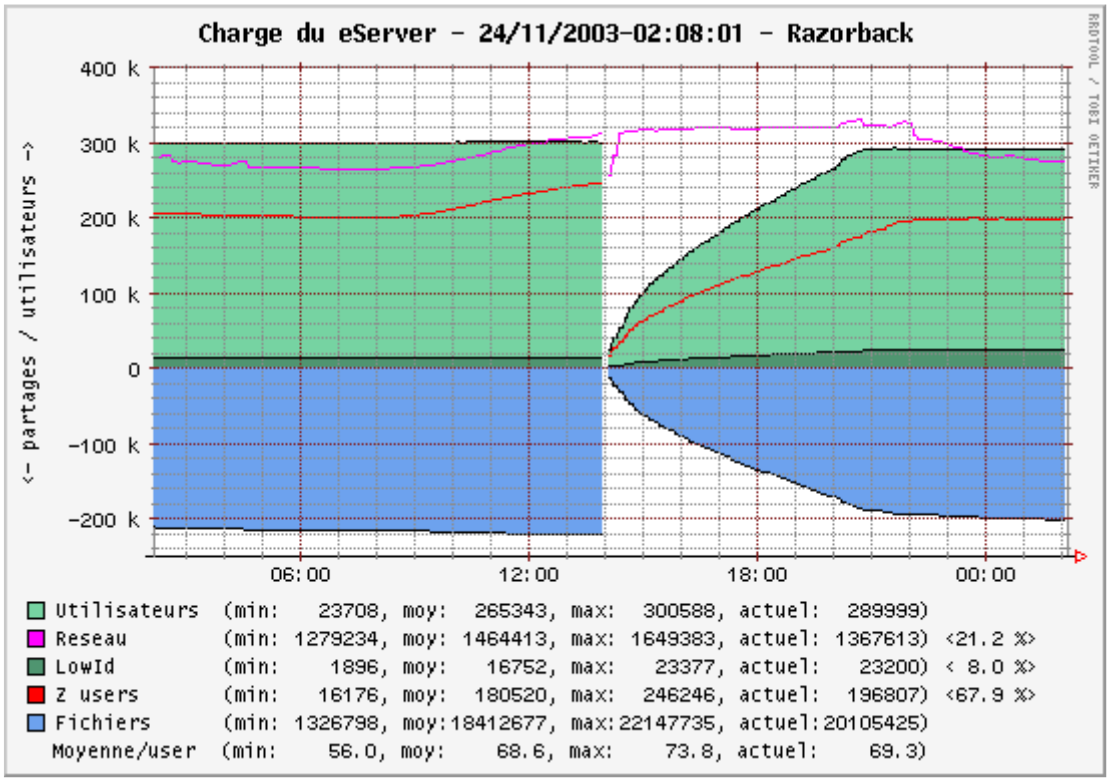


Abbildung 7: Ausfall des eDonkey Servers Razorback [61]

Fällt einer der größeren Server aus, sind gleich Tausende von Clients nicht mehr mit dem eDonkey Netz verbunden. Bei Razorback ist dies besonders kritisch, weil er fast ein Fünftel der Benutzer verwaltet und diese dann alle gleichzeitig versuchen sich bei einem anderen Server anzumelden, die durch die vielen Login Meldungen überlastet werden können. Abbildung 7 listet die Statistiken des Razorback Servers nach einem Ausfall auf. Interessant ist, dass es fast 30 Minuten dauert, bis die 300.000 Benutzer wieder mit dem Netz verbunden waren. In Grün ist die Zahl der mit dem Server verbundenen Benutzer (Utilisateurs), in Blau die Summe deren angebotenen Dateien (Fichiers) und in Violett die gesamte Anzahl aller Nutzer im eDonkey Netz (Réseau).

Die Benutzer die vom Netz getrennt sind können trotzdem weiter Herunterladen und behalten auch ihre Position in den Warteschlangen anderer Anbieter, zum Suchen nach Dateien müssen Sie sich aber wieder bei einem Server anmelden.

Auf der Seite TheDonkeyNetwork [17] kann man eine aktuelle Serverliste von ocbMaurice [18] finden.

Die Server sind wie auch bei Napster für die Verwaltung der Benutzer zuständig und bauen einen Index aller angebotenen Dateien auf.

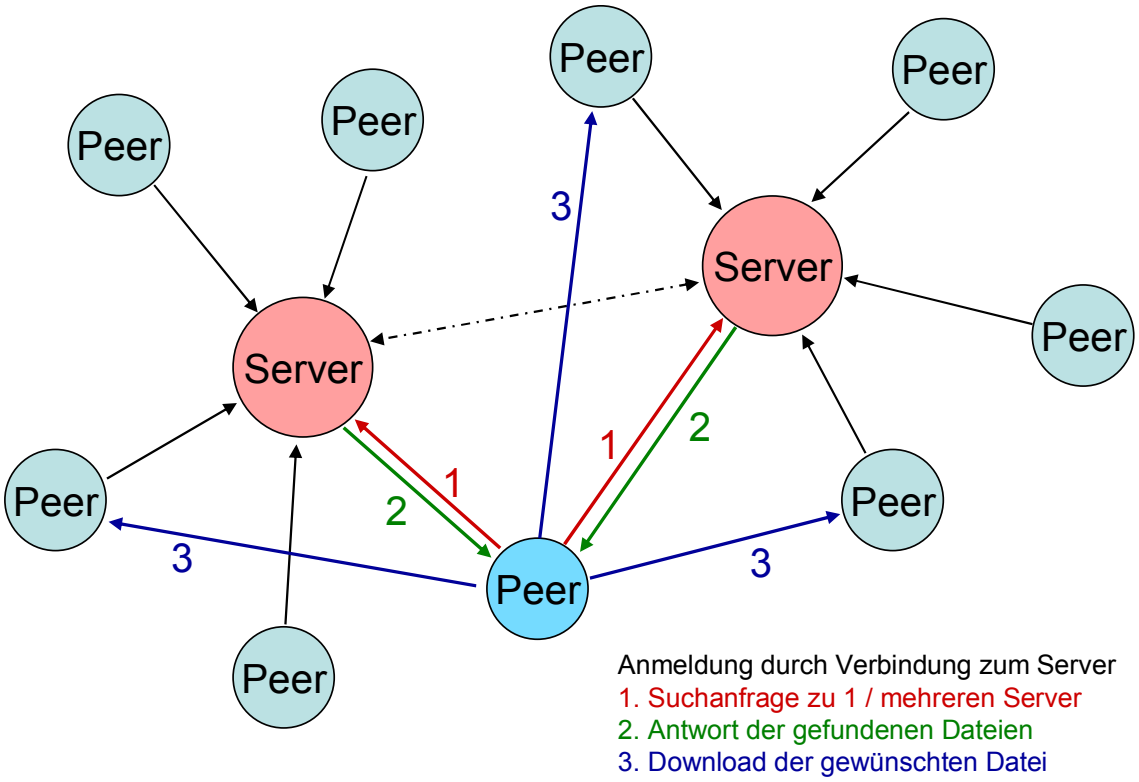


Abbildung 8: Struktur des eDonkey Netzwerks

Login:

Bei der ersten Verbindung zu einem Server ist der Client auf eine Server-Liste angewiesen, die ihm die IP-Adresse eines Servers liefert. Diese kann im Programm implementiert sein oder man liefert dem Client eine aktualisierte Server-Liste die man auf diversen Seiten im Internet herunterladen kann. Der Client versucht anschließend sich mit einem Server zu verbinden. Ist die Verbindung aufgebaut, sendet der Server eine aktuelle Server-Liste an den Client zurück. Der Client wiederum sendet dem Server eine Liste mit den Dateien, die er zum tauschen anbietet. Die Server kommunizieren untereinander nur, um die Server-Liste zu aktualisieren.

Suche:

Bei der Suche hat man die Möglichkeit nur bei dem Server zu suchen, zu dem man verbunden ist, oder eine globale Suche über das gesamte eDonkey-Netz zu starten. Anders als beim FastTrack Netz verarbeiten die Server nur die Suchanfragen und senden das Ergebnis zurück, sind aber nicht für das Weiterleiten der Suchanfragen zuständig. Bei der globalen Suche, sendet der Client die Suchanfrage einfach an mehrere Server aus der Liste.

Download:

Zum Download der gewünschten Datei sendet der Client eine Anfrage an mehrere Clients, die ihn anschließend in eine Warteschlange setzen oder ihm ein Teil der Datei senden.

3.2.2 Funktionalität

Das eDonkey Netz ist auf das Tauschen von großen Dateien mit mehreren Megabytes bis hin zu mit einer Größe von 1-5 Gigabytes spezialisiert, vom Anbieten vieler kleiner Dateien wird sogar abgeraten, da dies zu einer Belastung der Server führt. Es werden viele unterschiedliche Arten von Dateien getauscht, ganze Alben von Musikgruppen, raubkopierte CD-Images von PC-Spielen und Software, Filme in komprimierter Form sowie ganze DVD-Kopien.

Zur eindeutigen Kennung der Dateien wird ein Hashwert errechnet. Große Dateien werden von den Clients in Teile von 9500 KByte fragmentiert. Der Client kann mehrere Fragmente anfordern, aber auch ein einzelnes Fragment gleichzeitig von mehreren Clients herunterladen und somit die Downloadgeschwindigkeit steigern. Die Fragmente werden wiederum durch einen MD4 Hashwert auf ihre Authentizität überprüft, wobei MD4 wahrscheinlich einem sichereren Algorithmus wie MD5 aufgrund seiner schnelleren Berechnung vorgezogen wurde.

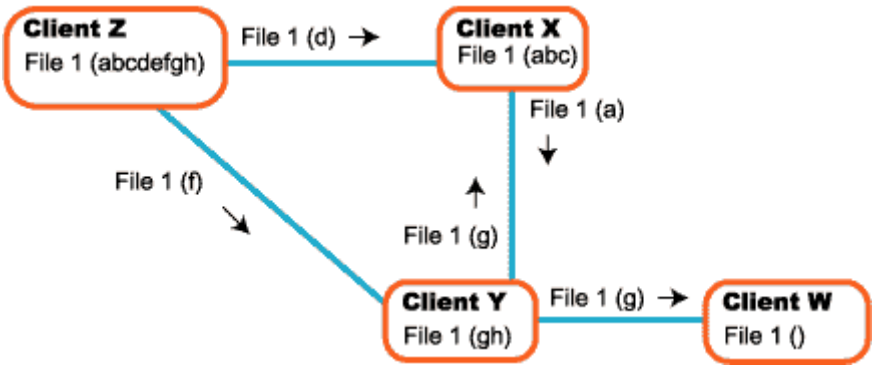


Abbildung 9: Download in eDonkey mit MFTP [62]

Eine andere Besonderheit der eDonkey-Clients ist, dass mit dem Upload einer Datei bereits begonnen werden kann, sobald ein Fragment vollständig vorliegt. Durch dieses Verfahren wird sichergestellt, dass sich auch große Dateien schnell im Netz verbreiten, und selbst wenn kein einziger Anbieter die Datei vollständig hat, kann sie doch erfolgreich heruntergeladen werden, sofern alle Fragmente einzeln noch zu finden sind. Ein Nachteil dieser Methode ist, wenn nicht mehr alle Fragmente einer Datei getauscht werden, z.B. weil der einzige Anbieter nicht mehr online ist. In dem Fall werden die einzelnen Fragmente weiter im Netz verbreitet, ohne dass eine Chance besteht die Datei vollständig zusammenzusetzen. [62]

In Abbildung 9 hat Client Z alle Fragmente der Datei 1 und W, X, Y wollen diese Datei herunterladen. Da X und Y schon unterschiedliche Fragmente von Datei 1 heruntergeladen haben können sie diese schon untereinander tauschen. Client W kann ebenfalls mit dem Download beginnen, selbst wenn die Bandbreite von Z ausgelastet ist, und lädt ein Fragment von Y herunter.

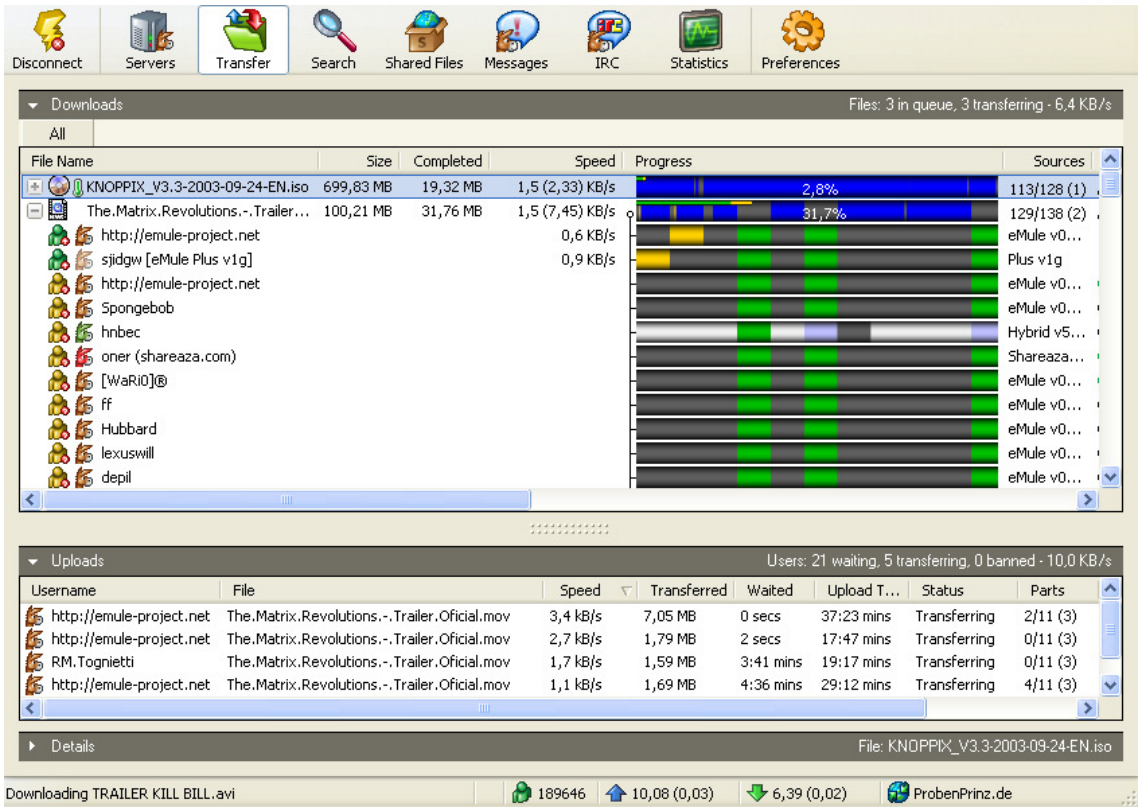


Abbildung 10: Download mit eMule Plus

Einige Clients benutzen außerdem intelligente Verfahren um die Verbreitung von Dateien zu verbessern und zu verhindern dass einige Fragmente sehr oft heruntergeladen werden, während andere sich nur sehr langsam verbreiten. Mit Jumpstart werden nur Fragmente angezeigt, die nur wenige Male heruntergeladen wurden, somit wird sichergestellt dass sich, gerade beim Anbieten neuer Dateien, diese uniform im Netz verbreiten.

Andererseits benutzen einige Clients auch intelligente Verfahren beim Download von Dateien. So werden mit Intelligent Chunk Request zuerst die Fragmente heruntergeladen, die die wenigsten Quellen anbieten.

Da eine Tauschbörse nur funktionieren kann, wenn viele Benutzer Dateien uploaden und nicht nur Dateien herunterladen ohne etwas zurück zu geben, ist in den Clients eine Beschränkung der Download-Geschwindigkeit eingebaut. Stellt der Benutzer weniger als 10kB/s für den Upload zur Verfügung, ist die Upload/Download Ratio auf 1:4 festgelegt, d.h. ein Upload-Limit von 5kB/s hat einen maximalen Download von 20kB/s zur Folge.

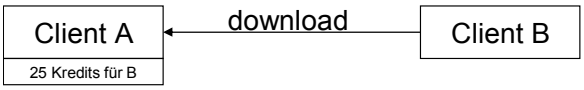


Abbildung 11: Kreditsystem

Um die Benutzer, die viele Dateien uploaden zu belohnen, benutzen die Clients außerdem ein Kredit-System. Die Kredits gelten nur zwischen zwei Benutzer: Wenn

Client A eine Menge von Daten von Client B herunterlädt, rechnet dafür Client A eine gewisse Anzahl von Kredits Client B zu. Wenn Client B später Daten von Client A herunterladen will, wird er bevorzugt behandelt und landet je nach Anzahl von Kredits in einer höheren Position in der Download-Warteschlange. Die Kredits werden in einer Datei bei dem „Schuldner“ gespeichert und können daher nicht manipuliert werden. [63]

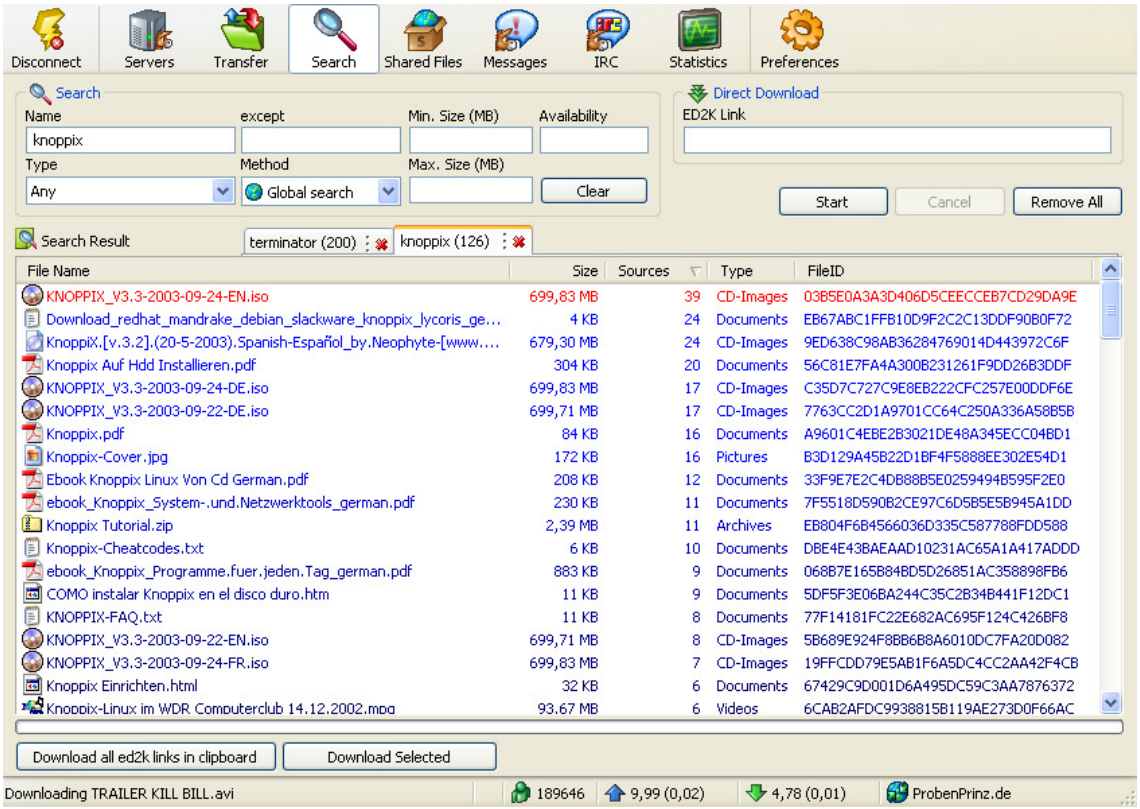


Abbildung 12: Suche in eMule Plus

Die Suche im eDonkey-Netz ist nicht die einzige Möglichkeit Dateien zu finden. Durch spezielle 'ed2k'-Links in html-Seiten, die den Namen, die Größe und den eindeutigen Hashwert beinhalten, ist es möglich aus dem Browser Dateien in die Downloadliste des Clients aufzunehmen.

```
ed2k://|file|Something.avi|126863828|7f2b985df22661f66449ee72d81b6daa|
```

Mit der neusten Version von KaZaA 2.6 unterstützt erstmals auch ein FastTrack Client mit Magnet Link direkte Links zu Dateien aus html-Seiten.

