Diplomarbeit

Filesharing Networks

Ausgeführt zum Zweck der Erlangung des akademischen Grades eines
Dipl.-Ing. (FH) Telekommunikation und Medien
am Fachhochschul-Diplomstudiengang Telekommunikation und Medien St. Pölten

unter der Erstbetreuung von

Dipl.-Ing. Grischa Schmiedl

Zweitbegutachtung von

FH-Prof. Dipl.-Ing. Georg Barta

ausgeführt von

Manuel Messerer
0310038041

St. Pölten, am 12. September 2007 Unterschrift

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der von den Begutachtern beurteilten Arbeit überein.

....
Ort, Datum Unterschrift

Kurzfassung

Die Arbeit befasst sich einleitend mit der Bedeutung, Definition und Geschichte von Filesharing Systemen.
Nach einer Einteilung und einer Ausführung bezüglich der Generationen folgt ein Kapitel über die rechtliche Situation im österreichischen sowie im amerikanischen Rechtsraum.
Anhand dieser Grundlagen werden die Basistopologien und deren Kombinationen präsentiert, welche bei Filesharing Netzwerken genutzt werden. Nur diese Technologien ermöglichen die verschiedenen vorhandenen Applikationen und beinhalten die Grundlagen dafür.
Anschließend werden neuere Techniken und insbesondere Distributed Hash Tables (DHTs) vorgestellt. Einen wesentlichen Schwerpunkt bildet das Chord System.
Es folgt ein Überblick über moderne Anwendungen von Filesharing Netzwerken und deren Technologien. Außerdem werden mögliche komplett neue Möglichkeiten erforscht.
Am Ende folgt noch das Proof of Concept, welches basierend auf ein Chord Netzwerk ein Filesystem aufbaut. Zum Schluss erfolgt noch eine technisch gehaltene Anleitung für Benutzer des erstellten Systems.

Abstract

This thesis is introductory concerned with the meaning, definition and history of filesharing systems.
After an organization into generations, a chapter takes place about the legal situation in Austria as well as the American area.
On these bases the base topologies and combinations of those are presented, which are used in Filesharing networks. Only these technologies offer the possibility for the different presented existing applications and their techniques. Subsequently, newer technologies and in particular Distributed Hash Tables (DHTs) are introduced. The Chord system forms a substantial importance.
An overview of modern uses of file sharing networks and their technologies follows. Therefore several possibilities are investigated.
Afterwards there’s the theory about the Proof of Conecpt which builds a file system based on a Chord Network.
In the end a technically held guidance for the user of the provided system is provided.

Inhaltsverzeichis

Einleitung

Erkenntnisgegenstand

Filesharing Networks sind aus der heutigen Computerwelt nicht mehr wegzudenken. Sie basierten ursprünglich auf einer reinen Client-Server Struktur.
Mittlerweile sind Filesharing Systeme jedoch vor allem aus rechtlichen Gründen dezentralisiert. Der Begriff Filesharing ist eng verbunden mit dem Begriff Peer-to-Peer. Mehr dazu im *Kapitel 1 Überblick*.

Diese Netzwerke sind aufgrund massiven Missbrauches (illegaler Dateiaustausch) in den letzten Jahren unter heftige Kritik geraten.
Rechtlich gesehen sind in Österreich und Amerika Unterschiede zu beobachten. Vor allem bei den Strafen für Vergehen gegen die einschlägigen Gesetze. Genaueres dazu im *Kapitel 4 Rechtliche Hintergründe*.

Den ersten Ansatz zum „Hype" Filesharing Networks legte das bekannte Programm Napster. Diese Applikation war noch auf einer Client-Server Struktur aufgebaut – eine erhebliche Schwachstelle dieses Systems, welche aber durch die nächste Generation im Wesentlichen behoben wurde. Informationen dazu im *Kapitel 3 Filesharing Generationen*.
In der zweiten Generation wurden die Netzwerke dezentralisiert. Jeder Client konnte nun auch als Server fungieren. Es gibt nur noch sogenannte „*Servents*".
Dadurch ist dieses Netzwerk nicht mehr abzuschalten, indem man gezielt einige Stellen deaktiviert. Genau aus diesem Grund gibt es derzeit mehrere Systeme, die diverse Vor- und Nachteile haben.

Inzwischen gibt es zahlreiche neue Technologien, die vollkommen dezentral arbeiten. Leider hängt Filesharing Systemen aufgrund der Geschichte das Klischee an, dass sie nur für den Tausch von Musik und Videos zu gebrauchen sind.

Diese Technologien eröffnen zahlreiche Möglichkeiten, die zurzeit aber noch zuwenig genutzt werden. BitTorrent zum Beispiel liefert aufgrund der durchdachten Algorithmen eine hervorragende Grundlage für Dateiverteilung. Dazu mehr im *Kapitel 6.4 BitTorrent.*
BitTorrent ist auch die Grundlage für viele Anwendungen mit großer Zukunft. Information darüber im *Kapitel 8 Neue dezentrale Technologien*.

Einige weiterführende Anwendungen könnten wesentlich von diesen Systemen profitieren. Die Robustheit und Verteilungsalgorithmen von Systemen wie Chord, machen Ideen, wie ein dezentrales Filesystem oder online Spiele ohne zentralen Server überhaupt erst möglich.

Um die durchdachten Technologien insbesondere von Chord zu präsentieren, wird eine „Proof of Concept" Applikation zeigen, ob es möglich ist, ein globales Dateisystem in einem Netzwerk zu erstellen.
Es sollen dabei alle Daten jederzeit verfügbar sein wie in einem Client-Server Modell, jedoch mit dem entscheidenden Unterschied, dass kein Server benötigt wird.

Forschungsproblem

Die Überlegung, welche Anwendungen ein dezentrales Filesharing Netzwerk nutzen könnte, benötigt eine sehr genaue Untersuchung. Denn es ist möglich, dass Firmen bereits mit diesen Möglichkeiten beschäftigt sind, aber in der Umsetzung auf Probleme stoßen. Beispiele für beginnende Umsetzungen und Erweiterungen sind in dem *Kapitel 9 Mögliche dezentrale Technologien* zu finden. Probleme oder Schwächen, die bei der Recherche aufgetreten sind, sind unter *Fazit* zu finden.
Dennoch wird versucht, sinnvolle Anwendungsmöglichkeiten zu finden und diese genauer auszuführen.
Viele Systeme, wie zum Beispiel Kademlia (KAD), sind zwar bereits genau definiert, aber im Vergleich zu Gnutella großteils nur sehr schlecht dokumentiert.

Um sich auf die wesentliche „Proof of Concept" Implementierung zu konzentrieren, ist es notwendig, sämtliche Teilnehmer in einem Programm zu simulieren. Eine genaue Ausführung hierzu befindet sich im *Kapitel 10 Proof of Concept-GCFS*

Forschungsleitende Fragestellung

Welche Geschichte steckt hinter dem File Sharing? Warum ist ein dezentrales System so wichtig? Sind, außer dem allseits bekannten Gnutella noch ausgeklügeltere Technologien überlegt worden, und wie funktionieren diese?
Ferner ist es eine der wichtigsten Fragen bei der Erstellung dieser Arbeit, ob es zu den modernen Technologien bereits Umsetzungen gibt. Dies wird im *Kapitel 8 Neue dezentrale Technologien* näher erläutert.
Des Weiteren wird untersucht, welche Technologien denkbar und implementierbar wären. Dies wird im *Kapitel 9 Mögliche dezentrale Technologien* betrachtet. Ebenso werden dort aufgetauchte Probleme und wenn möglich Lösungen dazu geschildert.
Aufgrund der Forschung in diesem Bereich ist ein Proof of Concept mit einer neuen Technologie implementierbar, welches eine neue Technik aufgreift und abseits von dem „Klischee-Filesharing" sinnvoll und funktionstüchtig macht.

Forschungsmethode

Für die theoretische Arbeit und das Hintergrundwissen bezüglich Filesharing Systeme dient hauptsächlich das Internet.
Für den Einstieg und ein besseres Verständnis bezüglich „Filesharing Networks" dienen die Gnutella White Papers [GTL1], da dies das am häufigsten verwendete Netzwerk ist und hier auch die meisten Ressourcen/Applikationen vorhanden sind.
Die Erforschung der Filesharing Netzwerke beginnt bei den simpleren Techniken. Dies schafft das Grundlagenwissen für komplexe Technologien.
Mit dem Wissen über diese Systeme soll erforscht werden, ob File Sharing Systeme typisch zentralisierte Anwendungen revolutionieren können.
Die Test-Implementation von „Chord" (Proof of Concept) kann regelmäßig durch die Funktionalität selber getestet werden.

1 Überblick

Die Begriffe Filesharing Networks und Peer-to-Peer sind eng miteinander verbunden.
Eine genaue Definition von Peer-to-Peer ist schwierig, weil P2P weder eine homogene Technologie, noch eine konkrete Anwendung, sondern ein Paradigma ist.
Daher werden in dieser Arbeit Systeme/Anwendungen, welche eine Filesharing Topologie (mehr dazu im *Kapitel 5 Filesharing Topologien*), typische Suchabfragen und den Austausch – oder das *Streaming* von Daten als Mittelpunkt haben, als Filesharing Netzwerke/Systeme bezeichnet.

Mit der zunehmenden Verbreitung des Internets wird das Tauschen von Daten wie Musik, Filme und Software über Netzwerke als Filesharing verstanden.
Eine Einschränkung des Begriffes Filesharing Netzwerke ist seine Gleichsetzung mit dem Tauschen urheberrechtlich geschützter Inhalte über Peer-to-Peer.

Filesharing bedeutet, digitale Informationen zur Verfügung zu stellen oder zu beziehen. Jeder Informationsaustausch zwischen Menschen, bei denen der Computer als Medium genutzt wird, ist damit Filesharing.

„The word ‚filesharing' is a euphemism and a serious misnomer. [...]
In fact, it's not really sharing at all, because if I share a piece of cake with you, we're each doing with a little less -- I have half a piece and you have half a piece. This doesn't hold true for digital distribution since I don't lose anything by ‚sharing' with you."
[KDC], S. 27

1.1 Bedeutung

Filesharing Systeme sind auf Grund der Popularität von Diensten wie Napster oder Gnutella vielfach als **die** neue Technologie des Internets anzusehen.

Ursprünglich, vor allem bei der Entwicklung von ARPANET, stand ein dezentrales Netzwerk im Vordergrund. Aus diesem entstand dann später das Internet.
Durch die vermehrte private Nutzung, veränderte sich jedoch das Anforderungsprofil. Es war nicht mehr nötig, dass jeder Rechner auf jeden anderen Rechner jederzeit Zugriff haben muss. Vielmehr entwickelte sich das Internet zu einem bloßen Downstream-Medium. Dies merkt man noch heute an den meist asymmetrischen Bandbreiten.

Filesharing Systeme erlangten erst 1999 durch den Erfolg von Napster allgemeine Bekanntheit. Doch Napster war eigentlich nur der Grundstein der modernen Filesharing Technologien. Dies war sozusagen lediglich das Einläuten einer Reihe von Netzwerken, wobei das Ende noch lange nicht erreicht ist und auch nicht abzusehen ist.
Auch die Integrierung von Filesharing Systemen in andere Programme steckt erst in den Kinderschuhen.

1.2 Eigenschaften von Filesharing Systemen

Aufgrund der Geschichte, kann auch eine Client-Server Architektur ein Filesharing Netzwerk darstellen. Heutzutage sind jedoch hauptsächlich dezentrale Filesharing Netzwerke verbreitet.

Charakteristische Eigenschaften einer Client-Server Architektur sind:

- Initiative zur Interaktion geht immer vom Client aus
- Kommunikation findet ausschließlich zwischen Clients und Servern statt
- Server bietet Dienste an, Clients nutzen diese

Charakteristische Eigenschaften eines dezentralen Netzwerkes sind:

- Teilnehmer können Clients und Server sein – so genannte Servents
- Die Servents können Information untereinander direkt austauschen und sind gleichberechtigt
- Das System muss auch dann funktionieren, wenn Servents häufig dem Netzwerk beitreten oder dieses verlassen

2 Geschichte

2.1 Bulletin Boards

Bulletin Board Systems (BBS) sind Computer, auf denen eine Software läuft, die anderen Rechnern ermöglicht, sich dort einzuwählen. Dies kann zum Beispiel auch über eine normale Telefonleitung geschehen. Über diese Verbindung kann man Daten downloaden, uploaden oder Nachrichten austauschen.
Software selbst wurde in der ersten Zeit der Bulletin Boards, am Ende der 1970 Jahre, kaum getauscht. Die dafür erforderliche Bandbreite war für den durchschnittlichen Teilnehmer nicht verfügbar.
BB wurden hauptsächlich als eine Art Kommunikationsmittel genutzt. Erst als Modems preisgünstiger wurden, konnten kleine Softwarepakete ausgetauscht werden. Vor allem in Großstädten der USA, wo Ortsgespräche kostenlos sind, stieg die Nutzung von Bulletin Boards enorm an.

2.2 IRC

Der Internet Relay Chat (IRC) wurde im August 1988 veröffentlicht. IRCs sind ein rein textbasiertes Nachrichtenaustauschsystem.
Es können mehrere Personen an einem Gespräch teilnehmen. Ebenfalls kann man sich in verschiedenen Channels aufhalten, die meist nach Interessen benannt sind. Neue Gesprächs-Channels können von jedem Teilnehmer frei eröffnet werden.
Zur Teilnahme ist ein spezielles Programm, nämlich ein IRC-Client, notwendig.
IRC wird in erweiterter Form noch heute sowohl allein stehend als auch von Filesharing Anwendungen genutzt.

2.3 Webwarez

Mit dem Aufkommen des Internets in den neunziger-Jahren waren innerhalb kurzer Zeit zahlreiche Webseiten zu finden, welche "*Webwarez*" anboten. Da diese Seiten von jedem benutzt werden konnten, der Zugang zum Internet besaß, wurde binnen kürzester Zeit eine breite Masse auf dieses Angebot aufmerksam. Da die Anzahl der Teilnehmer stetig wuchs, wurde auch das Angebot von Schwarzkopien zahlreicher.

Dies löste erste Diskussionen aus, dass das Internet kein rechtsfreier Raum sein dürfe. Bereits wenige Monate nach Beginn dieser Diskussionen musste der erste Provider einer Webwarez Seite seinen Dienst auf Grund rechtlichen Einschreitens vom Netz nehmen. Vier Jahre später – zum Jahrtausendwechsel – war die Webwarez Szene so gut wie ausgestorben. Seitdem ist auch das Internet Kontrollen ausgesetzt.
Jedoch hatte das Internet für die Idee an sich gesorgt, ein allzeit zur Verfügung stehendes Medium zur Verbreitung und Beschaffung von Schwarzkopien zu sein. So war es nur eine Frage der Zeit, bis andere Verbreitungswege gefunden wurden.

2.4 Release Groups

Mittlerweile hatten sich die sogenannten "FTP-Release Groups" gebildet. Durch neue Kodierungsmöglichkeiten konnten nun auch Video- und Musikdateien mit akzeptablem Speicherverbrauch digital gespeichert und kopiert werden.
Die Verbreitung der Warez selber geschah unter Ausschluss der Masse über passwortgeschützte FTP Server. Neue gekrackte Software wurde von Mitgliedern auf diese privaten FTP Server kopiert. Anderen Gruppen wurde sie zugespielt, indem eine Kopie auf einem öffentlichen FTP-Server vervielfältigt wurde. Von dort wurden sie dann von einem Mitglied einer anderen Gruppe, wenn benötigt, wieder auf den eigenen FTP-Server kopiert und dem Archiv hinzugefügt. Hier galt, ein Mitglied muss etwas tun, um sich Software herunterzuladen. (zum Beispiel eine neuen Film zu "rippen"). Neueste Software, Musik und Filme wurden durch dieses System innerhalb der Szene sehr schnell verbreitet.
Je nach ihrer Aktivität erhielten Mitglieder die Passwörter zu den Internet Seiten, wo die Adressen der FTP Server notiert sind. Die Topsites waren jedoch nur sehr wenigen Leuten vorbehalten.

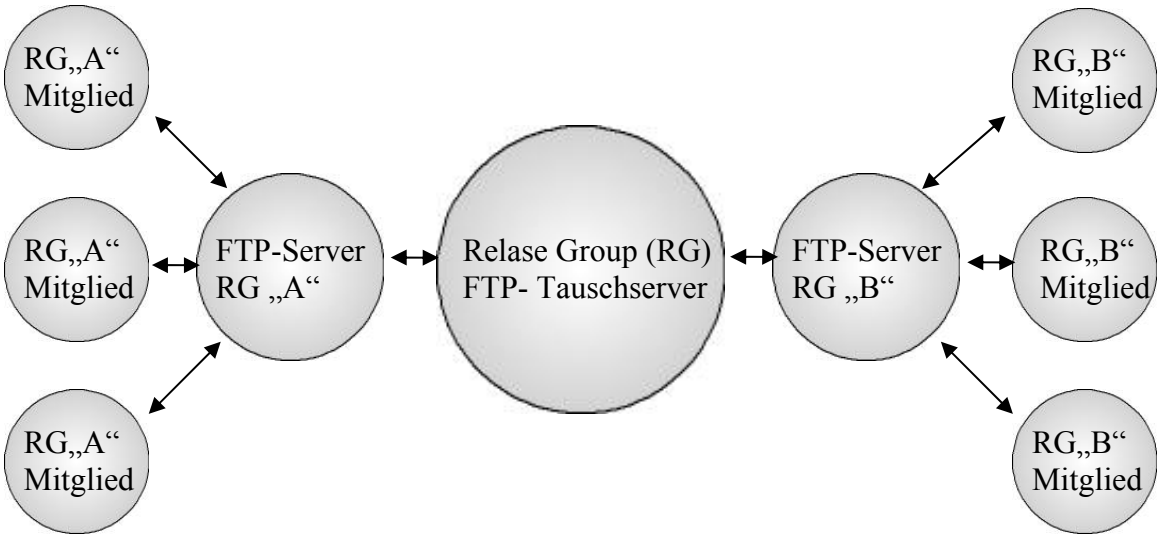

Abbildung 1: Datenaustausch bei Release Groups (RGs)

2.5 FXP Szene

Zugang zu der Release Group Szene hatten die wenigsten. Da aber das Publikum für Softwaretausch sehr zahlreich war, bildete sich eine weitere Gruppe namens FXP.
Den Namen verdanken sie einem Protokoll von FTP-Servern, dem "File eXchange protocol".

Während die Release Groups auf Software „kracken“ und untereinander verbreiten abzielten, entwickelte sich in der FXP-Szene eine regelrechte Sammelwut.
Da die Benutzer der FXP-Szene die Programme nicht selber „krackten“ oder per Insider von Firmen besorgten, war die Software, die getauscht wurde, nicht brandaktuell, jedoch wurde wesentlich reger getauscht. Zum Veröffentlichen selber benutzte die FXP Szene fremde FTP Server von Firmen, die gehackt wurden. Dort wurde die Software hinaufgeladen und anschließend die IP des Servers auf Boards veröffentlicht.
Diese Server waren dann nach kurzer Zeit wieder offline, zum Beispiel weil ein Administrator den ungewöhnlich hohen *Traffic* bemerkt hatte. Um dies so weit wie möglich zu verhindern, wurden die Adressen meist nur in passwortgeschützten Boards preisgegeben.
Der Kreis, der somit Zugang zu den Schwarzkopien hatte, war zwar wesentlich größer als bei den Release Groups, jedoch noch immer sehr eingeschränkt.
Da jedoch jeder gerne gratis Schwarzkopien besitzen wollte, kam es gegen Ende der neunziger Jahre zu einer revolutionären Erfindung, die schon bald als Schreckgespenst der Unternehmen gelten sollte. Das „Filesharing“, so wie es heute bekannt ist.

2.6 Das „Monster“ Napster

Ende der neunziger Jahre entwickelte der achtzehnjährige Shawn Fanning eine revolutionäre Software namens „Napster“.
Der Name des Programms stammt vom Spitznamen seines Schöpfers. Shawn Fanning gehörte nämlich einer Gruppe von Hackern mit dem Pseudonym „Napster“ an. Als Napster entwickelt wurde, studierte Fanning Informatik an der Northeastern University in Boston.
Durch Napster ist der Begriff „Filesharing“ erst bekannt geworden. Durch die einfache Bedienung revolutionierte Napster die „Warez“.
Es war nicht mehr notwendig, mühselig Webwarezseiten zu durchsuchen. Auch eine Einarbeitung in eine Szene war nun überflüssig. Damit war sozusagen das „Eldorado der Gelegenheitskopierer“ geboren worden.
Shawn Fanning bekam bei der Programmierung Hilfe durch seine Kollegen Sean Parker und Jordan Ritter. Sie erschufen die Struktur von Napster, welche dem Programm später auch zum Verhängnis wurde. Genaueres dazu im *Kapitel 6.1.1 Napster.*
Der Onkel von Shawn, John Fanning, erkannte das große Potential von Napster und gründete die Firma mit dem gleichlautenden Namen Napster.
Diese stellte bereits 1999 eine zum Download verfügbare, überarbeitete Version ins Internet. Der Andrang war so groß, dass die Server unter den Anfragen zusammenbrachen.

Nur durch Mundpropaganda wurde Napster zur am schnellsten verbreiteten legalen Software der Geschichte des Internets und im selben Zeitraum auch zum Albtraum der Musikindustrie. Bereits im gleichen Jahr reichte die RIAA (Recording Industry Association of America) Klage ein.
Die Forderung war hierbei $100.000 für jeden widerrechtlich getauschten Titel. Dies ergab bei den damals 200.000 verfügbaren Liedern eine Klagesumme von 20 Milliarden Dollar.

Diese Klage hatte jedoch einen Nebeneffekt, den die Musikindustrie nicht geahnt hatte. Durch die Medienberichterstattung stieg der Bekanntheitsgrad von Napster explosionsartig und die Anzahl der Benutzer erreichte astronomische Höhen.

Dennoch musste Napster im Mai 2000 die erste juristische Niederlage gegen Anwälte der Heavy Metal Band „Metallica“ einstecken. Die Band erstellte eine Benutzerliste mit Pseudonymen von über 300.000 Teilnehmern, die ihre Lieder zum Download anboten. Als Folge wurden diese von der Benutzerliste entfernt, was jedoch keinen dieser Benutzer daran hinderte, sich einfach mit einem anderen Namen neu anzumelden.
Diesem erstmaligen Erfolg gegen Napster folgten jedoch andere Künstler und die Kritik wurde immer schärfer. Napster selber wurde dadurch jedoch immer bekannter und zählte noch im Jahr 2000 bis zu 20 Millionen Teilnehmer.

Im Juli 2000 untersagte dann ein amerikanisches Bezirksgericht erstmals den Tausch urheberrechtlich geschützter Musik.

Auf die Anmerkung des Anwalts von Napster, dass dieses kritische Urteil einer Schließung der Tauschbörse gleichkomme, antwortete die Richterin Marilyn Hall Patel mit: *„Das ist Ihr Problem, Sie haben dieses Monster erschaffen“*. [KJSE], S.83

Das Urteil sorgte für einen neuerlichen Ansturm und die Server von Napster brachen erneut zusammen.
Gegen dieses Urteil wurde Berufung eingelegt und die Server blieben bis zu einer endgültigen Klärung der Streitfrage online. Zu diesem Zeitpunkt konnte die Tauschbörse bereits geschätzte 40 Millionen Benutzer aufweisen.
Im Oktober desselben Jahres sorgte der Medienkonzern Bertelsmann für Aufsehen, als er sich überraschend bei Napster einkaufte obwohl er selbst an der Klage beteiligt war. Bertelsmann kündigte an, alle Klagen fallen zu lassen und bis Mitte 2001 einen zu bezahlenden Downloadservice einzurichten. Der Medienkonzern konnte sich aber bis dahin nicht mit den großen Plattenfirmen einigen.

Am 1. Juli 2001 musste sich Napster geschlagen geben und schloss die Pforten. Die Server wurden nach nur 2 Jahren abgeschaltet. Zuletzt konnte man mit bis zu 60 Millionen Benutzern Musikdateien tauschen.

2.7 Napsters Erbe

Bereits vor dem Tode des freien Napster waren zahlreiche Alternativen von findigen Programmierern in Arbeit und warteten bereits in den Startlöchern.
Die Musikindustrie musste erkennen, dass sie den Feind nicht besiegt, sondern nur kurzfristig in die Knie gezwungen und lediglich zerteilt hatte.
Neue Programme wie Audiogalaxy und WinMX waren die bekanntesten Vertreter des direkten Erbes. Sie waren bereits etwas dezentraler gehalten und somit etwas sicherer gegen das Gesetz und Attacken der Industrie*.
Es wurden nach der Veröffentlichung des Source Codes von Napster einige Open Source Implementationen geschrieben. Ein bekannter Vertreter hierfür ist OpenNap.

*Anmerkung des Autors: Zuerst wird der Ausdruck „Musikindustrie“ verwendet. Danach nur noch „Industrie“, um zu zeigen wie sich im Laufe der Zeit das „Problem“ für die Industrie ausgeweitet hat.

Außerdem brachten diese Programme noch weitere Vorteile, als nur eine bessere Dezentralisierung.
Mit der neuen Tauschbörsengeneration wurde erstmals auch das Austauschen von anderen Dateien möglich. Außerdem brachten die Programme eine Geschwindigkeitserhöhung mit sich, indem sie ermöglichten, dass von mehreren Benutzern gleichzeitig eine bestimmte Datei downloadet.
Dies alles kam mit der neuen Option einher, abgebrochene Downloads neu aufzunehmen, was vor allem Besitzern einer „dial up-Verbindung“ den Zugang zu den Tauschbörsen erleichterte.

Damit wurde auch das Tauschen von Filmen ermöglicht, was nun auch die Filmindustrie auf den Plan rief.

Geschockt von der Fortsetzung des Filesharings, zog die Industrie nun wieder in den „Krieg“. Dabei konnten einige Börsen gestoppt werden, wie zum Beispiel Audiogalaxy. Viele dezentrale Systeme wie das beliebte KaZaA [KZA] konnten jedoch nicht aufgehalten werden.
Daher entschied sich die Industrie für einen neuen juristischen Weg. Sie begann gezielt einzelne Privatbenutzer zu verklagen und hofft bis heute, damit eine abschreckende Wirkung zu erzielen.
Dies änderte jedoch nichts daran, dass bereits im August 2001 die vier beliebtesten Tauschbörsen zusammen einen insgesamt höheren Download hatten als Napster jemals zuvor.
Ende 2004, Anfang 2005 wurde das BitTorrent Netzwerk von Bram Cohen vorgestellt. Die Technologie dieses Systems ist so gut durchdacht, dass sie immer mehr in Applikationen implementiert wird.
Durch diese nicht klagbaren Ansätze ist die Industrie nun vermehrt darauf umgestiegen, einzelne Benutzer zu verklagen. Ebenso versucht man durch Botschaften in Werbungen wie „Raubkopierer sind Verbrecher“ abzuschrecken.

3 Filesharing Generationen

Im Wesentlichen kann man 4 Generationen unterscheiden, wobei den Beginn, also die erste Generation „Napster“ darstellt (Möglichkeiten wie FTP nicht mitgezählt, da das Publikum vergleichsweise verschwindend gering ist bzw. war).
Anzumerken sei, dass bei dieser Generationseinteilung nur bedingt zu erkennen ist, wie „modern“ ein Netzwerk ist. Da die zweite Generation am verbreitetsten ist, werden diese Verfahren am Stärksten weiter entwickelt. Bestes Beispiel hierfür ist die relativ neue Technologie BitTorrent, welche eben dieser zweiten Generation angehört.

3.1 Erste Filesharing Generation

Die erste Generation von Filesharing erkennt man vor allem durch eine eindeutige Client-Server Struktur. Der Server bietet hierbei die Daten direkt zum Download an, oder enthält Informationen über andere, sogenannte *Nodes*. Diese Nodes sind andere Clients, von denen man Daten herunterladen kann. Entscheidend hierbei ist jedoch, dass im Mittelpunkt ein Server steht. Legt man diesen lahm, so fällt das gesamte Netzwerk in sich zusammen.
Bestes Beispiel hierfür ist Napster oder die damalige Serverversion von eDonkey2000. Dieses System wird auch von Instant Messenger (IM) benutzt. Die IMs werden aber nun zusehends dezentralisierter gestaltet. Entwicklungen zu IMs, sind im *Kapitel 8.2 Instant Messenger* zu finden.