Die große Popularität von eDonkey ist darauf zurückzuführen, dass viele Webseiten diese Links auf ihre Echtheit untersuchen und kategorisieren. Die Seite ShareReactor [64] z.B. bietet Links über 2000 Dateien mit einem Gesamtvolumen von 1.74 Terabyte an, sortiert nach Filmen, Spielen, Programmen, Bücher, etc. Die Seite Saugstube [65] bietet exklusiv Links zu deutschen Dateien an.

3.3 Das SoulSeek Netzwerk

SoulSeek [19] ist ein relativ neues Netzwerk und wurde erst Ende 2002 von Nir Abel, einem ehemaligen Mitarbeiter von Napster entworfen. Im Netz befinden sich etwa 100.000 Benutzer. Das Netzwerk wird hauptsächlich zum Tauschen von Musik verwendet, man kann aber auch andere Dateien herunterladen. SoulSeek ist nach der Schließung von AudioGalaxy, einer anderen Musiktauschbörse, populär geworden. Über die Seite SoulSeek Records [66] wird freie Musik von unabhängigen Musikern angeboten, die man im SoulSeek Netz herunterladen kann. Es gibt nur eine offizielle Version des Clients für Windows Systeme, es gibt aber open-source Projekte, die auch Clients für MacOS [21] oder Linux [20][22] entwickeln.

3.3.1 Struktur des Netzes

Das SoulSeek Netzwerk basiert auf dem Peer-to-Peer Modell mit einem zentralen Server, verwendet aber auch Aspekte der dezentralen Netze. Wie auch bei Napster gibt es nur einen zentralen Server mit dem sich alle Peers verbinden. In der Vergangenheit gab es die Möglichkeit einen eigenen Server zu betreiben, doch dieser Service wurde nur selten genutzt und da die Benutzer das Programm fast ausschließlich verwenden um mp3-Dateien zu tauschen, ist es besser nur eine große Community zu bilden, indem sich alle Benutzer mit einem gemeinsamen Server zu verbinden.

Der Community-Aspekt hat in SoulSeek eine viel größere Rolle als in anderen Filesharing-Programmen. Viele Funktionen in SoulSeek, die den Community-Aspekt bilden, lassen sich in dezentralen Server nur schwer implementieren. Bei einem Client/Server Modell ist es aber möglich alle Benutzer zu erreichen, ohne dafür die Netzlast der einzelnen Clients wesentlich zu erhöhen. Funktionen die von diesem Modell profitieren, sind insbesondere Räume, in denen alle Benutzer miteinander chatten können oder das Recommendation System, mit dem der Benutzer Gruppen mit ähnlicher Stilrichtung von Musik leicht finden. [67]

Da ein einzelner Server aber einen Single-point-of-failure darstellt, lässt sich das Soulseek Netzwerk genauso leicht schließen, wie Napster.

Zurzeit experimentiert der Autor ständig an neuen Funktionen und Verbesserungen der Struktur, was aber auch nicht selten zu Instabilitäten führt.

Nach Aussage des Autors skaliert der Server viel besser als die von Napster. Nicht zuletzt auch deswegen, weil er sich nicht mit dem Aufbau eines Index der angebotenen Dateien beschäftigen muss, und die Antworten auf die Suchanfragen nicht selber generiert, sondern nur die Suchanfragen an alle Benutzer weiterleitet. Bei der Suche wird auch der dezentrale Einfluss verdeutlicht: der Server sendet die Suchanfragen nur and ein paar übergeordnete Peers, die im SoulSeek Netz Parents heißen, diese verteilen die Nachrichten anschließend an alle Peers.

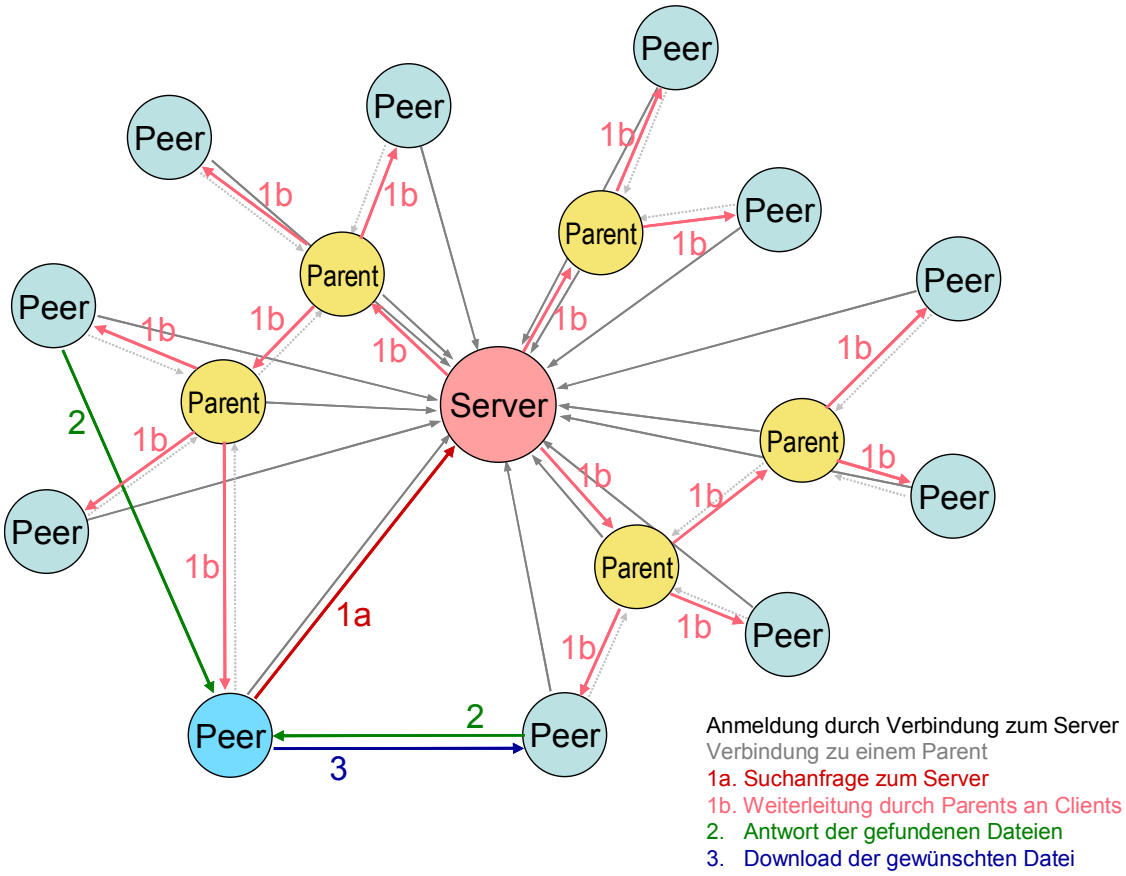


Abbildung 13: Struktur des SoulSeek Netzwerks

Login:

Beim Login verbindet sich der Client mit dem Server, dessen Adresse im Programm implementiert ist oder die er zuvor von der Homepage bezieht. Der Server sendet dem Client eine Liste der Räume, und die Anzahl der Benutzer, die sich darin aufhalten. Die Räume werden in SoulSeek verwendet um die Benutzer in verschiedene Gruppierungen, nach der Art der Musikrichtung die getauscht wird, aufzuteilen. Der Client sendet schließlich die Anzahl der von ihm angebotenen Dateien und Verzeichnissen an den Server zurück. Anschliessend fordert er eine Liste von Parents an und verbindet sich mit einem von diesen, oder beantragt selbst Parent zu werden.

Suche:

Zur Suche sendet der Client die Suchanfrage an den Server. Dieser bearbeitet sie nicht, sondern verschickt die Anfrage eingekapselt in einer Nachricht an die Parents weiter. Die Parents senden die Suchanfrage anschließend an alle ihre Children, die normale Peers sind, oder wiederum Parents. Somit werden die Suchanfragen über das gesamte Netz geflutet und die Suche erreicht alle Clients.

Bietet ein Client die gesuchte Datei an, sendet er eine Nachricht direkt an den Suchenden zurück. Die Ergebnisliste der Suchanfrage erweitert sich bei SoulSeek, bis alle Antworten eingetroffen sind.

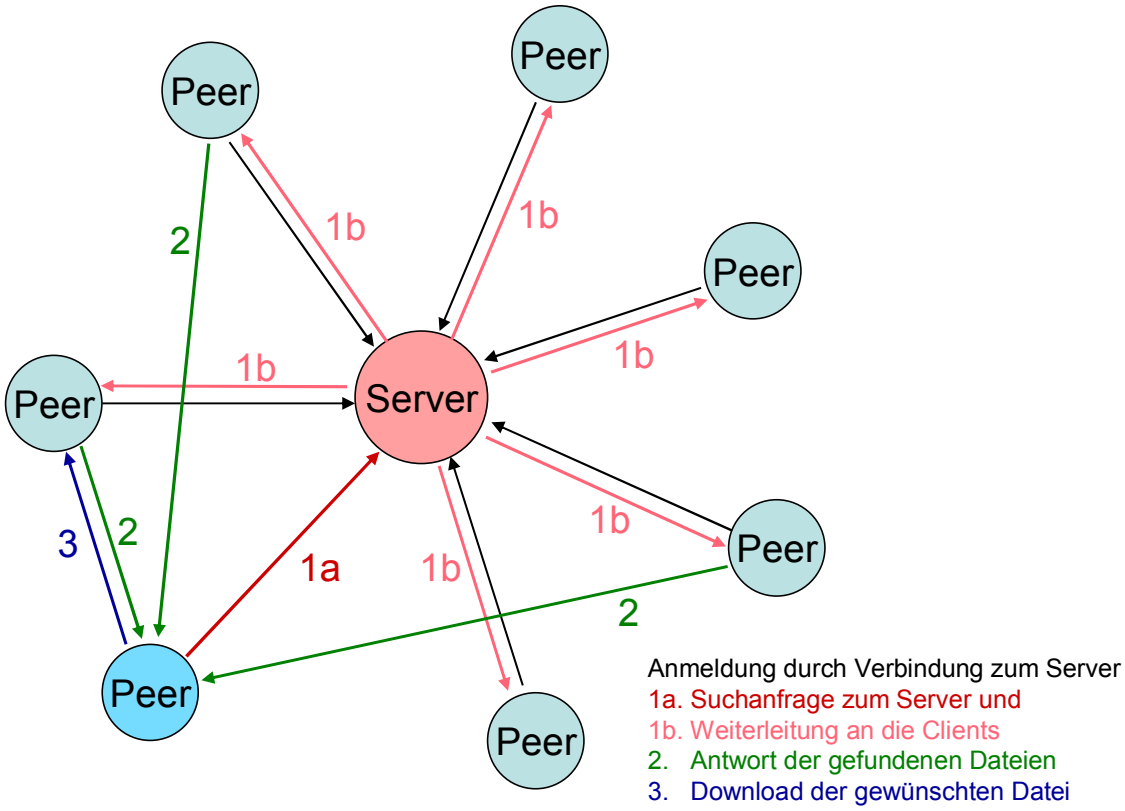


Abbildung 14: SoulSeek Suche nach Dateinamen

SoulSeek unterstützt aber auch eine Suche die auf dem Client/Server Modell beruht und nur bei der Suche nach einem exakten Dateinamen benutzt wird. Bei dieser Suche sendet der Server selbst die Suchanfrage an alle Clients weiter. Bei dieser Suche sind die Parents völlig unbeteiligt. Die Antwort auf die Suche erfolgt genauso wie bei der dezentralen Suche.

Download:

Zum Herunterladen der Datei kontaktiert der Client dann einen der Anbieter direkt.

3.3.2 Funktionalität

SoulSeek setzt großen Wert auf den Community Aspekt. Die Benutzer sollen sich in einer Gemeinschaft näher stehen in der sie miteinander reden können, ihre Vorlieben angeben und Musik austauschen.

Der SoulSeek Client bietet viele Räume an, in denen sich Benutzer, die sich für die gleiche Musikrichtung interessieren, treffen können und miteinander chatten. Es gibt mehr als 170 verschiedene Räume für die verschiedenen Interessensgebiete, z.B Jazz, Metal, Techno, HipHop, MovieMusic, … Die Benutzer können ebenfalls neue Räume gründen.

Wenn der Benutzer in einen Raum eintritt, sendet der Server die Liste aller Benutzer die sich ebenfalls dort aufhalten. Es gibt ein globales Fenster zum chatten, man kann aber auch private Nachrichten an andere Nutzer schicken. In den Räumen kann man die angebotenen Dateien anderer Benutzer durchforsten und einzelne Dateien oder ganze Verzeichnisse zum Download anfordern.

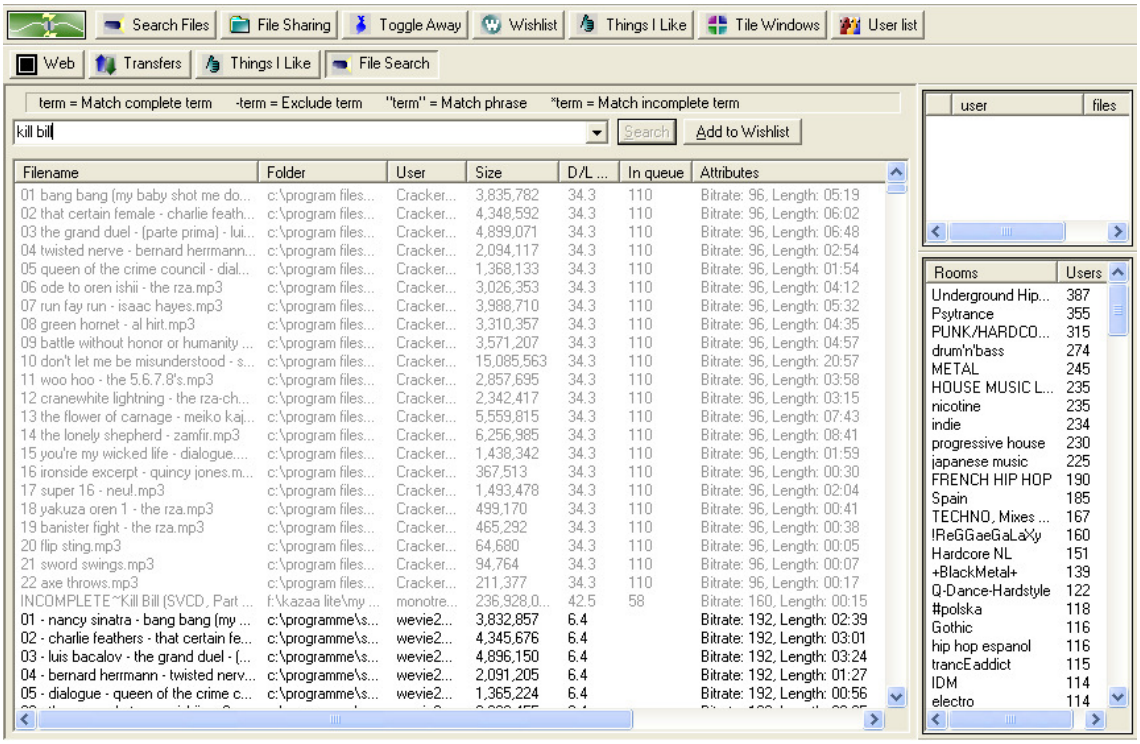


Abbildung 15: Suche in SoulSeek

Man kann aber auch gezielt nach einem bestimmten Begriff suchen und erhält eine Liste mit den Dateinamen und den User die diese anbieten von Server zurück. Die Liste enthält außerdem die Länge und Bitrate der Lieder, sowie die mittlere Uploadgeschwindigkeit der anderen Benutzer und die Länge der Warteschlange, in der man sich zum Download einreihen müsste.

Findet man die gesuchte Datei nicht, kann man auch den Titel oder den Namen der Musikgruppe in seine persönliche Wishlist aufzunehmen. Diese Wishlist wird in periodischen Abständen vom Server an alle Benutzer gesendet.

SoulSeek ist aber auch hervorragend geeignet um neue Musikgruppen kennen zu lernen. Eine Möglichkeit besteht darin in dem Chatraum der Musikrichtung mit anderen Benutzern zu reden. SoulSeek bietet aber auch ein ausgeklügeltes Recommendation System an. Hier werden die Vorzüge des Client/Server Modells ausgenutzt: Die Benutzer tragen unter 'Things I like' im Client ihre bevorzugten Musikrichtungen oder Gruppen ein, diese werden dann zum Server gesendet, der sie in einem Index verwaltet. Die Benutzer können im Client die Favoriten anderer Benutzer sehen, aber auch für einen bestimmten Begriff vom Server eine Liste von ähnlichen Gruppen und Musikrichtungen anfordern oder eine Liste der Benutzer, die diese Musik ebenfalls mögen. Mit ’get similar Users’ erhält man eine Liste von Benutzern die die gleichen Favoriten haben als man selber.

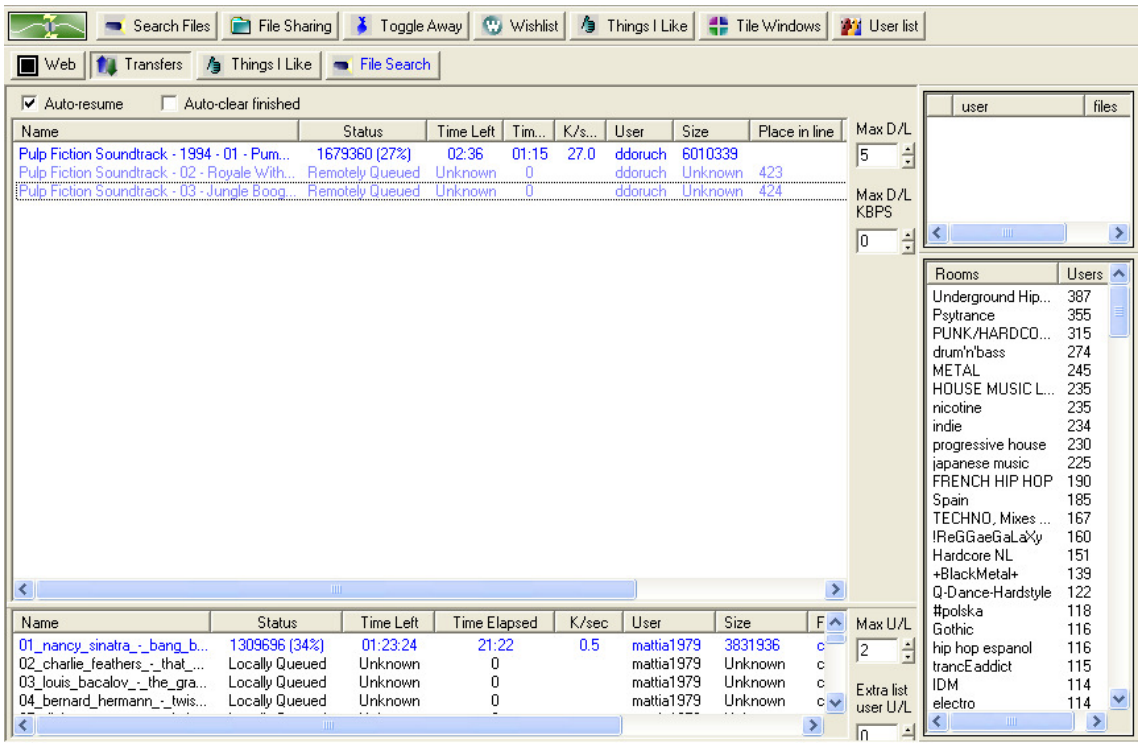


Abbildung 16: Download in SoulSeek

SoulSeek enthält keinerlei Werbebanner oder Spyware, um aber dennoch ein wenig Geld damit zu verdienen, kann der Benutzer durch eine Überweisung mittels PayPal sich Privilegien kaufen, für 5 Dollar erhält er für 30 Tage Privilegien. Als privilegierter Benutzer wird man in den Download-Warteschlangen bevorzugt behandelt, und kann somit die Dateien früher herunterladen.

3.4 Vergleich

Suche:

Die Peer-to-Peer Netze mit zentralem Server haben den Vorteil, dass der Server alle Clients kennt und einen Index aller angebotenen Dateien erstellen kann. Somit ist es Möglich das gesamte Netz zu durchsuchen ohne die Clients zu belasten. Das Suchergebnis wird schnell vom Server zurückgesendet.

Bei dezentralen Netzen kennen die Servents nicht alle Benutzer. Auch die Suche ist auf einen festgelegten Radius limitiert, eine Suche über alle Benutzer ist nicht Möglich. Die Suche erzeugt eine hohe Netzlast bei den Clients, da diese die Nachrichten weiterleiten müssen. Die Suchergebnisse werden von den Clients wieder zurück an den Suchenden geleitet, wodurch eine Suche relativ lange dauern kann und es für den Suchenden schwer abzuschätzen ist, wann die Suche beendet ist.

Bei FastTrack senden die Clients die Suchanfrage an den SuperPeer mit dem sie verbunden sind. Dieser erzeugt das Suchergebnis und sendet es zurück an den

Suchenden. Da ein SuperPeer aber nur die Benutzer kennt, die mit ihm verbunden sind, beruht auch das Suchergebnis nur auf den Dateien die dieser SuperPeer indiziert hat. Um die Suche auszuweiten sendet der SuperPeer die Suchanfrage an andere SuperPeers, mit denen er verbunden ist. Die Suche ist auch hier begrenzt und erreicht nicht das gesamte Netz.

Das eDonkey Netz sind die Server hingegen nicht miteinander verbunden. Der Server generiert auch hier nur ein Suchergebnis über die angebotenen Dateien der Benutzer, die bei ihm angemeldet sind. Im Gegensatz zu den SuperPeers bei FastTrack, die nur zirka 100 Benutzer verbinden, verwalten die Server von eDonkey aber oft mehrere Hundertausend Benutzer. Um die Suche auszuweiten, kann der Client außerdem andere Server kontaktieren. Eine Suche über das gesamte Netz ist also möglich.

Im Gegensatz zu den anderen Server-basierten Peer-to-Peer Systemen erstellt bei SoulSeek der Server keinen Index der angebotenen Dateien. SoulSeek setzt einen zentralen Server ein, mit dem alle Benutzer verbunden sind, benutzt aber zwei unterschiedliche Suchverfahren: Bei der Suche nach exakten Dateinamen sendet der Server die Suchanfrage einfach an alle Benutzer weiter, diese senden dann gegebenenfalls ein Suchergebnis direkt an den Suchenden zurück. Diese Suche wird aber nur benutzt um neue Downloadquellen zu finden. Bei der normalen Suche, die viel häufiger benutzt wird, sendet der Server die Suchanfrage lediglich an ein paar Parents, die anschließend die Suchanfragen über das gesamte Netz fluten.

Skalierung und Robustheit:

Bei Systemen mit zentralem Server, stellt dieser außerdem den Flaschenhals dar und muss den Grossteil des Nachrichtenverkehrs bewältigen. Die Kapazität des Servers limitiert somit die Anzahl der Benutzer. Eine Vergrößerung des Netzes ist nur durch Optimierung des Servers möglich. Der Server ist ein Single-point-of-failure, der bei Ausfall das gesamte Netz auflöst.

Bei dezentralen Systemen wird die Skalierung maßgeblich durch die hohe Netzlast limitiert, die durch das Weiterleiten der Ping-, Such- und Push-Nachrichten entsteht. Clients mit einer langsamen Internetanbindung, verbrauchen somit schnell ihre gesamte Bandbreite, deshalb sind heutige Netze meistens hybrid und verwenden verschiedene Hierarchieebenen der Peers. Das Netz ist resistent gegen den Ausfall der Peers, da die Peers untereinander mehrfach redundant verbunden sind, was den Nachrichten verkehr noch zusätzlich erhöht.

Das FastTrack Netzwerk skaliert aufgrund der automatischen Addierung neuer SuperPeers sehr gut. Da nur Peers mit der erforderlichen Rechenkapazität und Bandbreite zu einem neuen SuperPeer werden können, ist sichergestellt, dass die SuperPeers den höheren Datenverkehr bewältigen können. Beim Ausfall eines SuperPeers verbinden sich die Clients einfach mit einem anderen SuperPeer aus ihrem Host-Cache.

Im eDonkey Netz sind die Server meistens dedizierte Rechner, die abhängig von der Hardware auf eine feste Benutzerzahl limitiert sind. Zusätzlich lassen sich noch verschiedene Parameter einstellen, die aggressive Clients, die während einer kurzen Zeitspanne viele Requests senden, auf eine Blacklist setzen und ihnen den Login

verweigern. Die Server haben meistens eine feste IP-Adresse und bilden somit ein Ziel der Antipiraterie Firmen. Fällt ein Server aus, verbinden sich die Clients automatisch mit einem anderen Server aus der Serverliste.

Der SoulSeek Server skaliert besser als andere zentrale Server wie beispielsweise bei Napster, weil der Server keinen Index über die angebotenen Dateien erstellen muss und die Suchanfragen mit Hilfe von Parents verteilt werden. Doch der Server ist mittlerweile auf 100.000 Benutzer limitiert worden. Recommendation System führte aber zu einer so großen Belastung für den Server, dass es sogar mittlerweile wieder abgeschaltet worden ist. Um den Server weiterhin zu entlasten kann man sich vorstellen, dass in Zukunft weiterer Funktionalitäten auf die Parents übertragen wird. Denkbar wären z.B. die Suche nach exakten Dateinamen, oder ein spezieller Parent pro Chatroom der die Verteilung der Chatnachrichten übernimmt.

Funktionalität

KaZaA ist besonders beliebt, wegen der großen Benutzerzahl und dem daraus resultierenden Dateiangebot. Die Supernodes speichern viele Metadaten, so dass man präzise nach einem Medientyp, Kategorie oder Autor suchen kann. Man sieht die Bitrate, Auflösung und Dauer der Audio- oder Video-Dateien und kann dann die gewünschten Dateien zum Herunterladen auswählen. Das Protokoll ermöglicht den automatischen Download einer Datei von mehreren Quellen gleichzeitig.

Das eDonkey Netz eignet sich besonders zum Tauschen von großen Dateien, wie z.B. Kopien von CD's oder DVD's. Das Multisource File Transfer Protocol teilt die Dateien in Fragmente von 9.500 Kilobyte auf und kann diese Fragmente von mehreren Quellen gleichzeitig herunterladen. Nach dem Download eines Fragmentes wird dieses mit einem Hashwert überprüft und kann bereits mit anderen Peers getauscht werden, selbst wenn die gesamte Datei noch nicht ganz heruntergeladen worden ist, somit stehen selbst bei großen Dateien schnell viele Quellen bereit. Ein Kreditsystem stellt sicher, dass man in den Warteschlangen der Peers, an die man Daten gesendet hat, bevorzugt behandelt wird. Das Netzwerk funktioniert nach dem Prinzip Summe der Downloads ist gleich Summe der Uploads. Durch Anklicken spezieller html-Tags, kann man vom Browser aus den Download einer Datei einleiten. Eine Vielzahl von Internetseiten bieten solche Links an und überprüfen sie auf ihre Echtheit.

SoulSeek ist besonders geeignet um nach seltenen Liedern oder Musikgruppen zu suchen, aber auch um neue Gruppen einer bestimmten Musikrichtung zu finden. Benutzer können Räume einrichten, in denen sich Liebhaber einer bestimmten Musikrichtung treffen, miteinander chatten können und die angebotenen Dateien anderer Benutzer ansehen kann. Mit einem Recommendation-System kann der Benutzer seine bevorzugten Gruppen oder Musikrichtungen eintragen, und nach anderen Gruppen oder Benutzer derselben Kategorie suchen.

4. Ethereal

4.1 Überblick

Ethereal [23] ist ein open-source Packet Sniffer und Network Analyser. Das Programm kann den Nachrichtenverkehr auf einem Medium aufzeichnen und die verschiedenen Packete analysieren. Ethereal kann zusätzlich die Aufzeichnungen von mehr als 18 anderen Packet Sniffer importieren. Die aufgezeichneten Nachrichten werden in einer GUI dargestellt.

Ethereal kann mehr als 400 verschiedene Protokolle erkennen und analysieren. Die Daten der einzelnen Packete werden in einer Baumstruktur dargestellt und können nach verschiedenen Kriterien sortiert und gefiltert werden.

Die Programmoberfläche besteht aus drei Fenstern:

- Das obere Fenster zeigt eine Liste der Packete an, die aufgezeichnet wurden. Bei den Packeten wird die Zeit der Übertragung, die Quell- und Ziel-Addresse, sowie der Name des Protokolls und einer Bezeichnung der Nachricht angezeigt. Durch anklicken kann man ein Packet auswählen und sich detaillierte Informationen über die darin enthaltenen Nachrichten in den anderen zwei Fenster anzeigen lassen.

- Das mittlere Fenster ist in einer Baumstruktur organisiert und gibt detailierte Informationen über die Protokolle, die zur Übertragung der Nachricht benutzt wurden, sowie über die einzelnen Datenfelder, die in der Payload der Nachricht übertragen werden.

- Das untere Fenster zeigt die Daten des Packets an. Man kann durch anklicken im mittleren Fenster einzelne Datenfelder selektieren.

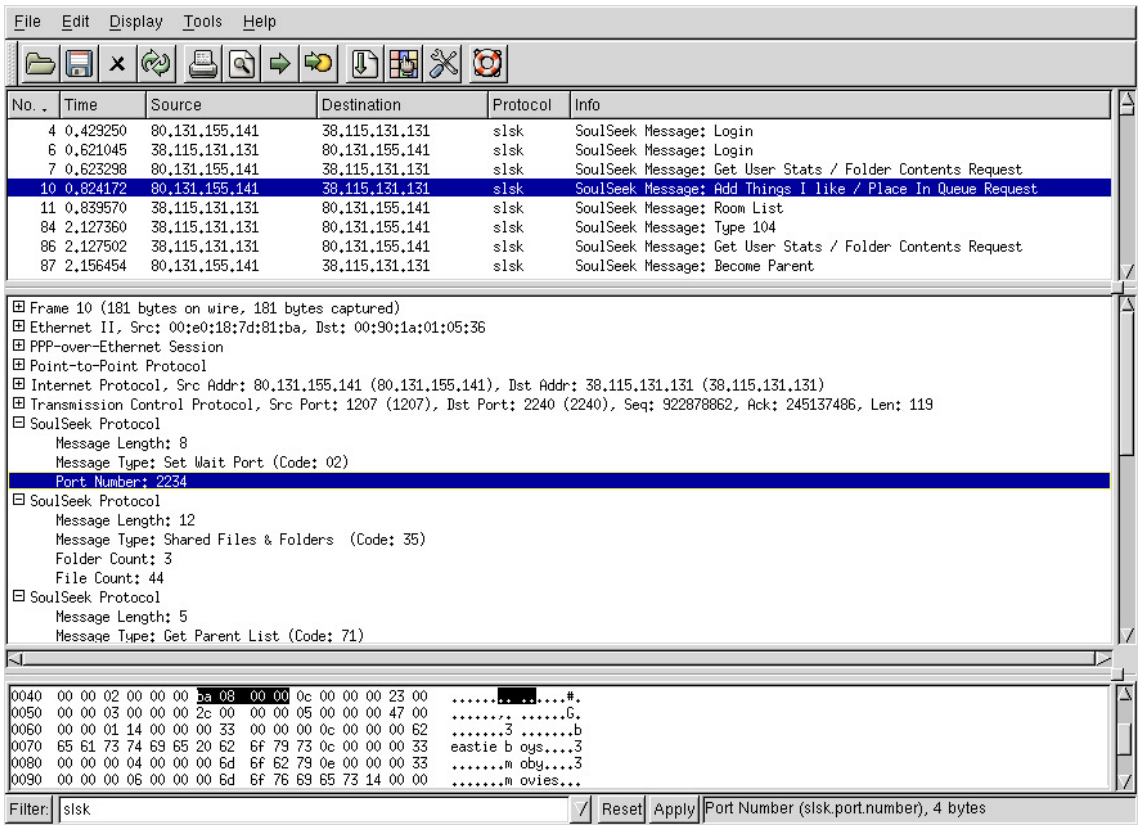


Abbildung 17: Ethereal GUI

4.2 Implementierung neuer Protokoll Dissektoren

Die verschiedenen Protokolle werden in Ethereal mit so genannten Dissektoren erkannt und analysiert. Da der Quellcode von Ethereal offen ist, ist es möglich neue Protokolle zu analysieren, indem man einen neuen Dissektor speziell für das Protokoll implementiert.

Die Daten der einzelnen Packete werden in einem Puffer bereitgestellt. Dieser tvbuffer (testy virtual buffer) wird dem Dissektor zur Analyse übergeben. Der tvbuffer enthält nur die Daten, die zur eigentlichen Nachricht gehören, wird ein Packet z.B. mittels TCP übertragen, sieht der Dissektor, der das Packet analysieren soll nur die Payload, nicht aber den TCP-Header.

Die Daten in einem tvbuffer werden mit speziellen Accessoren ausgelesen. Es gibt verschiedene Accessoren, für das Auslesen von einzelnen Bytes, von 16-, 24-, oder 32-Bit Integer, zur Unterscheidung von Network-to-host order und Little-endian-to-host order, oder das Auslesen von Gleitkommazahlen.

Es gibt Funktionen um die Bezeichnung des Protokolls und der Nachricht im oberen Fenster hinzuzufügen, und zu verändern.

Hauptbestandteil eines Protokoll Dissektors ist der Code, der die Daten analysiert und die Baumstruktur im mittleren Fenster aufbaut. Die einzelnen Datenfelder werden mit einem eigenen Bezeichner gekennzeichnet, der später benutzt werden kann um in der GUI nach diesen Felder zu suchen und zu filtern. Ethereal unterstützt viele verschiedene Datentypen, die dann automatisch mit speziellen Funktionen in der gewünschten Form angezeigt werden, wie z.B. boolsche Werte, verschiedene Integer und Gleitkommazahlen, Zeitangaben, Strings, IP-Adressen, …

Desweiteren kann man die genaue Position des Datenfeldes im Packet angeben, die benutzt wird um dann im unteren Fenster die Bytes zu markieren.

Der nachfolgende Code liest z.B. die Länge eines Strings als 32-Bit Integer aus, und zeigt diese Länge und den String, der zuvor als Username registriert wurde in der Baumansicht im slsk_tree an:

```
proto_tree_add_uint(slsk_tree, hf_slsk_string_length, tvb, offset,
                    4, tvb_get_letohl(tvb, offset));
proto_tree_add_item(slsk_tree, hf_slsk_username, tvb, offset+4,
                    tvb_get_letohl(tvb, offset), FALSE);
```

Die Länge des Strings wird mit `proto_tree_add_uint()` angezeigt. Der Parameter `slsk_tree` gibt dabei die Stelle im Baum an, in der die Information ausgegeben werden soll. `hf_slsk_string_length` ist der Bezeichner unter dem das Datenfeld „String Length“ in Ethereal registriert wurde und zeigt den Namen des Datenfeldes in der Baumansicht als Präfix an. Der Bezeichner wird ausserdem benutzt um in Ethereal nach dem Datenfeld zu filtern. `tvb` kennzeichnet den Puffer aus dem die Nachricht augelesen wird. Die Variablen `offset` und die nachfolgende Zahl bestimmen die Position und die Länge des Datenfeldes im Puffer und werden benutzt um das Datenfeld in dem unteren Fenster in Ethereal hervorzuheben. Mit der Callback Funktion `tvb_get_letohl()` wird schließlich das Integer aus dem Puffer ausgelesen. Die Funktion stellt `proto_tree_add_item()` anschliessend den Inhalt des Strings dar, der von Typ `hf_slsk_username` ist. Der Offset ist um 4Bytes erhöht worden, und `tvb_get_letohl(tvb, offset)` gibt die Länge des Strings zurück

Eine weitere interessante Funktion ist das Desegmentieren von TCP Paketen. Ist eine Nachricht zu groß, um in einem TCP-Frame übertragen zu werden, wird sie automatisch in mehrere Segmente aufgeteilt. Diese Segmente können in Ethereal automatisch wieder zusammengefügt und dann analysiert werden.

Man kann so 4 verschiedene Fälle unterscheiden:

a) Eine Nachricht wird in einem TCP-Frame übertragen.
b) Eine Nachricht ist so groß, dass sie in mehrere TCP-Segmente aufgeteilt ist.
c) Ein TCP-Frame enthält mehrere kleine Nachrichten.
d) Eine Kombination aus b) und c).

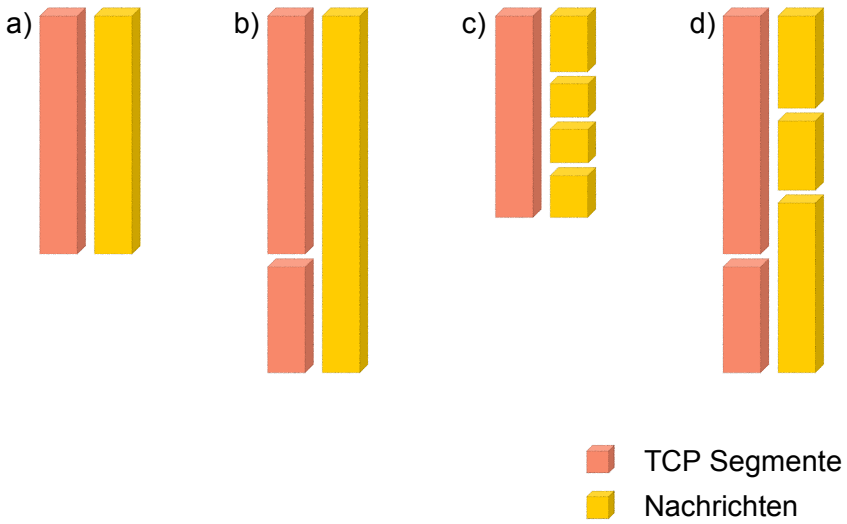


Abbildung 18: Übertragung von Nachrichten in TCP-Segmenten

Anstatt die Pakete gleich zu analysieren, muss man vorher überprüfen, ob die Nachricht aufgeteilt wurde, und diese gegebenenfalls von dem TCP-Dissektor dann komplettieren lassen.

```
static void dissect_slsk(tvbuff_t *tvb, packet_info *pinfo,
      proto_tree *tree)
{
   tcp_dissect_pdus(tvb, pinfo, tree, slsk_desegment, 4,
                    get_slsk_pdu_len, dissect_slsk_pdu);

}
```

Als Beispiel ist hier der Dissektor vom SoulSeek Protokoll angeführt, der als erstes die Funktion `tcp_dissect_pdus` aufruft. Dieser Funktion übergibt er die Daten der Payload im Puffer `tvb`, sowie Paket-Informationen und die Position in der Baumstruktur. `slsk_desegment` ist ein boolscher Wert, mit dem man das Desegmentiern in der GUI ein- oder ausschalten kann. Die Funktion `tcp_dissect_pdus()` benötigt des Weiteren die Länge des Pakets, die mit dem Callback `get_slsk_pdu_len()` übergeben wird. Die Zahl `4` gibt dabei an wie viel Bytes benötigt werden um die Paketlänge zu ermitteln (im SoulSeek Protokoll steht die Länge des Pakets gleich am Anfang des Paketkopfes). Das komplette Packet wird nach dem Zusammenfügen wieder an den SoulSeek Dissektor übergeben und mit der Funktion `dissect_slsk_pdu()` analysiert.