3.2 Zweite Filesharing Generation

Das Ziel der zweiten Filesharing Generation war es, das Hauptmanko der ersten Generation auszumerzen – den Server. Dieser musste nicht nur hohen Belastungen standhalten (Napster Server sind des Öfteren aufgrund des hohen Andrangs zusammengebrochen), sondern der Provider muss auch mit hohem Traffic rechnen.
In der zweiten Generation fällt der Server weg, jedoch wurden die Systeme dadurch enorm komplexer. Nun sind Verbindungen zu jedem anderen Servent direkt möglich und erforderlich. Dies ermöglicht, ohne zentralen Server auszukommen. Gleichzeitig ist damit das System sehr robust, da man es nicht mehr einfach abschalten kann. Dies ist vor allem in rechtlicher Hinsicht wichtig.
Die bekanntesten Vertreter dieser Generation sind Gnutella, KaZaA (FastTrack) und eMule (Kademlia). Alle diese Filesharing Netzwerke sind vollständig dezentralisiert, mit Ausnahme von KaZaA, das zwar theoretisch ohne Server funktioniert, jedoch für die Benutzeranmeldung einen zentralen Server benötigt.
Später wurden weitere Technologien dieser Generation genutzt, die nach wie vor die weiteste Verbreitung mit über 20 Millionen Benutzern aufweist. Eine bekanntere Technologie ist hierbei das häufig eingesetzte „Fasttrack“ Verfahren.
Ferner kann man auch BitTorrent zu dieser Generation zählen, wobei hier die Benutzerzahl nicht effektiv gemessen werden kann, auf jeden Fall aber stetig ansteigt.

3.3 Dritte Filesharing Generation

Die dritte Generation zeichnet vor allem dadurch aus, dass sie neue Prinzipien mit den Technologien der zweiten Generation zu kreuzen versucht.

Diese Generation weist vor allem eine hohe Anonymität der Benutzer auf.
Die Daten werden zuerst verschlüsselt, bevor sie versendet werden. Ferner gelangen die Daten nicht direkt von einem Servent zum nächsten, sondern werden über Umwege an das Ziel geleitet.

Anstatt wie sonst üblich, werden die Daten nicht von „Servent A“ direkt zum „Servent C“ (strichlierter Pfeil) gesendet. Vielmehr sendet „Servent A“ die Daten an „Servent B“, der diese wiederum an „Servent C“ weitergibt. Der Empfänger hat den ursprünglichen Sender somit nie kennen gelernt und dieser bleibt vollständig anonym (Abbildung 2).

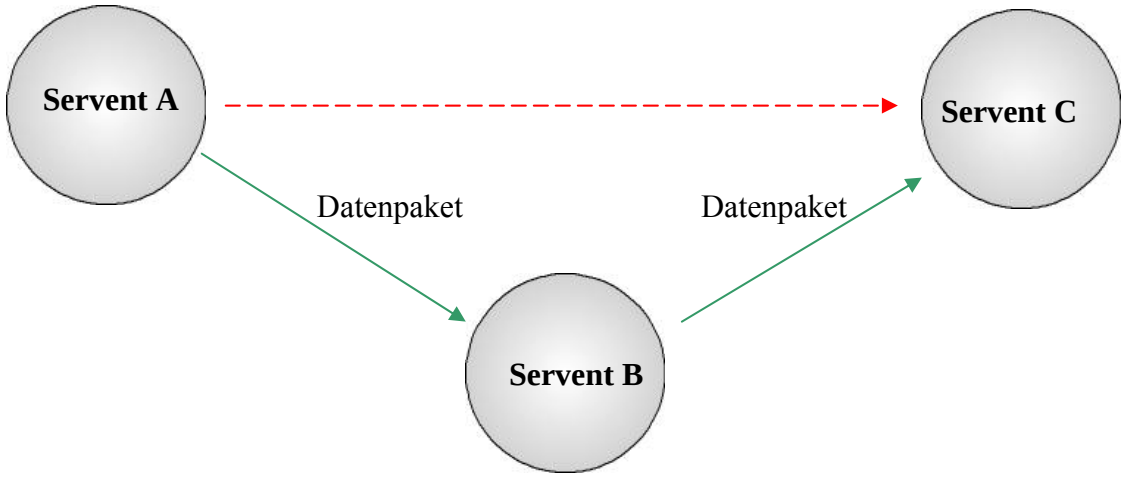

Abbildung 2: Indirekter Datenverkehr

Bekannte Beispiele hierfür sind die in einem noch sehr frühen Status befindlichen Netzwerke Freenet, WASTE, Mute* und I2P. Diese Netze unterscheiden sich stark voneinander und stecken noch in den Kinderschuhen. Außerdem ist hier noch sehr viel Dokumentationsaufwand von nöten, um diese Netzwerke besser zugänglich zu machen.

* Projekt stillgelegt am 03.06.2007

3.4 Vierte Filesharing Generation

In der vierten Generation von Filesharing Systemen werden statt Dateien sogenannte Streams über ein Netzwerk gesendet.

Diese Technologie ermöglicht es zum Beispiel, Musik zu hören oder fernzusehen ohne einen Server zu kontaktieren.
Der Stream selber wird dabei einfach über das Filesharing Netzwerk verteilt.
Diese Technik setzt eine so genannte „Swarming“ Technologie voraus, wie die von BitTorrent. (Weiterführende Erklärungen zu BitTorrent im *Kapitel 6.4 BitTorrent*)
Würden die Daten nicht über die Swarming Technologie verteilt, wären die benötigten Bandbreiten zu groß.
Zu dieser Generation existieren bereits neue Umsetzungen mit beachtlichen Möglichkeiten.
Eine immer bekannter werdende Anwendung ist PeerCast. Mehr Informationen zu PeerCast im *Kapitel 8.1 Audio/Video Broadcasting*.

4 Rechtliche Hintergründe

4.1 Österreichisches Recht

4.1.1 Die Privatkopie

Nach österreichischem Recht ist eine Herstellung einzelner Vervielfältigungsstücke außer bei Software, erlaubt. Ebenso ist eine Weitergabe im Bekanntenkreis erlaubt.
Erst durch die Urheberrechtsnovelle im Jahr 2003 wird in §42 das Recht auf Vervielfältigung für eigenen Gebrauch eingeschränkt. Ebenso dürfen die Kopien weder für unmittelbare noch für mittelbare Zwecke eingesetzt werden. Daher darf ein geschütztes Werk nicht dazu verwendet werden, es der Öffentlichkeit zugänglich zu machen.
Eine Ausnahme besteht für Ton- und Bildaufnahmen, da die Leerkassettenvergütung einen gewissen Ausgleich hierfür schafft.

4.1.2 Sonderproblem Tauschbörsen

Filesharing Netzwerke im Internet dienen dem Austausch von Daten, meistens in Form von Dateien. Dieser Vorgang ist mit der Zustimmung des Urhebers oder nicht urheberrechtlich geschütztem Material völlig legal. Werden jedoch urheberrechtlich geschützte Daten ohne Zustimmung des Urhebers angeboten, verstößt man gegen das Vervielfältigungsrecht und dem Zurverfügungstellungsrecht. Beides steht nach §91 unter Strafe.

Download:

Es stellt sich jedoch die Frage, ob es strafbar ist, geschütztes Material ohne Bezahlung über Filesharing Netzwerke zu beziehen.
Wird somit ein solches Werk bezogen und man besitzt zusätslich eine legal erworbene Kopie davon, so fällt diese Handlung unter Privatkopie. Das Gesetz schreibt keinerlei bestimmte Anforderungen an die Qualität der Kopie vor.

Upload:

Die RBU (Revidierte Berner Übereinkunft) zum Schutze von Werken der Literatur und Kunst ist ein völkerrechtlicher Vertrag.
Umgesetzt wurden diese Richtlinien in der Urheberrechtsnovelle 2006.
Entscheidend ist, dass es nicht strafbar ist, passiv (d.h.: nicht explizites Herunterladen von geschützten Daten) an Filesharing Netzwerken teilzunehmen, obwohl man dabei andere bei illegalen Aktivitäten unterstützt.
Jedoch ist bei fast jedem Filesharing Teilnehmer auch der Upload aktiviert, nachdem er geschützte Werke heruntergeladen hat.
Ein Upload von geschützten Werken ist generell strafbar, da hier das Zurverfügungsstellungsrecht nach $18a verletzt wird.

Vergehensstrafen:

Es genügt, gegen Uploader von urheberrechtlichem Material vorzugehen, wie es die Industrie praktiziert. Tatsächlich wird jedoch hierzulande nur gegen Großanbieter vorgegangen, denn diese beeinträchtigen die Interessen der Urheber unzumutbar.

4.2 Amerikanisches Recht

Der Grossteil von rechtlichen Problemen mit Filesharing Netzwerken wird in Amerika durch den „Digital Millenium Copyright Acts“ (DMCA) geregelt, welcher am 28. Oktober 1998 rechtskräftig wurde.
Im Vergleich zu österreichischen Gesetzen ist dieser sehr „firmenfreundlich“
DMCA ermöglichte zum Beispiel Epson, allen Drittanbietern die Herstellung von Druckerkartuschen für deren Drucker zu verbieten.
Dieser Akt löste heftige Diskussionen aus. Der Chaos Computer Club (CCC) äußerte sich umfassend zu diesem Thema. Die genaue Aussage ist im *Anhang 15.2 Chaos Computer Club: Boycott the music industry* zu finden.

Der DCMA wird für die den Filesharing Netzwerken relevanten Gesetze durch den „Intellectual Property Protection Act“(IPPA) erweitert.

Insgesamt ist die rechtliche Grundlage nicht so verschieden zwischen Amerika und Österreich. Stark variiert jedoch das Strafausmaß für illegale Kopien. Ein Kavaliersdelikt bei uns kann dort schnell Haftstrafen nach sich ziehen.
Ein wesentlicher Unterschied besteht noch in den Möglichkeiten, die der Exekutive gegeben werden. So müssen zum Beispiel auf Verdacht Providerdaten preisgeben werden.

Der IPPA 2007 soll wieder eine deutliche Verschärfung an Strafen und noch mehr Rechte für die Exekutive beinhaltet. Im Vergleich zu österreichischen Gesetzen wirken die Maßnahmen in diesem Akt erschreckend und zugleich belustigend.

Einige Neuheiten in der IPPA 2007:

1. Strafbar soll nicht nur eine kriminelle Handlung hinsichtlich Copyright sein, sondern bereits der Versuch dazu. Ein Beispiel wäre hierfür jemand, der einen Kopierschutz zu knacken versucht.
2. Wird eine Klage gegen einen Downloader von geschütztem Material eingeleitet, kann sein Rechner vollständig beschlagnahmt werden.
3. Die Exekutive ist berechtigt, einen verdächtigen Filesharing Teilnehmer direkt abzuhören.

Umsetzung

Die jährliche Erweiterung des IPPA ist immer eine der am heftigst diskutierten Novelle in Amerika. Innerhalb dieser Auseinandersetzungen wird auch immer wieder DCMA kritisiert. Ein Beispiel hierfür befindet sich im *Anhang 15.3 Congress readies broad new digital copyright bill*.

5 Filesharing Topologien

Im folgenden Kapitel werden Topologien vorgestellt, die in Filesharing Netzwerken eingesetzt werden.

Den Ausgangspunkt für komplexere Topologien bieten zentrale hierarchische, sowie dezentrale Systeme.
Diese Basistopologien sind entscheidend, da die meisten modernen Filesharing Netzwerke auf Mischformen aus diesen basieren.
Gnutella besitzt eine Struktur, die aus den Basistopologien Zentral und Dezentral besteht und damit ein Beispiel für eine Mischform darstellt.

5.1 Basis Topologien

5.1.1 Zentrale Topologie

Zentrale Systeme sind die gebräuchlichste Form von Topologien im Internet. Diese Topologie ist auch unter dem Namen Client/Server Architektur bekannt und Vorbild für die Arbeitsweise von Webservern und Datenbank-Anwendungen.

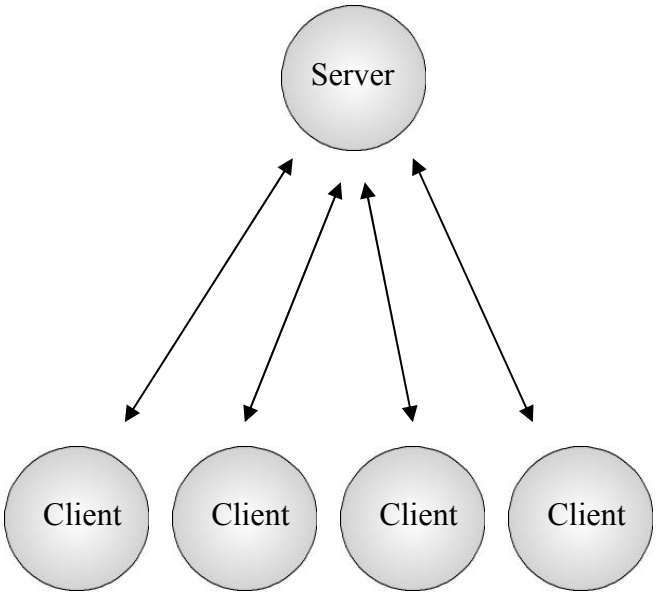

Abbildung 3: Zentrale Topologie

Die Daten werden in diesem System zentral am Server gelagert. Die Clients können diese meist nur abrufen oder ergänzen, wenn sie die nötigen Rechte hierfür haben.
Die Kontrolle, wer auf welche Daten zugreifen darf, unterliegt bei dieser Topologie allein dem Server. Lehnt dieser einen Client ab, gibt es für ihn keine Möglichkeit, an die entsprechenden Daten zu gelangen.

Der entscheidende Vorteil des zentralen Systems ist die Verwaltung. Alle Daten, die von irgendeinem Client zu einem beliebigen Zeitpunkt benötigt werden, müssen lediglich dem Server bekannt sein. Durch diese Art der Datenverwaltung kann auch die Konsistenz der Daten und deren Kohärenz gewährleistet werden.
Dies ist zum Beispiel ausschlaggebend dafür, dass das SETI@home Projekt diese Form verwendet. Denn bei einem zentralen System kann der Server die Daten nach der etwaigen Verarbeitung durch die Clients überprüfen, was einen guten Schutz vor *Exploitern* bietet.

Der Vorteil, nur einen Rechner vor Angriffen und Viren schützen zu müssen, ist auch gleichzeitig die *Achillesferse* des Systems. Sollte der Server abstürzen oder überlastet sein, legt dies das gesamte Netzwerk lahm.

5.1.2 Hierarchische Topologie

Eine Hierarchische Struktur wird bereits seit langer Zeit in der Computerwelt benutzt und zählt generell zu den ältesten Topologien.

Ein bekanntes System für diese Architektur ist das Domain-Name-System (DNS). Dieses basiert auf einer hierarchischen Struktur, ist jedoch von der Aufgabe her nicht direkt eine Filesharing Technologie.
Die DNS-Namensbereiche sind in einer Baumstruktur aufgeteilt. Sekundäre Namensserver können primäre Namensserver auf der Suche nach Informationen kontaktieren und umgekehrt.
Jede Hierarchie ist dabei für die Informationsweitergabe an die direkt höher oder tiefer gelegene Ebene zuständig (Abbildung 4).

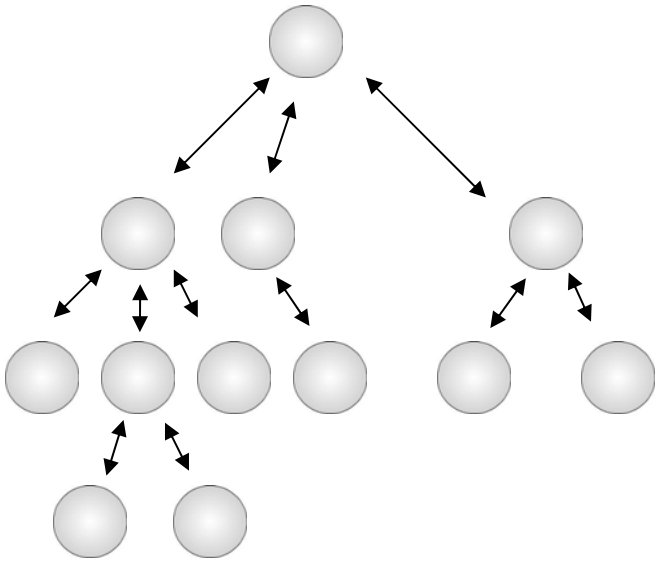

Abbildung 4: Hierarchische Topologie

Ein bekannter Vertreter der hierarchischen Systeme ist das Usenet. Die diversen unterschiedlichen Newsgroups des Usenets sind in einer übersichtlichen Hierarchie gegliedert. Die oberste Ebene stellt die „offizielle“ Hierarchie dar, die sich in weitere Hierarchien mit vielen Untergruppen aufgliedern. Hierarchische Systeme haben einen klar definierten Aufbau, der die Verwaltung eines solchen Systems erleichtert.
Die Größe eines solchen Systems kann zu Problemen führen, wenn es um das Auffinden einzelner Fehlfunktionen geht. Sie sind jedoch insgesamt gesehen weit weniger fehleranfällig, als ein zentral strukturiertes System.

Der wesentlichste Vorteil dieser Topologie ist die uneingeschränkte Skalierbarkeit.
Ein gutes Beispiel ist das DNS, das vor über 15 Jahren aus wenigen tausend Hosts bestand und nun Hunderte von Millionen Hosts verwaltet.

5.1.3 Dezentrale Topologie

In dezentralen Systemen wird auf einen zentralen Server komplett verzichtet. In einem solchen Netzwerk kann jeder mit jedem Informationen als Gleichgestellter austauschen. Die einzelnen Nodes dieses Systems sind somit zugleich Client und Server (Servents).

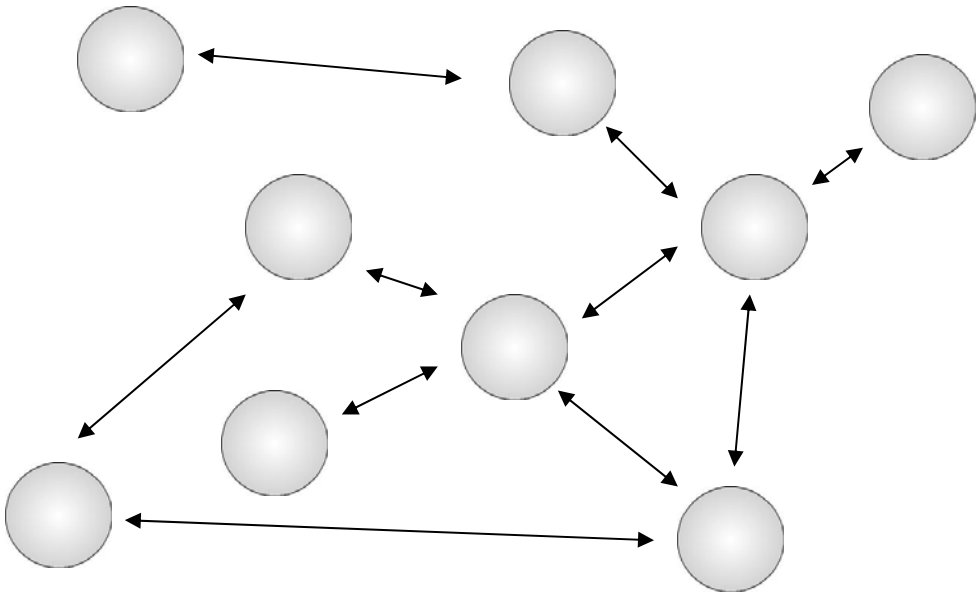

Abbildung 5: Dezentrale Topologie

Das Gnutella Netzwerk ist wahrscheinlich das „reinste“ dezentrale System, das in der Praxis verwendet wird. Das Filesharing System Gnutella ist derart dezentral verteilt, dass das System praktisch nicht zusammenbrechen kann, auch wenn große Teile ausfallen würden.

Die Eigenschaften der dezentralen Systeme stellen das Gegenteil von zentralen Systemen dar. Ihre Verwaltung ist fast unmöglich und es ist schwierig, die Inhalte zu kontrollieren. Für die Sicherheit muss jeder Servent selber sorgen, da jeder Teilnehmer die Inhalte selber bestimmt. Das dezentrale System hat im Allgemeinen keine beschränkte Größe und wird daher nur von der Anzahl der Benutzer bestimmt.

5.2 Hybride Topologien

Die Kombination von unterschiedlichen Basis Topologien lässt völlig neue Systeme entstehen – sogenannte hybride Netzwerke.
Durch die Zusammenführung den unterschiedlichen Topologien lassen sich Stärken von verschiedenen Systemen integrieren, ohne die Schwächen, die eine Topologie alleine beinhaltet, in Kauf nehmen zu müssen.
Bei modernen Filesharing Netzwerken wird inzwischen fast ausschließlich auf hybride Systeme gesetzt.

5.2.1 Zentral kombiniert mit zentral

Ein Server in einem zentralen System fungiert hier gleichzeitig auch als Client für andere zentrale Server (Abbildung 6).

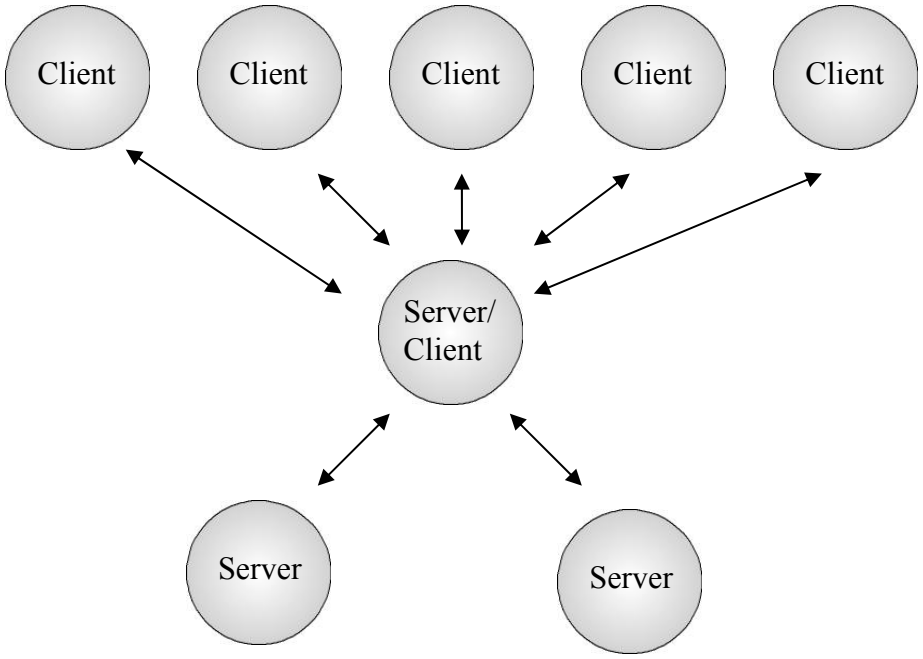

Abbildung 6: Zentral - mit Zentral Topologie

Wenn ein Browser einen Server kontaktiert, formatiert dieser Server oft lediglich Dateien in HTML. Die Daten selbst erhält er von einem oder mehreren einzelnen, zentralen Datenbankservern. Diese Architektur wird häufig verwendet, um die Funktionalitäten verschiedener Systeme mit nur einem öffentlichen Server zu ermöglichen.
Auch viele Filesharing Netzwerke nutzen diese Topologie, indem einzelne Server einen Verbund eingehen. Server, die voneinander unabhängig sind, können sich mit dieser Methode zusammenschließen und damit eine größere Nutzerbasis erreichen.
Anfragen, die von Clients an einen Server gestellt sind, können an benachbarte Server weitergegeben werden. Untereinander stehen die Server in einem Client/Server Verhältnis.

5.2.2 Zentral kombiniert mit dezentral

Eine zentrale Topologie mit einem dezentralen Netzwerk zusammengeschaltet, erscheint zunächst unsinnig. Ist dieses aber unabhängig voneinander angeordnet, ergibt sich daraus eine sehr gute Architektur.
Diese Topologie findet ihre Anwendung im FastTrack Netzwerk, das unter anderem von der populären Tauschbörse KaZaA eingesetzt wird. Mehr zu der Technologie FastTrack in dem *Kapitel 6.2.4 FastTrack*.

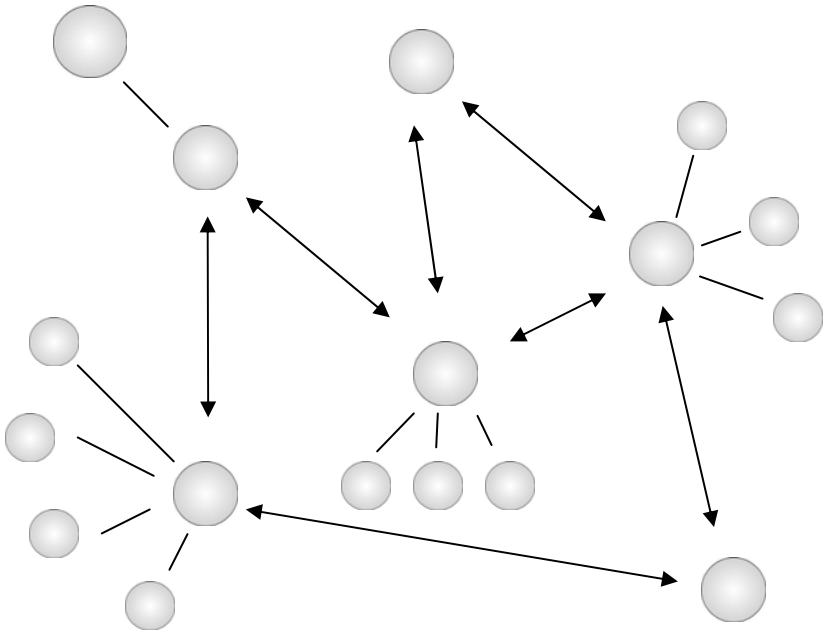

Abbildung 7: Zentral - mit Dezentraler Topologie

Das eMail-System des Internet basiert ebenfalls auf einem hybriden Zentral-Dezentral-System. Die eMail-Anwendungen auf den Rechnern kontaktieren stets ihren zentralen Mail-Server. Die Server tauschen anschließend den eMail Verkehr dezentral aus.

Ein solches System kann die Vorteile aus beiden Topologien nutzen. Der dezentrale Teil lässt eine große Erweiterbarkeit zu, ohne dass eine Kontrolle oder Einflussnahme möglich ist. Der zentrale Teil hilft dem System, die Datenströme besser zu verwalten, als dies ein dezentrales System alleine könnte.

6 Filesharing Applikationen und deren Technologie

Es werden hier nun Filesharing Netzwerk Topologien erklärt und danach ein bekanntes Beispiel angeführt, welches diese Technologie in der Praxis verwendet.

Die meisten Anwendungen sind dezentral oder als Hybrid aufgebaut. Jedoch weisen sie wesentliche Variationen auf.

Die moderne Entwicklung geht immer stärker zu den sogenannten Distributed Hash Tables (DHT). Diese Technologie ist auch abseits des reinen Filesharings interessant, da sie gute Skalierbarkeit und auch erweiterbare Sicherheit bietet.
DHTs sind mit Abstand am komplexesten, dafür aber meist sogar mathematisch genau durchdacht.

6.1 Zentral

In diesem Modell verwaltet und bedient ein Server eine große Anzahl an Nodes.
Nodes sind Teilnehmer, die eine Verbindung ins Internet haben. Die Teilnehmer loggen sich zunächst über den Server in das Netzwerk ein.
Dabei werden sie als Benutzer mit ihren Daten vom Server registriert und in eine zentrale Datenbank aufgenommen. Die Daten der Nodes können nun über das Netzwerk gefunden und abgerufen werden. Ein eingeloggter Node kann nun Suchanfragen an den Server senden, die er mit seiner Datenbank über die aktiven Nodes vergleicht.

Ist der Node mit dem Ergebnis der Anfrage zufrieden, stellt der Server eine Verbindung zwischen dem Besitzer der gewünschten Daten und dem Suchenden her. Der Datenaustausch findet autonom zwischen den Nodes statt, ohne weitere Teilnahme des Servers. Die Geschwindigkeit, mit der sich Peers austauschen können, ist abhängig von mehreren Faktoren. Primär sind es die Bandbreiten der Internetverbindungen, die die Geschwindigkeit des Datenaustausches zwischen den beteiligten Nodes bestimmen. Außerdem kommt hinzu, dass weitere Verbindungen von anderen Teilnehmern zum Node die Austauschgeschwindigkeit weiter verringern. Für Nodes mit langsamen Verbindungen ist es deshalb sinnvoll, keine oder nur wenige Daten anzubieten. Der Vorteil dieser Architektur ist das zentrale Verzeichnis mit der ständig aktualisierten Datenbank auf dem Server. Eine Suche ist hier sehr effizient, da sie sehr schnell ausgeführt werden kann und alle angeschlossenen Teilnehmer berücksichtigt sind. Ein Ergebnis von Anfragen ist somit für den aktuellen Zeitpunkt aussagekräftig.