Dieser Vorgang ist in Abbildung 19: Datenfluss - Desegmentierung von TCP Paketen noch einmal verdeutlicht:

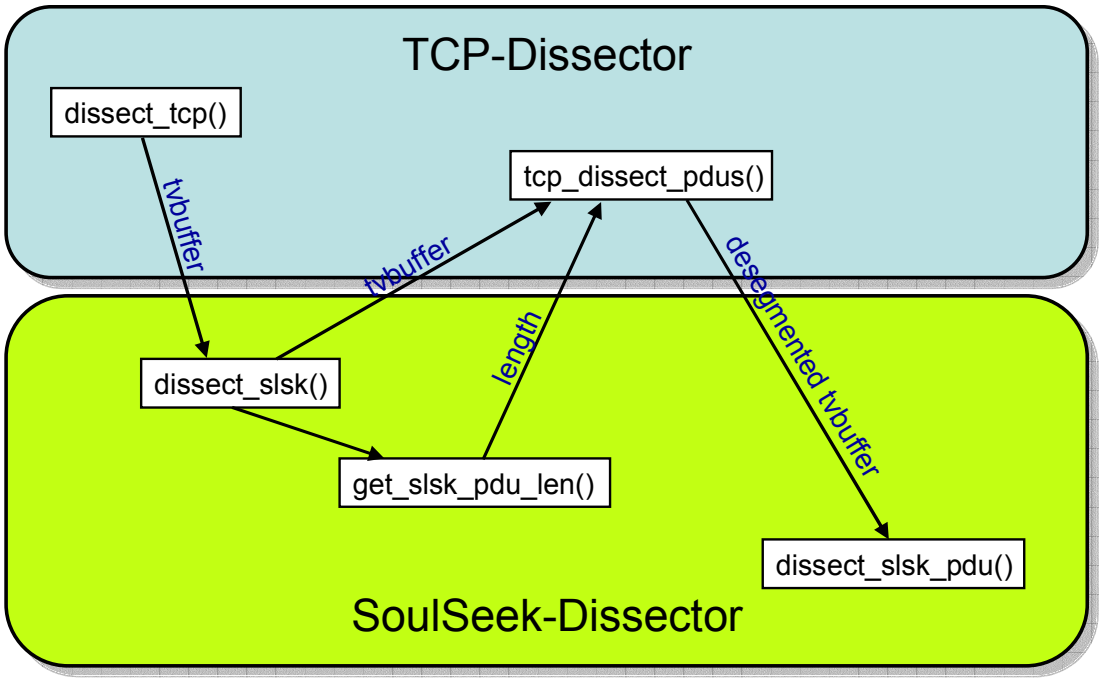


Abbildung 19: Datenfluss - Desegmentierung von TCP Paketen

4.2.1 FastTrack-Dissektor

Die ersten Clients des FastTrack Netzes waren alle kommerzielle closed-source Programme. Da es sich um ein proprietäres Netz handelt, liegen die Protokollspezifikationen nicht offen. Des Weiteren werden die Nachrichten im FastTrack Protokoll verschlüsselt übertragen. [86]

Das open-source Project giFT (giFT: internet File Transfer) [6] wurde ursprünlich als FastTrack Client für Linux gegründet, und konnte durch Reverse-Engineering der Usernode-Supernode Kommunikation einen Client implementieren, indem das Suchen und Herunterladen von Dateien funktionierte. Doch FastTrack reagierte schnell und änderte den Verschlüsselungsalgorithmus mit einem Security-Update, das alle offiziellen Clients automatisch installierten, so dass der open-source Client vom Netz ausgeschlossen war.

Seit dem der Digital Millennium Copyright Act (DMCA) in den Vereinigten Staaten das Reverse-Engineering verbietet, droht Sharman Networks den Internet Seiten, die Informationen über das Protokoll verbreiten mit rechtlichen Schritten. Somit ist es schwierig einen Dissektor für Ethereal zu schreiben.

Heute gibt es mit xMule [13], mlDonkey [14] und dem gift-FastTrack Plugin [7] jedoch wieder drei open-source Projekte, die das FastTrack Protokoll teilweise unterstützen, doch die Supernode-Supernode Kommunikation ist weiterhin unerforscht. [72]

4.2.2 eDonkey-Dissektor

eDonkey2000 ist zwar ein kommerzielles closed-source Programm, doch seit der Entwicklung vom open-source Client eMule sind die Protokoll Spezifikationen bekannt, und wurden sogar für eMule noch erweitert, deshalb stand der Implementierung eines Dissektors nichts im Weg. [87][88]

Seit der Version 0.9.14 vom 23. July 2003 beinhaltet Ethereal auch einen eDonkey-Dissektor und unterstützt die Analyse der Nachrichten vom original eDonkey2000 Protokoll sowie den verschiedenen Erweiterungen vom eMule Client und Overnet.

4.2.3 SoulSeek-Dissektor

SoulSeek ist ebenfalls ein closed-source Programm. Es gibt aber auch für das SoulSeek Netzwerk ein Projekt, das einen open-source Client entwickelt und durch Reverse-Engineering einen Teil des Protokolls documentiert hat. [89]

Die Protokollspezifikationen waren aber unvollständig und die Struktur des Netzes und meisten Nachrichtensequenzen waren unbekannt. Aufgrund der schnellen Änderungen des Clients haben außerdem viele Nachrichten das Format geändert.

Die Implementierung eines SoulSeek Dissektors war also sinnvoll zur besseren Verständnis des Protokolls und ist Teil dieser Arbeit gewesen. Mit Hilfe des neuen Dissektors kann man den Nachrichtenverkehr zwischen dem SoulSeek Client und dem Server, sowie zwischen zwei Clients aufzeichnen und somit den Ablauf der Nachrichtensequenzen ermitteln. Im Dissektor kann man ebenfalls für jede Nachricht das Format und die in ihr enthaltenen Datenfelder sehen.

Die Erkennung der Nachrichtentypen wird beim SoulSoulSeek Protokoll dadurch erschwert, dass der Message Code einer Nachricht nicht eindeutig deren Typ bestimmt, außerdem gibt es Nachrichten deren Message Code als Integer angegeben ist, bei anderen als Byte. Somit haben Nachrichten mit demselben Message Code je nach Kontext verschiedene Bedeutungen und Formate. Deshalb musste eine Funktion implementiert werden, die überprüft ob eine Nachricht, einem bestimmten Format entspricht. Erst dann kann man den Nachrichtentyp bestimmen und die Bedeutung der einzelnen Datenfelder beschreiben.

In Anhang B steht als Beispiel der Quellcode wichtiger Funktionen des Dissektors und ein Ausschnitt des Quellcodes, der die Nachrichten des Message Code 36 analysiert. Mit `get_message_type()` wird getestet ob der Message Code als Integer angegeben ist, indem man ihn mit einer Liste bekannter Codes vergleicht und der Typ als String zurückgegeben. Ist er unbekannt, so wird getestet ob es sich um einen Nachrichtentyp handelt bei dem der Message Code als Byte angegeben ist. `check_slsk_format()` testet ob es sich bei einem gegebenen Puffer um eine Nachricht eines bestimmten Formats handert. Das Format wird hierbei als Character-Array angegeben, wobei '`i`' für Integer, '`b`' für Byte und '`s`' für String steht. Zusätzlich kann am Ende noch ein Stern '*' angegeben werden, um weitere Bytes zu vernachlässigen.

Der Dissektor erkennt die Nachrichten die über Port 2234, 2240 oder 5534 gesendet werden als SoulSeek Protokoll. Packete werden standartmäßig wie in 4.2 beschrieben desegmentiert und wenn der Inhalt komprimiert ist entpackt. In der Benutzeroberfläche kann man unter Edit|Preferences...|Protocols|SoulSeek das Desegmentieren und Dekomprimieren aber auch ausschalten.

Man kann nur die SoulSeek Nachrichten anzeigen, indem man nach der Abkürzung '`slsk`' filtert. Man kann aber auch präzise nach Datenfeldern filtern, mit '`slsk.message.code == 9 and slsk.token == 52`' werden z.B. nur die 'File Search Result' Antworten auf die Sucheanfrage mit Token 52 angezeigt.

5. Praktische Untersuchung des SoulSeek Protokolls

Die in diesem Kapitel enthaltenen Informationen über das SoulSeek Protokoll wurden mit Hilfe des SoulSeek Dissektors ermittelt, der als Teil dieser Arbeit implementiert wurde. Die verschiedenen Funktionalitäten des Protokolls, wurden mit dem offiziellen SoulSeek Client für Windows getestet, und der so entstandene Nachrichtenverkehr mit Ethereal aufgezeichnet. Die Protokoll Spezifikationen beziehen sich auf den offiziellen SoulSeek Client für Windows in der Version 151 vom 21. Oktober 2003.

Die Implementierung des Protokoll-Dissektors für Ethereal wurde durch Reverse-Engineering durchgeführt. Das Format der meisten Nachrichten konnte durch Ausprobieren im Client und Analysieren der Änderungen im Nachrichtenverkehr ermittelt werden. Als Quelle des Nachrichtenaufbaus war die Protokoll-Dokumentation [89] des SoleSeek [21] Clients für MacOS hilfreich, jedoch waren Informationen unvollständig und entsprachen in vielen Bereichen nicht mehr der aktuellen Version des SoulSeek Clients.

5.1 Aufbau der Nachrichten

Das SoulSeek Protokoll besteht aus verschiedenen Nachrichtentypen, die die Kommunikation zwischen Server, Parent, und Client ermöglichen.

Die Nachrichten in SoulSeek beginnen immer mit einem 4 Byte Integer, der die Länge der Payload angibt, die vier Bytes der Länge selbst sind dabei nicht mit eingerechnet.
Die eigentliche Nachricht wird im Payload übertragen.

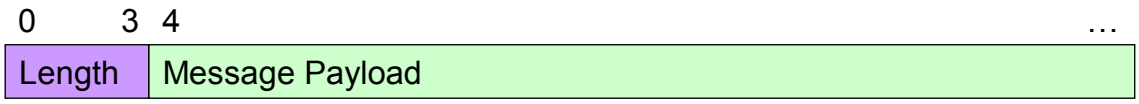


Abbildung 20: Aufbau einer SoulSeek Nachricht

Es gibt zwei mögliche Formate:

Die meisten Nachrichten beginnen mit einem Integer, der den Message Code beinhaltet und den Typ der Nachricht angibt, gefolgt von verschiedenen Datentypen, deren Anzahl und Reihenfolge je nach Nachrichtentyp verschieden ist.

Lediglich drei Nachrichtentypen weichen von diesem Format ab. Sie beginnen mit einem einzelnen Byte für den Message Code anstatt eines Integers. Diese Nachrichten werden exklusiv für die Peer-to-Peer Kommunikation genutzt.

Der Message Code bestimmt nicht eindeutig den Nachrichtentyp und somit das Format einer Nachricht, es wird ebenfalls je nach Kommunikations-Partner und Richtung unterschieden. So kann eine Nachricht mit gleichem Message Code unterschiedliche Bedeutungen und Formate haben, je nachdem ob er vom Server zum Client gesendet wird, vom Client zum Server oder von Client zu Client.
Das SoulSeek Protokoll scheint derzeitig, bei den Nacherichtentypen nicht zu unterscheiden ob der Kommunikations-Partner bei einer Client-Client Verbindung ein Parent ist oder ein normaler Peer.

In den SoulSeek Nachrichten können drei unterschiedliche Datentypen beinhaltet sein:

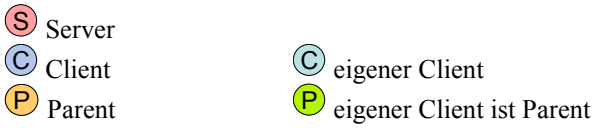
- Bytes: die bisher nur die Werte 0 oder 1 einnehmen, und für boolsche Werte benutzt werden
- Integer: die aus 4 Bytes bestehen und in Little Endian übertragen werden
- String: die aus einem 4-Byte Integer bestehen, der die Länge n des Strings angibt, gefolgt von n Bytes, die den eigentlichen String bilden. Die Strings sind nicht Null-terminiert.

Nachrichten, die zwischen zwei Clients gesendet werden und große Listen enthalten, sind zusätzlich noch mit zlib komprimiert. Dies gilt für die Nachrichten Shared File List (5), File Search Result (9) und Folder Contents Request (37), die eine Auflistung von Verzeichnissen und Dateien enthalten und mehrere Zehntausend Einträge haben können.

5.2 Nachrichtensequenzen

Die Nachrichtensequenzen wurden mit Ethereal aufgezeichnet und dem SoulSeek Dissektor analysiert.

Bedeutung der Ikone in den Diagrammen:



Neben den Ikonen steht der Name der gesendeten Nachricht und in Klammern der zugehörige Message Code. Ein voranstehendes B bedeutet dass der Message Code aus einem Byte besteht, ansonsten aus einem Integer.

5.2.1 Loginsequenz

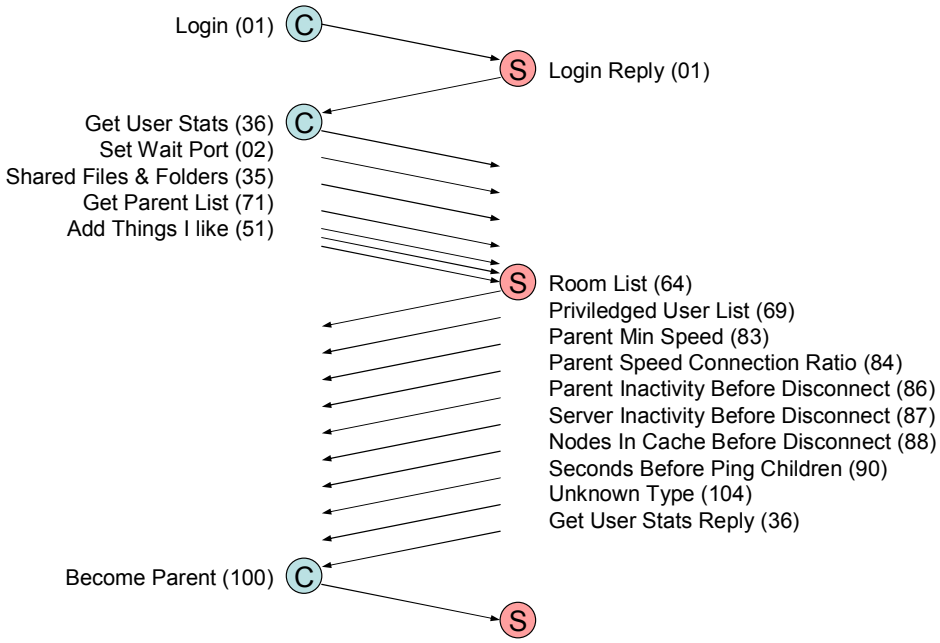


Abbildung 21: Loginsequenz

=> Login (01)
Der Client sendet zum Login, seinen Namen, sein Passwort und die Versionsnummer seines Clients.

<= Login Reply (01)
Der Server antwortet mit einem Login Reply und sendet bei erfolgreichem Login einen Willkommensgruss, bei fehlgeschlagenem Login den Grund, z.B. „Passwort ungültig“.

=> Get User Stats (36)
Der Client fragt nach seinen eigenen Statistiken.

=> Set Wait Port (02)
Der Client sendet dem Server die Portnummer die er benutzt.

=> Shared Files & Folders (35)
Der Client sendet die Anzahl der Dateien und Verzeichnissen, die er zum Tauschen anbietet, diese werden beim betreten eines Raumes zusammen mit dem Benutzernamen angezeigt.

=> Get Parent List (71)
Der Client fordert eine Liste der Parent Nodes an.
Diese Nachricht besteht aus nur einem Byte, das hier 1 ist.

=> Add Things I like (51)
Wenn der Benutzer im Client unter 'Things I like' seine Favoriten eingetragen hat, werden die beim Login ebenfalls übertragen.

<= Room List (64)
Der Server sendet eine Liste mit den Namen der Räumen und der Anzahl der Benutzer, die sich darin aufhalten, zurück.

<= Priviledged User List (69)
Dies ist die Liste der Benutzer, die über Privilegien verfügen, und in den Warteschlangen vom Client bevorzugt behandelt werden sollen.

Die nachfolgenden Nachrichten bestehen aus nur jeweils einem Integer, für Einstellungen im Client, die vom Server geändert werden können.

<= Parent Min Speed (83)
Wert 1

<= Parent Speed Connection Ratio (84)
Wert 10

<= Parent Inactivity Before Disconnect (86)
Timeout: 120 Sekunden

<= Server Inactivity Before Disconnect (87)
Timeout: 20000 Sekunden

<= Nodes In Cache Before Disconnect (88)
Wert 3

<= Seconds Before Ping Children (90)
Timeout: 100 Sekunden

<= Unknown Type (104)
Wert 720

<= Get User Stats Reply (36)
Diese Nachricht enthält Statistiken, über einen bestimmten Benutzer: den Namen, die Anzahl der Dateien und Verzeichnissen die er anbietet und die mittlere Upload-Geschwindigkeit.

=> Become Parent (100)
Die Nachricht besteht aus nur einem Byte 1 und wird optional nach dem Login übertragen. Die Nachricht scheint nur gesendet zu werden, wenn der Client die Erforderungen erfüllt um Parent zu werden.

5.2.2 Suche nach Dateien

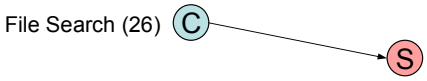


Abbildung 22: Suche nach Dateien

=> File Search (26)
Bei der Suche sendet der Client den Suchtext sowie einen zugehörigen Token an den Server.

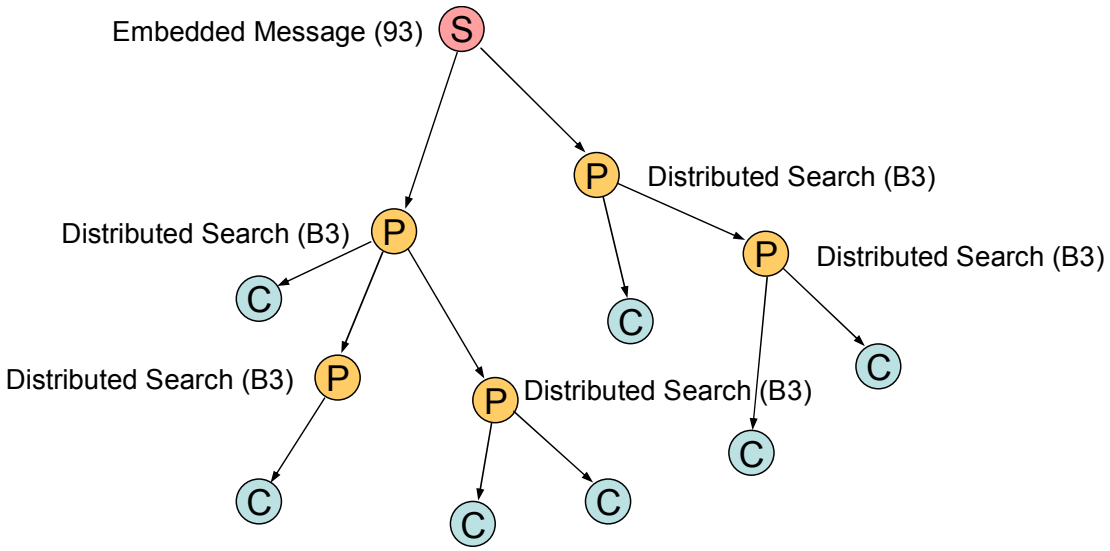


Abbildung 23: Propagierung der Suchanfragen

Die Suchanfrage wird anschließend vom Server in eine Nachricht eingebettet und an auserwählte höhergestellte Peers gesendet, die so genannten Parents.

Die Parents wiederum verbreiten die Nachricht an andere Parents, und senden sie an die normalen Peers.

=> Embedded Message (93)
Diese Nachricht gilt als eine Art Behälter, in den der Server die Suchnachricht, die die Parents an die Peers weiterleiten, einkapselt.

=> Distributed Search (B3)
Die Distributed Search Nachricht wird von den Parents propagiert und erreicht schließlich alle Peers.