Das zentrale Filesharing System, das durch Napster populär wurde, wird heutzutage von immer weniger kommerziellen Anwendungen unterstützt. Grund dafür ist, dass Anbieter wie Napster und AudioGalaxy gerade wegen ihrer zentralen Server gerichtlich stillgelegt werden konnten.

6.1.1 Napster

Napster funktioniert auf einer vergleichsweise simplen Client/Server Architektur. Folgendes Sequenzdiagramm (Abbildung 8) zeigt den vollständigen Vorgang eines Downloads im Netzwerk von Napster.

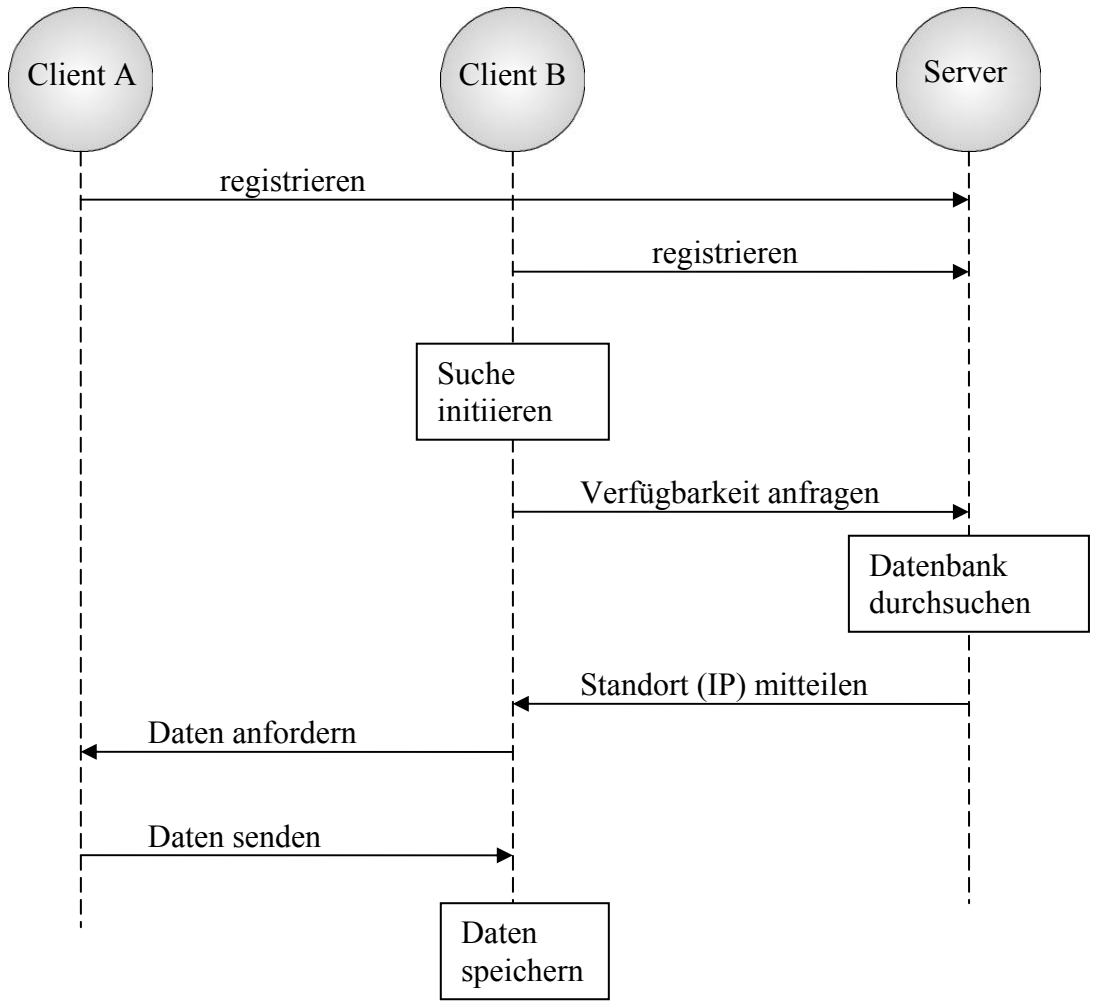

Abbildung 8: Downloadvorgang in einem zentralen Filesharing Netzwerk (Napster)

OpenNap

Seit der Schließung der Napster Server bilden die OpenNap [NA1] (Open Source Napster)-Server das neue Rückgrat des Napster-Netzwerks. Mit der Offenlegung des Protokolls gründete sich eine Open Source Gemeinde, die OpenNap ins Leben rief.

OpenNap ist inzwischen eine Multiplattform, die erfolgreich auf viele Betriebssysteme wie Linux, Windows, OS/2 oder auch Solaris portiert wurde.
Die frei verfügbare und gut dokumentierte Server Software ermöglicht es, auch ohne Expertenkenntnisse einen Computer in einen OpenNap-Server zu verwandeln.

Besitzer einer Internetverbindung mit hoher Bandbreite können ohne weiteres einen Server für den Dauerbetrieb erstellen. Die Entwickler haben inzwischen viele der Defizite und Einschränkungen der Originalserver von Napster beseitigt.

- Der Server Index ist erweitert worden und kann nun jeden beliebigen *MIME-Type* verwalten. Damit ist nun jeder Dateityp möglich, nicht mehr nur „mp3“.

- OpenNap-Server können ihre Indizes nun synchronisieren, was zu einer enormen Erhöhung der zur Verfügung stehenden Dateimenge führt.

- Für die Neuaufnahme in einem Netzwerk müssen Server Mindestanforderungen an Bandbreite und Erreichbarkeit besitzen. Operatoren kontrollieren diese Regeln, um eine dauerhafte Verfügbarkeit des Netzwerks zu garantieren und um Sabotageversuchen der Unterhaltungsindustrie vorzubeugen.

JNerve

Neben OpenNap existiert noch ein weiteres Open Source Projekt: JNerve [NA2].
Das Java-Programm JNerve basiert ebenfalls auf dem Napster-Protokoll, hat aber in Bezug auf Installation und Betriebssicherheit noch nicht den Reifegrad des OpenNap-Projekts. Erfahrene Java-Anwender können mit JNerve einen lauffähigen Java-Server erstellen, der alle wesentlichen Napster-Funktionen unterstützt.

OpenNap Client Software

Die Verbindung zu einem OpenNap-Server wird über eine Client-Software hergestellt. Mittlerweile gibt es etliche Napster-Client-Clones für die verschiedensten Betriebssysteme.

Die zahlreichen Client-Programme unterscheiden sich lediglich in ihren graphischen Oberflächen und ihrem Funktionsumfang, während die Arbeitsweise immer die gleiche ist.

In vielen Programmen können die Nutzer auch ihnen selbst bekannte Server eintragen. Im Internet existieren dazu diverse Seiten, die Serveradressen mit den dort getauschten Inhalten vorstellen. Meistens sind diese „offiziellen“ Server völlig überlastet und eine Einwahl ist erst nach einiger Zeit und vielen Versuchen möglich.

Populäre Filesharing Clients sind WinMX [WMX] und File Navigator [FNV].
Während WinMX neben seinem eigenen Netzwerk auch die Einbindung von OpenNap-Server erlaubt, kann File Navigator seine Suche auf OpenNap und Gnutella erweitern.

6.2 Dezentral/Hybride

In einem dezentralen oder hybriden Netzwerk können Nodes direkt miteinander kommunizieren, während in zentralen Systemen immer erst die Kommunikation mit einem Server vorausgehen muss.

Außerdem sind Nodes zugleich Server und Client, weswegen sie auch „Servents“ genannt werden. Servents erfüllen eine Serverfunktion, indem sie Suchanfragen prüfen und weiterleiten, während sie gleichzeitig als Client eigene Anfragen aussenden können. Jeder Servent fungiert als Knotenpunkt im Netzwerk, der mit weiteren Knoten verbunden ist und diese wieder mit weiteren. Der Ausfall eines oder mehrerer Knoten hat keine Auswirkungen auf den Rest des Netzwerkes, da jeder Node wie in einem Spinnennetz mit vielen Knotenpunkten verwebt bzw. verbunden ist. Das Fehlen einer zentralen Komponente bietet den Vorteil, nicht mehr von Leistung oder Ausfall eines Servers abhängig zu sein.
Ein weiterer Vorteil ist die Einwahl in die dezentralen Netzwerke, wo sofort eine Verbindung zustande kommt, während in zentralen Netzwerken der Server nur eine Kommunikation zulässt, wenn die nötigen Ressourcen für den Client vorhanden sind.

In der Praxis ist aber auch in dezentralen Systemen die erstmalige Teilnahme über eine zentrale Komponente nötig.
Durch die ständige Fluktuation von sich ein- und abwählenden Nodes gibt es nur wenige permanente Servents im Netzwerk, die bereits bekannt sind.
Anwendungen, die dezentrale Netze benutzen, verwenden deshalb lokale Host-Caches.

Die Host-Caches bestehen aus einem Pool von IP-Adressen und Ports von Servents.
Ein neuer Teilnehmer, der einsteigt, wird mit einer zufälligen Auswahl verbunden.
Suchanfragen werden zunächst an die direkt verknüpften Servents geschickt, die diese dann an weitere bekannte Nodes weiterleiten. Eine Antwort wird über denselben Weg zurückgeschickt. Der Datenaustausch selbst geschieht anschließend durch eine direkte Verbindung mit dem ausgewählten Servent.

6.2.1 Gnutella

Bei Gnutella handelt es sich um ein Netzwerk, das heute Grundlage für viele Filesharing Anwendungen ist. Gnutella war der Initiator für dezentrale File Sharing Netzwerke, so wie es Napster für den Bereich der zentralen Netzwerke war.
Das ursprüngliche Programm Gnutella wurde bereits während seiner Beta-Phase eingestellt. Der Entwickler Justin Frankel, der mit seiner Firma Nullsoft bereits den beliebten MP3-Player Winamp schuf, musste dem Druck des Mutterkonzerns AOL/TimeWarner nachgeben. Die Entwicklung einer potentiellen, urheberrechtsverletzenden Tauschbörse und der gleichzeitige Rechtsstreit mit Napster führten nach nur einem Tag zur Einstellung des Downloads der Betaversion.

Die Verbreitung der Gnutella-Software konnte dennoch nicht mehr rückgängig gemacht werden. Die wachsende Fan-Gemeinde von Gnutella wurde durch ihren Entwickler Frankel unterstützt, der anonym in einer IRC-Chat Sitzung wichtige Details über das Gnutella-Netzwerk preisgab, die für die Entwicklung von Gnutella-Klonen von elementarer Bedeutung waren. Mittlerweile ist das Protokoll vollständig dokumentiert und öffentlich zugänglich.

Der Einstieg in das Gnutella-Netzwerk ist inzwischen sehr einfach. Mit Hilfe eines der vielen Programme wie Limewire [LMW], die im Internet kostenlos erhältlich sind, kann ein Anwender problemlos Zugang in das Netzwerk erhalten. Ein spezieller Login oder die Wahl eines Benutzernamens für das Netzwerk ist nicht notwendig.
Für den Einstieg in das Netzwerk benutzen die Gnutella-Programme einen Host-Cache. Ein Programm kennt die jeweilige Adresse des Host-Cache, den jeder Anbieter eigenständig verwaltet. Der Host Cache ist ein Pool von IP-Adressen und Ports, der aus vorherigen und neuen Verbindungen mit dem Netzwerk gebildet wird. Eine zufällig ausgewählte Liste von IP-Adressen und Ports wird vom Host-Cache an den Gnutella-Servent gesendet. Die erhaltenen IP-Adressen und Ports sind die Andockpunkte an das Netzwerk. Das Versenden von Nachrichten an eine größere Anzahl an Adressen ist notwendig, da keine Garantie bestehen kann, ob einzelne Verbindungen noch aktiv sind. Gnutella-Anwendungen versuchen immer, eine bestimmte Anzahl an aktiven Verbindungen zu anderen Servents aufrecht zu erhalten.

Suchanfragen gehen zunächst über die verbundenen Servents, die sie an ihre bekannten Nodes weiterleiten, bis ein Time To Live (*TTL*)-Mechanismus das Weiterreichen der Anfrage stoppt. Jede Suchanfrage erhält dafür eine TTL-Variable. Bei der Weiterleitung über einen Servent dekrementiert dieser den Wert. Sobald TTL den Wert 0 erreicht, wird die Anfrage verworfen.
Ein sogenanntes *Flooding* würde auftreten, wenn ein Servent eine gleichlautende Anfrage die jedoch den gleichen Urpsrungsservent haben, von verschiedenen Servents erhält. Um dies zu verhindern, werden die Pakete verglichen und bei einer Redundanz nur eine Anfrage weitergeleitet.

Da nicht das gesamte Netzwerk durchsucht wird, bekommt man je nach Einstiegspunkt und Suchhorizont unterschiedliche Ergebnisse.
Der Datei-Austausch selber findet jedoch direkt zwischen den beteiligten Servents statt, ohne Auswirkungen für das Gnutella Netzwerk.

In Abbildung 9 leitet der Servent A eine Suchabfrage mit TTL=2 ein:

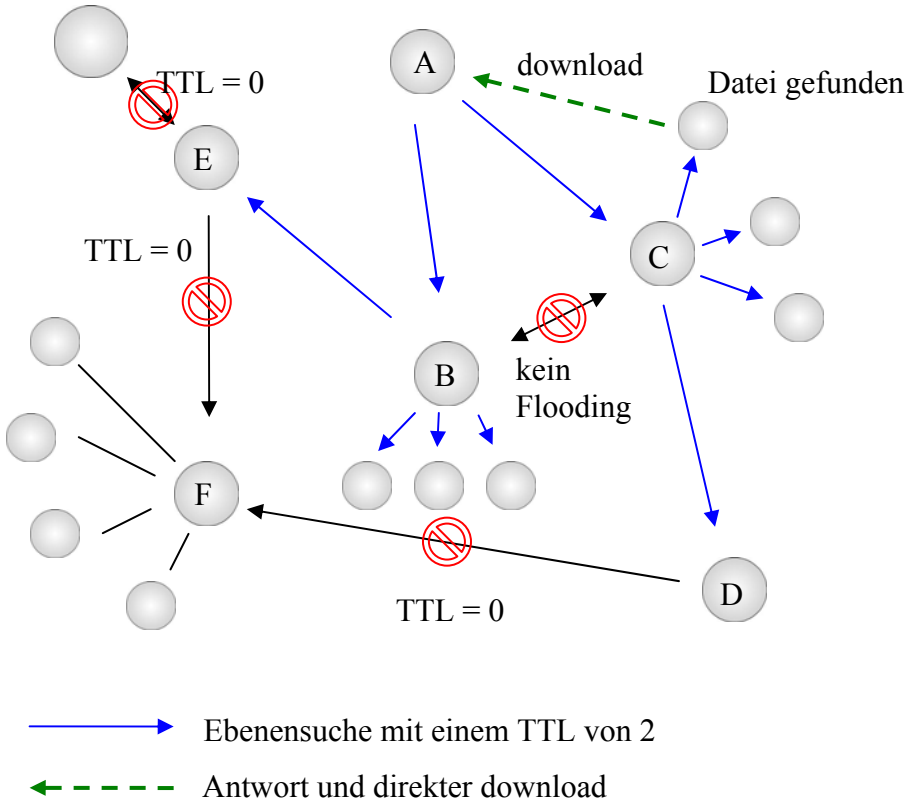

Abbildung 9: Suchabfrage im Gnutella Netzwerk

Als Übertragungsprotokoll für den Datentransfer wird das HTTP (Hypertext Transfer Protokoll) genutzt, das auch im world wide web (www) zum Einsatz kommt. Einige Web-Suchmaschinen machen sich dies zu Nutze und greifen auf Gnutella-Dateien zu. Da hierbei jedoch nicht das gleichzeitige Anbieten von Dateien möglich ist, sind solche Zugriffe sehr unbeliebt in der Community.
Eine besondere Funktion haben *Ping*-, *Pong*- und Push Pakete im Gnutella-Netzwerk. Die Servents schicken in regelmäßigen Abständen Ping-Pakete an all ihre Nachbarknoten. Die Empfänger antworten mit einem Pong-Paket, das dem Sender ihr Vorhandensein bestätigt. Die Paktete werden wie Suchanfragen verteilt. Auch sie haben eine begrenzte TTL. Pong-Pakete enthalten neben IP-Adresse und Port auch statistische Informationen über den Datenbestand. Diese Informationen werden ebenfalls an die Host-Cache gesendet und füllen die Listen mit neuen Adressen. Push-Pakete sind speziell notwendig für Servents, die sich hinter Firewalls befinden.

Das ständige Routen der Nachrichten nimmt in einem Gnutella -Netzwerk einen erheblichen Teil der Bandbreite ein. Als dezentrales Filesharing Netzwerk, das auf die feste Verbindungsstruktur von Client/Server Netzen verzichtet, muss eine intensivere Kommunikation betrieben werden, was zu einer höheren Netzauslastung führt. Trotzdem

hat dieser dezentrale, verteilte Ansatz im Hinblick auf die Schwächen und Anfälligkeiten von Client/Server seine Berechtigung.

Folgender partielle Ausschnitt des Gnutella Netzwerkes (Abbildung 10) soll die vollkommen dezentrale Anordnung verdeutlichen. Jeder Eckpunkt bildet hier einen Servent im Netzwerk. Eine Linie ausgehend von einem Punkt zu einem anderen stellt eine aufrecht erhaltene Verbindung dar.

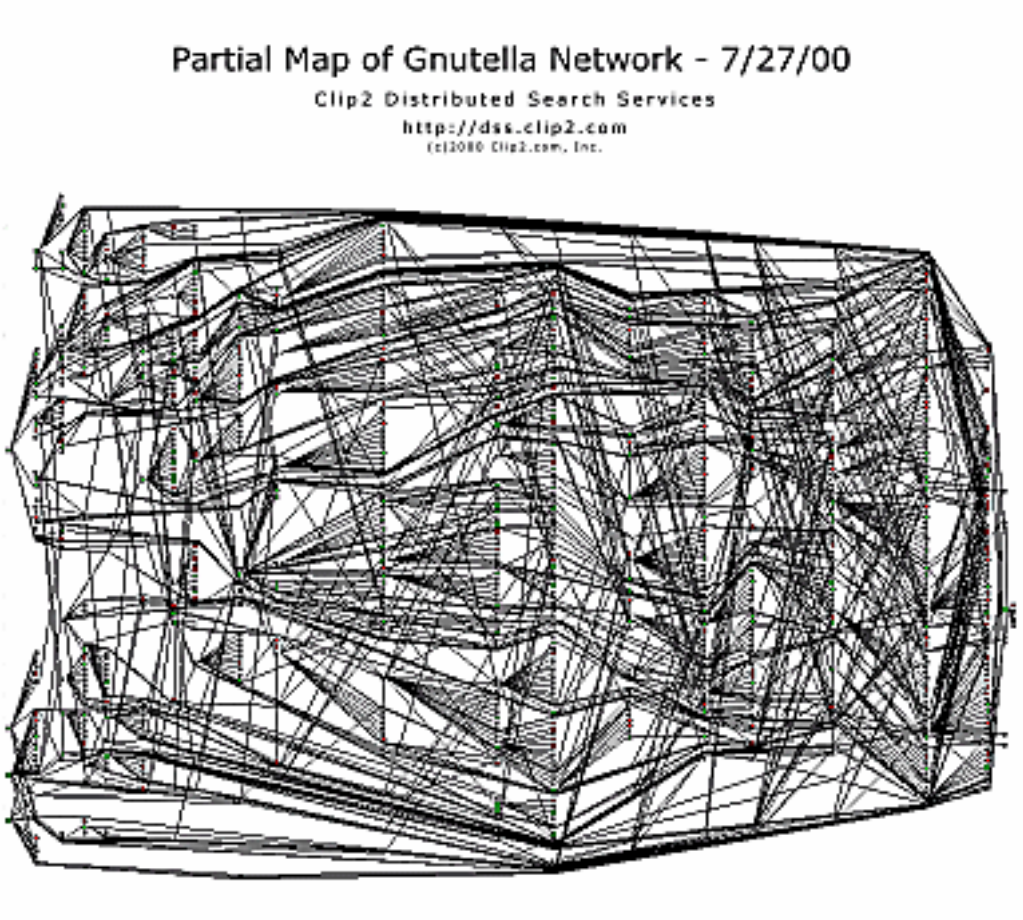

Abbildung 10: Ausschnitt des Gnutella Netzwerkes [GTL3]

6.2.2 Mike's Protocol (Gnutella2)

Im November 2002 wurde im Gnutella Developers Forum [GTL2] (GDF), das Gnutella2 Protokoll von Michael Strokes vorgestellt. Dies führte sofort zu heftigem Disput unter den Entwicklern von führenden Filesharing Applikationen.

Es gab zwei Meinungen unter den Entwicklern. Eine davon sagte, dass das Ziel von Gnutella2 ist, nicht das Protokoll entscheidend weiter zu entwickeln, sondern mit dem alten Gnutella0.6 zu brechen und alte Behelfslösungen zu überwinden.
Eine andere Meinung unter Entwicklern von LimeWire war, dass die Weiterentwicklung fast ausschließlich ein Publicity Gag sei und kündigten an, es nicht zu unterstützen. Auch wurde dem Protokoll in Entwicklerkreisen der Name „Mike's Protocol" anstatt Gnutella2 gegeben.

Jedoch benutzt auch die neue Version den alten Handshake „GNUTELLA CONNECT/0.6“ zum Verbindungsaufbau, welcher bereits in den alten Spezifikationen dokumentiert ist. Dies sahen einige Entwickler als Schritt um rückwärtskompatibel zu sein, während andere dies als Versuch werteten, ein eigenständiges Netzwerk mit Verbindung zum alten Netzwerk zu etablieren.
Durch den Zwiespalt bei den Entwicklern entstand recht schnell ein *Flamewar*.
Nichtsdestotrotz wurden am 26.März 2003 die Spezifikationen freigegeben. Später folgten nur noch genauere und leicht abgeänderte Details.
Aufgrund dieser Streitigkeiten ist Gnutella2 bis heute nur in sehr wenigen Filesharing Applikationen vertreten.

Funktionsweise:

Während Gnutella alle Nodes gleich behandelt, teilt Gnutella2 diese in zwei Gruppen. In so genannte leaves (Blätter) und *Hubs*. Während die Blätter nur ein oder zwei Verbindungen zu Hubs besitzen, halten Hubs hunderte von Verbindungen zu Blättern und anderen Hubs aufrecht. Dies sorgt für eine Umwandlung des rein dezentralen in ein hybrides Netzwerk.

Ein Node versucht zunächst einmal eine Liste von Hubs zu bekommen. Erfolgt nun eine Suchanfrage, kontaktiert der Teilnehmer dann die Hubs in der vorher erstellten Liste. Dabei merkt er sich, welcher Hub schon besucht wurde. Dies wird solange fortgeführt, bis die Liste abgearbeitet worden ist, oder aber eine genau definierte Suchschranke überschritten wird.
Damit kann sichergestellt werden, dass ein Blatt, ohne das Netzwerk stark zu belasten, eine verbreitete Datei findet. So können sogar Daten aufgefunden werden, die im gesamten Netzwerk nur einmal vorhanden sind.
Wenn sich ein Blatt bei einem Hub anmeldet, erstellt dieser einen Index, mit dem er weiß, welche Dateien die Blätter anbieten.

Der Index wird auch Query Routing Table genannt und ist ein Hash-Index der Suchbegriffe. Der Hub generiert mit diesen Informationen eine kombinierte Version mit allen Schlüsselwortlisten der Blätter, die sich bei ihm angemeldet haben.

Diese optimierte Liste wird nun an die Nachbar Hubs verteilt, um die Anzahl der weitergeleiteten Suchabfragen zu verringern.

6.2.3 eDonkey2000

Die genaue Einordnung von eDonkey2000 [EDK] bereitet einige Schwierigkeiten. Von manchen wegen seiner zentralen wie dezentralen Komponenten als hybrides Netzwerk dargestellt, bezeichnet sich eDonkey2000 gerne als dezentrales Netzwerk.
Grundsätzlich handelt es sich auch um ein dezentrales Filesharing Netzwerk, das große Ähnlichkeiten mit dem OpenNap Modell aufweist.

eDonkey2000 bestand aus zwei Programm-Komponenten. Die eDonkey2000 Server Anwendung ermöglicht es, den eigenen Rechner zur Server-Plattform für das Filesharing Netzwerk werden zu lassen und selber als Administrator zu fungieren.
Das Gegenstück ist der eDonkey2000 Client, ein Programm mit graphischer Oberfläche und Funktionen, wie man sie auch von anderen Client Programmen gewohnt ist. Ein

Anwender kann sowohl Client- bzw. Server-Programm, als auch beide zugleich betreiben.
Das Client Programm verfügt bereits über eine rudimentäre Liste aus Servern, in die sich bequem weitere Server mit ihren Adressen eintragen lassen. Normalerweise beginnt der Client damit, für ein Login in das Netzwerk alle Server aus der Liste abzuklappern. Damit verfährt er nach der vorgeschriebenen Reihenfolge, bis die Einwahl in einen Server erfolgreich ist. Der Benutzer kann auch versuchen, gezielt einen Server zu wählen.
Der erreichte Server ist nun der Hauptserver für den Client. Er verwaltet den Index aller Dateien von Nodes, die bei ihm eingeloggt sind. Eine normale Suchanfrage bleibt für die Nodes auf ihren Hauptserver beschränkt. Der Filetransfer selber findet direkt zwischen den Clients statt. Siehe Abbildung 11.

Die Kommunikation zwischen eDonkey2000-Servern findet nur sporadisch statt. Die einzige Verbindung, die von Zeit zu Zeit aufgebaut wird, dient dazu, eine aktuelle Liste von aktiven, bekannten Servern zu pflegen. Diese Liste wird auch an das Client Programm weitergegeben. Um eine Suchanfrage über den Hauptserver hinaus zu initiieren, muss eine erweiterte Suche gestartet werden. Das Programm versucht dann, weitere Server aus der Liste zu kontaktieren.

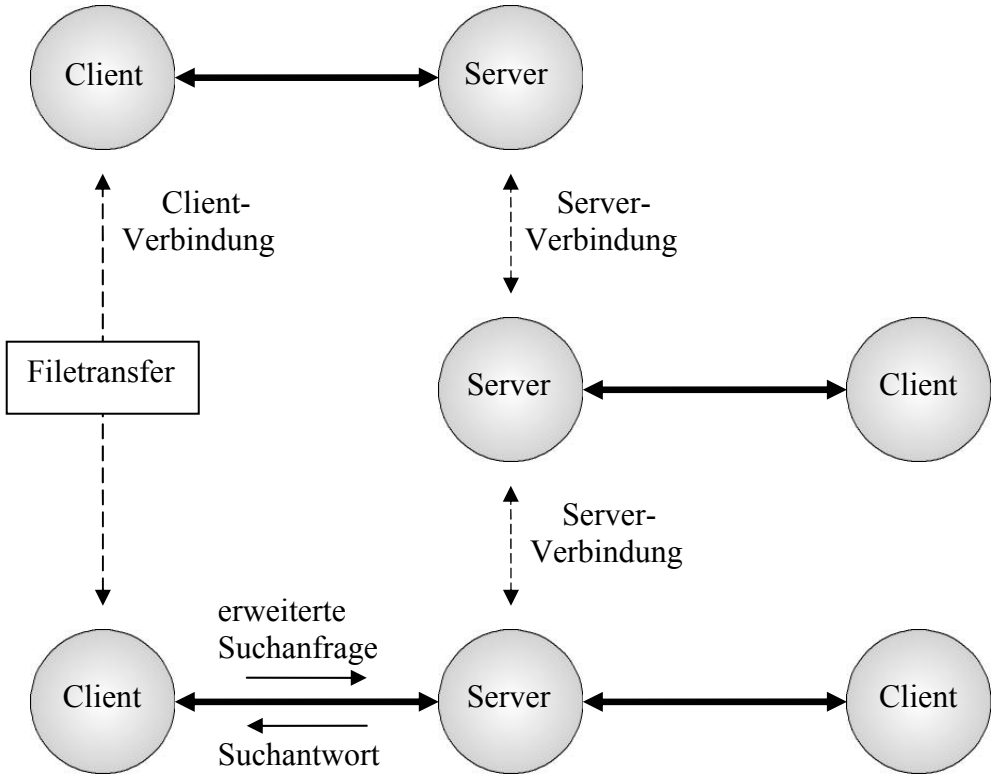

Abbildung 11: Struktur und Verbindungsarten im eDonkey2000-Netzwerk

Das Herzstück des eDonkey2000 Netzwerks ist das „Multisource File Transfer Protocol“ (MFTP). Dieses System erlaubt nicht nur das Wiederaufnehmen von abgebrochenen Downloads (auto resuming) und das Herunterladen aus mehreren Quellen, sondern geht noch darüber hinaus. Es ist möglich, dass Nodes, obwohl sie nur unfertige Bruchstücke einer Datei besitzen, die jeweils noch fehlenden Teile austauschen. Dazu indiziert der

Client alle freigegebenen Dateien mit einem Hash-Wert. Anschließend kann eDonkey2000 anhand der eindeutigen Identifikation das Herunterladen von verschiedenen Nutzern koordinieren. Dies ist ein großer Fortschritt im Filesharing System, denn bisher war es nur möglich gewesen, Teile aus kompletten Dateien zu erhalten. Die Architektur ist damit sehr effektiv.