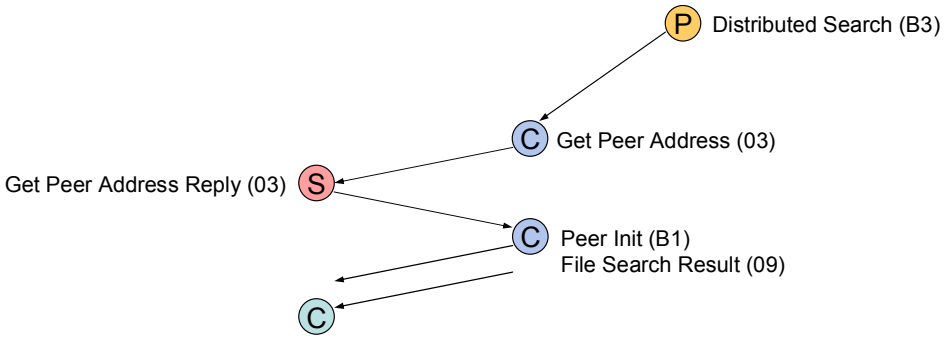


Abbildung 24: Antwort auf die Suchanfrage

Empfängt ein Client eine Suchanfrage nach einer Datei, die er anbietet, fordert er die IP-Adresse des Suchenden und versucht eine direkte Verbindung zu ihm aufzubauen um eine Liste mit den gefundenen Dateien zu senden.

=> Get Peer Address (03)
Der Client sendet den Namen eines Users zum Server um dessen Adresse zu erfahren.

<= Get Peer Address Reply (03)
Der Server sendet die IP-Adresse und die Portnummer des Users zurück.

=> Peer Init (B1)
Peer Init wird vom Client gesendet um einen Verbindung zu initialisieren. Die Nachricht besteht aus einem String der den Typ der Verbindung angibt und einem Token. Als Verbindungstyp wird Peer Connection ('P') angegeben.

=> File Search Result (09)
Diese Nachricht ist komprimiert mit zlib. Es wird der eigene Benutzername und der Token der Suchanfrage übertragen, gefolgt von einer Liste mit den Treffern aus der Suche. Für jede Datei wird der volle Name mit Pfad angegeben, die Größe und einige weitere Attribute wie z.B. die Bitrate, Länge und ob es mit variabler Bitrate komprimiert wurde, bei mp3-Dateien. Die Nachricht gibt außerdem die mittlere Upload-Geschwindigkeit an und ob nach ein Upload-Slot frei ist, bzw. wie lang die Warteschlange ist.

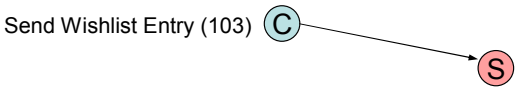


Abbildung 25: Send Wishlist Entry

Findet man bei der Suche keine Treffer, besteht die Möglichkeit, den Suchtext in seine Wishlist aufzunehmen. Die Einträge der Wishlist werden periodisch an den Server gesendet, und dann wie normale Suchanfragen an alle Peers propagiert.

=> Send Wishlist Entry (103)
Die Nachricht enthält, wie die normale Suche, den Suchtext und einen Token.

5.2.3 Propagierung der Suchanfragen

Es gibt drei unterschiedliche Positionen, die man im SoulSeek Netz einnehmen kann:

- Ein normaler Peer mit Verbindung zu einem Parent
- Ein Parent mit Verbindung zu einem anderen Parent
- Ein Parent mit Verbindung zum Server.

5.2.3.1 Verbindung Peer-Parent

Im Normalfall ist der Client ein gewöhnlicher Peer, der nur Suchanfragen empfängt, aber keine weiterleitet.

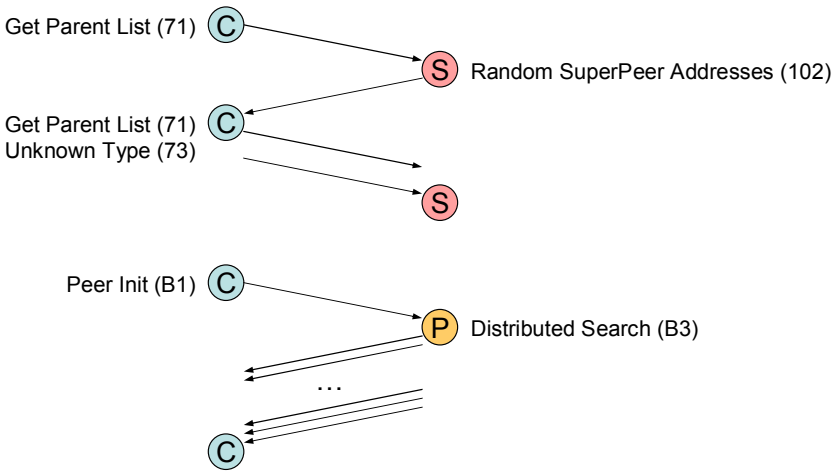


Abbildung 26: Verbindung Peer-Parent

=> Get Parent List (71)
Diese Nachricht wird gleich beim Login gesendet um eine Liste von Parent Nodes anzufordern. Diese Nachricht besteht aus nur einem Byte, das bei der Anforderung 1 ist.

<= Random Parent Addresses (102)
Der Server sendet eine Liste mit dem Namen, der IP-Adresse und der Portnummer von drei Parents zurück.

=> Get Parent List (71)
Diese beiden Nachrichten werden gleich nach dem Empfang der Parent Liste aber noch vor dem Verbindungsaufbau mit einem der Parents gesendet.
Das Byte ist diesmal auf 1 gesetzt.

=> Unknown Type (73)
Die Bedeutung dieser Nachricht ist noch unklar, sie besteht aus nur einem Integer, der immer 0 zu sein scheint.

Nachdem der Client die Liste mit Parents empfangen hat, baut er eine Verbindung mit einem dieser auf.

=> Peer Init (B1)
Der Client sendet Peer Init um eine Verbindung zu einem Parent herzustellen. Der Verbindungstyp ist mit 'D' für Distributed Search gekennzeichnet.

Ist die Verbindung zu einem Parent hergestellt, sendet dieser fortwährend die Suchanfragen anderer Benutzer.

<= Distributed Search (B3)
Für jede Suchanfrage eines Benutzers wird eine Nachricht gesendet, mit dem Usernamen des Suchenden, einem Token und dem Suchtext.

5.2.3.2 Verbindung Parent-Parent

Wenn der Client die Rolle eines Parent einnimmt, ist er außerdem zuständig für die Weiterleitung von Suchanfragen.

Der Client scheint die Bereitschaft Parent zu werden mit dem Nachrichtentyp 100 beim Login zu übermitteln, wahrscheinlich nachdem festgestellt worden ist, dass er die nötigen Erforderungen, wie z.B. eine schnelle Internetverbindung erfüllt.

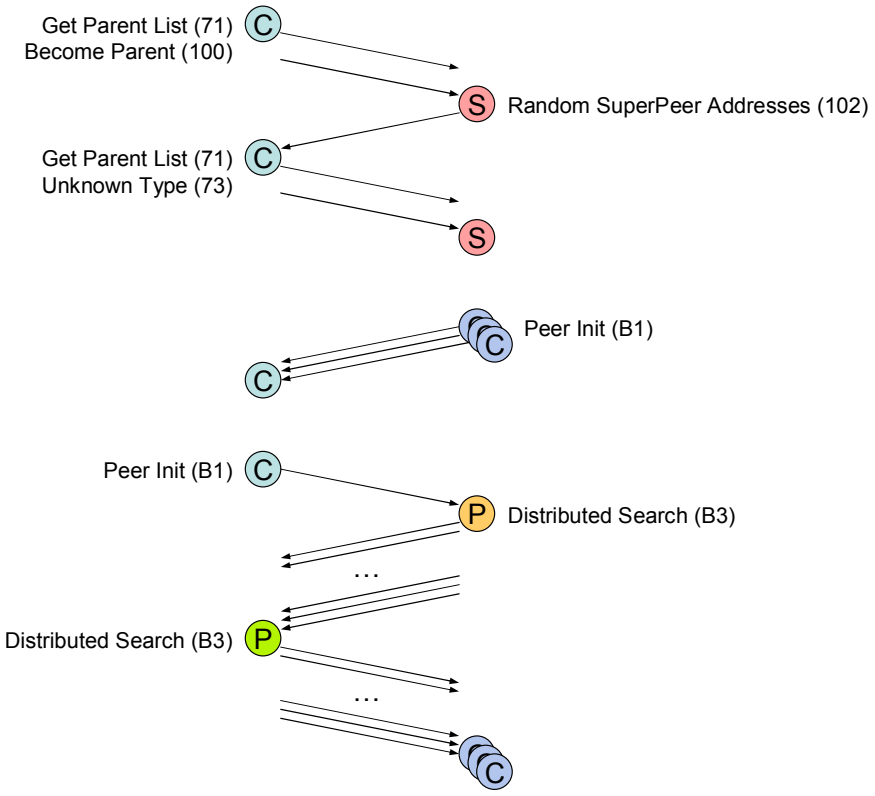


Abbildung 27: Verbindung Parent-Parent

Der Client fordert wie im vorherigen Beispiel eine Liste von Parents an.

<= Peer Init (B1)
Nach dem senden der Nachricht Become Parent (100), senden mehrere Peers ein Peer Init, um eine Verbindung zu initialisieren. Der Verbindungstyp ist mit 'D' für Distributed Search gekennzeichnet und signalisiert, dass der Peer Suchanfragen gesendet bekommen will.

=> Peer Init (B1)
Der Client baut zuerst eine Verbindung mit einem Parent auf, mit einem Peer Init vom Typ 'D'.

<= Distributed Search (B3)
Er empfängt vom Parent fortwährend Suchanfragen.

=> Distributed Search (B3)
Der Client nimmt selbst die Rolle eines Parents ein, und sendet sofort jede empfangene Suchanfrage an seine 'Children' weiter.

5.2.3.3 Verbindung Server-Parent

Wenn der Client die Rolle eines Parent einnimmt, kann er auch in direkter Verbindung mit dem Server sein, anstatt mit einem anderen Parent verbunden zu sein.
Es ist anzunehmen, dass der Client in diesem Fall die Suchanfragen nur an andere Parents weiterleitet und nicht an normale Peers.

Bei der Verbindung Server-Parent scheint der Nachrichtentyp (100) keinen Einfluss zu haben.

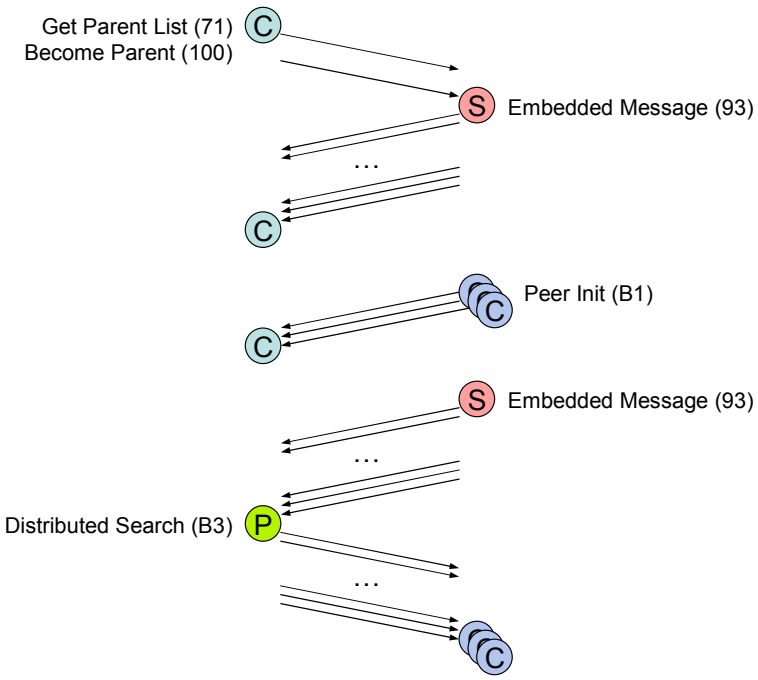


Abbildung 28: Verbindung Server-Parent

Der Server fängt gleich nach dem Empfang von Get Parent List an eingebettete Nachrichten zu senden, ohne die Nachricht Become Parent (100) abzuwarten.

<= Embedded Message (93)
Bei den eingebetteten Nachrichten handelt es sich um eine Nachricht mit einem Integer für den Message Code, an die komplette die Payload einer anderen Nachricht angehängt ist.

<= Peer Init (B1)
Nach dem senden von Become Parent (100), senden mehrere Peers wie im vorherigen Fall ein Peer Init, um Suchanfragen anzufordern.

=> Distributed Search (B3)
Für jede empfangene eingebettete Nachricht, entfernt der Client den anführenden Integer und sendet die so gewonnene Nachricht an alle Children weiter.

5.2.4 Exact File Search

In SoulSeek besteht auch die Möglichkeit nach exakten Dateinamen zu suchen. Dies ist sinnvoll um neue Quellen für eine bestimmte Datei zu finden, um einen abgebrochenen Download wieder aufzunehmen, um schneller Quellen zu finden, oderman sich nicht in eine lange Warteschlange einreihen will.

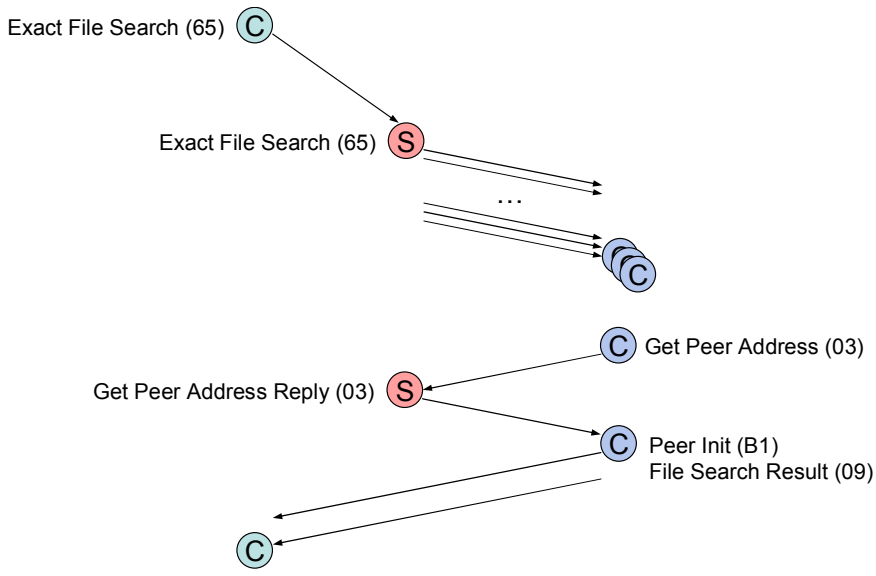


Abbildung 29: Exact File Search

=> Exact File Search (65)
Der Client sendet den exakten Namen einer Datei und dessen Verzeichnis zusammen mit einem Token.

=> Exact File Search (65)
Der Server fügt noch zusätzlich den Username hinzu, sendet die Nachricht an alle Peers.

Im Gegensatz zur normalen Suche, übernimmt der Server die Propagierung der exakten Dateisuche selbst.

<= Get Peer Address (03)
=> Get Peer Address Reply (03)
<= Peer Init (B1)
<= File Search Result (09)
Die Peers antworten bei einem Treffer mit einer File Search Result Nachricht, wie bei einer normalen Suche (siehe 5.2.2 Suche nach Dateien).

5.2.5 File Transfer

Wenn der Client eine Datei von einem bestimmten Benutzer herunterladen will, prüft er erst ob es den Benutzer noch gibt. Kennt er dessen Adresse nicht sondern nur den Benutzernamen, dann fordert er sie vom Server. Anschließend werden die Statistiken und der Status des Benutzers angefragt.

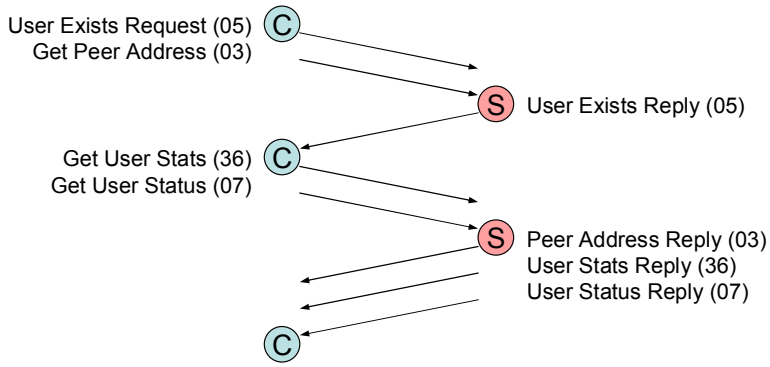


Abbildung 30: User exists

=> User Exists Request (05)
Der Client sendet diese Nachricht um zu ermitteln ob der Benutzer überhaupt existiert.

=> Get Peer Address (03)
Der Client fordert die Adresse des Peers an.

<= User Exists Reply (05)
Der Server antwortet mit einem Byte 0 oder 1.

=> Get User Stats (36)
Der Client fordert die Benutzerstatistiken an.

=> Get User Status (07)
Der Client fordert den Status des Benutzers an.

<= Peer Address Reply (03)
Der Server sendet die IP-Adresse und die Portnummer des Peers zurück.

<= User Stats Reply (36)
Der Server sendet die mittlere Uploadgeschwindigkeit und die Anzahl seiner angebotenen Dateien und Verzeichnissen.

<= User Status Reply (07)
Der Server sendet einen Integer mit dem Status des Benutzers.
Dieser kann online, offline, away oder unknown sein.

Wenn der Client eine Verbindung zum Anbieter aufbauen kann, wird die Downloadanfrage direkt übertragen:

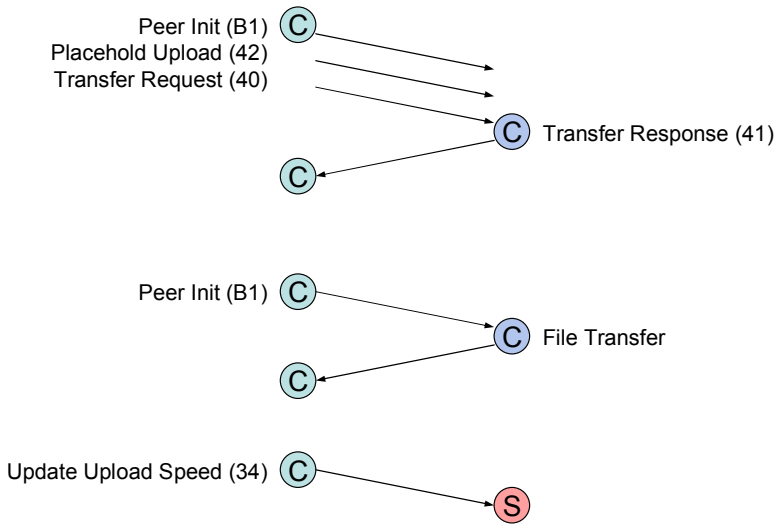


Abbildung 31: File Transfer

=> Peer Init (B1)
Der Client sendet ein Peer Init zur Initialisierung der Verbindung.
Der Verbindungstyp ist 'P' für Peer Connection.

=> Placehold Upload (42)
Diese Nachricht enthält den Dateinamen.

=> Transfer Request (40)
Der Client sendet eine Downloadanfrage mit dem Namen der Datei und einem Token.

<= Transfer Response (41)
Der Anbieter antwortet auf den Transfer Request. Die Nachricht enthält den Token der Downloadanfrage, die Größe der Datei sowie einem Byte ob der Download genehmigt ist oder nicht.

=> Peer Init (B1)
Wurde der Datei Austausch genehmigt, baut der Client eine Verbindung auf und initialisiert sie mit 'F' für den Verbindungstyp File Transfer.
Der Anbieter sendet daraufhin die Datei.

=> Update Upload Speed (34)
Ist der Download der Datei erfolgreich abgeschlossen, sendet der Client, der die Datei empfangen hat, eine Nachricht an den Server zurück um die mittlere Upload Geschwindigkeit in den Statistiken des Anbieters zu aktualisieren.