Ein Beispiel für den Downloadmechanismus (Abbildung 12):
Servent 1 hat alle Teile einer Datei ($p_1p_2p_3p_4p_5p_6p_7p_8$). Da Servent 2 und 3 verschiedene Teile besitzen, können sie gegenseitig die fehlenden Teile austauschen.
Servent 4 kann nun bereits p7 von Servent 3 downloaden.

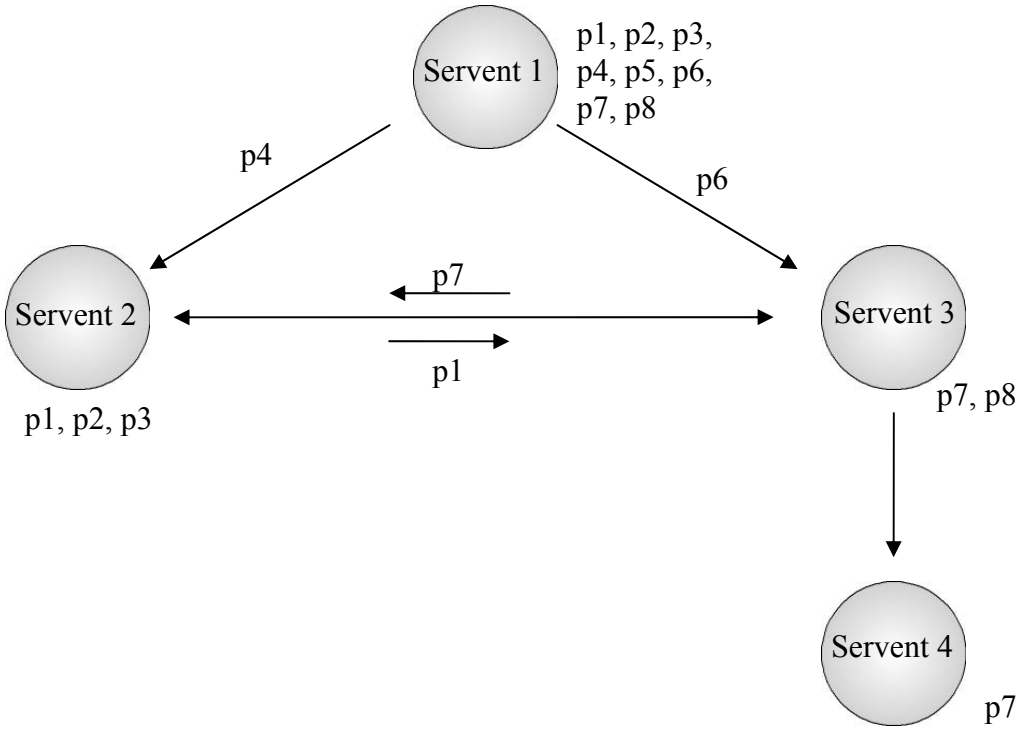

Abbildung 12: Der Downloadmechanismus von eDonkey2000

Mittlerweile gibt es eDonkey2000 jedoch nur noch in inoffiziellen Versionen von Programmierern der Open-Source-Gemeinde. Diese sind vollständig dezentralisiert und auf „Distributed Hash Tables“ umprogrammiert worden, ähnlich Kademlia.

6.2.4 FastTrack

FastTrack [FST] ist mittlerweile einer der häufigsten benutzten Techniken in Filesharing Netzwerken.
Der bekannteste Nutzer dieser Technologie ist KaZaA.
Der Aufbau des Netzwerks besteht bis auf einen zentralen Login Server, ähnlich wie

bei „Mike's Protocol", auf einer flexiblen Struktur, die den Nodes verschiedene Aufgaben zuweist.
Als neuer Benutzer ist zunächst eine einmalige Anmeldung unter einem Nicknamen erforderlich. Später erledigt der Servent Anmeldung bzw. Login automatisch. Vor einem Login stellt das Programm fest, ob ein Rechner in seiner Performance und Bandbreite für Serveraufgaben geeignet ist und ernennt sich in diesem Fall zum „Superpeer".

Das Netzwerk ist so ausgelegt, dass leistungsfähige Rechner die SuperPeers, die weniger leistungsstarken Rechner „Peers" darstellen. Somit wird die Bandbreite des Netzwerks entlastet, da nicht mehr jeder Peer mit allen anderen kommunizieren muss.
Die automatisch ernannten SuperPeers bilden das eigentliche FastTrack-Netzwerk.
Nach dem Login erhalten Rechner, die eher als Peer geeignet sind, IP-Adressen und Ports von einem oder mehreren SuperPeers. Eine Liste von Adressen von festen SuperPeers sind den Filesharing-Anwendungen beigelegt.
Peers senden die Informationen über ihre angebotenen Dateien, wie auch Suchanfragen an ihren Superpeer. Die Suchanfragen werden allen anderen Superpeers weitergereicht (Abbildung 13).

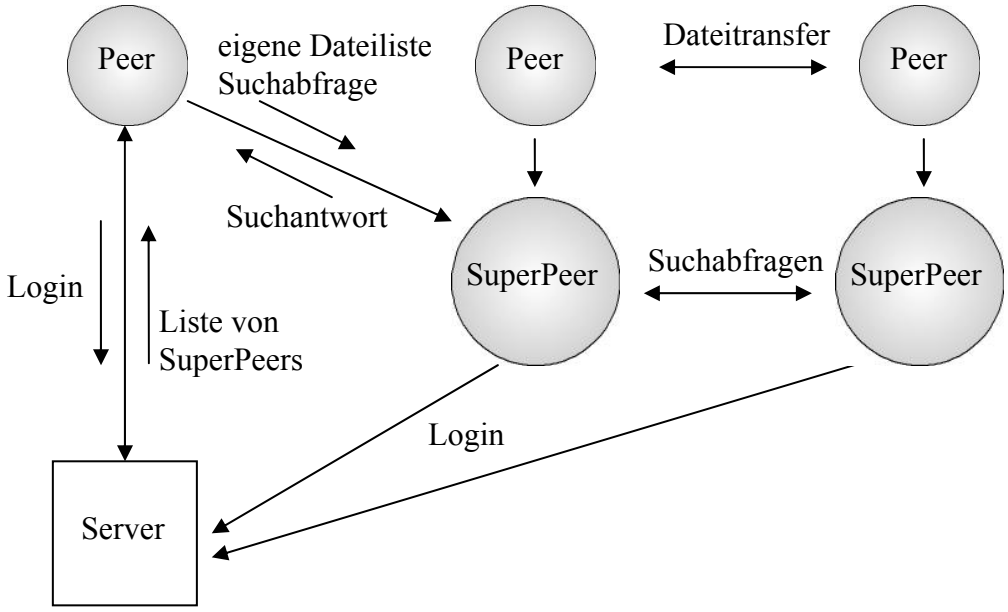

Abbildung 13: Struktur und Verbindungsarten im FastTrack-Netzwerk

Die Suchergebnisse sind sehr ausführlich, da sie die speziellen Informationen in den Dateien auslesen. Diese „Tags", die man als Aushängeschild einer Datei verstehen kann, enthalten diverse Informationen, wie Autor oder Titel.
Durch eine Baumstruktur werden identische Dateien aufgeschlüsselt, die von mehreren Peers bereitgestellt sind. So ist es möglich, zwischen vielen Benutzern gleichzeitig eine Datei auszutauschen oder gezielt auf einzelne Peers zuzugreifen. Dabei kann auch das gesamte Verzeichnis eingesehen werden, welches ein Peer anbietet. Der Austausch von Dateien erfolgt wie üblich direkt zwischen den Peers.

Das FastTrack-Netzwerk eignet sich für beliebige Dateitypen.
Das Netzwerk arbeitet so effizient, dass es täglich und zu jeder Zeit von über zwei Millionen Menschen genutzt werden kann. Versuche seitens der Medienindustrie, FastTrack zu schließen, waren bisher erfolglos.

6.3 Distributed Hash Tables (DHTs)

Eine Distributed Hash Table (DHT) ist eine Datenstruktur, die das allgemeine Problem in herkömmlichen Filesharing Netzwerken löst, den Speicherort einer Datei sicher zu bestimmen. Ein Ziel ist es auch, möglichst effizient zu sein. Die Daten sollen hierbei möglichst gleichmäßig über alle Nodes im Netzwerk verteilt werden.
Jeder Servent ist dabei ein Behälter einer *Hashtabelle*.

Das Netzwerk muss ständige Ausfälle und Beitritte von Nodes überstehen. Außerdem soll es sich selbst organisieren und gut skalieren.

Die Node Adressen, die meist aus einem Hash Wert der IP-Adresse und des Ports bestehen, bieten eine wichtige Grundlage.
Die zu speichernden Daten werden ebenfalls nach diesem System gehasht. Der Servent, dessen ID am ehesten dem Daten-Hash entspricht, speichert somit ein *Tupel* aus key und value. Der key bezeichnet somit den Inhalt und dient für Suchabfragen. Im value können nun die Daten direkt stehen oder er bezeichnet eine Adresse, wo diese gespeichert sind.

Die Netzwerke Chord und Kademlia basieren grundlegend auf DHTs.
Nicht eingegangen wird darauf, wie die Daten letztendlich auf einem Knoten abgelegt werden, da dies ein Problem ist, das unabhängig von der Struktur des verwendeten DHT ist. Ebenso nicht betrachtet werden hier Aspekte der Sicherheit eines DHT, also die Frage, ob man Daten böswillig verändern kann oder das Auffinden von Schlüsseln stören kann.

6.3.1 Chord

Chord [SMKKB] ist im Jahre 2001 am Massachusetts Institute of Technology (MIT) entwickelt worden. Es stellt ein Verfahren zur Organisation einer DHT dar.
Wie eine Anwendung auf Chord aufbauen kann, zeigt das *Kapitel 10 Proof of Concept-GCFS*.

Bei den meisten herkömmlichen Filesharing Netzwerken kann es passieren, dass eine bestimmte Information existiert, diese aber nicht gefunden wird.
In Chord jedoch kann garantiert werden, dass jede Suchanfrage definitiv beantwortet wird. Die so genannten Lookups (Suchanfragen) sind hierbei sehr effizient und schnell. Es ist eindeutig nach einer gewissen Zeit feststellbar, ob eine Information existiert oder nicht. Falls diese existiert, gibt es auch einen Servent, der die angeforderten Daten bereitstellen kann. Auch sorgen die Algorithmen hinter Chord dafür, dass die Redundanz der Daten gering gehalten wird, aber dennoch kann eine Duplizierung, falls notwendig, jederzeit erfolgen.

6.3.1.1 Grundlagen

Das Entwicklungsziel von Chord war, Mängel in den bestehenden Filesharing Netzwerken auszumerzen.
Die größten Probleme der älteren Netzwerke sind:

- Performance von Suchabfragen
- Antworten auf Suchabfragen sind nicht eindeutig

Lastenausgleich

Die Idee hinter Chord ist, eine Performance zu liefern, die statistisch beweisbar ist. Zugleich soll das System nicht zu komplex und ebenso universell einsetzbar sein.
Um all dies zu erreichen, benutzt Chord nur Schlüssel und Werte. Es funktioniert also wie eine Hashtabelle, die verteilt auf die Nodes im Netzwerk ist.
Die Applikationsschicht über Chord kann frei entscheiden, welche Art von Daten sie als Werte verwendet. Hierbei sind viele Varianten möglich, wie weitere Schlüssel bis hin zu kompletten Dateien. Durch die Verwendung von guten Hashingalgorithmen wird dafür gesorgt, dass die Schlüssel bzw. deren Werte auf die im Netzwerk befindlichen Knoten gleichmäßig verteilt werden.

Dezentralisation

Die Schlüsselverteilung über das Netzwerk wird durch einen guten Hash Algorithmus geregelt (im Original wird hierbei die Hash Funktion *SHA1* verwendet). Entscheidend in einem freien Netzwerk ist hierbei, dass der generierte Hashschlüssel nicht rückwirkend auf das Ursprungsdokument schließen lässt.
Weichen zwei Dokumente nur im Geringsten voneinander ab, führt dies zu unterschiedlichen Schlüsseln. Statistisch gesehen verteilen sie sich somit gleichmäßig auf das Netzwerk.

Gleichberechtigung

Ein wichtiger Punkt von Chord ist, dass jeder Servent gleich behandelt wird. Es gibt also keinerlei Server-Client Strukturen oder Einteilungen in Supernodes. Alle Nodes sind somit Servents.

Skalierbarkeit

Die Laufzeit von Suchabfragen ist fast komplett unabhängig von der Anzahl der Teilnehmer. Prinzipiell gilt, dass eine Suche nach einem Schlüssel ohne Optimierung in S(ld n) Schritten ausgeführt wird. „n“ beschreibt hierbei die maximale Anzahl von Nodes im Netzwerk.

Verfügbarkeit

Chord garantiert, dass jeder Schlüssel, falls vorhanden in der zugesicherten Laufzeit gefunden wird. Die Suchantworten sind damit eindeutig. Im Gegensatz zu Netzwerken wie Gnutella existiert eine Information im aktuellen Zustand des Netzes nicht, wenn sie nicht beim ersten Mal gefunden wird.

6.3.1.2 Formale Eigenschaften

Das Chord Protokoll sieht vor, dass jeder Servent jedes Dokument, sowie jeder sonstige gespeicherte Wert durch einen eindeutigen Schlüssel identifiziert wird. Hierbei besteht kein Unterschied für den Schlüssel zur Identifizierung von Nodes und denen von Werten.

Die Servents in einem Chord Netzwerk sind in einem virtuellen Kreis angeordnet. Der Knoten, dessen Identifikationsschlüssel gleich dem Schlüssel des gesuchten Wertes ist, hat auch eben diesen Wert, oder die Information, wo die Daten zu finden sind, gespeichert. Sollte der Knoten, der diesen Schlüssel speichern sollte, nicht existieren, so erhält der Servent mit dem nächsthöheren Schlüsselwert diesen. Das entspricht der nächsten Node im Uhrzeigersinn im Kreis. Sollte kein Servent mit einer höheren ID existieren, so findet ein Kreisübersprung statt.

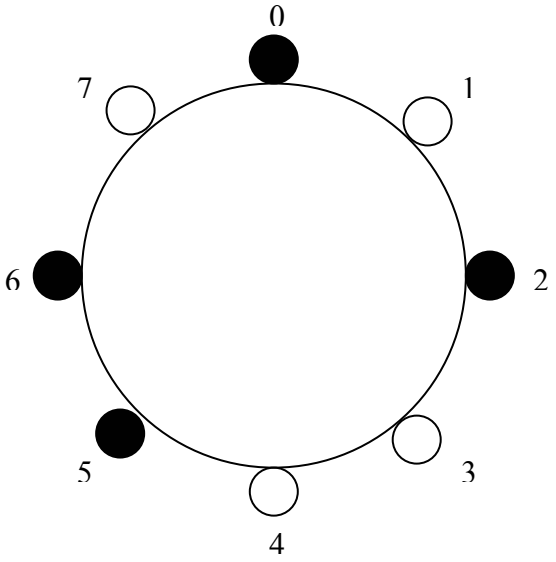

Abbildung 14: Ein Chord Netzwerk mit den Knoten 0,2,5 und 6

Jeder Servent im Kreis merkt sich dabei seinen Nachfolger darin (Successor). Somit ist es möglich, bei einer Suche den kompletten Kreis zu durchlaufen und sicher zu stellen, ob eine Information existiert. Ohne eine Optimierung kann damit eine Suche maximal n-1 Schritte dauern. Wobei $n = 2^m$ den Adressraum, also die maximale Anzahl der Servents im Netzwerk darstellt. *ld(n)* oder m erhält hierbei eine entscheidende Bedeutung für die Menge an Nodeadressen, die gespeichert werden müssen.

6.3.1.3 Anfragen

Diese Suchweise wäre jedoch zu langsam, daher ist es nötig, dass jeder Servent mehr Informationen über seine Umgebung erhält.
Eine Suche soll nicht mehr als m Schritte benötigen und dabei nicht mehr als m Informationen über andere Servents benötigen.

Es wird also nicht nur der Successor und der Predecessor gespeichert, sondern auch eine sogenannte Fingertabelle. Diese enthält m Einträge.
Für einen Servent s berechnen sich die einzelnen Finger wie folgt:
i gibt jeweils den Finger an (1,2,… m).
s_{ID} gibt den Schlüssel der Node an.
Die Fingerstartwerte ergeben sich nun aus: $fi = s_{ID} + 2^{i-1} \bmod 2^m$ wobei $1 \le i \le m$.

Es ergeben sich somit als Finger $s_{ID}+$ 1, 2, 4, 8, … n/2. Die tatsächlichen verlinkten Knoten können dabei variieren. Falls ein Servent nicht vorhanden ist, wird der nächste im Uhrzeigersinn im Kreis verwendet.

Folgendes Beispiel (Abbildung 15) soll die Funktion einer Suche im Chord Netzwerk mit Hilfe der Fingertabelle erläutern. Existiert ein vorgesehener Node nicht, so zeigt der entsprechende Zeiger auf den nächsten vorhandenen Node. Ein vorhandener Teilnehmer im Netzwerk wird in der folgenden Illustration als fester Punkt dargestellt.
Um die Suche besser darzustellen, wird sie in mehrere Schritte eingeteilt.

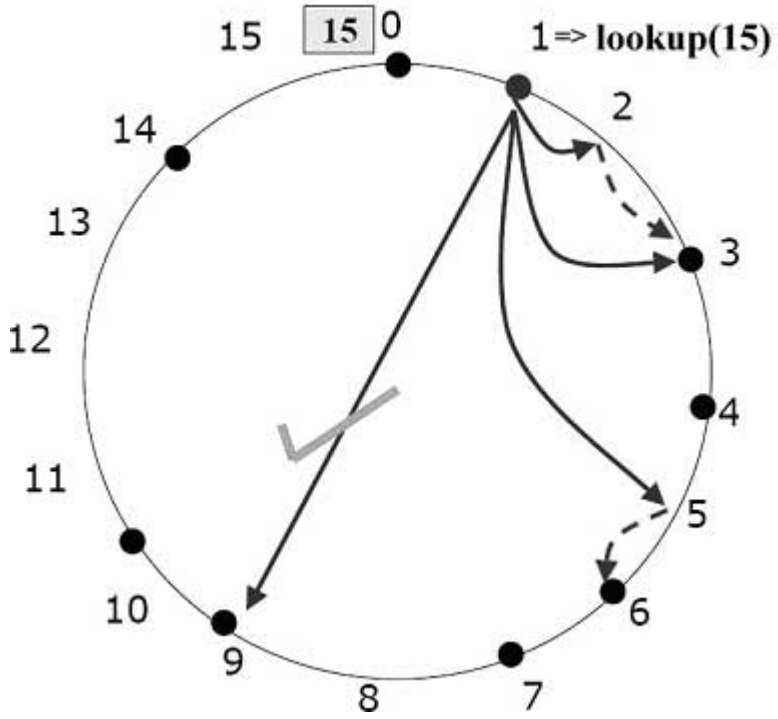

Abbildung 15: Chord Lookup, Schritt 1

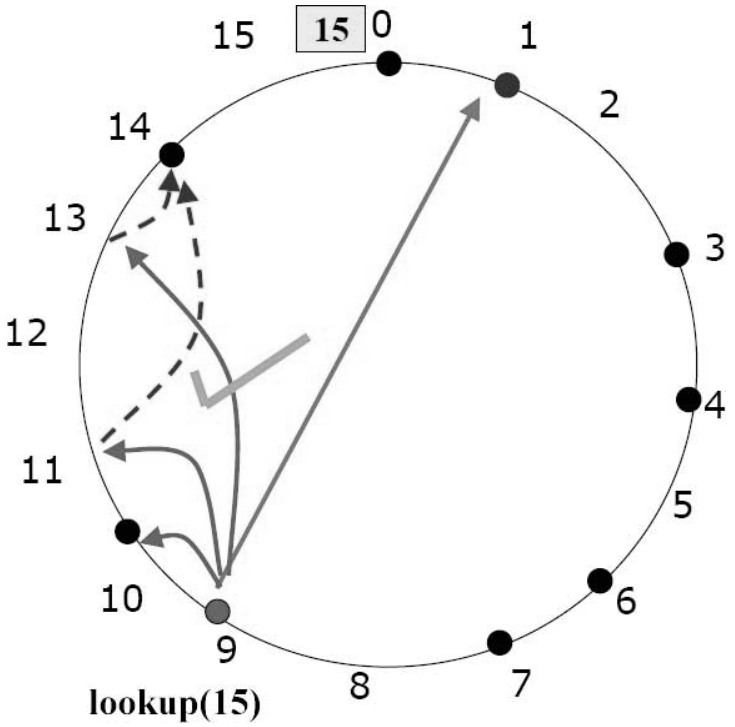

Abbildung 16: Chord Lookup, Schritt 2

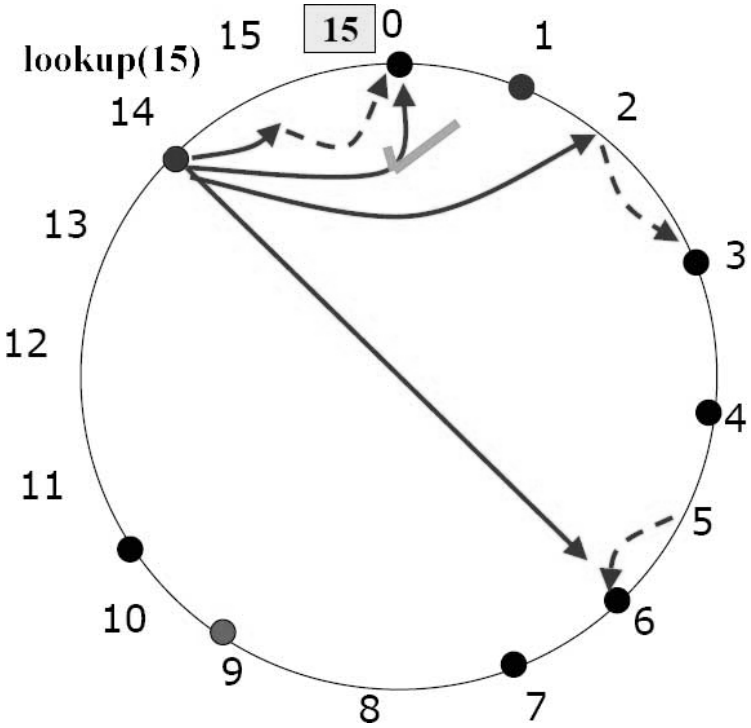

Abbildung 17: Chord Lookup, Schritt 3

Anhand der Funktionsweise der Finger kann man erkennen, dass Chord bei Suchabfragen hervorragend skaliert. Selbst bei wesentlich höherer Teilnehmerzahl halbiert sich der verbleibende Abstand jeweils um die Hälfte. Wird ein Schlüssel nicht direkt gefunden, so wird jeweils der Servent gefragt, der am besten erscheint, die Anfrage weiter zu verarbeiten.

6.3.1.4 Netzwerkdynamik

Ein Chord Netzwerk ist nicht statisch. Es kann ständig passieren, dass ein neuer Knoten hinzukommt und ein anderer verschwindet.
Somit kann es passieren, dass sich Successor und Finger eines Servents ändern.
Eine Änderung im Netzwerk zieht eine Stabilisierung nach sich.
Nur dadurch kann eine Suchabfrage in maximal S(ld n) Schritten garantiert werden.
Da sich nun auch die Knoten, die die entsprechenden Schlüssel beherbergen ändern, müssen mehrere Schlüssel-Werte Paare übertragen werden. Es lässt sich jedoch zeigen, dass wenn der N'te Knoten zu einem Netzwerk hinzugefügt wird, nur etwa 1/N Schlüssel-Werte verschoben werden müssen.

Um das Einfügen eines neuen Knotens zu erleichtern, dient die Speicherung des Vorgängers (Predecessor). Somit lässt sich im Kreis auch gegen den Uhrzeigersinn navigieren. Es sind nun folgende Schritte notwendig, einen Servent in das Netzwerk zu integrieren:

1. Successor und Predecessor von Servent initialisieren
2. Finger, Successoren und Predecessor der relevanten Nodes aktualisieren
3. Nachricht an die darüber liegende Applikation, um die Schlüssel-Werte Paare zu aktualisieren und zu verschieben.

Die folgenden Abbildungen 18,19,20 demonstrieren, wie ein Servent dem Netzwerk hinzugefügt wird (join). Die Blöcke stellen die Schlüssel auf den jeweiligen Servents dar.

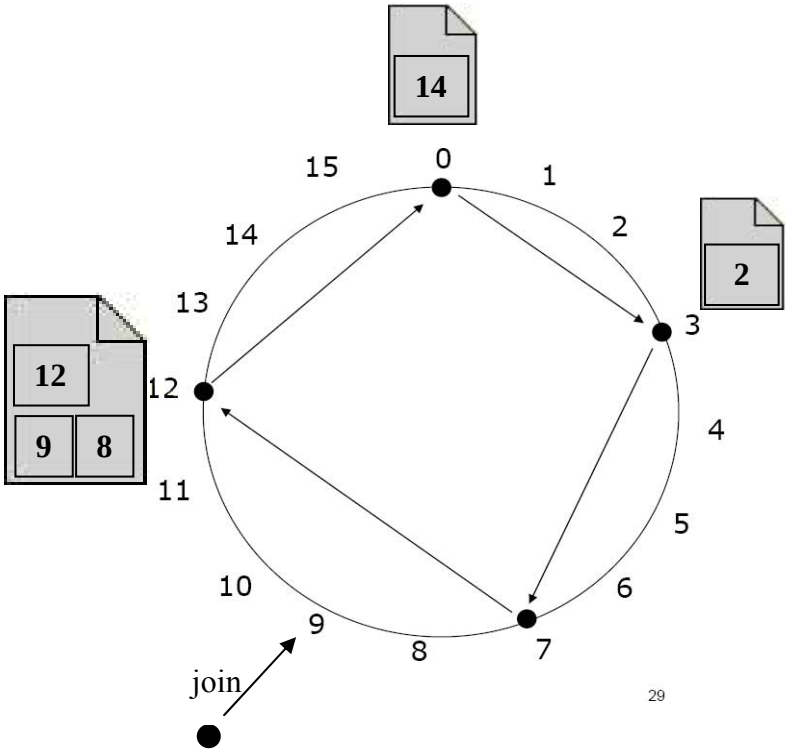

Abbildung 18: Join eines Servents in einem Chord Netzwerk, Schritt 1

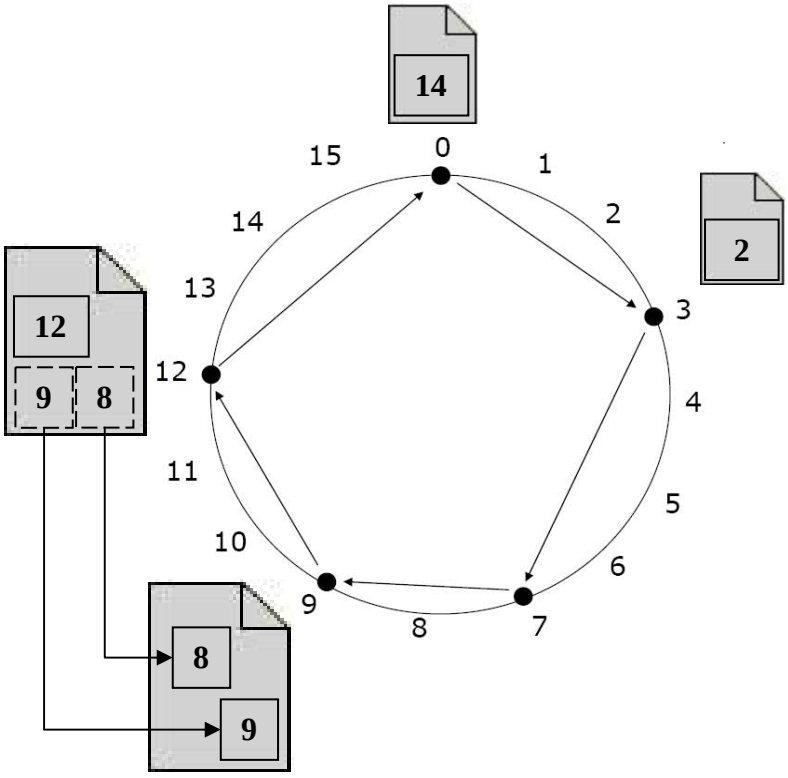

Abbildung 19: Join eines Servents in einem Chord Netzwerk, Schritt 2

Ein Servent kann nicht nur das Chord Netzwerk joinen, sondern natürlich auch verlassen. Wie dies vor sich geht, demonstrieren folgende Abbildungen 20, 21:

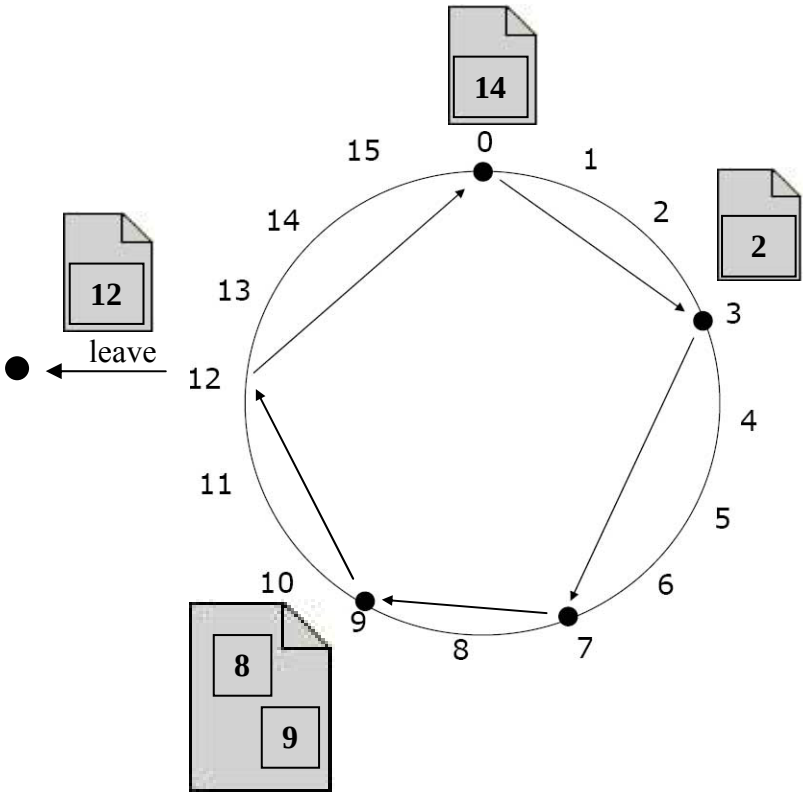

Abbildung 20: Leave eines Servents in einem Chord Netzwerk, Schritt 1

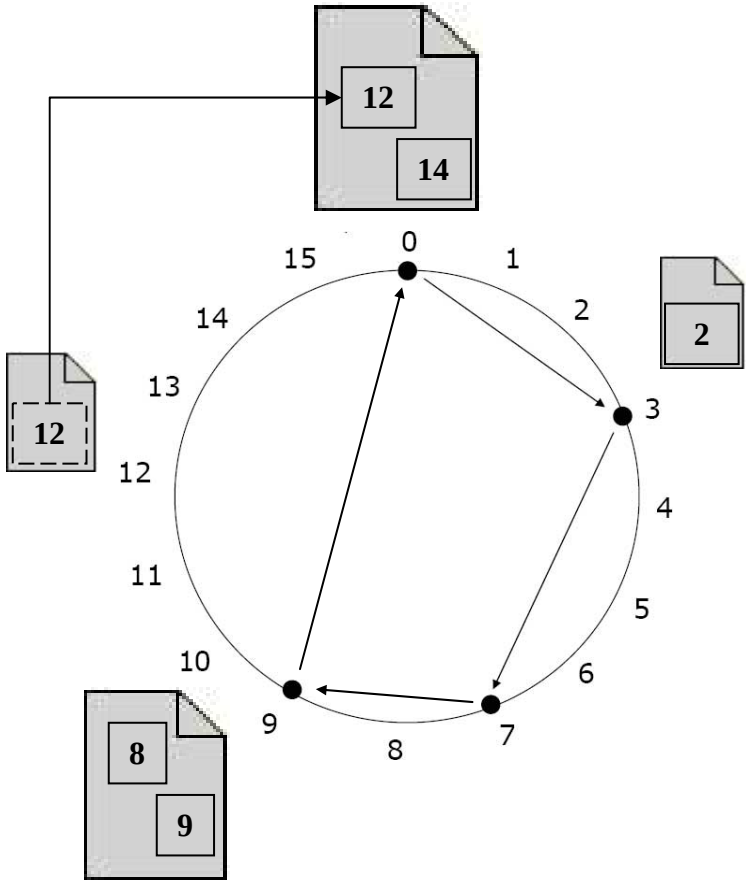

Abbildung 21: Leave eines Servents aus einem Chord Netzwerk, Schritt 2

Verlässt nun ein Knoten durch Ausfall das Netzwerk, so würde der Predecessor von diesem seinen Successor verlieren und eine Lücke würde entstehen.
Um diese Problematik in den Griff zu bekommen, speichert ein Servent nicht nur einen Successor, sondern auch „m“ (ld(n)) Nachfolger. Dies sorgt dynamisch je nach Netzwerkgröße für eine ausreichende Sicherheit.

Problematisch für die Geschwindigkeit von Chord sind häufige joines und leaves. Daher benutzt dieses Netzwerksystem einen Stabilisierungsalgorithmus.
Dieser wird nach gewisser Zeit, je nach verfügbarer freier Bandbreite beliebig oft ausgeführt.
Mindestens jedoch wenn ein join oder leave festgestellt wird, sollten die direkt betroffenen Servents (Successor und Predecessor) diesen ausführen. Der Algorithmus sorgt dafür, dass die Zeiger auf Successor, Predecessor und Finger bei einer Veränderung im Netzwerk (leave, join) richtig gestellt werden.
Nodes, die den Stabilisierungsalgorithmus ausführen, können ihren Nachfolger dazu veranlassen oder empfehlen, diesen ebenfalls auszuführen.

6.3.2 Kademlia (KAD)

Kademlia [MMK] wurde an der New York Universität im Jahre 2002 entwickelt.
Diese Technologie ist bereits in dem bekannten Filesharing System eMule umgesetzt.

Bei Kademlia handelt es sich um ein DHT System. Es baut ein selbstorganisierendes, strukturiertes Netzwerk auf und regelt Kommunikation und Austausch zwischen den Nodes.

Das besondere Kennzeichen von Kademlia ist ein gemeinsamer virtueller Adressraum. Die Knoten IDs und Schlüssel werden auf eine Länge von 160 Bit gehasht. Damit eine Information im Netzwerk gefunden werden kann, wird der Hashwert dieser Information als key aufgefasst. Key-value Paare werden auf mehreren Nodes mit einer zum Schlüssel nahen ID gespeichert (siehe übernächsten Absatz). Ein Servent, der den dort gespeicherten Schlüssel sucht, kann dann den Wert zum entsprechenden key erhalten und anhand dessen bestimmen, von welchem Netzwerkteilnehmer er die gewünschten Daten erhalten kann. Um einen Abstand zu ermitteln, wird bitweises *XOR* verwendet. Da der Schlüssel und die Servent-IDs auf dem gleichen Raum definiert sind, kann auch der Abstand eines keys von einer Node-ID ermittelt werden.

Der Abstand wird mittels XOR Metrik berechnet:
a XOR b = c
dies würde zum Beispiel wie folgt aussehen:
1001 XOR 1110 = 0111
Das Ergebnis dieser Operation wird als Abstand aufgefasst.
Das MSB (most significant bit) ist in diesem Beispiel die führende 0. Das LSB (least significant bit) ist die letzte 1. Ein Abstand von 1000 ist damit wesentlich größer als 0001.

Abbildung 22 zeigt den Aufbau eines kleinen Kademlia Netzwerks.

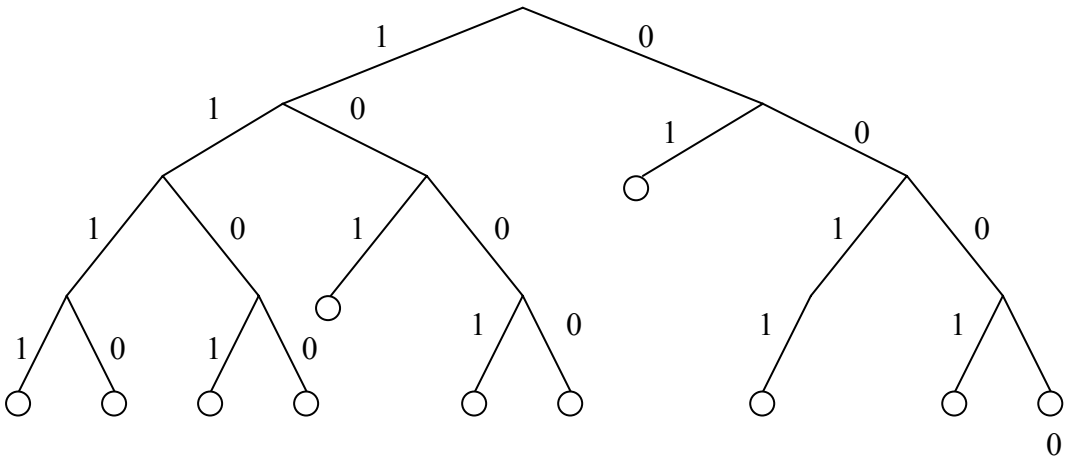

Abbildung 22: Beispielhafter Kademlia Aufbau

Bekannte Nodes werden in Kademlia in Listen gehandhabt. Dafür wird der vollständige Adressraum in Intervalle zerlegt. Diese reichen von je 2^i bis $2^i + 1$ mit $0 <= i < 160$. Jeder Servent führt nun 160 solcher Listen, um den ganzen Adressraum erfassen zu können.
In einer solchen Liste ist jeweils Platz für k viele Tupel, die aus IP-Adresse, UDP-Port und Node-ID bestehen. Jedes Tupel beschreibt einen Servent des Netzwerks.

Diese Liste ist nach dem zeitlichen Abstand des Kontaktes mit dem jeweiligen Tupel sortiert. Kürzlich bestätigte Kontakte stehen am Ende der Liste.
"k" ist ein systemweiter Parameter. Dieser wird so gewählt, dass ein Ausfall aller Knoten einer solchen Liste innerhalb einer Stunde so gut wie unmöglich ist.
In der Praxis ist k=20 ein guter Richtwert.
Aus diesem Aufbau ergibt sich, dass ein Servent über ein hohes Wissen über seine nahen Nachbarn verfügt, jedoch nur wenig über Nodes weiß, die sehr weit entfernt sind.

Um diese entscheidenden Listen aktuell zu halten, werden aus dem Netzwerk eingehende Nachrichten zur Aktualisierung der „k-*Buckets*" (alle gespeicherten Tupel) verwendet.

- Ist der Absenderservent bereits in den k-Buckets vorhanden, wird er in der Liste an das Ende sortiert
- Ist der Absenderservent noch nicht in den k-Buckets vorhanden und das entsprechende Bucket noch nicht voll, wird der Servent am Ende der Liste angefügt
- Ist der Absenderservent noch nicht in den k-Buckets vorhanden und ist das entsprechende Bucket außerdem bereits voll, wird der am Beginn der Liste stehende Node kontaktiert. Reagiert diese jedoch nicht, wird der neue Servent am Ende der Liste eingefügt, andernfalls wird er verworfen. Sollte er eingefügt werden, muss der erste in der Liste entfernt werden

Dieses Verfahren besitzt folgende Eigenschaften:

- Geringer Suchaufwand, mit maximal S((log(n)) Schritten
- Ausgleich des hohen Netzwerkverkehrs durch key/value-Paar Caching
- Bei Nodeausfall folgt eine Schlüsselsuche in Kademlia ohne Latenzsteigerung durch parallele Anfragen
- Besonders geeignet wenn sich die Teilnehmerzahl häufig ändert
- Hohe Effizienz des Datenverkehrs

Kademlia realisiert eine Suchfunktion in einem strukturierten Filesharing Netzwerk mit logarithmischer Komplexität, was für DHT-Ansätze allgemein typisch ist.

6.4 BitTorrent

BitTorrent war eigentlich nur dazu gedacht, große Datenmengen in den Griff zu bekommen, die zum Beispiel bei Verwendung von FTP-Servern auftreten. Dies wurde jedoch so gut gelöst, dass es massiv für Filesharing Netzwerke benutzt wird und eigentlich nur dadurch seinen hohen Bekanntheitsgrad erlangte.
BitTorrent hat einige wesentliche Grundbestandteile. Diese werden in den folgenden Kapiteln genauer geschildert, da sie Grundlage für neue Umsetzungen sind, die diese Technologie immer beliebter werden lässt. Eine Implementation von BitTorrent findet man zum Beispiel im *Kapitel 8.1 Audio/Video Broadcasting*.

6.4.1 Bestandteile

BitTorrent Client

Von BitTorrent Clients gibt es mittlerweile einige Implementierungen. Die bekannteste Version ist der Originalclient des Autors Bram Cohen. Dieser ähnelt in der Verwendung sehr dem Windows Schema und dessen Dialogen.

Torrent

Als Torrent wird eine Datei bezeichnet, die mittels BitTorrent angeboten wird und für die auch eine jeweilige .torrent Datei existiert (Metainfo Datei). Ebenso muss ein funktionierender Tracker existieren (mehr dazu im Absatz *Tracker*).

.torrent Datei (Metainfo Datei)

Die .torrent Datei stellt den Einstiegspunkt dar. Sie befindet sich auf dem jeweiligen Server und ist dort mit dem entsprechenden MIME-Type „application/x-bittorrent“ eingetragen. Somit kann der Client die Datei erkennen und den Download entsprechend starten.

Tracker

Der Tracker hilft allen, die an einer bestimmten Datei interessiert sind, einander zu kontaktieren. Hierfür wird das geläufige HTT Protocol benutzt. Der Tracker selber dient dabei ausschließlich für das Vermitteln der Clients und hat keinen Einfluss auf das Verteilen der Daten selbst.

Seed

Ein Seed ist ein Client, der bereits die komplette Datei besitzt, aber dennoch für andere Teilnehmer für den Upload zur Verfügung steht. Der Provider der Datei muss zu Beginn einen Seed stellen, der mindestens einmal die komplette Datei verteilt. Theoretisch kann dieser anschließend abgeschaltet werden. Tatsächlich sollte er aber weiterhin noch zur Verfügung stehen (je beliebter die Datei selber, desto kürzer, da sich schnell neue Seeds bilden).
Nach einem vollständigen Downloadvorgang sind zwar sämtliche Teile der Datei unter den Teilnehmern vorhanden, jedoch könnte einer dieser Clients vorzeitig offline sein, ohne seinen Teil weiter verbreitet zu haben, was den gesamten Download verzögern, oder sogar unterbrechen würde.

Swarm

Sämtliche Interessenten, die an einem Torrent File interessiert sind und den Download gestartet haben, bezeichnet man als Swarm (Schwarm).
Lediglich der Tracker kennt sämtliche Clients des Schwarms. Er wird den einzelnen Nodes jedoch nicht alle anderen Teilnehmer mitteilen, sondern lediglich eine benötigte Teilmenge.

6.4.2 Download

Damit eine Datei mittels BitTorrent heruntergeladen werden kann, muss zuerst eine .torrent Datei (Metainformationen) erstellt werden. In dieser Datei finden sich Informationen über Länge, Name und Sonstiges über den Inhalt. Außerdem wird darin der zu verwendende Tracker und die SHA1 Hashwerte der Teilstücke, in die die Datei aufgeteilt wird, angegeben. Die nötigen Einstellungen können dabei automatisiert von einem Tool gemacht werden, wobei der Ersteller selber nur noch wenig verändern muss.

Bevor der Download für alle Interessenten verfügbar ist, muss der Provider der Datei einen Client zu einem Seed ernennen. Des Weiteren wird die .torrent Datei zum Beispiel auf einem Webserver publiziert.

Nun kann ein Client mit dem Tracker, der in der .torrent Datei eingetragen ist, Kontakt aufnehmen, um weitere Teilnehmer zu erhalten. Der Tracker selber stellt dem Client eine zufällig generierte Liste von ihm bekannten Teilnehmern aus dem Schwarm zusammen und übermittelt diese. Von jedem dieser Clients können nun einzelne Teile downgeloaded werden. Sie werden somit zu Servern für Clients, die von ihnen downloaden.

Der restliche Vorgang findet nun komplett innerhalb des Schwarms selber statt und der Tracker erhält nur noch Statusinformationen für die statistische Auswertung des Torrents. Auch informiert jeder Teilnehmer die anderen im Schwarm über seinen Status und welche Teile von ihm noch benötigt werden.

Ein Client versucht nun bei möglichst vielen ihm bekannten Partnern Teile zu bekommen. Zur gleichen Zeit erhält derselbe aber die Anfragen der anderen. Der Client kann nun entscheiden, ob er denjenigen, die anfragen, den Teil senden oder den Upload schlichtweg verweigert. Diese Negierung einer Abfrage wird als „Choking“ bezeichnet. Mehr Informationen zu Choking im *Kapitel 6.4.3.4 Choking*.

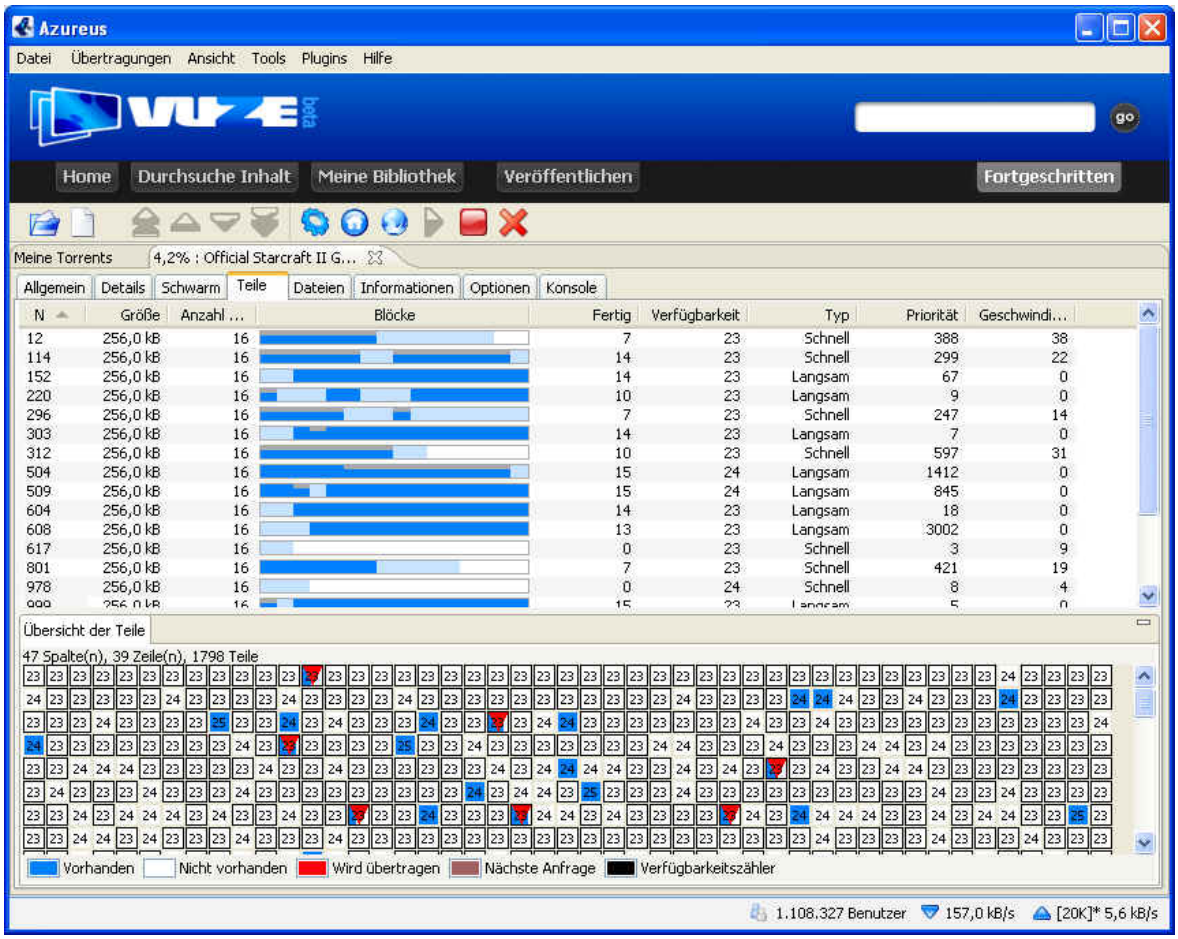

Abbildung 23: Download durch den Azureus3 BitTorrent Client

6.4.3 Funktionalität

6.4.3.1 Fragmentierung

Damit eine Datei noch schneller übertragen werden kann, werden die einzelnen Teile noch weiter verkleinert und in sogenannte Fragmente aufgeteilt. Diese sind normalerweise zwischen 16 und 512 Kilobytes groß.
Um die Wartezeiten bei Anfragen zu verringern, behält der Client immer mehrere Nachfragen auf Fragmente offen.
Sobald ein Fragment angekommen ist, startet der Client eine neue Anfrage. So entstehen keine Wartezeiten und der Client ist andauernd beschäftigt.

6.4.3.2 Auswahl der Teile

Die richtige Auswahl und die Reihenfolge, in der die Teile herunter geladen werden, ist von entscheidender Bedeutung für die gesamte Geschwindigkeit des Filetransfers.
BitTorrent hat dabei einige Technologien auf Lager, um dies zu erreichen.

Strict Priority

Sobald ein Fragment eines Teiles angefordert wurde, werden die anderen Fragmente dieses Teiles priorisiert.
Dies sorgt für eine möglichst schnelle Vervollständigung eines Teils. Vollständige Teile sind entscheidend, da sie wiederum zum Upload zur Verfügung gestellt werden können. Durch diese Priorisierung wird der gesamte Download wesentlich beschleunigt, da die Downloadrate streng von der gesamten Uploadrate abhängig ist.

Rarest First

Hat ein Client die Wahl zwischen mehreren Teilen, die er herunterladen könnte, so entscheidet er sich immer für den Teil, der unter den ihm bekannten Teilnehmern am seltensten vorhanden ist. Diese Strategie bietet den Vorteil, dass dadurch seltene Teile weiter verbreitet werden. Somit können andere Clients diesen Teil nun ebenfalls leichter und mit höherer Geschwindigkeit herunterladen.

Außerdem ist dieser Mechanismus ein Schutz dagegen, dass ein seltener Teil unvorhergesehen aus dem Netz verschwinden könnte. Dies ist vor allem dann von entscheidender Bedeutung, wenn es keine Seeds im Netz gibt.
Zusätzlich sorgt Rarest First dafür, dass Clients immer verschiedene Teile zu Beginn von einem Seed anfordern, da nach dem erstmaligen Download von Teilen ein anderer Teil den Platz des Rarest First einnimmt.
Somit wird die Wahrscheinlichkeit insgesamt erheblich verringert, dass es zu einem Totalausfall einer Datei kommen kann, da diese möglichst schnell dupliziert wird.

Random First

Eine Ausnahme zu Rarest First ist direkt nach dem Start des Clients gegeben. Da noch kein Teilnehmer außer dem Seed Teile besitzt, die er uploaden könnte, ist es lediglich wichtig, so schnell wie möglich an einen vollständigen Teil zu gelangen.
Die seltensten Teile im Netz sind aber mit hoher Wahrscheinlichkeit nicht die, die der Client mit der besten Geschwindigkeit herunterladen kann. In diesem Fall muss also mit Rarest First gebrochen werden und es wird ein Teil per Zufall ausgewählt.

6.4.3.3 Endgame Modus

Während des Downloadvorgangs ist es nicht weiter tragisch, wenn ein Teil der Datei von einem Partner geladen wird, der signifikant langsamer ist als der Client.
Ärgerlich wird dies jedoch, wenn es sich um das letzte Teil des Downloads handelt, da dann der Abschluss des Downloads von dieser langsamen Node abhängt.
Daher existiert der Endgame Modus. Dieser setzt ein, sobald alle Fragmente, die der Client nicht besitzt, einmal angefragt wurden.
Die verbliebenen Fragmente werden nun bei allen anderen Nodes nachgefragt, statt wie bisher bei dem Node, bei dem der passende Teil angefragt wurde.
Sollten daraus redundante Sendevorgänge entstehen, so werden diese abgebrochen, sobald das Fragment das erste Mal vollständig eingetroffen ist.

6.4.3.4 Choking

Bei BitTorrent, wie bei jedem anderen Filesharing System natürlich auch, ist jeder Downloader bestrebt, die Datei, an der er interessiert ist, möglichst schnell zu erhalten. In einem Filesharing Netzwerk stellt sich also die Frage, wie Upload und Download einzustellen sind, damit möglichst jeder dieses Ziel erreicht. Da es keine zentrale Ressourcenverwaltung gibt, muss jeder Client selbst dafür sorgen, seine Downloadgeschwindigkeit zu maximieren. Hierfür versucht er, bei so vielen Partnern wie möglich Teile der Datei downzuloaden.
Welchen Clients Upload gestattet wird, wird mit einer Variante von *tit-for-tat* entschieden. Fällt die Entscheidung, einem Uploadrequest zu folgen negativ aus, so spricht man von ”choking”. Es wird dann temporär nicht zu dem anfragenden Client hochgeladen.
Der Algorithmus, nach dem entschieden wird, ob hochgeladen wird oder nicht – der Choking-Algorithmus also – ist demnach der zentrale Teil des Protokolls, der seine Geschwindigkeit maßgeblich beeinflusst, da Upload und Download direkt zusammenhängen. Ein optimaler Algorithmus dafür würde den Schwarm zu einem *pareto-effizienten* System erheben.

6.4.3.5 Optimistic Unchoking

Würde ein Client immer nur zu beliebig vielen Partnern uploaden, bei denen er gerade die beste Downloadgeschwindigkeit erreicht, so wäre das nicht optimal.
Es könnten ja noch Clients existieren, die eine bessere Downloadgeschwindigkeit anbieten könnten, dies aber nicht tun, da der Client gerade nicht zu ihnen hochlädt.
Um dieses Problem zu umgehen, existiert das Optimistic Unchoking. Dabei wird alle 30 Sekunden ein Client, der eigentlich nicht in das Choking Schema passt, frei geschaltet, um zu prüfen, ob dies die Performance verbessert.

6.4.3.6 Anti-Snubbing

Es kann passieren, dass ein Client von allen Partnern, von denen er bisher herunter geladen hat, blockiert wird. Sollte dieser Fall eintreten und ein Client für eine gewisse Zeit (Standard: 1 Minute) kein einziges Fragment mehr erhalten, so geht er davon aus, dass er von diesem Partner „gesnubbt” ist.
Er blockiert dann den entsprechenden Client und versucht einen anderen Partner freizuschalten, um wieder pareto-effizient zu sein. Das heißt, dass das System nicht verändert werden kann, ohne dass nicht mindestens ein Teilnehmer hinterher unglücklicher ist. Dies stellt eine Abweichung vom Optimistic Unchoking dar, bei dem immer zufällig ein Partner frei geschaltet wird. Durch dieses Vorgehen erholen sich die Downloadraten deutlich schneller, als wenn darauf gewartet würde, dass per Optimistic Unchoking ein passender Partner gefunden wird.

6.4.4 Sicherheit

Für Filesharing Netzwerke ist es heutzutage nicht nur wichtig, einen Download in möglichst kurzer Zeit zu ermöglichen. Faktoren wie Vertraulichkeit, Integrität und Verfügbarkeit werden immer wichtiger. Anonymität ist für den rechtlichen Aspekt sehr interessant.

6.4.4.1 Vertraulichkeit

Bei BitTorrent ist es unmöglich zu verhindern, dass persönliche freigegebene Dateien zumindest theoretisch von anderen zum eigenen Nachteil benutzt werden können.
Die Kommunikation von BitTorrent ist generell komplett ungeschützt und kann problemlos mitnotiert werden.
Entscheidend hierbei sind jedoch weniger die Daten an sich, als die Tatsache, welche Personen an dem Austausch beteiligt sind.

6.4.4.2 Integrität

Die Integrität der zu verteilenden Daten ist ein wichtiger Bestandteil eines Verteilungsprotokolls.
Um die Integrität der Teile zu gewährleisten, sind in der Metainfo Datei die SHA1 Hashes jedes Fragmentes vermerkt. Der Client kann also direkt nach dem Download eines Teiles prüfen, ob dieser in Ordnung ist, oder ob die Daten korrupt sind. In letzterem Fall wird der Teil verworfen und der Client, von dem er geladen wurde, gesperrt. Anschließend wird erneut versucht, betreffende Fragmente von den restlichen im Netzwerk verbliebenen Clients zu erhalten.
Dateien durch Einwirkung von Aussen dauerhaft zu schädigen, ist damit so gut wie unmöglich, während dies in anderen Filesharing Netzwerken bereits Gang und Gäbe ist.