5.2.6 Firewall

Befindet sich ein Peer hinter einer Firewall, erweist sich die Kommunikation etwas schwieriger. Die Versuche eine Verbindung von außen aufzubauen, werden von der Firewall geblockt.

Um dennoch mit den Benutzer hinter einer Firewall kommunizieren zu können, muss also jede Verbindung von diesem Benutzer eingeleitet werden. Man sendet zuerst ein Nachricht an den Server, der dann den Peer beauftragt eine Verbindung aufzubauen.

Befinden sich beide Peers hinter einer Firewall, ist kein direkter Datenaustausch zwischen den beiden möglich, sie können nur über den Server miteinander kommunizieren.

Als Beispiel zeigt Abbildung 32 wie das Senden einer File Search Result Nachricht funktioniert, wenn der Suchende hinter einer Firewall ist:

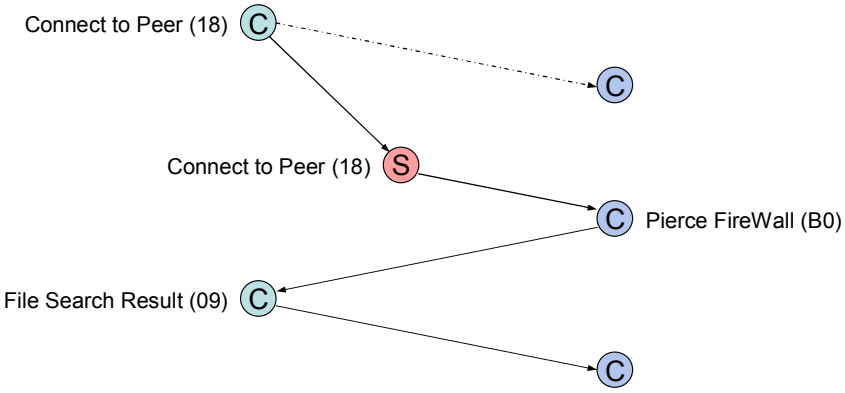


Abbildung 32: Kommunikation hinter einer Firewall

Der Client versucht zunächst eine direkte Verbindung zum suchenden Peer aufzubauen um dann wie in 5.2.2 eine Peer Init Nachricht zu senden, dies funktioniert jedoch nicht aufgrund der Firewall.

=> Connect to Peer (18)
Die Connect to Peer Nachricht enthält alle Informationen, die auch in der Peer Init Nachricht enthalten waren, also den Verbindungstyp und einen Token. Zusätzlich ist auch noch der Username von dem Peer enthalten, den der Server benachrichtigen soll.

=> Connect to Peer (18)
Der Server sendet die Nachricht an den im Username angegebenen Benutzer, und fügt zusätzlich noch den Username, die IP-Adresse sowie die Portnummer des Auftraggebers in die Nachricht ein.

<= Pierce FireWall (B0)
Der Peer sendet diese Nachricht an den Client zurück zum Verbindungsaufbau. Die Nachricht enthält den Token der Connect to Peer Nachricht.

=> File Search Result (09)
Der Client kann schließlich die Nachricht an den Peer senden.

Die anderen Nachrichtentypen werden meistens als Request an den Server oder an einen anderen Peer gesendet, die dann einen Reply zurücksenden. Reply Nachrichten vom Server haben den gleichen Message Code wie der Request, während bei Client-Client Nachrichten der Message Code des Reply um 1 höher ist als beim Request.
Client-Server Nachrichten:

- **Join/Add Room – Join Room User List (14)**
- **Room List Request (64)**
- **Check Privileges (92)**
- **Nachrichten des Recommendation-Systems (54), (56), (57), (110), (111), (112)**

Client-Client Nachrichten:

- **Get Shared File List (4), (5)**
- **User Info Request (15), (16)**
- **Folder Contents Request (36), (37)**
- **Transfer Request (40), (41)**

Es gibt auch Nachrichten, die vom Server an alle Peers im Netz oder in einem bestimmten Raum gesendet werden.

- **Say ChatRoom (13)**
- **User Joined Room (16)**
- **User Left Room (17)**
- **Room Added (62)**
- **Room Removed (63)**
- **Add To Privileged (91)**

Eine komplette Protokoll Spezifikation mit der Auflistung aller Nachrichtenformate ist in Anhang A angegeben.

6. Zusammenfassung und Ausblick

Es wurde gezeigt, dass die Peer-to-Peer Systemen der ersten Generation auf zwei grundverschiedenen Modellen basierten: Systeme mit einem zentralen Server und dezentrale Systeme.

Bei den Systemen mit einem zentralen Server verwaltet der Server die Benutzer und erstellt einen Index der angebotenen Dateien. Dadurch reagiert das Netz schnell bei Suchanfragen, da der Server gleich eine Liste der gefundenen Dateien zurücksenden kann. Der Server limitiert aber auch die Anzahl der Benutzer. Eine Vergrößerung des Netzes ist nur durch Optimierung des Servers möglich. Dieser stellt einen Single-point-of-failure dar, der eine Schließung des Netzes einfach ermöglicht.

Dezentrale Systeme Peer-to-Peer Systeme benutzen hingegen keinen Server. Jeder Client ist erfüllt gleichzeitig die Funktionalität eines Servers und ist ein so genannter Servent. Das Netz wird aufgebaut indem die Servents durch ping-Nachrichten neue Nachbarn suchen, mit denen sie sich verbinden können. Suchanfragen werden im Netz geflutet, sind aber auf einen festgelegten Radius limitiert. Somit ist es unmöglich das gesamte Netz zu durchsuchen. Der hohe Nachrichtenverkehr bei den Servents führt außerdem zu einer schlechten Skalierung. Die Netze sind aber sehr resistent gegen Ausfälle.

Die aktuellen Peer-to-Peer Tauschbörsen unterscheiden sich deutlich von denen der ersten Generation. Sie sind ausfallsicher geworden und skalieren besser, was in erster Linie darauf zurückzuführen ist, dass die meisten Programme eine hybride Netzwerkarchitektur verwenden. Die Zahl der Benutzer und der angebotenen Dateien steigt stetig.

Das FastTrack Netz fügt automatisch neue SuperPeers hinzu und skaliert somit fast unbegrenzt. Die SuperPeers verwalten nur ein paar Hundert Benutzer. Sie sind miteinander verbunden und leiten Suchanfragen an andere SuperPeers weiter. Eine Suche über das gesamte Netz ist aber nicht mehr möglich.

Im eDonkey Netz gibt es mehrere dedizierte Server, die mehrere Hundertausende Benutzer verwalten können. Sie sind nicht untereinander verbunden. Zur Ausweitung der Suche muss der Client die Suchanfrage an mehrere Server senden.

Mit einem neu implementierten Dissektor für das SoulSeek Protokoll, wurde in Ethereal der Nachrichtentransfer von SoulSeek aufgezeichnet und die Nachrichtensequenzen analysiert. Der Dissektor kann die aufgezeichneten Pakete desegmentieren sowie deren Inhalt dekomprimieren. Durch Reverse-Engineering ist der Aufbau der Nachrichtenformate und die Bedeutung der enthaltenen Datenfelder untersucht worden. Die Informationen werden angezeigt und man kann nach Datenfeldern filtern.

Durch Analyse des Nachrichtenverkehrs wurde festgestellt, dass das SoulSeek Netz, obwohl es auf einem zentralen Server beruht, spezielle Peers als Parent erklärt und die Suche über diese Parents geflutet wird. Der Server skaliert somit zwar besser, weil er keinen Index der angebotenen Dateien verwalten muss, wird aber durch ein Recommendation System dennoch stark belastet.

Ausblick

Peer-to-Peer Systeme werden längst nicht mehr nur für File-Sharing eingesetzt. Das Konzept, das die freien Ressourcen am Rand der Netze benutzt, um an einem gemeinsamen Ziel vereint zu arbeiten, wird heute auch bei Instant Messaging – zum schnellen Austausch von Nachrichten oder Groupware – zur Koordination von Arbeitsgruppen angewendet. Ein anderes Beispiel ist Grid-Computing, das die freien Rechenzyklen der Peers benutzt, um an einem gemeinsamen Projekt zu arbeiten, sinnvoll, wie im Fall der AIDS-Bekämpfung oder nur zum Spaß als verteiltes Aquarium.

Heute ist die Zukunft von Peer-to-Peer Tauschbörsen trotzdem ungewiss, auch wenn die Systeme der neusten Generation viele Schwachstellen ihrer Vorfahren wegräumen konnten. Die meisten Systeme sind nicht mehr auf einen zentralen Server angewiesen und haben daher keinen Single-point-of-failure, der z.B. das einfache Abschalten von Napster ermöglichte. Dennoch skalieren die aktuellen Netze besser, ohne aber zu einer hohen Netzlast bei den Peers zu führen, was zu großen Problemen bei Gnutella führte.

Da die File-Sharing Programme benutzt werden um meistens illegalen Inhalt zu tauschen, engagieren sich die Rechteinhaber dieser Daten aber mehr denn je, und reagieren immer häufiger mit rechtlichen Konsequenzen. Besonders die amerikanische Musik- und Filmindustrie sehen ihre Felle davonschwimmen und machen das illegale Tauschen verantwortlich für sinkende Einnahmen beim Verkauf von CD's und DVD's.

Sharman Networks, die Betreiberfirma von FastTrack, wurde schon mehrmals in Amerika angeklagt, obwohl sich der Firmensitz jetzt in einem pazifischen Inselstaat befindet. Die Firma versucht jetzt auf legalem Weg, mit dem Verkauf von rechtlich geschützten Daten, die über das FastTrack Netz verteilt werden eine neue Zukunft für KaZaA zu finden und bietet seit kurzen mit Skype eine P2P-Telefonie-Software an.[78] Auch mehrere Betreiber von eDonkey Server wurden von der Antipiraterie-Firma Retspan [79] verwarnt und müssten, laut Retspan, in den USA sogar mit Strafen von mehreren Millarden Dollar rechnen.[80] Einzig SoulSeek blieb bisher von rechtlichen

Attacken verschont. Eine Plattenfirma forderte aber schon das Filtern der Suchanfragen, was aber technisch nicht möglich sei, so der Gründer von SoulSeek Nir Abel. Dieser hat aber schon deutlich gemacht, dass er im Falle rechtlicher Maßnahmen aufgeben würde und den Quellcode veröffentlichen würde.

Überdies versucht die Musikindustrie die Tauschbörsen zu sabotieren, indem sie z.B. SuperNodes ins Netz stellen, die falsche Informationen verbreiten, Viren versenden die Fehler in der Programmierung der Clients ausnutzten, die PC's der Benutzer hacken oder versuchen mit Suchanfragen das Netz zu überfluten. [85]

Aber selbst wenn die Tauschbörsen nicht abgeschaltet werden können, so bleibt das Tauschen von rechtlich geschützten Daten doch illegal. Um die Millionen Benutzer darauf aufmerksam zu machen, hat die Musikindustrie in den Vereinigten Staaten in der bis Dato größten Anklagewelle hunderte von KaZaA Benutzer angeklagt und droht mit mindestens 750 US-Dollar Schadensersatz pro illegal angebotenem Song.[81] Dies kann man jedoch nur als abschreckende Maßnahme betrachten, da es praktisch unmöglich ist alle Benutzer die illegal Musik tauschen zu verklagen.[76]

Dreieinhalb Jahre nach dem Durchbruch von Napster kann man jetzt mit Napster 2 [77] und Apples iTunes erstmals legale Musik aus dem Internet herunterladen, jedoch sind die Dateien DRM-geschützt (Digital Rights Management).[84] Besonders bei Napster 2 ist die Benutzung der Musikdateien sehr stark eingeschränkt: Die Musikdateien lassen sich nur 30 Tage lang auf maximal 3 Windows-PC's abspielen und können nicht auf CD gebrannt werden.[83] Aber auch viele CD's die man im Musikladen kaufen kann sind mit einem Kopierschutz versehen und so gibt es die skurrile Situation, dass der Käufer die gerade gekaufte CD nicht im Autoradio hören kann oder auf seinen mp3-Player überspielen kann.

Die Benutzer von mp3 Dateien sind an die vielfältigen Nutzungsmöglichkeiten von diesem ungeschützten Format gewöhnt und können nur schwer diese Einschränkungen akzeptieren. Des Weiteren ist das Angebot der illegalen Dateien viel größer und so laden viele weiterhin Musik kostenlos von den Tauschbörsen herunter. Darauf setzt Sharman Networks und hat einen Vorschlag unterbreitet zukünftig Downloads lizenzierter Musikdateien in den P2P Netzen automatisch zu erkennen, die dann mit der Rechnung des ISP (Internet Service Provider) bezahlt werden können.[82]

Doch auch wenn das Tauschen von illegalen Dateien verhindert werden kann, so bleiben File-Sharing Programme eine interessante Alternative, um Dateien schnell zu verbreiten, ohne dass ein teuerer Server zum Upload benötigt wird. In den Tauschbörsen findet man deshalb auch legale Dateien, wie zum Beispiel Linux-Distributionen als CD-Image, Trailer von Kinofilmen oder freie Musikalben von Gruppen, die auf diesem Weg versuchen möglichst viele Hörer zu erreichen.

Peer-to-Peer Netze sind heute nicht mehr vom Internet wegzudenken und auch wenn das Tauschen in den File-Sharing Netzen eingeschränkt werden sollten, so bleiben diese doch die meistgenutze Anwendeung dieser Technik.

Offene Fragen

Die Peer-to-Peer Netze werden ständig weiterentwickelt und die Protokolle erweitert. Gerade bei SoulSeek, werden immer neue Funktionen implementiert und fast monatlich erscheint eine neue Version des Clients. Der in dieser Arbeit implementierte Dissektor kann somit nur den augenblicklichen Zustand wiedergeben.

Da die offiziellen Protokollspezifikationen nicht veröffentlicht wurden, ist es fast unmöglich die Bedeutung aller existierenden Nachrichtentypen durch Reverse-Engineering herauszufinden. So gibt es in den hier beschriebenen Protokollspezifikationen noch einige Nachrichtentypen sowie Datenfelder deren Bedeutung unklar ist.

Des Weiteren gibt es noch keinen Dissektor für das FastTrack Protokoll, obwohl FastTrack heute das wohl meistgenutzte File-Sharing Netz ist. Doch es ist bislang nur wenig über das Protokoll bekannt und die Nachrichten sind dazu noch verschlüsselt.

Literatur

Internetseiten der Clients

FastTrack

[1] **Sharman Networks**
http://www.sharmannetworks.com/

[2] **Kazaa Media Desktop**
http://www.kazaa.com/

[3] **Kazaa Lite**
http://www.kazaalite.tk/

[4] **iMesh**
http://www.imesh.com/

[5] **Grokster, p2p or person to person file sharing**
http://www.grokster.com/

[6] **giFT: Internet File Transfer**
http://gift.sourceforge.net/

[7] **giFT-FastTrack - BerliOS Developer: Project Info**
http://developer.berlios.de/projects/gift-fasttrack/

eDonkey

[8] **eDonkey2000 - Overnet**
http://www.edonkey2000.com/

[9] **Overnet**
http://www.overnet.com/

[10] **eMule-Project.net - Offizielle Homepage von eMule. Mit Downloads,Hilfe,Doku,Neuigkeiten...**
http://www.emule-project.net/

[11] **eMule Plus - The eye candy eMule Mod**
http://emuleplus.sourceforge.net/

[12] **eMule-MODs.de - eMule MOD's InfoSeite Download**
http://www.emule-mods.de/

[13] **xMule.org - Home of the xMule P2P Client**
http://unthesis.web.aplus.net/index.php

[14] **mldonkey**
http://www.nongnu.org/mldonkey/

[15] **Lugdunum - edonkey/emule servers**
http://lugdunum2k.free.fr/index.html

[16] **Edonkey2000-Switzerland - Site dédié au réseau eDonkey / Razorback**
http://www.ed2k.ch/

[17] **TDN - TheDonkeyNetwork**
http://tdn.no-ip.org/pl/tdn.pl

[18] **edonkey2000 serverlist by ocbMaurice**
http://www.ocbmaurice.com/slist/serverlist.html.gz

SoulSeek

[19] **Soulseek**
http://www.slsknet.org/

[20] **PySoulSeek**
http://www.sensi.org/~ak/pyslsk/

[21] **SoleSeek Project**
http://soleseek.sourceforge.net/

[22] **Nicotine**
http://nicotine.thegraveyard.org/

Ethereal

[23] **The Ethereal Network Analyzer**
http://www.ethereal.com/

P2P News Portale

[24] **Slyck - File Sharing News and Info**
http://www.slyck.com/

[25] **Open-files.com | Aides, Tutoriaux emule edonkey overnet |**
http://www.open-files.com/news/index0.htm

[26] **Peer-to-peer - Wikipedia**
http://en.wikipedia.org/wiki/Peer-to-peer

[27] **@-Web - Napster und Co. - P2P Netze. Die andere Art von Suchmaschinen**
http://p2p.at-web.de/p2p-dienste.htm

[28] **Kefk-Network - File Sharing-Tools nach Netzwerken**
http://www.kefk.net/P2P/Spektrum/Teilen/File-Sharing/Netzwerke/index.asp

[29] **OpenP2P.com -- p2p development, open source development**
http://www.openp2p.com/

[30] The File Sharing Portal - p2p programs reviews and downloads, file sharing forums
http://www.zeropaid.com/

[31] p2pnet.net - The original digital media and p2p news site
http://p2pnet.net/ez/index.php/news/

Literaturreferenzen

[32] Peer-to-Peer: Harnessing the Power of Disruptive Technologies - Chapter 1: A Network of Peers - Andy Oram
http://www.oreilly.com/catalog/peertopeer/chapter/ch01.html

[33] OpenP2P.com: What Is P2P ... And What Isn't - Clay Shirky [Nov. 24, 2000]
http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html

[34] ZDNet.de - Peer-to-Peer: Die Erneuerung des verteilten Rechnens
http://www.zdnet.de/itmanager/tech/0,39023442,2107183-3,00.htm

[35] Why Gnutella Can't Scale. No, Really.
http://www.darkridge.com/~jpr5/doc/gnutella.html

[36] Gnutella: To the Bandwidth Barrier and Beyond
http://web.archive.org/web/20010316013718/www.clip2.com/gnutella.html

[37] OpenP2P.com: Distributed Systems Topologies: Part 1 [Dec. 14, 2001]
http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html

[38] E-Business - Wissenspool - Artikel: Peer-to-Peer - Anwendungsbereiche und Herausforderungen
http://www.netskill.de/ebusiness.nsf/0/6ae6ecb8a739c970c1256c7c004a2169?OpenDocument

[39] Peer-to-Peer Anwendungsbereiche und Herausforderungen - Detlef Schoder und Kai Fischbach
http://www.netskill.de/ebusiness.nsf/6AE6ECB8A739C970C1256C7C004A2169/$File/e-business_schoder%20-%20p2p%20anwendungsbereiche.pdf

[40] ICQ instant messenger, chat, people search and messaging service!
http://www.icq.com/

[41] AOL (R) Instant Messenger (TM)
http://www.aim.com/

[42] MSN Messenger
http://messenger.msn.com/

[43] Jabber: Open Instant Messaging and a Whole Lot More
http://www.jabber.org/

[44] Yahoo! Messenger
http://messenger.yahoo.com/

[45] Skype
http://www.skype.com/

[46] Groove Networks
http://www.groove.net/

[47] Heise News-Ticker: Peer-to-Peer-Netzanwendung von Notes-Guru Ray Ozzie
http://www.heise.de/newsticker/data/jk-25.10.00-006/

[48] Groove updates peer-to-peer software | CNET News.com
http://news.com.com/2100-1001-880508.html

[49] SETI@home: Search for Extraterrestrial Intelligence at home
http://setiathome.ssl.berkeley.edu/