6.4.4.3 Verfügbarkeit

Die Verfügbarkeit dürfte derzeit vielleicht das größte Manko von BitTorrent darstellen. Eine Datei ist logischerweise nur so lange verfügbar, wie im gesamten Netzwerk mindestens einmal alle Teile vorhanden sind.
Es muss also entweder ein Client existieren, der die gesamte Datei besitzt, oder die Summe der Teile aller Clients muss einmal die gesamte Datei ergeben. Es hängt also von der Beliebtheit einer Datei oder dem guten Willen der Teilnehmer ab, wie lange eine Datei verfügbar ist. Teilnehmer die nach dem kompletten Download den Client noch geöffnet lassen, dienen dann als Seeds.
In seltenen Fällen kommt es auch vor, dass der Tracker irgendwann nicht mehr verfügbar ist. Damit endet natürlich auch die Verfügbarkeit aller Dateien, die diesen Tracker verwendet haben.
Derzeit werden „trackerlose“ BitTorrent Systeme entwickelt. Die Trackerfunktion wird dabei von der Clientsoftware übernommen.

6.4.4.4 Anonymität

In einem Interview sagte Bram Cohen, der Autor von BitTorrent einmal, dass Anonymität für ihn bei der Entwicklung von BitTorrent keine Rolle gespielt hat und er sogar absichtlich darauf verzichtet, Möglichkeiten hierfür zu entwickeln. Er wolle damit verhindern, dass BitTorrent zu sehr in den illegalen Sektor abrutscht und vermehrt für die illegale Verbreitung von Filmen oder Musik verwendet wird (vgl. Bram Cohen; 23.11.2005). Demzufolge ist es nicht verwunderlich, dass BitTorrent keinerlei Techniken zur Verfügung stellt, die es dem Nutzer ermöglichen, anonym am Verteilungsprozess teilzunehmen.
Einziger kleiner Vorteil von BitTorrent gegenüber anderen Filesharing Systemen ist die Tatsache, dass immer nur eine Datei geteilt wird. Es ist also nicht so einfach möglich, die Verteilung aller Dateien, die über BitTorrent verbreitet werden, zu überwachen.

6.4.5 Vorteile (the edge over…)

Der entscheidende Vorteil bei BitTorent ist nicht nur, dass ein einzelner Teilnehmer seine Daten schneller bekommen kann, sondern auch der Schwarm als Gesamtes gesehen, braucht wesentlich weniger Zeit, um die Daten zu erhalten.
Hierbei wird die gesamte Bandbreite genutzt, um die geforderten Dateien schnell downzuloaden. Das Diagramm soll mit verschiedenen Parametern zeigen, wie sich je nach den Parametern des Systems die Zeit im Vergleich zu einem normalen Download verkürzt.

Die einzelnen Testumgebungen wurden mit einer variablen Anzahl von Nodes, Bandbreite und maximalen Verbindungen pro Client simuliert (Abbildung 24).
Als fix wurde die Anzahl der Teile der Datei mit 20 festgesetzt. Ebenso wurde ein Teil einer Datei mit 256KB festgelegt. Somit ist die Testdatei genau 5MB groß.
Die Nettozeit die ein Download auf herkömmliche Weise benötigt, lässt sich wie folgt berechnen:

$$t = \frac{Teilchengröße[bytes]}{download[bytes/s]} \cdot Teile \cdot (Nodes - 1)$$

Da jede herunterzuladene Datei von einer zentralen Komponente geleitet wird, kann durch den Ausfall dieser Komponente diese Datei nicht mehr bezogen werden. Ein Benutzer mit der gesamten Datei muss eine neue Torrent-Datei und damit einen Tracker erstellen. Das zeigt, dass man hier Ausweichmöglichkeiten hat und es zu keinem totalen Ausfall des Netzwerks kommt.

Das Netzwerk an sich ist gut skalierbar, da beliebig viele Tracker online gestellt werden können, die eine bestimmte Anzahl von Benutzern aufnehmen, aber die Beschränkung der Benutzeranzahl für den Tracker ist dennoch gegeben.

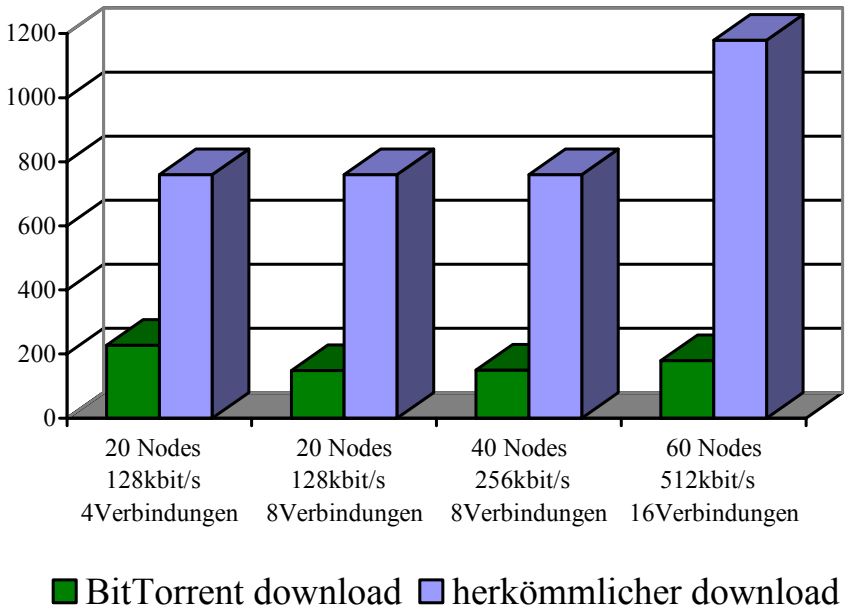

Abbildung 24: Vergleich von Downloadzeiten in s

7 Bootstrapping

Unter Bootstrap versteht man „das Ladeprogramm“.
Problematisch bei dezentralen Filesharing Netzwerken ist der erstmalige Einstieg.
Installiert man eine Filesharing Anwendung auf seinem Computer, so hat dieser noch keine Verbindung zu anderen Nodes.
Wie erhält dieser nun diese Verbindungen?

Mögliche Methoden dafür sind, mit der Anwendung eine Liste mit IP Adressen und Portnummern auszuliefern, die zuvor häufig online waren.

Eine eher „unschöne“ Möglichkeit ist das Pingen, bis man so per Zufall einen anderen Servent findet. Dies wird aber nicht gerne gesehen, da es viel Traffic erzeugt, bis ein Servent in das Netzwerk integriert ist. Außerdem ist diese Methode nur bei weit verbreiteten Netzwerken akzeptabel, da der erstmalige Bootstrap Vorgang sonst zu lange dauert.

Auch das weltweit genutzte IRC ist für manche Filesharing Dienste eine Einstiegsmöglichkeit, um sich in das Netzwerk einzufinden. (Das Netzwerk Ants [ANT] benutzt unter anderem diese Methode)

Meistens jedoch werden so genannte „Gnutella WebCaches“ für den Einstieg in Netzwerke genutzt.

Gnutella WebCache

Die inzwischen geläufigste Methode stammt aus dem Gnutella System. Diese Einstiegsmethode ist jedoch nie im Protokoll selber definiert worden.
Hierbei befindet sich einfach eine PHP, Perl oder ähnliche Implementierung des Gnutella WebCaches auf einem Webserver. Dieser speichert Clients, die diese Liste abfragen und auch andere WebCache Adressen. Somit braucht ein Servent nur einmal einen WebCache erreichen, um dauerhaften Einstieg ins Netzwerk zu besitzen.
Meistens wird den Anwendungen einfach eine aktuelle Liste von WebCaches beigelegt. Einmal einen WebCache befragt, sollte der Servent ihn nicht mehr benutzen, da er jetzt bzw. über seine aktive Laufzeit genug andere Servents im Netzwerk kennt. Dies soll gewähren, dass ein Webserver, der einen solchen Dienst anbietet, keinen zu hohen Traffic erleidet und damit dieser Dienst gratis angeboten werden kann.
Da Anfragen über das HTT Protocol stattfinden, sind diese Listen auch mit einem einfachen Internet Browser abzurufen. Die richtigen Parameter müssen jedoch händisch angefügt werden.

Ein Beispiel, um sich solche Listen mit einem Browser anzusehen:
http://>Adresse<?get=1&client=gnutella
wobei „Adresse“ durch einen gültigen Pfad ersetzt werden muss. Dies ist ein einfacher Aufruf, der den Webserver im Glauben lässt, dass ein Gnutella Servent anfragt.
Auf der Seite „http://gcachescan.jonatkins.com/“ (letzter Zugriff am 4.Juli.2007) kann man eine Auflistung von aktuellen WebCaches und deren Zugriffe ansehen.

8 Neue dezentrale Technologien

Im diesem Kapitel sollen beginnende oder vollständige Umsetzungen auf Basis von neuen Technologien gezeigt werden. Außerdem soll genauer auf die Funktionalität der Anwendung oder der Technologie eingegangen werden.

Ein Fazit soll besondere Merkmale herausheben und die Filesharing Technologie bewerten. Ferner wird, wo im Laufe der Forschung eine Idee für eine Erweiterung aufkam, diese näher erläutert werden.

8.1 Audio/Video Broadcasting

WinAMP, einer der beliebtesten Mediaplayer für Windows, bot Möglichkeiten, um Internetradio über Shoutcast Server zu empfangen. Shoutcast Server sind meist sogar werbefrei und für die Allgemeinheit zugänglich. Oft sind diese Server jedoch überbelegt. Noch schlechter sieht es bei Video Streams aus, da hier noch höhere Bandbreiten benötigt werden.
Damit ist es für eine kleine Gruppe oder gar nur einen Einzelnen sehr schwierig, einen Radio oder Videostream zu veröffentlichen. Die vierte Filesharing Generation bietet genau hierfür eine Lösung, die auf einem Filesharing Netzwerk basiert.
Eine Umsetzung hierfür ist PeerCast. Dieses Programm soll genau dies bewerkstelligen. Es wird vor allem durch die swarming Technologie ermöglicht, wie es BitTorrent besitzt. Dadurch kann jeder einzelne Teilnehmer seinen eigenes Radio-, oder mittlerweile sogar Videostream veröffentlichen. Durch dieses Verfahren ist ein Audiobroadcast bereits mit einem 56 Kilobit Modem möglich.

PeerCast wurde im Jahr 2002 ohne finanziellen Hintergedanken als Audio Broadcasting Anwendung entwickelt. Das Ziel war es, jedermann zu ermöglichen, eigene Audio Dateien in Form eines Streams zu veröffentlichen. Das Filesharing Netzwerk sollte also komplett ohne teure Serverhardware oder großer Bandbreite funktionieren.

Funktion

PeerCast funktioniert im Wesentlichen auf einer Stream-Swarming Technologie.
Ein Teilnehmer, der einen Stream veröffentlicht, erstellt wie bei BitTorrent eine Art Tracker. Je nach Bandbreite wird ein Teil des Streams in kurze (geringe Bandbreite) oder längere (höhere Bandbreite) Teile zerlegt. Diese werden danach an die einzelnen Teilnehmer verteilt. Die Teile des Streams werden nun unter den Clients ausgetauscht. So ist es möglich, dass im Optimalfall der Broadcaster nur die Bandbreite für einen Stream benötigt.
Damit auch Broadcaster mit einem sehr schwachen Computer den Service nutzen können, ist es möglich, den Stream zu einem anderen PeerCast Nutzer zu senden. Dieser übernimmt dann die Tracker Funktion.
PeerCast kommt somit komplett ohne zentralen Server aus, da jeder Benutzer normaler Teilnehmer (Client) oder Broadcaster (Server) sein kann.

Vor allem im Bezug auf die Rechtliche Situation bietet PeerCast einen weiteren Vorteil gegenüber herkömmlichen Broadcasts.
Aufgrund des Systems ist es nicht leicht, den Originalstream als Teilnehmer nachzuvollziehen. Damit ist gleichzeitig eine hohe Anonymität gewährleistet. Der Stream ist außerdem auf gewisse Teilnehmer beschränkbar.

Fazit

Die Idee des Systems spricht für sich. Es wurden die wichtigsten Punkte von BitTorrent gut implementiert. Ebenso ist die Anonymität vorhanden, die mittlerweile vielen Teilnehmern wichtig ist.

Dennoch merkt man an einigen Punkten, dass es sich noch immer in der Entwicklungsphase befindet.
Die größten Probleme sind folgende:

- Wenn bei einem kleinen Publikum eine Zwischenstation verloren geht, verschwindet ebenso ein Streamteil. Dies hat den unangenehmen Nebeneffekt, dass der Stream aussetzt oder gar die Verbindung abgebrochen wird.
- Da die Upload Bandbreite eines Teilnehmers nicht immer im vollen Ausmaß zur Verfügung steht, kann es schnell zu Aussetzern kommen.
- Je nach Gesetzeszone gibt es eine rechtliche Grauzone, wenn rechtlich geschützte Daten veröffentlicht werden. Da jeder Teilnehmer beim Hören gleichzeitig ein Verteiler wird, verteilt man diese auch, was häufig illegal ist.
- Zahlreiche Sicherheitslücken, die nach und nach auftauchen.

8.2 Instant Messenger

Als ICQ 1997 der Öffentlichkeit vorgestellt wurde, war vielen der nachfolgende Erfolg des Konzeptes unvorstellbar.
Viel zu sehr orientierte man sich an Web und eMail. Doch während E-Mail das direkte Kommunikationsbedürfnis nur beschränkt erfüllen konnte, wurden Instant Messenger (IM) immer populärer und sind heute Ausgangspunkt für viele interessante neue Dienste und Anwendungen.

Die ersten Instant Messenger, die von einer großen Masse benutzt wurden, basierten auf einer reinen Client/Server Architektur.
Clients meldeten sich an einem Server an. Dieser verwaltet also eine Liste von allen aktiven Teilnehmern und ihren aktuellen Status. Will ein Client an einen anderen eine Nachricht senden, so wird diese zuerst an den Server gesendet. Anschließend leitete dieser die Botschaft an den Zielclient weiter (Push-Verfahren).

Mittlerweile wurden auch hier Prinzipien von Filesharing Systemen integriert.
Ein Client muss hier nur noch den Server kontaktieren, um die Zieladresse zu erlangen. Danach werden sämtliche Nachrichten immer direkt zwischen den beiden Teilnehmern, die nun innerhalb der Verbindung Servents sind, gesendet.

Fazit

IM waren eine der ersten Anwendungen, wo Filesharing Technologien implementiert worden sind. Dies liegt nahe, da es nicht nur den Server entlastet, sondern auch integrierte Dienste, wie Audio- und Videochat erst ermöglicht.

Große Erweiterungen mittels neuer Filesharing Technologien sind hier jedoch kaum noch sinnvoll.
Stream-Swarming wäre für Ausnahmefälle noch integrierbar. Es werden jedoch selten Audio- oder Videokonferenzen in Instant Messenger abgehalten, an denen mehrere Personen teilnehmen. Dies erscheint daher äußerst unrentabel, da nur sehr wenige Leute diesen Vorteil wirklich nutzen würden.

8.3 SET

Einen Ansatz Downloads zu beschleunigen bietet SET [SET] (Similarity Enhanced Transfer). Diese Technik soll den Dateientransfer zwischen Servents erheblich erhöhen. Entwickelt wurde diese Methode von David G. Andersen an der Carnegie Mellon University.

Funktion

Bei den bisherigen Filesharing Systemen erfolgt ein Download von Teilnehmern, die eine identische Datei besitzen. Dies gilt ebenso für BitTorrent.
SET sucht nun auch nach Dateien, wo Teile mit der gesuchten Datei übereinstimmen. Dies könnte zum Beispiel bei einem Film, der in verschiedenen Sprachen vorliegt auftreten. Hier wird dann die Audio Spur von einer anderen Quelle downgeloaded, als die Videospur. Eine Gigabyte Datei wird in 64000 Teile zu je 16 Kilobytes zerteilt. Diese Zerteilung in so kleine Stücke sorgt dafür, dass viele Teile aus ähnlichen Dateien gefunden werden können. Gleichzeitig wird jedoch ein hoher Aufwand für Vergleiche benötigt.

Bei einem Download einer Applikation ist eine wesentliche Beschleunigung möglich, wenn diese in verschiedenen Versionen vorliegt. Da sich die Anwendungen nicht von Version zu Version komplett ändern, sind viele Files und damit Teile davon identisch. Dies steigert die Zahl der Anbieter von Dateiteilen enorm.

Fazit

Die Technologie selber beschleunigt die Downloads nur dann erheblich, wenn es viele Anbieter gibt. In der Praxis problematisch ist hierbei, dass die Geschwindigkeit bei genügend Quellen sowieso ausreichend ist. Bei Dateien, die jedoch nur in geringer Anzahl und wenigen Versionen vorhanden sind, nutzt dieses Verfahren kaum etwas.
Soll jedoch eine beliebte Datei schnellstmöglich auf einem Servent seinen Platz finden, so ist dies eindeutig der schnellste Mechanismus hierfür. Da heutzutage jedoch viele

Filesharing Systeme mit ihrer Downloadgeschwindigkeit werben, wird diese Technologie wohl sehr bald Einzug halten.

Der Algorithmus selber wurde von den Entwicklern frei zur Verfügung gestellt.
„This is a technique that I would like people to steal. [...] In tests based upon real files […], SET improved the transfer time of an MP3 music file by 71 percent. A larger 55-megabyte movie trailer went 30 percent faster using the researchers' techniques to draw from movie trailers that were 47 percent similar. The researchers hope that the efficiency gains from SET will enable the next generation of high-speed online multimedia delivery.” [ADS]

Erweiterung

Durch die ähnlichen Dateiaufteilungen wäre eine interessante Symbiose von BitTorrent und SET möglich. Wobei die Verteilung mit dem BitTorrent Algorithmus funktionieren würde, jedoch SET gleichzeitig Server mit möglichen Teilen sucht.

Für den sinnvollen Einsatz eines solchen komplexen Filesharing Netzwerks würden sich aber nur relativ große Dateien eignen. Bei kleinen Datenmengen wie Quelltexte würde der *Overhead*, der durch die multiplen Suchvorgänge entsteht, ein Vielfaches der Datei selber betragen.

9 Mögliche dezentrale Technologien

Im folgenden Kapitel wird erforscht, welche Umsetzungen auf Basis von Filesharing Netzwerken sinnvoll wären.
Es wird dazu jeweils eine klassische Client/Server Anwendung verwendet. Danach werden je nach Komplexität die wichtigsten Punkte analysiert, warum diese zentral umgesetzt sind.
Zu jedem dieser Punkte wird dann ein dezentraler oder hybrider Ansatz überlegt.
Neuartige Ideen benötigen natürlich kein direktes Vorbild.
Das Fazit soll zeigen, ob die Überlegungen sinnvoll sind.

9.1 Semantisches Web

Webseiten sind eigentlich eindeutig auf eine Zentrale Technik ausgelegt. Die Seiten werden auf einen Server oder einen Verbund gestellt und sind nun vom Client aufrufbar. Gerade in Zeiten des Web2.0 gibt es zahlreiche Inhalte auf jeder Seite, die downloadbar sind. In den letzten Jahren besonders hervorgetan haben sich hier die Blogs im Internet.

9.1.1 Integration

Gerade dieses Gebiet würde sich für eine Nutzung von Filesharing Technologien anbieten. Ein Servent könnte zum Beispiel in einem Browser integriert sein. Ladet man sich hier nun eine Datei, zum Beispiel ein Video herunter, so wird dieses vom Servent, der im Hintergrund läuft indiziert. Ladet nun ein anderer Servent die gleiche Datei down, wäre es vorstellbar, dass dieser nicht nur den Server, sondern auch den anderen Servent kontaktiert und von diesem ebenfalls herunterlädt. Diese Technologie entspricht im Wesentlichen BitTorrent.

Eine wesentliche Erweiterung könnte dieses System jedoch erfahren, wenn eine Webseite zum Beispiel über PHP einen Servent selbst darstellt. Damit könnten sämtliche freigegebenen Dateien wie in einem herkömmlichen Filesharing Netzwerk downgeloaded werden.

Der Server, der nun als eine Art SuperPeer agiert, schreibt sich alle Clients auf, die einen gewissen Bereich betreffen, oder ein gewisses Thema behandeln. Jeder Client kann nun Dateien oder Informationen über dieses Thema veröffentlichen.
Meldet sich nun ein neuer Client, der sich für diesen Bereich interessiert, so kann er nicht nur die Daten des ursprünglichen Servers downloaden, sondern auch die der anderen Servents, die an dem Netzwerk teilnehmen.

Dies würde einen nie dagewesenen Komfort vor allem auf Webseiten bieten, die eine Unmenge von kleinen Dateien übersichtlich anbieten könnte, da Suchfunktionen sehr leicht zu implementieren wären. Nicht zu verachten wären die zusätzlichen Möglichkeiten, die mit geringem Aufwand indirekt vom Provider zum Beispiel durch Blogs angeboten werden können. Dies geht auch über ein reines Kommentieren von Newseinträgen weit hinaus, da so komplette Dateien von jedem Teilnehmer verfügbar sind.
Auch eine Begrenzung auf bestimmte Benutzer wäre leicht, da die Authentifizierung durch den Servent geschieht.

Ein erster kleiner Schritt dahingehend, ist bereits geschehen. In der Version 9 des Opera Browsers [OPR] ist erstmals BitTorrent direkt in den Browser integriert worden.
Dies dient jedoch nur dem Komfort bei den Downloads selbst und eröffnet nicht direkt neue Möglichkeiten.

9.1.2 Fazit

Ohne Teilnahme von anderen würde diese Technik sich für den Benutzer kaum von FTP unterscheiden, so wie es in jeden modernen Browser integriert ist.
Dennoch gibt es ein paar Argumente, die für ein solches System sprechen:

- FTP Zugang ist bei billigen Webhosts nicht verfügbar.
- Kaum Kosten für die Implementierung auf einer Webseite.
- Dateiinformationen anfügen durch Metasprachen wie XML.

Wenn jedoch das komplette System gut umgesetzt ist und damit gleichzeitig Leute motiviert, daran teilzunehmen, stehen zahlreiche Möglichkeiten offen.
Ein Webserver muss damit lediglich eine Implementation verwenden und hätte somit ganz nebenbei Anwendungen wie eine vollständige Blogging Software.

9.2 Botnet

Ein Botnet ist die Kurzform von Ro**bot**-**Net**work. Hierbei versteht man Applikationen, die sich auf Computern befinden und fernsteuerbar sind, gleichzeitig aber auch für sich allein arbeiten können. Ferner können Bots untereinander kommunizieren. Dies ähnelt nicht nur den Filesharing Systemen, vielmehr sind moderne Botnets auf Basis von rudimentären WASTE [WST], Pastry [PST] und Chord Implementationen programmiert.
Botnetze sind leider in letzter Zeit durch Missbrauch sehr in Verruf gekommen.

9.2.1 Work Bots (Positiver Aspekt)

Bisher sind große Rechenaufgaben immer mittels Server verteilt worden. Viele Projekte wie SETI@home [SC1] und evolution@home [SC2] benutzen dies. Im Vordergrund dieses Filesharing Systems steht ein Server. Teilnehmer können nun eine Worker Software downloaden. Diese fragt nun regelmäßig beim Server an und ladet eine bestimmte Aufgabe. Zum Beispiel die Berechnung eines Teilproblems. Die berechneten Daten werden dann in herkömmlicher Client/Server Filesharing Technik gesandt. Diese Anwendung zur Verteilung von Aufgaben wird immer beliebter und neue Projekte sind zahlreich im Entstehen.

Warum nicht vollkommen dezentralisiert?

Hierfür sind einige wesentliche Gründe verantwortlich:

- Sicherheit
- Verteilung
- Ergebnisse

9.2.1.1 Sicherheit

Sicherheit ist einer der wesentlichsten Punkte bei Workbot Systemen. Schließlich könnte ein einziger Worker, der gezielt falsche Daten liefert, die gesamten Ergebnisse zerstören. In einer Client/Server Architektur kann man diesem Problem relativ leicht vorbeugen. Der Server braucht lediglich stichprobenartig die Ergebnisse untersuchen.
Falls es mathematische Berechnungen sind, wären zum Beispiel Überschläge denkbar. Ebenso könnte er von jedem Worker eine errechnete Aufgabe genau nachprüfen.
Eine relativ sichere Methode ist es, verschiedenen Workers die gleiche Aufgabe zu erteilen. Die Ergebnisse können dann auf diese Weise leicht überprüft werden.

Nur vereinzelt möglich: Wenn zum Beispiel wie bei SETI@home Signale überprüft werden. Es werden einfach zwei Signalteile, die sich teilweise überschneiden, an zwei verschiedene Worker gesendet. Dadurch gehen wenige Ressourcen verloren und man erreicht eine hohe Sicherheit.

Abbildung 25 soll diese Idee verdeutlichen*:

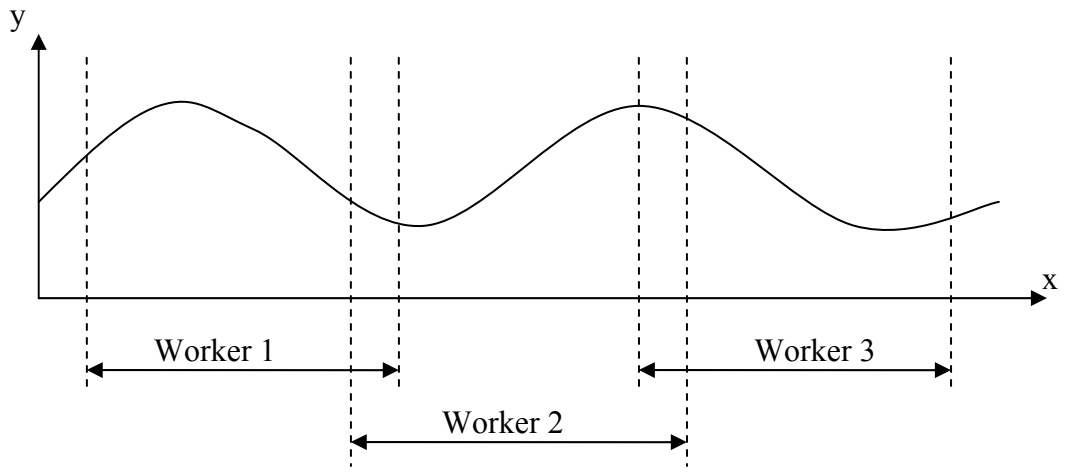

Abbildung 25: Signaleinteilung für Worker

Welche Methode tatsächlich verwendet wird, ist zumeist ein Geheimnis, um sich gegen gezielte Angriffe zu schützen, die das Sicherheitssystem ad absurdum führen würden.

*Das Signal ist zufällig gewählt und dient nur für eine gute Anschauung.

Dezentraler Lösungsansatz:

In einem vollkommen dezentralisierten Botnet könnten immer 4 Worker ein und dasselbe Paket berechnen. Nach der Byzantinischen fault tolerance (Mehr dazu im *Anhang 15.1 Byzantine-fault-tolerance*) ist somit eine gute Sicherheit gegeben.
Die Technologie des Netzwerkes ist hierbei irrelevant.
Nachteil eines solchen Systems ist, dass hier davon ausgegangen wird, dass jeder vierte Worker ein Betrüger ist. Normalerweise liegt diese Zahl bei seriösen Netzwerken im Promille oder kleineren Bereich. Dadurch entsteht eine sehr hohe Datenredundanz, wodurch viele Ressourcen verschwendet werden.

9.2.1.2 Verteilung

Die Verteilung der einzelnen Datenpakete erfolgt herkömmlich durch den Server. Je nach Arbeitsvorgang nummeriert er diese und teilt sie aus. Bei der Verteilung kann ein Server den Workern helfen, indem er ihnen Arbeitspakete gibt, die deren Performance entgegenkommen. So erhalten schwache Rechner kleine und schnelle Rechner große Pakete.

Dezentraler Lösungsansatz:

Eine Verteilung ohne Server könnte wie folgt aussehen:
Das entwickelnde Institut stellt einen Worker in das Netz. Dieser enthält eine ganze Reihe von Arbeitspaketen. Die einzelnen Aufgabenpakete werden wie Dateien gehasht und dann auf die entsprechenden Worker im Filesharing Netzwerk übertragen. Der Hashing Algorithmus sollte dafür sorgen, dass sämtliche Pakete möglichst gleichmäßig auf alle Teilnehmer verteilt werden.
Dieser Ansatz setzt somit ein DHT-Filesharing Netzwerk voraus.

9.2.1.3 Ergebnisse

Berechnete Pakete werden von den Workern zum Server gesandt. Diese brauchen anschließend nur noch zwecks Sicherheit geprüft werden. Sind diese Daten einwandfrei, können die Ergebnisse zum Beispiel in eine Datenbank gespeichert werden. Sind sie unkorrekt, wird das Paket verworfen und wieder neu in die *Queue* eingereiht.

Dezentraler Lösungsansatz:

Bei der Verwaltung der Ergebnisse der Berechnungen treten die größten Probleme für einen Ansatz mittels DHT Netzwerk auf. Für die Ergebnisverteilung, die hier genauer geschildert wird, bietet sich ein Chord System an.

Ein Ansatz wäre der Folgende:
Der Verteiler der Pakete, welcher vom Entwickler gestellt wird, entspricht einem gleichberechtigten Worker. Dadurch sind die errechneten Daten nicht direkt an ihn sendbar, da er ein gewöhnliches Mitglied ist. Jedes berechnete Paket muss deswegen an den nächsten weitergeschoben werden. Dies geschieht so lange, bis das Paket am eigenen Vorläufer angelangt ist. So ist sichergestellt, dass der auslösende Worker auch sämtliche Ergebnisse erhält.
Da die Ergebnisse selbst wahrscheinlich nicht sehr groß sind, ist der Bandbreitenaufwand hierbei nicht weiter schlimm. Das eigentliche Problem besteht darin, dass nach diesem Vorgang jeder Worker die Ergebnisse besitzt.

Ein besserer Ansatz ist hier, die Servents im Netzwerk zu gewichten.
Diese Lösung hat nun auch mit der Ergebnisverteilung keine Probleme mehr. Institute könnten einfach SuperNodes im Netzwerk benutzen. Diese SuperPeers teilen jedoch das Netzwerk nicht in ein gemischtes, dezentrales-DHT Netzwerk auf. Vielmehr dient die Ernennung zum SuperNode einfach dazu, dass Ergebnisse bei normalen Nodes nicht mehr gespeichert werden. Erreichen berechnete Daten die SuperNodes, so speichern sie diese und reichen jedoch die Daten auch weiter. Dies hat den Sinn, dass weitere SuperNodes, die sich im Netzwerk befinden, ebenfalls die Ergebnisse erhalten.

9.2.1.4 Fazit

Der Ansatz, dass jeder Teilnehmer die Ergebnisse erhält, ist für wirtschaftliche Zwecke nicht sinnvoll. Denn die Tatsache, dass die Ergebnisse jeder bekommt, entspricht zwar absolut dem Prinzip des Filesharings, jedoch ist dies weniger attraktiv für Firmen. Selten hat ein Konzern Interesse daran, dass die Ergebnisse anschließend jeder besitzt.

Dennoch könnte ein solches Projekt in den nächsten Jahren im Open Source Bereich entstehen. Die Möglichkeiten wären in einem solchen System immens. Von kleinen Berechnungen bis hin zu mathematischen Berechnungen von Primzahlen wäre alles denkbar.
Dies wäre jedoch nur möglich, wenn es eine Art Skriptsprache gäbe, mit der auch nicht so versierte Benutzer interessante Problemberechnungen aufteilen könnten. Als Motivation, an diesem Netzwerk teilzunehmen, dienen die Ergebnisse, die jeder erhält.

Mit dem Ansatz, in dem DHT Netzwerk SuperPeers einzuführen, erscheinen umfangreiche Umsetzungen sehr wahrscheinlich.

Tatsächlich wird ein ähnliches Netzwerk bereits an einigen Universitäten entwickelt. Es tragt den Namen EGEE [EGEE]. Dieses verbindet die Rechnersysteme an verschiedenen geographischen Orten und dient zur Verteilung von Rechenarbeiten.

9.2.2 Slave Bots (Negativer Aspekt)

Bisher hatten sich Schädlinge, wie Computer-Viren und -Würmer, entweder mit einem zentralen Server verbunden, über den sie steuerbar waren, oder einfach nach einer bestimmten Zeit eine Aktion gestartet. Server sind hier meist beliebige IRC-Server. Ähnlich wie das Filesharing Netzwerk Napster kann man dieses System jedoch recht einfach beseitigen, indem man den Server selbst entfernt.
Falls eine Aktion innerhalb oder ab einer bestimmten Zeit stattfindet, kann man sich zumindest darauf vorbereiten. Dies ist möglich, da der Schädling immer gleich „aussieht" und damit durch Inspektion des Quellcodes berechenbar ist.

Hier ist jedoch bereits die nächste Generation im Anmarsch. Sie benutzt vorhandene Filesharing Netzwerke und verschlüsselt im schlimmsten Falle sogar die Kommunikation.

9.2.2.1 Storm-Worm

Ein erstes, trotzdem bereits furchterregendes Botnet, ist mit Storm-Worm im Jänner 2007 aufgetaucht.
Dieses Botnet kennt zwar noch keine Updates, hat aber bereits mehrere mögliche Modis, welche beliebig getriggert werden können.
Darunter DDoS Attacken, Backdoor Funktionalität und eMail Verbreitungstool.
Da sich der Slave Bot mit dem eDonkey Netzwerk verbindet, bekommt dieser auf einfache Weise massenhaft Adressen von Rechnern, die er zu infizieren versucht.

Ferner ist bereits eine adaptierte Version von Storm-Worm aufgetaucht. Diese besitzt jedoch noch keine neuen Techniken, sondern „nur" einen neuen Modus. Dieser durchsucht den Computer nach Benutzernamen und Passwörtern für Foren. Dort trägt er sich dann mit einem Text und einem infizierten Link ein. Folgt ein anderer Benutzer diesem, kann er sich ebenso infizieren. Der Wurm kann somit nicht nur das Filesharing System eDonkey, sondern auch Foren ausnutzen.

9.2.2.2 Fazit

Auch wenn die neue Variante von Storm-Worm nichts wirklich Neues bringt, sieht man bereits den erschreckenden Weg, den Botnets nehmen können. Durch die bereits vorhandenen Infrastrukturen von Filesharing Systemen erhalten sie genug Adressen, die angreifbar sind.

Ein Albtraum wären Botnets, die sich gegenseitig aktualisieren. Mehr Informationen dazu im *Kapitel 9.5 Programmupdates*.
Diese Botnets könnten dann jederzeit neue Sicherheitslücken ausnutzen.

9.3 MMOGs

MMOGs steht für „massive multiplayer online games“.
Die soziale Komponente spielt bei allen Anwendungen im Internet immer eine große Rolle. Die Entwicklung geht hierbei immer mehr in Richtung Interaktivität.
Herkömmliche Chatrooms wurden zum Beispiel zu interaktiven Welten, in denen auch Interaktion mit Gegenständen möglich ist.
Diese sind jedoch meist in den einzelnen Bereichen auf wenige Teilnehmer gleichzeitig beschränkt.
Anders bei MMOGs. Diese sind bisher auf einer reinen Client/Server Architektur aufgebaut.

MMOGs wurden bisher eher belächelt. Durch die Sozialisierung im Internet werden sie jedoch immer beliebter. Das MMOG World of Warcraft zum Beispiel, zählt bereits über 8 Millionen Spieler.
Dies sorgt für enorme Kosten für Server. Somit ist es wegen der notwendigen Infrastruktur für Entwicklergemeinden schwierig, solche Spiele zu entwickeln.

Wäre ein solches Online Spiel komplett dezentralisiert denkbar?
Folgende Punkte sind entscheidende Merkmale für MMOGs:

- Client/Server Interaktion
- Latenz
- Verfügbarkeit
- Sicherheit
- Updatefähigkeit

9.3.1 Client/Server Interaktion

In einem herkömmlichen Client/Server MMOG ist eine Interaktion leicht durchzuführen. Die Clients speichern Elemente wie die Spielwelt und nicht manipulierbare Gegenstände. Der Server verwaltet die Charactere, deren Eigenschaften, Besitz und alle dynamischen Daten.
Ebenso weist der Server dem Client seine Koordinaten und sonstige Charakterzustände zu. Will ein Teilnehmer nun eine Aktion ausführen, wird diese an den Server geleitet. Der entscheidet nun, ob tatsächlich eine Aktion ausgeführt werden soll und sendet bei einer Veränderung die Daten an alle Spieler, die sich geografisch auf der virtuellen Weltkarte in der Nähe befinden.

Dezentraler Lösungsansatz

In einem komplett dezentralen System gibt es keinen Spielleiter bzw. Server. Ebenso kann es wenigen oder einem Servent nicht zugemutet werden, all diese Daten zu speichern. Auch die Benachrichtigungen über Aktionen an die richtigen Teilnehmer kann nicht einem Teilnehmer überlassen werden.

Deswegen ist hier ein etwas komplexerer Ansatz auf Basis von Filesharing Netzwerken zu finden.

Die Weltkarte an sich wird wie herkömmlich bei jedem Servent gespeichert. Diese wird in kleinere Fragmente zerteilt. Die Größe der einzelnen Teile hängt von der Anzahl der aktuell teilnehmenden Spieler ab. Je mehr Servents damit an diesem Netzwerk teilnehmen, desto kleiner die Kartenfragmente.
Jeder Spieler, der nun in einem Kartenteil agiert, übernimmt auch Berechnungen für den jeweiligen Teil. Dies betrifft insbesondere Nicht Spieler Charaktere.
Ebenfalls benötigt er auch Aktionsdaten von anderen Teilnehmern in diesem Areal.
Falls möglich, erhält jeder Servent zwei Verbindungen mit Teilnehmern aus einem anderen Kartenfragment aufrecht. Wechselt der Spieler nun von einem Kartenteil in einen anderen, so übernimmt er nicht länger Berechnungen des alten Teils, erhält jedoch diese Aufgabe für den neu betretenen.
Interaktive Objekte und Nicht Spieler Charaktere werden ebenfalls von Servents innerhalb dieses Kartenfragments gespeichert.

Der Charakter, sowie Eigenschaften und Besitztümer eines Teilnehmers werden auf Servents gespeichert, die von der ID dem Hashwert am nächsten kommen. Natürlich müssen auch hier Duplikate angelegt werden, da sonst ein Einstieg nicht mehr möglich wäre, wenn der Spieler mit der Speicherung nicht im Netzwerk zu finden ist.

Dieser Ansatz ist ein Dezentrales Netzwerk, gemischt mit einem DHT Netzwerk.

Dezentral: Jeder Servent innerhalb eines Kartenfragmentes hält eine Verbindung zu einem beliebigen Teilnehmer in einem anderen Kartenteil aufrecht..

DHT: Innerhalb eines Kartenfragmentes sind die Servents in einem Chord Netzwerk angeordnet.

Abbildung 26 erläutert dieses Netzwerk näher:

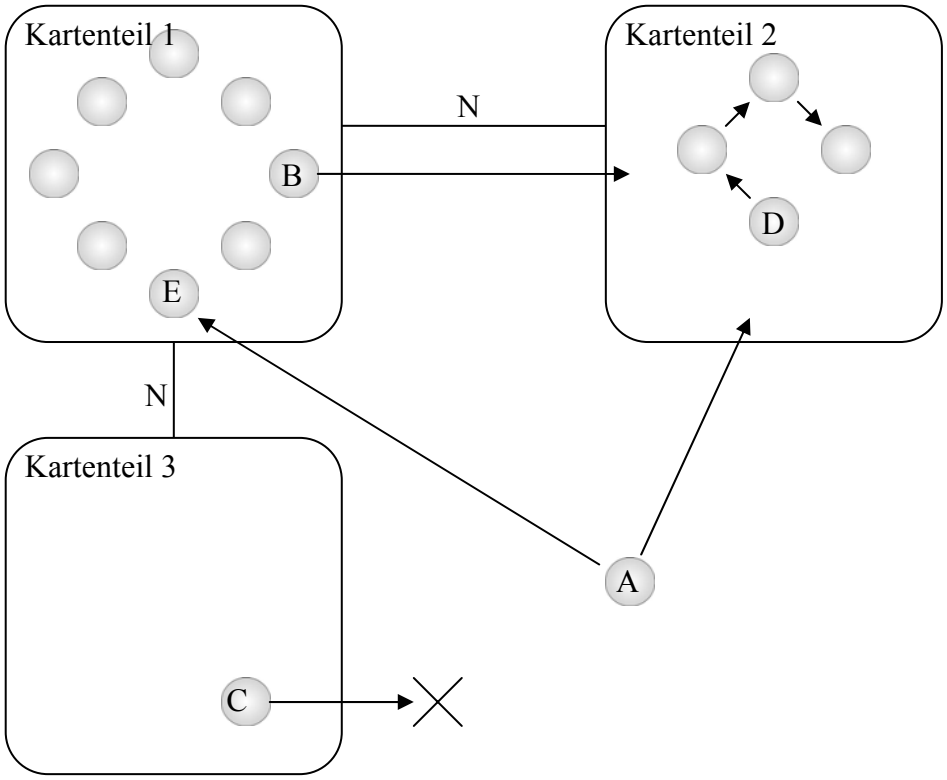

Abbildung 26: Vorgänge in einem Dezentralen MMOG

A: Ein Servent, der an dem MMOG teilnehmen will. Zuerst muss er hier seinen Charakter laden. Diese Eigenschaften hat zum Beispiel der Servent E gespeichert. Da er bereits beim letzten Mal mittels Hashverfahren ermittelt worden ist, befindet er sich logischerweise in einem Host Cache von A. Dies gilt auch für Servents, welche Duplikate gespeichert haben. Mit diesen Informationen hat der Servent nun festgestellt, dass er in Kartenteil 2 positioniert ist. Kennt er zufällig einen Teilnehmer, der sich dort aufhält, kann er sich sofort ins dortige Subnetzwerk einfügen. Ansonsten muss er einen anderen Servent fragen. Dieser besitzt ja eine Verbindung zu einem Teilnehmer in einem anderen Fragment. So kann sich der Einsteiger schnell in das nötige Subnetzwerk bewegen.

B: Hier wandert im Spiel ein Charakter vom Kartenteil 1 nach 2. Zuerst muss er sich in das Subnetzwerk im Kartenteil 2 einfügen. Sobald er alle Zustände von den anderen Servents erhalten hat, kann er ebenso Berechnungen übernehmen und Aktionen auslösen. Er nimmt ab dann nicht mehr an Berechnungen von Kartenteil 1 teil.

C: Dieser Servent will aus dem Spiel aussteigen. Falls der Servent nicht ungeplant aus dem Netzwerk aussteigt, wird vorher überprüft, ob ausreichend Servents gecacht wurden, damit ein Einstieg in das Netzwerk leicht wieder möglich ist. Hierfür cached er auch alle Servents, die Duplikate der Charakterdaten besitzen.
Diese Daten werden außerdem regelmäßig aktuell gehalten, um bei einem überraschenden Ausstieg weiterhin in Zukunft teilnehmen zu können.

D: Dieser Teilnehmer führt eine Aktion aus. Da jeder in diesem Bereich davon erfahren muss, wird die Aktionsnachricht alle Servents durchwandern.

9.3.2 Latenz

Für Online Spiele ist eine gute Latenz unverzichtbar. Diese ist, egal ob entweder ein Chat stattfindet oder eine komplexere Spieler/nicht Spieler – Charakter Interaktion. Diese sollten bei allen Teilnehmern, die sich in unmittelbarer Nähe befinden, möglichst zur gleichen Zeit passieren. Wenn ein Servent eine Aktion auslöst, darf es als Richtzeit maximal 200 ms dauern, bis alle beteiligten Spieler von der Änderung Bescheid wissen.

Bei einem Client/Server Modell ist diese Latenz von folgenden Zeiten abhängig:

- Netzwerklaufzeit (Client → Server)
- Serverberechnungszeit
- Netzwerklaufzeit (Server → Client)

Dezentraler Lösungsansatz:

Mit dem Ansatz, eine Aktionsnachricht durch den kompletten Chord Ring zu schieben, ist eine vernünftige Latenz nicht zu halten.
Hier wird die Fingertabelle in dem Chord System genutzt. Dadurch wird der Ring geteilt. Damit wird eine Nachricht nicht nur an den direkten Nachfolger im Ring gesendet, sondern auch an die Finger, die den Ring halbieren, vierteln, und so weiter. Der jeweilige empfangende Servent verfährt nun genauso mit der Aktionsnachricht.
Dadurch erhält jeder Teilnehmer den Status in wesentlich kürzerer Zeit.
Eine weitere Verbesserung erhält man, wenn die Nachricht gleichzeitig an die Successoren und dem Predecessor gesendet wird. Dies würde aber eine genaue Untersuchung benötigen, damit nicht zu viele redundante Pakete gesendet werden.

9.3.3 Verfügbarkeit

In einer herkömmlichen Client/Server Architektur ist Verfügbarkeit gegeben, wenn der Server aktiv ist. Ebenso sind Charaktere und Zustände auf dem Server gespeichert.

Dezentraler Lösungsansatz:

Ein dezentraler Ansatz hat genau hier seinen Schwachpunkt. Die Daten dürfen nicht auf dem eigenen Servent gespeichert werden. Dies hat zwei wesentliche Gründe:

1. Die Welt kann sich weiterentwickeln, und damit können sich Zustände ändern.
2. Charaktere und deren Besitz könnten verändert werden (siehe Sicherheit)

Berechnungen innerhalb eines Kartenteils werden von den entsprechenden Servents selber vorgenommen, die dort interagieren. Die Speicherung von Zuständen in diesem Kartenfragment erfolgt auf Teilnehmer, die mittels Hashing der Zustandsliste ermittelt werden. Ebenso werden Duplikate verteilt.
Das gleiche Prinzip gilt auch für den eigenen Charakter.

9.3.4 Sicherheit

Sicherheit spielt auch bei MMOG eine große Rolle. Gerade weil die meisten komplexeren Spiele Gebühren kosten, muss sichergestellt werden, dass keiner betrügen kann.
Dies ist bei klassischen Client/Server Architekturen kaum ein Problem. Aufgrund des Systems kann der Server alleine entscheiden und ebenfalls überprüfen. Ebenso können log Dateien über die Vorgänge in der Virtuellen Welt erstellt werden.

Dezentraler Lösungsansatz:

Hier müssen zwei wesentliche Punkte unterschieden werden:

- Aktionssicherheit

Da es in MMOGs vorkommen kann, dass ein Spieler sich alleine in einem Kartenfragment aufhalten kann, ist kein anderer Teilnehmer in der Lage, dessen Aktionen von diesem zu kontrollieren. Somit wäre hier eine Manipulation möglich. Der Servent könnte zum Beispiel einen Gegenstand als aufgenommen melden, an den er eigentlich nicht alleine kommen könnte.

Für diese Problematik wurde vom Autor dieser Arbeit leider keine Lösung gefunden. Mehr dazu im *Kapitel 9.3.6 Fazit*.

- Datensicherheit

Unter Datensicherheit fallen Zustände von Kartengebieten. Da diese nicht auf dem eigenen Servent gespeichert werden, können diese kaum manipuliert werden. Durch Duplikate wird auch sichergestellt, dass diese Zustände nicht verfallen.

9.3.5 Updatefähigkeit

MMOGs leben davon, dass die Welt sich regelmäßig verändert und erweitert.
In einem Client/Server System sind Updates sehr einfach zu bewerkstelligen. Wird ein Update von den Entwicklern fertig gestellt, so muss zuerst die Serversoftware aktualisiert werden. Jeder Client, der daraufhin in das Spiel einsteigen will, muss zuerst die neueste Version downloaden.
Moderne MMOGs benutzen für diesen Downloadvorgang, der oft einige Hundert MB groß sein kann, BitTorrent.

Dezentraler Lösungsansatz:

Es muss sichergestellt werden, dass jeder Servent, der an dem MMOG teilnehmen will, das Update erhält. Die Entwickler stellen dazu eine gewöhnliche Node mit der neuesten Version in das Netzwerk. Dieser Servent verweigert nun die Zusammenarbeit mit jedem anderen Teilnehmer, bis dieser sich das Update überspielt hat. Dies sorgt in einer Art *Schneeballsystem* dafür, dass jeder Servent die neueste Version verpflichtend erhält.
Optimiert kann der Download hier werden, indem die BitTorrent Technik eingesetzt wird.

9.3.6 Fazit

Ein vollkommen dezentrales MMOG ist sehr aufwändig. Viele grundlegende Funktionen, die in einem Client/Server Netzwerk sehr simpel sind, können im dezentralen Ansatz zu einem großen Hindernis werden. Besonders gilt dies für die Bereiche Sicherheit und Verfügbarkeit.
Vor allem im Bereich Aktionssicherheit wurde hier keine Lösung gefunden.

Technisch gesehen also ist der Bereich MMOGs mit der Technik von dezentralen Filesharing Netzwerken eine umfangreiche Herausforderung. Im vorgestellten Ansatz wurden gleich mehrere verschiedene Strukturen verwendet.

Aufgrund dieser Hindernisse wird wohl in absehbarer Zeit keine Implementation umgesetzt werden. Der Aufwand und die Entwicklungskosten stehen in keinem Verhältnis zu den Server Kosten.
Eine Umsetzung wäre jedoch eine große Bereicherung für kleine Entwicklungsteams. Wäre diese Arbeit einmal geleistet, könnten freie MMOGs generell darauf aufbauen. Somit könnten auch kleine Gruppen von Entwicklern Spiele veröffentlichen, da sie sich keine Gedanken um teure Server zu machen brauchen. Die gesamte Entwicklungsarbeit kann somit ausschließlich in das Spiel investiert werden.

Ein Grund, warum eine solche Umsetzung nicht sehr bald möglich sein wird, sind die geringen wirtschaftlichen Vorteile, wenn die Umsetzungskosten eingerechnet werden müssen. Außerdem müssen Charakterdaten jederzeit verfügbar sein. Solange sich in einem dezentralen Ansatz nur wenige Teilnehmer befinden, ist dies nicht sicher gegeben. Je mehr Duplikate, bis zu einer gewissen Grenze, auf verschiedenen Servents erstellbar sind, desto höher wird die Verfügbarkeit in diesem System.

9.4 Video Portale (am Beispiel YouTube)

Eine Studie der Firma Ellacoya Networks hat ergeben, dass der Dienst YouTube ein Zehntel des gesamten Transferaufkommens im Internet ausmacht.
„Web Traffic Overtakes Peer-to-Peer (P2P) as Largest Percentage of Bandwidth on the Network" [ELL].

YouTube selber verwendet ein Flash-Video-Format. Die Videos können online als Stream im Webbrowser betrachtet oder lokal gespeichert und mit Flash Video-fähigen Softwareplayern abgespielt werden. Zum Betrachten der Videos im Webbrowser ist die Installation des Adobe-Flash-Plugins erforderlich.
Am 9. Oktober 2006 wurde YouTube vom Suchmaschinenbetreiber Google für umgerechnet 1,31 Milliarden Euro (in Aktien) gekauft.

YouTube (Google) muss für diesen Aufwand bis zu $1,000,000 pro Monat an Serverkosten bezahlen [SYT].
Angesichts dieser Kosten stellt sich die Frage, ob es für genau solche Portale nicht eine Möglichkeit gibt, die Unkosten durch Filesharing Netzwerk Technologien zu senken..

9.4.1 Mögliche Erweiterung: Stream-Swarming

Die YouTube Server haben genauen Einblick, wie oft welcher Clip angesehen wurde. Gerade oft besuchte Videos wie Trailer von Kinofilmen könnten durch eine Stream Swarming Technologie verbreitet werden. Nähere Informationen zu Stream Swarming im *Kapitel 8.1 Audio/Video Broadcasting*.

Jeder, der an diesem System teilnehmen will, benötigt den Client.
Weil sich nach wie vor jeder Teilnehmer durch Besuchen der Webseite bzw. das Aussuchen des entsprechenden Clips beim Server meldet, hat der Server Zugriff zu allen Clients. Das Client/Server System besteht also weiterhin, jedoch verhalten sich die einzelnen Clients untereinander wie Servents.

Dieses Verteilungssystem würde umso besser funktionieren, je mehr Teilnehmer es nutzen. Es wäre außerdem noch möglich, dass es neben einem herkömmlichen Browser Stream koexistieren kann. Dadurch ist kein plötzlicher Bruch notwendig.

9.4.2 Fazit

Ein solches System könnte Aufgrund der Verbreitung derzeit nur YouTube (Google) durchführen, da sie den nötigen Kundenstamm haben, um einen solchen Client auch unters Volk zu bringen (Marktpräsenz). Anbieten würde sich, nachdem der Besitzer Google selbst ist, dieses System in die Google Toolbar im Browser einzubauen.

9.5 Programmupdates

Heutzutage werden fast alle Programme direkt durch einen Server upgedatet. Für jeden Client, der ein verfügbares Update annimmt, ist sichergestellt, dass er die neuste Version benutzt. Bei den heutigen Computerschädlingen wie im *Kapitel 9.2.2.1 Storm-Worm*, ist es enorm wichtig, dass Betriebssysteme und Virenschutzanwendungen auf dem aktuellen Stand gehalten werden.

9.5.1 Funktion

Ein Update ist in einem Client-Server-System leicht zu bewerkstelligen. Die Software muss dafür einfach den Server kontaktieren und erhält bei Genehmigung der Anforderung die notwendigen Daten. Die benötigten Bandbreiten dafür sind enorm. Diese Server müssen damit hohen Anforderungen genügen; Beispiel: jene, die für Windows-Update Pakete zuständig sind.

Dezentraler Ansatz

Durch geschickten Einsatz von Filesharing Netzwerken könnte man Software ohne zentrale Stelle updaten.
Das für diesen Zweck wohl sinnvollste Netzwerk ist FastTrack. SuperPeers könnten hier die Download Anlaufstellen darstellen. Die Anmeldung an den Server stellt dabei den Sicherheitsschutz dar. Dies ist vor allem bei Updates für Betriebssysteme oder sonstiger heikler Software notwendig. Der Anmeldeserver teilt dem Peer dann mit, welche Version aktuell ist. Durch Hashen der Dateien kann sichergestellt werden, dass kein Betrug stattfinden kann. Der Peer kann in diesem Netzwerk somit beruhigt und schnell das Update downloaden, ohne dass die Sicherheit dabei zu kurz kommt.

Für Open Source Software kann der Server komplett entfallen. Dadurch ist jedoch keine so große Sicherheit mehr gegeben. Durch das Hashen der Daten und mithilfe der Byzantine-fault-tolerance bleibt dieses Verfahren dennoch relativ sicher. Mehr dazu im *Anhang 15.1 Byzantine-fault-tolerance*.

9.5.2 Fazit

Viren und ähnliche Schädlinge nisten sich häufig in Systemdateien als Code ein. Oft sind diese Dateien in häufiger Verwendung, da sie einen Teil des Betriebssystems darstellen. Würde das Betriebsystem jedoch mit anderen Servents regelmäßig die Dateien-Konsistenz über Hash Werte mit den SuperPeers abgleichen, könnte eine schädliche Veränderung sofort festgestellt und auch entfernt werden.

Bei einer Implementierung eines solchen Systems sollte jedoch größte Vorsicht Vorrang haben, denn bei Sicherheitslücken könnte es zu einem massiven Ausbruch von Problemen führen.

9.6 Dateisysteme

Mit Filesharing Netzwerken wie Chord ist bereits die Basis gelegt für ein vollkommen dezentrales Dateisystem.

Ein Ansatz hierfür ist ein Chord (Cooperative) File System (CFS). Ein Dateisystem, welches auf Chord basiert. Einen Ansatz eines solchen CFS wird im folgenden *Kapitel 10 Proof of Concept-GCFS* beschrieben.

10 Proof of Concept-GCFS

10.1 Global Cooperative File System (Theorie)

Ein Globales Dateisystem basierend auf Chord?

Prinzipiell dient Chord lediglich dazu, das Auffinden eines Servents, der für einen Schlüssel zuständig ist, zu ermöglichen.
An diesem Schlüssel können weitere Nutzdaten hängen. Ein solches System wurde bereits testweise implementiert, jedoch mit einem Dateisystem das **ausschließlich** lesen kann (CFS [CFS]) und somit sehr statisch ist.

Der Proof of Concept – „Global Chord File System“ – soll eine Simulation sein, welche in einer Testumgebung das Lesen und Schreiben, welches eigentlich ein rudimentäres Dateisystem ausmacht, zu ermöglichen. Aufgrund des Umfangs wird hierbei auf Rechte und erweiterte Funktionen keine Rücksicht genommen. Auch Verzeichnisstrukturen werden außer Acht gelassen, da diese lediglich weiteren Aufwand darstellen, aber keine echten Probleme aufweisen. (Da eine Verzeichnisstruktur bei einer Indizierung sowieso unerheblich ist). Außerdem soll die Komplexität für den Proof of Concept gemindert werden und die Möglichkeit, eine Datei auf mehreren Rechnern in Teilen abzuspeichern, nicht möglich sein.