[50] Mersenne Prime Search
http://www.mersenne.org/prime.htm

[51] FightAIDS@Home
http://fightaidsathome.scripps.edu/

[52] The Internet Movie Project
http://www.imp.org/

[53] MoneyBee
http://uk.moneybee.net/

[54] DALiWorld
http://www.daliworld.net/index.html

[55] Download.com - Kazaa Media Desktop (International)
http://download.com.com/3000-2166-10049974.html?part=kazaa&subj=dlpage

[56] Fasttracks wundersame Wandlungen
http://www.heise.de/tp/deutsch/inhalt/musik/12592/1.html

[57] TP: Sieg für Grokster und Morpheus
http://www.heise.de/tp/deutsch/inhalt/musik/14685/1.html

[58] Heise News-Ticker: Industrie verklagt Musiktauschbörse iMesh
http://www.heise.de/newsticker/data/mw-20.09.03-006/

[59] iA Wiki: KaZaA
http://www.infoanarchy.org/wiki/wiki.pl?KaZaA

[60] SourceForge.net: Top Downloads
http://sourceforge.net/top/toplist.php?type=downloads

[61] Statistique du nombre d'utilisateurs et de fichiers partagés sur Razorback
http://www.ed2k.ch/modules/freecontent/index.php?id=4

[62] eDonkey2000 - What is MFTP (multisource file transmission protocol)?
http://www.edonkey2000.com/documentation/mftp.html

[63] eMule FAQ :: Kredit System
http://www.emule-project.net/faq/de_credits.htm

[64] ShareReactor
http://www.sharereactor.com/

[65] Saugstube die beste Eselseite im Netz
http://www.saugstube.to/

[66] Soulseek records
http://www.soulseekrecords.com/

[67] phlow.net - Magazin für Musik, Blog- und Netzkultur, Design, Pop, Film und Technologie: Interview mit Soulseek-Gründer Nir Abel
http://www.phlow.net/archives/000181.html

[68] Das Deutsche Soulseek-Portal
http://www.soulseek.de.vu/

[69] Deutsches Soulseekportal
http://slsk.mip.mikemth.de/slsk/index.php?site=faqof

[70] MP3newswire.net - SoulSeek review
http://www.mp3newswire.net/stories/2002/soulseek.html

[71] Dachboden-WG : Soulseek Forum
http://www.dachboden-wg.de/forum/viewforum.php?f=7

[72] Linux NetMag - FastTrack Netzwerke unter Linux
http://www.linuxnetmag.com/de/issue9/m9fasttrack1.html

[73] The Register - eDonkey rides like the wind in P2P protocol races
http://www.theregister.com/content/22/33385.html

[74] Slyck News - The Rise of eDonkey2000 - Good News for P2P
http://www.slyck.com/news.php?story=279

[75] Heise News-Ticker: Napster 2 plant gemischtes Bezahlmodell
http://www.heise.de/newsticker/data/anw-28.07.03-000/

[76] RIAA will take 2191.78 years to sue everyone
http://www.theinquirer.net/?article=10733

[77] Napster 2.0
http://www.napster.com

[78] Heise News-Ticker: P2P-Telefonie von den Kazaa-Entwicklern
http://www.heise.de/newsticker/data/vza-02.09.03-000/

[79] RetSpan
http://www.retspan.info/

[80] Heise News-Ticker: Antipiraterie-Firma verwarnt Edonkey-Server
http://www.heise.de/newsticker/data/jk-12.06.03-000/

[81] Heise News-Ticker: USA: Neue Klagedrohungen gegen Musiktauscher
http://www.heise.de/newsticker/data/hag-18.10.03-001/default.shtml

[82] Kazaa backs plan that could spell an end to the days of free music
http://www.theage.com.au/articles/2003/10/10/1065676130907.html

[83] Heise News-Ticker: Napster 2.0 gestartet
http://66.102.11.104/search?q=cache:3qcDPgWI4koJ:www.heise.de/newsticker/data/anw-29.10.03-002/

[84] Heise News-Ticker: Apple macht Musik
http://www.heise.de/newsticker/data/jk-28.04.03-011/

[85] Wired 11.09: START
http://www.wired.com/wired/archive/11.09/start.html?pg=12

Protocol Spezifikationen

KaZaA

[86] The FastTrack Protocol - giFT-FastTrack
http://cvs.berlios.de/cgi-bin/viewcvs.cgi/*checkout*/gift-fasttrack/giFT-FastTrack/PROTOCOL?rev=1.3

eDonkey

[87] eDonkey Protocol v0.6 - 2002 Alexey Klimkin
http://savannah.nongnu.org/download/mldonkey/docs/Edonkey-Overnet/edonkey-protocol.txt

[88] E-Donkey Protokoll / OpenDonkey - hacked by w3seek .:: RAZER 2000 ::.
http://www.w3seek.de/opendonkey/

SoulSeek

[89] **SoulSeek Protocol Documentation – Project SoleSeek**
http://cvs.sourceforge.net/viewcvs.py/*checkout*/soleseek/SoleSeek/doc/protocol.html?rev=HEAD&content-type=text/html

weitere P2P Programme

WinMX - The best way to share your media
http://www.winmx.com/

KLassphere DICE
http://exe.adam.ne.jp/dice/

BearShare - The World's Best Gnutella Client
http://www.bearshare.com/

XoloX - Client
http://www.xolox.nl/client/index.html

FREE Grokster for p2p or person to person file sharing
http://www.grokster.com/

Direct Connect - NeoModus
http://www.neo-modus.com/

Gnucleus
http://www.gnucleus.net/

LimeWire: The Most Sophisticated File-Sharing Application
http://www.limewire.com/

Morpheus - MusicCity.com
http://www.musiccity.com/

audioGnome
http://www.audiognome.com/

P h e x
http://phex.kouk.de/

URLBlaze: the world's first URL Sharing Network!
http://www.urlblaze.net/

Diet Kaza: Supercharging KaZaA!
http://www.dietk.com/

Piolet - The 3rd Generation Peer-to-Peer Filesharing Client for free music downloads - Download music
http://www.piolet.com/

PinPost | Buy and Sell | Peer-to-Peer Network | Download
http://www.pinpost.com/

Gnotella
http://www.gnotella.com/

Ares - Softgap - Introducing Ares and peer to peer technology
http://www.softgap.com/

Blubster - Share Your Music
http://www.blubster.com/

Nova
http://novap2p.sourceforge.net/?do=about

NapMX Homepage
http://perso.wanadoo.fr/speedfire/napmx/index.html

Shareaza
http://www.shareaza.com/

.: FileNavigator :.
http://www.filenavigator.com/

songspy.com
http://www.songspy.com/

FileShare's website
http://www.blueboxnetworks.com

ShareMonkey.com - Free fileshare movie download links from P2P networks including Kazaa, plus Speedup software
http://www.zope.org.uk/sharemonkey/index.php

Locutus :: overview
http://locut.us/

Taxee Homepage
http://www.boolean-technologies.com/taxeepeng.php

Filetopia: Your secure file sharing and communications tool
http://www.filetopia.org/

FileAngel - SourceForge.net: Project Info
http://sourceforge.net/projects/fileangel/

MusIRC
http://musirc.sourceforge.net/

Frost - Freenet messaging and filesharing
http://jtcfrost.sourceforge.net/

Myster - Networks
http://www.mysternetworks.com/

The Freenet Project - quickstart - beginner
http://freenetproject.org/qcms/?page=quickstart

GLT Poliane
http://www.gltpoliane.cjb.net/baixar.php?POLIANE_ID=615de3c7d7949f224517e3dc1972bebc

Andromeda Streaming Jukebox: MP3 server for PHP
http://www.turnstyle.com/andromeda/home.asp

Clustone
http://www.clustone.com/en/home.html

Napigator
http://www.napigator.com/

PeerGuardian - XS P2P
http://xs.tech.nu/

File Rogue - FAQ
http://www.filerogue.com/faq.htm

BadBlue - peer-to-peer (P2P) file sharing web servers for home and business
http://www.badblue.com/

Gnutmeg - SourceForge.net: Project Info
http://sourceforge.net/projects/gnutmeg/

Phosphor File Sharing
http://phosphor.sourceforge.net/

Toadnode LLC
http://www.toadnode.com/

MyNapster
http://www.mynapster.com/

Sexter! The Adult Entertainment Network
http://www.sexter.com/

Voodoo Vision
http://www.voodoovision.com/

FileEx Homepage
http://www.piragissystems.com/fileex.htm

Napster
http://www.napster.com/

konspire2b: a revolution in mass-scale content distribution
http://konspire.sourceforge.net/

File Miner - Deep Forest Software
http://192.216.113.29/deepforest/miner/miner.htm

Wrapster
http://www.unwrapper.com/

The next revolution in P2P file sharing
http://www.es5.com/

The Freenet Project - index - beginner
http://freenetproject.org/

Abbildungsverzeichnis

Anhang A
SoulSeek Protokoll Spezifikation

Die folgenden Konstanten sind Datentypen, die in den Nachrichten als Byte oder als String der Länge 1 definiert sind:

```
  Status Code:
         -1      "Unknown"
         0       "Offline"
         1       "Away"
         2       "Online"

  Transfer Direction:
         0       "Download"
         1       "Upload"

  Attribute Type:
         0       "Bitrate"
         1       "Length"
         2       "VBR"


  Connection Type:
         D       "Distributed Search"
         P       "Peer Connection"
         F       "File Transfer"
```

Client-Server Nachrichten

Message Code 1:

```
=> Client-to-Server
Message Type: "Login"
  int    1       Message Code
  string         Username
```

```
  string          Password
  int             Version
```
Login-Nachricht

```
<= Server-to-Client
Message Type: "Login Reply"
  int    1        Message Code
  byte            Login successfull (1/0)
  string          Login Message
         int           Client IP      (if Login successfull = 1)
```
Antwort auf die Login-Nachricht

Message Code 2:

```
=> Client-to-Server
Message Type: "Set Wait Port"
  int    2        Message Code
  int             Port Number
```
Senden der Portnummer beim Login

Message Code 3:

```
=> Client-to-Server
Message Type: "Get Peer Address"
  int    3        Message Code
  string          Username
```
Anforderung der IP-Adresse eines Peers

```
<= Server-to-Client
Message Type: "Get Peer Address Reply"
  int    3        Message Code
  string          Username
  int             IP Address
  int             Port Number
```
IP-Adresse eines Peers

Message Code 5:

```
=> Client-to-Server
Message Type: "User Exists Request"
  int    5        Message Code
  string          Username
```
Test ob der Benutzer existiert

```
<= Server-to-Client
Message Type: "User Exists Reply"
  int    5        Message Code
  string          Username
  byte            user exists (1/0)
```
Antwort auf die obige Nachricht

Message Code 7:

```
=> Client-to-Server
Message Type: "Get User Status"
```

```
   int     7       Message Code
   string          Username
```
Anforderung des Status eines Benutzers

```
<= Server-to-Client
Message Type: "Get User Status Reply"
   int     7       Message Code
   string          Username
   int             Status Code
```
Status eines Benutzers (Status Code siehe oben)

Message Code 13:

```
=> Client-to-Server
Message Type: "Say ChatRoom"
   int     13      Message Code
   string          Room
   string          Chat Message
```
Senden einer Nachricht im Chatraum

```
<= Server-to-Client
Message Type: "Say ChatRoom"
   int     13      Message Code
   string          Room
   string          Username
   string          Chat Message
```
Empfang einer Nachricht im Chatraum

Message Code 14:

```
=> Client-to-Server
Message Type: "Join/Add Room"
   int     14      Message Code
   string          Room
```
Betreten eines Raums oder Erstellen eines neuen Raums

```
<= Server-to-Client
Message Type: "Join Room User List"
   int     14      Message Code
   string          Room
   int     U       Users in Room
   for each User (1 ≤ u ≤ U)
          string       User #u Username
   int     U       Users in Room
   for each User (1 ≤ u ≤ U)
          string       User #u Status Code
   int     U       Users in Room
   for each User (1 ≤ u ≤ U)
          int          User #u Average Speed
          int          User #u Downloadnum
          int          User #u Integer
          int          User #u Files
          int          User #u Folders
   int     U       Users in Room
   for each User (1 ≤ u ≤ U)
          int          User #u Slots full
```
Liste der Benutzer in einem Raum mit Statistiken, als Antwort auf die Join Room Nachricht

Message Code 15:

```
=> Client-to-Server
<= Server-to-Client
Message Type: "Leave Room"
   int    15      Message Code
   string         Room
```

Verlassen eines Raums, wird vom Server an alle im Raum befindlichen Benutzer gesendet

Message Code 16:

```
<= Server-to-Client
Message Type: "User Joined Room"
   int    16      Message Code
   string         Room
   string         Username
   int            Total uploads allowed
   int            Average Speed
   int            Download Number
   int            Integer
   int            Files
   int            Directories
   int            Slots full
```

Neuer Benutzer betritt einen Raum, mit Statistiken des Benutzers, wird vom Server an alle im Raum befindlichen Benutzer gesendet

Message Code 17:

```
<= Server-to-Client
Message Type: "User Left Room"
   int    17      Message Code
   string         Room
   string         Username
```

Benutzer verlässt einen Raum, wird vom Server an alle im Raum befindlichen Benutzer gesendet

Message Code 18:

```
=> Client-to-Server
Message Type: "Connect To Peer"
   int    18      Message Code
   int            Token
   string         Username
   string         Connection Type (P/D/F)
```

Anforderung an den Server, dem Peer mitzuteilen eine Verbindung eines bestimmten Typs zum eigenen Client herzustellen, weil der Peer z.B. wegen einer Firewall nicht direkt erreichbar ist

```
<= Server-to-Client
Message Type: "Connect To Peer"
   int    18      Message Code
   string         Username
   string         Connection Type (P/D/F)
   int            IP Address
   int            Port Number
   int            Token
```

Anforderung vom Server eine Verbindung eines bestimmten Typs zu einem Peer herzustellen, weil der Peer z.B. wegen einer Firewall nicht direkt eine Verbindung zum eigenen Client aufbauen konnte

Message Code 22:

```
=> Client-to-Server
Message Type: "Message User Send"
   int     22       Message Code
   string           Username
   string           Chat Message
```
Senden einer privaten Nachricht an einen Benutzer

```
<= Server-to-Client
Message Type: "Message User Receive"
   int     22       Message Code
   int              Chat Message ID
   int              Timestamp
   string           Username
   string           Chat Message
```
Empfang einer privaten Nachricht von einem Benutzer

Message Code 23:

```
=> Client-to-Server
Message Type: "Message User Receive Ack"
   int     23       Message Code
   int              Chat Message ID
```
Bestätigung des Empfangs einer privaten Nachricht, als Antwort auf Nachricht 22

Message Code 26:

```
=> Client-to-Server
Message Type: "File Search"
   int     26       Message Code
   int              Token
   string           Search Text
```
Suchanfrage

Message Code 28:

```
=> Client-to-Server
Message Type: "Set Status"
   int     28       Message Code
   int              Status Code
```
Änderung des Status, zB. away (Status Codes siehe oben)

Message Code 32:

```
=> Client-to-Server
Message Type: "Ping"
   int     32       Message Code
```
Ping-Nachricht an den Server

Message Code 34:

```
=> Client-to-Server
Message Type: "Update Upload Speed"
   int     34       Message Code
   string           Username
   int              Average Speed
```

Update des Upload-Speed in den Statistiken eines Benutzers, wird nach erfolgreichem Download gesendet

Message Code 35:

```
=> Client-to-Server
Message Type: "Shared Files & Folders"
   int     35       Message Code
   int              Folder Count
   int              File Count
```

Anzahl der angebotenen Dateien und Verzeichnissen, wird beim Login gesendet

Message Code 36:

```
=> Client-to-Server
Message Type: "Get User Stats"
   int     36       Message Code
   string           Username
```

Anforderung der Statistiken eines Benutzers, wird nach dem Login gesendet, um die eigenen Statistiken zu erhalten

```
<= Server-to-Client
Message Type: "Get User Stats Reply"
   int     36       Message Code
   string           Username
   int              Average Speed
   int              Download Number
   int              Integer
   int              Files
   int              Directories
```

Statistiken eines Benutzers, als Antwort auf obige Nachricht

Message Code 40:

```
<= Server-to-Client
Message Type: "Queued Downloads"
   int     40       Message Code
   string           Username
   int              Slots full
```

Anzahl der belegten Upload-Slots

Message Code 50:

```
=> Client-to-Server
Message Type: "Make Own Recommendation"
   int     50       Message Code
   string           Recommendation
```

Hinzufügen einer neuen Recommendation von einem Benutzer

```
=> Client-to-Server
Message Type: "Remove Own Recommendation"
   int     50       Message Code
   string           Recommendation
   int              Ranking
```
Entfernen einer neuen Recommendation von einem Benutzer

Message Code 51:

```
=> Client-to-Server
Message Type: "Add Things I like"
   int     51       Message Code
   string           Recommendation
```
Hinzufügen einer bekannten Recommendation in den Vorlieben eines Benutzer

Message Code 52:

```
=> Client-to-Server
Message Type: "Remove Things I like"
   int     52       Message Code
   string           Recommendation
```
Entfernen einer bekannten Recommendation in den Vorlieben eines Benutzer

Message Code 54:

```
=> Client-to-Server
Message Type: "Get Recommendations"
   int     54       Message Code
```
Anforderung einer Liste von Recommendations

```
<= Server-to-Client
Message Type: "Get Recommendations Reply"
   int     54       Message Code
   int     R        Number of Recommendations
   for each Recommendation (1 ≤ r ≤ R)
           string       Recommendation
           int          Ranking
```
Liste von Recommendations, als Antwort auf obige Nachricht

Message Code 55:

```
=> Client-to-Server
Message Type: "Type 55"
   int     55       Message Code
```
Unbekannte Nachricht

Message Code 56:

```
=> Client-to-Server
Message Type: "Get Global Rankings"
   int     56       Message Code
```
Anforderung der Liste aller bekannten Recommendations

```
<= Server-to-Client
Message Type: "Get Global Rankings Reply"
   int     56       Message Code
   int     R        Number of Recommendations
   for each Recommendation (1 ≤ r ≤ R)
           string       Recommendation
           int          Ranking
```
Liste aller bekannten Recommendations, als Antwort auf obige Nachricht

Message Code 57:

```
=> Client-to-Server
Message Type: "Get User Recommendations"
   int     57       Message Code
   string           Username
```
Anforderung der Liste der Recommendations eines Benutzers

```
<= Server-to-Client
Message Type: "Get User Recommendations Reply"
   int     57       Message Code
   string           Username
   int     R        Number of Recommendations
   for each Recommendation (1 ≤ r ≤ R)
           string       Recommendation
```
Liste der Recommendations eines Benutzers, als Antwort auf obige Nachricht

Message Code 58:

```
=> Client-to-Server
Message Type: "Admin Command"
   int     58       Message Code
   string           String
   int     S        Number of Strings
   for each String (1 ≤ s ≤ S)
           string       String
```
Spezielle Nachricht für Kommandos des Administrators

Message Code 60:

```
=> Client-to-Server
<= Server-to-Client
Message Type: "Place In Line Response"
   int     60       Message Code
   string           Username
   int              Token
   int              Place in Queue
```
Erfragung der Stelle in der Warteschlange

Message Code 62:

```
<= Server-to-Client
Message Type: "Room Added"
   int     62       Message Code
   string           Room
```
Neuer Raum gegründet

Message Code 63:

```
<= Server-to-Client
Message Type: "Room Removed"
   int     63       Message Code
   string           Room
```
Ein Raum wurde entfernt

Message Code 64:

```
=> Client-to-Server
Message Type: "Room List Request"
   int     64       Message Code
```
Anforderung der Liste aller Räume

```
<= Server-to-Client
Message Type: "Room List"
   int     64       Message Code
   int     R        Number of Rooms
   for each Room (1 ≤ r ≤ R)
           string       Room
   int     R        Number of Rooms
   for each Room (1 ≤ r ≤ R)
           int          Users in Room #r
```
Liste der populärsten Räume wird beim Login übertragen
Komplette Liste aller Räume nach obiger Nachricht

Message Code 65:

```
=> Client-to-Server
Message Type: "Exact File Search"
   int     65       Message Code
   int              Token
   string           Filename
   string           Directory
   (+13 0 bytes)
```
Suche nach dem exakten Dateinamen, zum Suchen nach anderen Quellen

```
<= Server-to-Client
Message Type: "Exact File Search"
   int     65       Message Code
   string           Username
   int              Token
   string           Filename
   string           Directory
   (+12 0 bytes)
```
Anforderung auf obige Suche

Message Code 66:

```
<= Server-to-Client
Message Type: "Admin Message"
   int     66       Message Code
   string           Chat Message
```
Spezielle Nachricht vom Administrator

Message Code 67:

```
=> Client-to-Server
Message Type: "Global User List Request"
   int     67       Message Code
```
Anforderung der Liste der Benutzer in einem Raum

```
<= Server-to-Client
Message Type: "Global User List"
   int     67       Message Code
   string           Room
   int     U        Users in Room
   for each User (1 ≤ u ≤ U)
          string       User #u Username
   int     U        Users in Room
   for each User (1 ≤ u ≤ U)
          string       User #u Status Code
   int     U        Users in Room
   for each User (1 ≤ u ≤ U)
          int          User #u Average Speed
          int          User #u Downloadnum
          int          User #u Integer
          int          User #u Files
          int          User #u Folders
   int     U        Users in Room
   for each User (1 ≤ u ≤ U)
          int          User #u Slots full
```
Liste der Benutzer in einem Raum mit Statistiken, als Antwort auf obige Nachricht

Message Code 68:

```
Message Type: "Tunneled Message"
   int     68       Message Code
   string           Username
   int              Code
   int              Token
   int              IP Address
   int              Port Number
   string           Chat Message
```
Spezielle Nachricht zum Tunneln einer Chatnachricht

Message Code 69:

```
=> Client-to-Server
Message Type: "Privileged User List Request"
   int     69       Message Code
```
Anforderung der Liste aller privilegierten Benutzer

```
<= Server-to-Client
Message Type: "Privileged User List"
   int     69       Message Code
   int     P        Number of Priviledged Users
   for each Priviledged User (1 ≤ p ≤ P)
          string       Username #p
```
Liste aller privilegierten Benutzer, wird auch beim Login gesendet

Message Code 71:

```
=> Client-to-Server
Message Type: "Get Parent List"
   int    71      Message Code
   byte           0 to get Parent List
                  1 after receiving the list
```

Anforderung einer Liste von Parent Adressen

Message Code 73:

```
=> Client-to-Server
Message Type: "Type 73"
   int    73      Message Code
   int            Integer
```

Unbekannte Nachricht, wird nach dem Erhalt der Liste von Parent Adressen gesendet

Message Code 83:

```
<= Server-to-Client
Message Type: "Parent Min Speed"
   int    83      Message Code
   int            Parent Min Speed
```

Einstellung des Netzes: Minimal erforderliche Geschwindigkeit eines Parents

Message Code 84:

```
<= Server-to-Client
Message Type: "Parent Speed Connection Ratio"
   int    84      Message Code
   int            Parent Speed Connection Ratio
```

Einstellung des Netzes

Message Code 86:

```
<= Server-to-Client
Message Type: "Parent Inactivity Before Disconnect"
   int    86      Message Code
   int            Seconds Parent Inactivity Before Disconnect
```

Einstellung des Netzes: Parent Timeout

Message Code 87:

```
<= Server-to-Client
Message Type: "Server Inactivity Before Disconnect"
   int    87      Message Code
   int            Seconds Server Inactivity Before Disconnect
```

Einstellung des Netzes: Server Timeout

Message Code 88:

```
<= Server-to-Client
Message Type: "Nodes In Cache Before Disconnect"
   int     88       Message Code
   int              Nodes In Cache Before Disconnect
```

Einstellung des Netzes

Message Code 90:

```
<= Server-to-Client
Message Type: "Seconds Before Ping Children"
   int     90       Message Code
   int              Seconds Before Ping Children
```

Einstellung des Netzes: Children Timeout

Message Code 91:

```
<= Server-to-Client
Message Type: "Add To Privileged"
   int     91       Message Code
   string           Username
```

Neuer privilegierter Benutzer

Message Code 92:

```
=> Client-to-Server
Message Type: "Check Privileges"
   int     92       Message Code
```

Erfragung, wie lange man noch privilegiert ist

```
<= Server-to-Client
Message Type: "Check Privileges Reply"
   int     92       Message Code
   int              Number of Days
```

Antfort auf obige Nachricht

Message Code 93:

```
<= Server-to-Client
Message Type: "Embedded Distributed Search Message"
   int     93       Message Code
           byte     3    Message Code
           int           Integer
           string        Username
           int           Token
           string        Search Text
```

Eingekapselte Nachricht vom Server an die Parents zur Weiterleitung

Message Code 100:

```
=> Client-to-Server
Message Type: "Become Parent"
```

```
  int    100     Message Code
  byte           Byte
```
Client wird selber zum Parent, Erfordernisse werden erfüllt

Message Code 102:

```
<= Server-to-Client
Message Type: "Random Parent Addresses"
  int    102     Message Code
  int    P       Number of Parent Addresses
  for each Parent Address (1 ≤ p ≤ P)
         string      Username #p
         int         IP Address
         int         Port Number
```
Liste von Parent Adressen, als Antwort auf Nachricht 71

Message Code 103:

```
<= Server-to-Client
Message Type: "Send Wishlist Entry"
  int    103     Message Code
  int            Token
  string         Search Text
```
Senden der Wishlist Einträge als Suchanfrage, nach periodischer Zeit automatisch

Message Code 104:

```
<= Server-to-Client
Message Type: "Type 104"
  int    104     Message Code
  int            Integer
```
Unbekannt Nachricht

Message Code 110:

```
=> Client-to-Server
Message Type: "Get Similar Users"
  int    110     Message Code
```
Anforderung einer Liste von Benutzern mit den gleichen Vorlieben

```
<= Server-to-Client
Message Type: "Get Similar Users Reply"
  int    110     Message Code
  int    U       Number of Users
  for each User (1 ≤ u ≤ U)
         string      User #u Username
         int         User #u Number of same Recommendations
```
Liste von Benutzern mit den gleichen Vorlieben, als Antwort auf obige Nachricht

Message Code 111:

```
=> Client-to-Server
Message Type: "Get Recommendations for Item"
  int    111     Message Code
```

```
   string          Recommendation
```
Anforderung einer Liste von Recommendations, die Benutzer, die diese Recommendation in ihren Vorlieben haben ebenfalls vorschlagen

```
<= Server-to-Client
Message Type: "Get Recommendations for Item Reply"
   int     111     Message Code
   string          Recommendation
   int     R       Number of Recommendations
   for each Recommendation (1 ≤ r ≤ R)
          string       Recommendation #r
          int          Ranking #r
```
Liste von Recommendations, als Antwort auf obige Nachricht

Message Code 112:

```
=> Client-to-Server
Message Type: "Get Similar Users for Item"
   int     112     Message Code
   string          Recommendation
```
Anforderung einer Liste von Benutzern, die eine bestimmte Recommendation in ihren Vorlieben haben

```
<= Server-to-Client
Message Type: "Get Similar Users for Item Reply"
   int     112     Message Code
   string          Recommendation
   int     R       Number of Users
   for each User (1 ≤ u ≤ U)
          string       Username #u
```
Liste von Benutzern, die eine bestimmte Recommendation in ihren Vorlieben haben, als Antwort auf obige Nachricht

Message Code 1001:

```
=> Client-to-Server
Message Type: "Can't Connect To Peer"
   int     1001    Message Code
   int             Token
   string          Username
```
Verbindung zu einem Peer kann nicht aufgebaut werden

```
<= Server-to-Client
Message Type: "Can't Connect To Peer"
   int             Message Code
   int             Token
```
Verbindung zu einem Peer kann nicht aufgebaut werden

Client-Client Nachrichten

Message Code 4:

```
<=> Client-to-Client
Message Type: "Get Shared File List"
   int     4       Message Code
```

Anforderung der Liste aller angebotenen Dateien

Message Code 5:

```
<=> Client-to-Client
Message Type: "Shared File List"
   int    5       Message Code

   [zlib compressed packet]
   int    D       Number of Directories
   for each Directories (1 ≤ d ≤ D)
          string      Directory #d Name
          int     F   Directory #d Number of Files
          for each Files (1 ≤ f ≤ F)
                  byte     Directory #d File #f Code
                  string       Directory #d File #f Filename
                  int      Directory #d File #f Size1
                  int      Directory #d File #f Size2
                  string       Directory #d File #f Extension
                  int A    Directory #d File #f Number of Attributes
                  for each Attribute (1 ≤ a ≤ A)
                      int      Directory #d File #f Attribute #a Type
                      int      Directory #d File #f Attribute #a Value
```

Liste aller angebotenen Dateien eines Benutzers, als Antwort auf Nachricht 4

Message Code 9:

```
<=> Client-to-Client
Message Type: "File Search Result"
   int    9       Message Code

   [zlib compressed packet]
   string         Username
   int            Token
   int    F       Number of Files
   for each File (1 ≤ f ≤ F)
          byte        File #f Code
          string      File #f Filename
          int         File #f Size1
          int         File #f Size2
          string      File #f Extension
          int     A   File #f Number of Attributes
          for each Attribute (1 ≤ a ≤ A)
                  int      File #f Attribute #a Type
                  int      File #f Attribute #a Value
   int            Free upload slots
   int            Upload speed
   int            In Queue
```

Liste aller angebotenen Dateien, die den gesuchten Begriff enthalten, als Antwort auf Nachricht 26

Message Code 15:

```
<=> Client-to-Client
Message Type: "User Info Request"
   int    15      Message Code
```

Anforderung der Informationen eines Benutzers

Message Code 16:

```
<=> Client-to-Client
Message Type: "User Info Reply"
   int    16      Message Code
   string         User Description
   byte   P       Picture exists (1/0)
   if P = 1
          bytes       Picture
   int            Total uploads allowed
   int            Queued uploads
   byte           Upload Slots available (1/0)
```

Informationen eines Benutzers, als Antwort auf Nachricht 15

Message Code 36:

```
<=> Client-to-Client
Message Type: "Folder Contents Request"
   int    36      Message Code
   int            Token
   string         Directory
```

Anforderung der Liste des Inhalts eines Verzeichnisses

Message Code 37:

```
<=> Client-to-Client
Message Type: "Folder Contents Response"
   int    37      Message Code

   [zlib compressed packet]
   int            Token
   string         Directory Name
   int    D       Number of Directories
   for each Directories (1 ≤ d ≤ D)
          string      Directory #d Name
          int     F   Directory #d Number of Files
          for each Files (1 ≤ f ≤ F)
                 byte     Directory #d File #f Code
                 string       Directory #d File #f Filename
                 int      Directory #d File #f Size1
                 int      Directory #d File #f Size2
                 string       Directory #d File #f Extension
                 int A    Directory #d File #f Number of Attributes
                 for each Attribute (1 ≤ a ≤ A)
                     int      Directory #d File #f Attribute #a Type
                     int      Directory #d File #f Attribute #a Value
```

Liste des Inhalts eines Verzeichnisses, als Antwort auf Nachricht 36

Message Code 40:

```
<=> Client-to-Client
Message Type: "Transfer Request"
   int    40      Message Code
   int            Transfer Direction
   int            Token
   string         Filename
```

```
  if (Transfer Direction = 1)
        int         Size
        int         Integer
```
Anfrage zum Download einer Datei

Message Code 41:

```
<=> Client-to-Client
Message Type: "Transfer Response"
  int    41      Message Code
  int            Token
  byte           Download allowed (1/0)
  if (Download allowed = 1)
        int         Size
        int         Integer
  if (Download allowed = 0)
        string      String
```
Download erlaubt oder nicht, als Antwort auf Nachricht 40

Message Code 42:

```
<=> Client-to-Client
Message Type: "Placehold Upload"
  int    42      Message Code
  string         Filename
```
Wird vor der Nachricht 40 zur Anfrage eines Downloads gesendet

Message Code 43:

```
<=> Client-to-Client
Message Type: "Queue Upload"
  int    43      Message Code
  string         Filename
```
Erfragung der Stelle in einer Warteschlange

Message Code 44:

```
<=> Client-to-Client
Message Type: "Place In Queue"
  int    44      Message Code
  string         Filename
  int            Place in Queue
```
Stelle in einer Warteschlange, als Antwort auf Nachricht 51

Message Code 46:

```
<=> Client-to-Client
Message Type: "Upload Failed"
  int    46      Message Code
  string         Filename
```
Upload gescheitert

Message Code 50:

```
<=> Client-to-Client
Message Type: "Queue Failed"
   int     50       Message Code
   string           Filename
   string           String
```
Warteschlange gescheitert

Message Code 51:

```
<=> Client-to-Client
Message Type: "Place In Queue Request"
   int     51       Message Code
   string           Filename
```
Erfragung der Stelle in einer Warteschlange

Message Code Byte 0:

```
<=> Client-to-Client
Message Type: "Pierce Fw"
   byte    0        Message Code
   int              Token
```
Von einem Peer gesendet um eine Verbindung aufzubauen und so die Firewall zu unterwandern

Message Code Byte 1:

```
<=> Client-to-Client
Message Type: "Peer Init"
   byte    1        Message Code
   string           Username
   string           Connection Type (P/D/F)
   int              Token
```
Initialisierung einer Verbindung eines bestimmten Typs (Verbindungstypen siehe oben)

Message Code Byte 3:

```
<=> Client-to-Client
Message Type: "Distributed Search"
   byte    3        Message Code
   int              Integer
   string           Username
   int              Token
   string           Search Text
```
Weierlaitung von den Parents einer Suchanfrage des Typs 26

Anhang B
Beispiele des SoulSeek Dissektor Quellcodes

```
static const value_string slsk_tcp_msgs[] = {
    { 1, "Login"},
    { 2, "Set Wait Port"},
    { 3, "Get Peer Address"},
(...)
};

static gboolean check_slsk_format(tvbuff_t *tvb, int offset, char format[]){
  /*
  * Returns TRUE if tvbuff beginning at offset matches a certain format
  * The format is given by an array of characters standing for a special field type
  *    i - integer  (4 bytes)
  *    b - byte  (1 byte)
  *    s - string  (string_length + 4 bytes)
  *
  *    * - can be used at the end of a format to ignore any following bytes
  */
  char remaining_format[strlen(format)-1];
  int i =0;
  while (i < (int)strlen(format)){remaining_format[i] = format[i+1]; i++;}
  switch ( format[0] ) {
    case 'i':
      if (tvb_length_remaining(tvb, offset) < 4) return FALSE;
      offset += 4;
    break;
    case 'b':
      if (tvb_length_remaining(tvb, offset) < 1) return FALSE;
      offset += 1;
    break;
    case 's':
      if (tvb_length_remaining(tvb, offset) < 4) return FALSE;
      if (tvb_length_remaining(tvb, offset) < (int)tvb_get_letohl(tvb, offset)+4) return FALSE;
      offset += tvb_get_letohl(tvb, offset)+4;
    break;
    case '*':
      return TRUE;
    break;
    default:
      return FALSE;
    break;
  }
  if (remaining_format[0] == '\0' ) {
    if (tvb_length_remaining(tvb, offset) != 0) return FALSE;
      return TRUE;
  }
  return check_slsk_format(tvb, offset, remaining_format);
```

```
}

static char* get_message_type(tvbuff_t *tvb) {
 /*
 * Checks if the Message Code is known.
 * If unknown checks if the Message Code is stored in a byte.
 */
 int msg_code = tvb_get_letohl(tvb, 4);
 gchar *message_type =  val_to_str(msg_code, slsk_tcp_msgs, "Unknown");
 if (strcmp(message_type, "Unknown") == 0) {
   if (check_slsk_format(tvb, 4, "bisis"))   message_type = "Distributed Search";
   if (check_slsk_format(tvb, 4, "bssi"))   message_type = "Peer Init";
   if (check_slsk_format(tvb, 4, "bi"))   message_type = "Pierce Fw";
 }
 return message_type;
};


static void dissect_slsk_pdu(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree) {
(...)
 if (tree) {

/* creates display subtree for the protocol */
   ti = proto_tree_add_item(tree, proto_slsk, tvb, 0, -1, FALSE);
   slsk_tree = proto_item_add_subtree(ti, ett_slsk);

   proto_tree_add_uint(slsk_tree, hf_slsk_message_length, tvb, offset, 4, msg_len);
   offset += 4;

   switch (msg_code) {
(...)
     case 36:
      if (check_slsk_format(tvb, offset, "isiiiii")) {
       /* Server-to-Client */
        message_type = "Get User Stats Reply";
        proto_tree_add_uint_format(slsk_tree, hf_slsk_message_code, tvb, offset, 4, msg_code,
              "Message Type: %s (Code: %02d)", message_type, msg_code);
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_string_length, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        proto_tree_add_item(slsk_tree, hf_slsk_username, tvb, offset+4, tvb_get_letohl(tvb, offset), FALSE);
        offset += 4+tvb_get_letohl(tvb, offset);
        proto_tree_add_uint(slsk_tree, hf_slsk_average_speed, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_download_number, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_integer, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_files, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_directories, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        offset += 4;
      }
      else if (check_slsk_format(tvb, offset, "is")) {
       /* Client-to-Server */
        message_type = "Get User Stats";
        proto_tree_add_uint_format(slsk_tree, hf_slsk_message_code, tvb, offset, 4, msg_code,
              "Message Type: %s (Code: %02d)", message_type, msg_code);
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_string_length, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        proto_tree_add_item(slsk_tree, hf_slsk_username, tvb, offset+4, tvb_get_letohl(tvb, offset), FALSE);
        offset += 4+tvb_get_letohl(tvb, offset);
      }
      else if (check_slsk_format(tvb, offset, "iis")) {
       /* Client-to-Client */
        message_type = "Folder Contents Request";
        proto_tree_add_uint_format(slsk_tree, hf_slsk_message_code, tvb, offset, 4, msg_code,
              "Message Type: %s (Code: %02d)", message_type, msg_code);
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_token, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        offset += 4;
        proto_tree_add_uint(slsk_tree, hf_slsk_string_length, tvb, offset, 4, tvb_get_letohl(tvb, offset));
        proto_tree_add_item(slsk_tree, hf_slsk_directory, tvb, offset+4, tvb_get_letohl(tvb, offset), FALSE);
        offset += 4+tvb_get_letohl(tvb, offset);
      }
     break;
(...)

   }
  }
}
```

```
static void dissect_slsk(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree)
{
  tcp_dissect_pdus(tvb, pinfo, tree, slsk_desegment, 4, get_slsk_pdu_len, dissect_slsk_pdu);

}


(...)
/* Setup list of header fields  */
  static hf_register_info hf[] = {
    { &hf_slsk_string_length,
      { "String Length", "slsk.string.length",
      FT_UINT32, BASE_DEC, NULL, 0, "String Length", HFILL } },
    { &hf_slsk_username,
      { "Username", "slsk.username",
      FT_STRING, BASE_NONE, NULL, 0, "Username", HFILL } },
    { &hf_slsk_token,
      { "Token", "slsk.token",
      FT_UINT32, BASE_DEC, NULL, 0, "Token", HFILL } },
(...)
  };
```

Anhang C
Statistiken

Statistiken der in Kapitel 3 beschriebenen File-Sharing P2P Netzen, gemessen mit den Clients am 29.11.2003

eDonkey Netz

```
          62  Server
   1.731.628  Benutzer
 127.854.791  Dateien
         ~74  Dateien / Benutzer
```

(gemessen mit KaZaA Lite K++)

FastTrack Netz

```
   3.632.159  Benutzer
 647.902.285  Dateien
   4.929.152  Gigabyte
        ~178  Dateien / Benutzer
        ~7,8  Megabyte / Dateien
```

(gemessen mit eMule Plus)

SoulSeek Netz

```
       2.566  Räume
     ~96.000  Benutzer
              Server limitiert auf 100.000 Benutzer
```

(gemessen mit SoulSeek Dissektor)
(siehe http://slsk.blogspot.com/)