Da das GCFS eine Schicht über den hardwarenahen Zugriffen erstellt, ist es egal, wie Dateien tatsächlich gespeichert sind. Die Servents werden simuliert und die Dateizugriffe werden über die Programmierschnittstelle (Java) geregelt. Somit ist es egal, in welcher Art oder welchem Medium die Daten tatsächlich gespeichert sind. Entscheidend ist, Dateien nach einem Schlüssel einem Servent zuzuteilen. Natürlich kann dieser auch weitere erstellen oder löschen.

Damit die Zuteilung bei der Simulation mehrerer Teilnehmer nicht zu komplex wird, werden sämtliche Dateien nur virtuell erstellt und gelöscht. Die genaue Implementierung ist unerheblich, da die Zugriffe sowieso über Java (beziehungsweise das entsprechende Betriebssystem) erfolgen.

10.2 Systemstruktur

Jede GCFS-Node enthält wesentliche Komponenten:

- Ein Filesystem

Dies stellt das lokale Dateisystem oder einen indizierten Bereich davon auf dem Computer dar. Mit diesem Ansatz könnten alle Servents unterschiedliche Betriebssysteme benutzen. GCFS würde somit auch funktionierten, wenn Windows, Linux und MAC Computer im Netzwerk teilnehmen.

- eine File-Metaliste

Diese Metaliste dient dazu, alle Dateien, die im System verfügbar sind, zu beschreiben. Dies sind lediglich Metadaten, um die Bandbreite bei einem Update zu schonen.

- Service Komponente

Da die Netzwerkzugriffe nur simuliert werden, übernimmt die Servicekomponente das Bootstrapping und enthält fixe Parameter für das System (z.B.: Adressraumgröße).

- *GUI* Komponente

Stellt den Servent auf der Simulationsumgebung dar.
Hier kann der Benutzer interaktiv Dateien erstellen, sowie Verbindungen des Servents einsehen, bis hin zur Entfernung des Nodes.

Im Proof of Concept wird davon ausgegangen, dass die Benutzer hinter den Servents miteinander arbeiten. Der Wert eines Schlüssels besagt also ohne Umwege, wo die Datei zu finden ist. Dies ist nach den Regeln von Chord in Ordnung, mit der Voraussetzung, dass kein Saboteur vorhanden ist, der die Datei händisch direkt im lokalen Dateisystem wieder löscht. Dies jedoch würde dem Sinn des GCFS widersprechen, da es für eine Zielgruppe wie zum Beispiel eine mittelgroße Firma konzipiert ist.

10.2.1 Systemdaten

Aufgrund der komplexen Struktur im Vergleich zu einem reinen Server hat dieses System natürlich auch Nachteile, wobei die Vorteile zwar in der Minderzahl sind, aber eindeutig schwerwiegender!

Vorteile:

Der entscheidende Vorteil bei diesem dezentralen Netzwerk liegt in der Robustheit. Es gibt keine zentrale Stelle, an der das Globale Dateisystem ausschaltbar wäre.
Nun gibt es bei herkömmlichen Client-Server Systemen die Möglichkeit, einfach Backup Server zu stellen. Leistungsfähige Server sind jedoch sehr teuer.

Interessant ist auch, dass das GCFS sehr gut skaliert. Ein Server kann schnell unter einer Last von mehreren hundert Clients zusammenbrechen, die alle gleichzeitig Dateien anfordern und oder schreiben wollen. Das GCFS im Gegenzug kann dies leicht verkraften, da die Last auf viele Rechner verteilt wird. Je mehr Rechner am Netzwerk teilnehmen, desto besser erfolgt auch die Lastverteilung. Eine Beschränkung des Systems besteht nur durch die jeweiligen Bandbreiten.

Nachteile:

Ein Nachteil des GCFS ist, dass Duplikate von Dateien nötig sind. Denn ein Servent könnte ohne Vorwarnung aus dem System aussteigen und wenn auf diesem Daten liegen, die keine Duplikate besitzen, wären diese nicht mehr vorhanden. Es müssen daher nach einem System eine Anzahl von Duplikaten erstellt und verteilt werden. Dies sorgt für erhöhten Platzbedarf auf den einzelnen Rechnern. Dies ist aber nur bedingt ein Nachteil, da meist sowieso mehr als genug freier Festplattenspeicher vorhanden ist. Auch im wirtschaftlichen Aspekt schneidet das GCFS nicht schlechter ab, da eine enorm große Festplatte (Client-Server Modell) das Vielfache von einigen kleinen, mit insgesamt gleichen Speicherkapazitäten, kostet (GCFS). Auch ist eine Erweiterbarkeit bei den Servents leichter gegeben.

Im Vergleich zum Client-Server Modell benötigt das GCFS jedoch ein Minimum von vier Rechnern aufgrund der Byzantine fault tolerance. Informationen dazu im *Anhang 15.1 Byzantine-fault-tolerance.*

Bedingte Nachteile

Alle weiteren Nachteile sind lediglich darauf zurückzuführen, dass diese aufgrund des Aufwandes nicht im Proof of Concept implementiert sind.
Des Weiteren sind einige dieser Funktionen auch nicht in einem Client-Server Modell zu finden!

- Dateirechte
- Verzeichnisse
- verteilte Dateien
- Datenredundanz

10.2.2 Netzwerkaufbau

Das grundlegende Filesharing Netzwerk bietet Chord. Dieses System daher, weil es einen sehr effizienten Suchalgorithmus besitzt

Der Adressraum mit dem Parameter n ($=2^m$) gibt den wichtigsten Parameter des Systems vor.

Jeder Servent besitzt eine Verbindung zu m Nachfolger (Successors) und besitzt eine Verbindung zu seinem Vorgänger (Predeccesor). Zur Beschleunigung von Zugriffen auf Dateien, die „im Kreis weit weg liegen“, dient eine Finger Tabelle.

Hashing Alghoritmus

Die Hash Werte werden nicht mit der im Chord üblichen Hash Funktion gebildet, sondern mittels einer Divisionsrestmethode. Der Hash Wert wird gebildet, indem die Zeichen des Namens der Datei in entsprechenden *ASCII* Zeichen umgewandelt werden. Diese werden dann multipliziert und anschließend modulo mit dem Adressraum gerechnet. Dadurch ergibt sich eine sehr gute Verteilung der Dateien auf die verfügbaren Nodes.

Um bei langen Dateinamen bei der Berechnung zu große Zahlen zu vermeiden, wird die Berechnung mit dem *Horner Schema* geteilt.

Eine Schwachstelle bei dieser Berechnung ist, dass Dateien mit denselben Buchstaben, die lediglich vertauscht sind, immer auf den gleichen Rechnern landen. Dies sollte aber selbst bei sehr vielen Dateien kaum auftreten.

Duplikate und Prioritäten

Dateien erhalten beim Erstellen, vom Benutzer gewählt oder voreingestellt, außerdem eine Priorität. Diese steuert gleichzeitig die Menge der Duplikate.
Eine Datei bekommt zum Beispiel eine Priorität von 1 (sehr gering) bis m (sehr hoch)

Bei m werden einfach m Duplikate erstellt (falls möglich, das heißt genug Servents vorhanden sind)

Die Speicherung der Duplikate, erfolgt nach folgendem Prinzip:

0. Duplikat(Original)...(ermittelte Hash Adresse) mod 2^m
1. Duplikat.....(Hash Adresse + 1*Adressraum/Duplikatanzahl) mod 2^m
2. Duplikat.....(Hash Adresse + 2*Adressraum/Duplikatanzahl) mod 2^m
3. Duplikat.....(Hash Adresse + 3*Adressraum/Duplikatanzahl) mod 2^m
m. Duplikat.....(Hash Adresse + m*Adressraum/Duplikatanzahl) mod 2^m

10.3 Umsetzunng

Die Umsetzung benötigt zum Starten folgende Systemvorausetzungen:

- Java Runtime Enviroment (JRE – mindestens 1.5 aka Version 5)
- 128 MB RAM

Der benötigte Prozessor hängt stark von der gewählten Größe des Netzwerks ab.
Ein aktuelles JRE ist unter http://www.java.com/en/download/ zu finden.

Als Programmiersprache wurde Java gewählt, um Systemunabhängigkeit zu erreichen. Diese erscheint wesentlich, da die „globale Festplatte“ unterschiedliche Computer und damit auch Betriebssysteme beinhalten könnte. Daher war selbst für eine Simulationsanwendung diese Systemunabhängigkeit entscheidend.

Das Programm, sowie der gesamte Quellcode liegt dieser Arbeit auf einer CD-ROM bei. Unter GCFS\executables ist eine ausführbare jar Datei zu finden. Der Code ist im Verzeichnis GCFS\src zu finden. Ebenfalls befindet sich im GCFS Ordner eine optionale Projektdatei für die Entwicklungsumgebung JetBrains-IntelliJ.

Der Quellcode ist nach dem Model-View-Controller (MVC) Konzept gegliedert.
Die wesentlichen Bereiche sind:

- controlling (Controller-Klassen)
- gui (View-Klassen)
- servent (Model-Klassen)
    - filesystem

10.3.1 Programmoberfläche

Abbildung 27 zeigt die grafische Benutzeroberfläche der GCFS Simulation. Dargestellt wird ein Netzwerk mit 128 Nodes (2^7). Der „geöffnete“ Servent zeigt, dass bereits Dateien angelegt wurden.

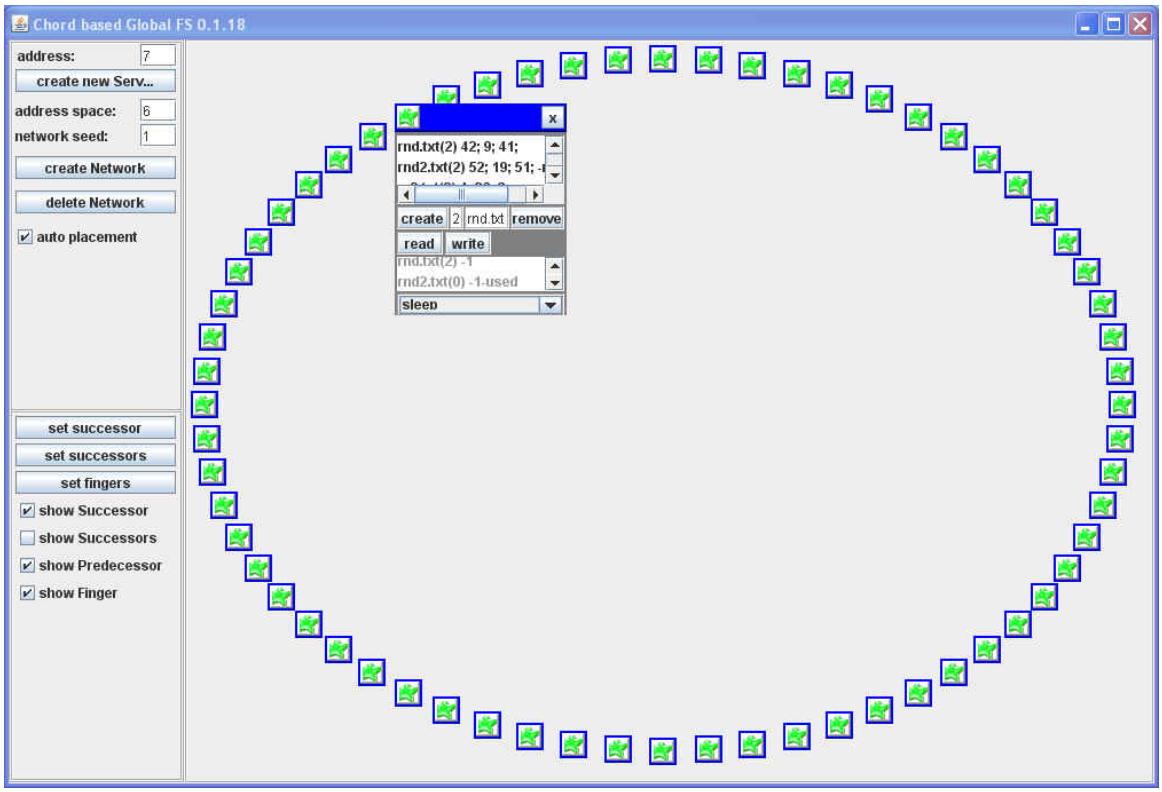

Abbildung 27: GCFS Simulations-Oberfläche

10.3.2 Technische Kurzanleitung

10.3.2.1 Controlling Panel

Abbildung 28: GCFS Controlling Panel

Create New Servent: Dies erstellt einen zu Beginn noch nicht im Netzwerk teilnehmenden Node. Die Adresse des Servent wird durch das Feld eingegeben. Ist ein Servent mit dieser Adresse bereits vorhanden, so muss diese geändert werden.

address space: Hier kann der Adressraum des zu erzeugenden Netzwerkes eingestellt werden. Achtung dieser Wert entspricht dem „m“ in folgender Gleichung: $n = 2^m$.

network seed: Bei der Erstellung eines Netzwerkes wird immer mit der Wahrscheinlichkeit des „seeds“ (zwischen 0 und 1) ein Servent mit einer Adresse platziert. Dies sorgt für ständig unterschiedliche Servents bei der Erstellung.

create Network: Erzeugt ein Netzwerk mit den eingestellten Parametern.

delete Network: Löscht das gesamte Netzwerk.
Vorsicht: auch Servents, die noch nicht mit dem Netzwerk verbunden sind.

auto placement: Diese Checkbox sorgt dafür, dass alle Servents visuell schön angeordnet werden. Sollte man händisch viele Servents bewegt haben und es sich empfiehlt einen Grundzustand wieder herzustellen, ist diese Funktion ebenso praktisch.

10.3.2.2 Servents und Darstellungs-Panel

Abbildung 29: GCFS Servents und Darstellungs-Panel

set successor: Dieser Button setzt sozusagen den Grundstein des Netzwerks. Er veranlasst jeden einzelnen Servent dazu, sich seinen Successor zu suchen (dies fordert sie gleichzeitig dazu auf, sich ihren Predecessor zu besorgen).*

set successors: Durch diese Funktion wird jeder Servent dazu aufgefordert seine Successoren richtig zu stellen.*

set fingers: Veranlasst die Servents dazu, ihre Fingertabelle anzulegen und richtig zu setzen.*

show Successor: Wenn der Mauszeiger über einen Servent gelangt, so wird sein Successor grafisch dargestellt, wenn diese Funktion aktiviert ist.

show Successors: Wenn der Mauszeiger über einen Servent gelangt, so werden seine weiteren Successoren grafisch dargestellt, wenn diese Funktion aktiviert ist.

show Predecessor: Wenn der Mauszeiger über einen Servent gelangt, so wird sein Predecessor grafisch dargestellt, wenn diese Funktion aktiviert ist.

show Finger: Wenn der Mauszeiger über einen Servent gelangt, so wird sein letzter Finger grafisch dargestellt, wenn diese Funktion aktiviert ist. Aufgrund der besseren visuellen Darstellbarkeit wird hier nur der Letzte gezeichnet. In dem *ToolTipText* der Servents sind jedoch alle einsehbar!

*Dieses Stabilisieren im Netzwerk wird normalerweise rein von den Servents initiiert. Für eine Simulation und damit Nachvollziehbarkeit ist es jedoch wichtig, dass diese Funktion vom Benutzer initiiert wird!

10.3.2.3 Servent Panel

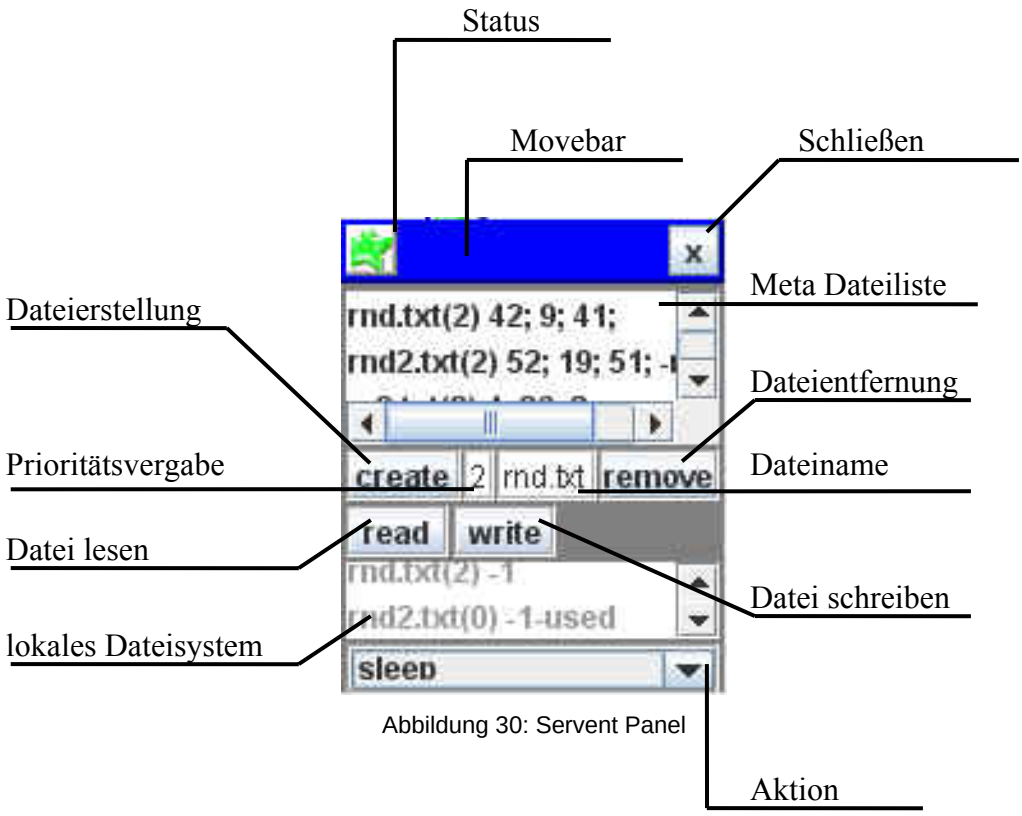

Abbildung 30: Servent Panel

Status: Dieses Feld zeigt interaktiv den Status des Servents an. Es kann drei Zustände annehmen. Nicht verbunden, teilweise verbunden (zum Beispiel: nur Successoren gesetzt) und voll im Netzwerk aktiv.

Movebar: Mit diesem Balken ist der Servent visuell grafisch bewegbar. Damit kann man ihn in eine schöne Position setzen, um ihn genauer zu betrachten.
Die Movebar verfügt ebenso über einen ToolTipText, welcher mehrzeilig ist und Informationen enthält. Darunter: ID, Successor(en), Predecessor und Fingertabelle.

Schließen: Entfernt den Servent aus dem Netzwerk.

Meta Dateiliste: Diese Liste zeigt die vollständige Meta Dateiliste an. Hierbei sieht man den Dateinamen und in Klammer die Anzahl der Duplikate, die existieren. Die Liste zeigt, wo die Duplikate (errechnet) liegen sollten. Anschließend kann noch ein "-read" oder "-write" stehen, welches einen Read und oder Write Lock anzeigt.

Prioritätsvergabe: Vor dem Erstellen einer Datei kann man hier die Priorität angeben. Dieses Feld ist voreingestellt, je nach Größe des Netzwerkes. Sollte jedoch eine Datei als extrem wichtig erscheinen, kann man hier händisch die Priorität setzen.

Dateiname: Der Dateiname der zu erstellenden Datei (Daraus wird auch der Hash Wert gebildet). Gleiche Dateinamen sind nicht erlaubt. Es sind sämtliche Zeichen erlaubt (für die Praxis empfiehlt es sich aber, keine Sonderzeichen zu verwenden).

Dateierstellung: Erzeugt die Datei lokal. Anschließend werden die Servents kontaktiert, auf dem die Datei tatsächlich gespeichert wird. Dies gilt danach auch für sämtliche Duplikate.

Dateientfernung: Entfernt die in der Meta Liste markierte Datei. Ist auf dieser jedoch ein Read oder Write Lock, so kann diese nicht entfernt werden.

Datei lesen: Erstellt einen Read Lock auf die entsprechende Datei und erzeugt beim lesenden Servent im Speicher ein Abbild. Es können mehrere Read Locks gleichzeitig existieren.

Datei schreiben: Erstellt einen Write Lock auf die entsprechende Datei und erzeugt beim schreibenden Servent im Speicher ein Abbild. Es kann immer nur ein Write Lock existieren.

lokales Dateisystem: Hier werden sämtliche Daten angezeigt, die tatsächlich auf der Festplatte des Servents gespeichert sind. Ebenso wie sämtliche Speicherabbilder, die bei Lese- oder Schreibvorgängen erzeugt wurden.
Neben dem Dateinamen wird in Klammer die Duplikatsnummer angezeigt, die lokal gespeichert ist.

Aktion: hier kann eine Aktion ausgeführt werden, die ausschließlich der betreffende Servent ausführt.
Zur Verfügung stehen:

- sleep (keine Aktivität)
- receive Successor
- receive Predecessor

10.4 Fazit

Die Simulation zeigt, dass ein GCFS auf Basis von Chord gut funktioniert. Das System skaliert sehr gut und bleibt bei der Auffindung von Daten, unabhängig von der Größe, sehr effizient. Problematisch sind nur Dateien, die sehr groß sind. Ab etwa 200 Megabyte belegt alleine die Duplikatsbildung bei standardmäßigen ein Gigabit (1000-MBit-Ethernet) Leitungen fast die komplette Bandbreite.
Die getesteten Systeme beinhalteten bis zu 1000 Nodes. Bei noch größeren Netzwerken benötigt es leistungsfähige Rechner.

Folgende sinnvolle Erweiterung, bzw. Änderung im System stellte sich als sehr sinnvoll heraus:
Im jetzigen System wird die Metaliste bei einer Veränderung im kompletten Netzwerk aktualisiert. Dieser Aufwand könnte erheblich optimiert werden, indem die Metalisten nur auf bestimmten Servents gespeichert sind. Welche dies sind, ist unerheblich, muss jedoch mathematisch genau definiert sein. Dadurch würde die Effizienz des Netzwerks weiter steigen, da die Metaliste nicht im kompletten System aktualisiert werden müsste.

Testsimulation

Abbildung 31 zeigt eine GCFS Simulation mit 256 Nodes. Insgesamt waren eintausend Dateien in dem Netzwerk verfügbar.
Diese Testumgebung wurde einem „Stresstest“ unterzogen. Jeweils drei zufällige Servents pro Sekunde wurden entfernt und die gleiche Menge wieder hinzugefügt. Der Stabiliserungsalgorithmus wurde von den Servents alle 20s asynchron (jeweils zu unterschiedlichen Zeitpunkten) durchgeführt.

Nach eintausend Testschritten wurde folgendes Ergebnis festgestellt:

- zwei Servents verloren den Kontakt zum Netzwerk
- sieben Dateien wurden nicht mehr aufgefunden

In der Realität sind so häufige joins/leaves und damit der momentane Verlust von Dateien oder Servents die den Kontakt zum Netzwerk verlieren, höchst unwahrscheinlich.
Bei einer Erhöhung des Duplikationsfaktors von drei auf vier, wurden alle Dateien nach dem Testlauf wieder aufgefunden

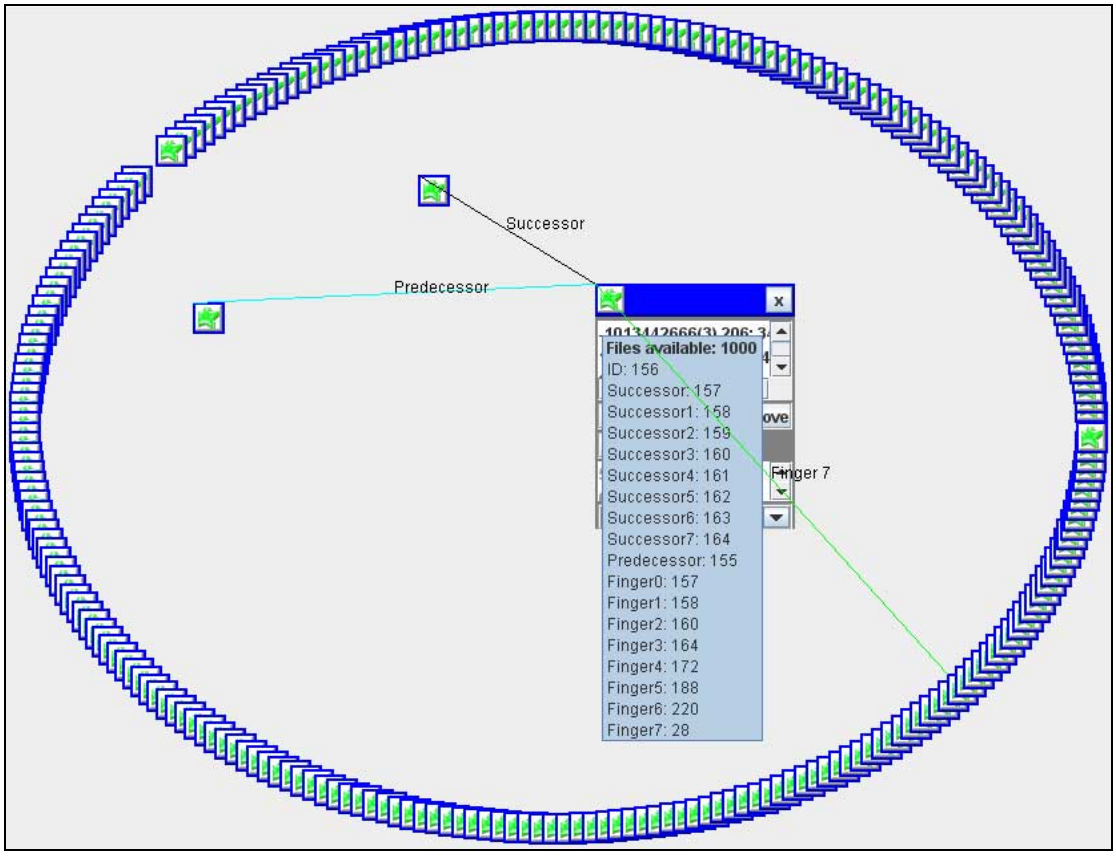

Abbildung 31: GCFS Simulation mit 256 Nodes

11 Resümee

Es wurde die Bedeutung von Filesharing Netzwerken geschildert, die nicht nur entscheidend von ihrer Geschichte und Rechtlichem geprägt war, sondern noch immer wesentlich beeinflusst wird. Ein Beispiel hierfür ist das Patent im Amerikanischen Raum für das Auffinden von Daten mittels Hashing.
Durch eine Einteilung in Generationen, sind die vorgestellten Filesharing Technologien eindeutig zuzuordnen.

Entwicklungsstadium

Einige Applikationen bauen bereits auf neue Techniken auf, sind aber aktuell noch im Entwicklungsstadium. Interessant ist, dass eine Technologie, die so nahe liegt wie SET, erst so spät umgesetzt wurde, da die Idee selber nicht sehr komplex und logisch ist.

Neue und mögliche Technologien

Bei der Untersuchung des Einsatzes von neuen Filesharing Technologien oder Kombinationen davon, ergaben sich viele Möglichkeiten.
Während bei den meisten Techniken die Vorteile überwiegen, erscheinen Botnetze als die problematischste. Denn so faszinierend diese Technik ist, so gefährlich könnte sie sein.

Eine besonders umfangreiche Analyse war in dem Kapitel MMOGs notwendig. Erschien es zu Beginn noch leicht, wurde bald klar, dass viele wichtige Details zu beachten sind. Ein besonderes Beispiel hierfür ist die Aktionssicherheit, für die leider keine Lösung gefunden wurde.
Alle überlegten Ansätze dafür scheiterten immer bei einem Detail der Aktionssicherheit, weswegen sie der Arbeit nicht beigefügt sind.

Bei den zahlreichen Updates, die von Programmen heutzutage auf dem Durchschnittsrechner durchgeführt werden, erscheint gerade in wirtschaftlicher Hinsicht die Technologie im Kapitel Programmupdates sehr lohnenswert.

Dass so wenige Möglichkeiten umgesetzt sind, liegt auch daran, dass diese Technologien noch sehr jung sind und sich erst durch Universitätsforschungen und Open-Source-Implementationen in der Praxis bewähren müssen, bevor sie von der Wirtschaft aufgegriffen werden.

Proof of Concept

Das Proof of Concept zeigt ein modernes DHT Filesharing Netzwerk, welches ein globales Dateisystem simuliert. Das System funktionierte bei den Simulationen einwandfrei. Es wurden nur die wichtigsten Aspekte eines Dateisystems berücksichtigt; Erweiterungen stellen jedoch keine weiteren Anforderungen an das Netzwerk selber, sondern sind reine Aufwandssache. Vor allem die Skalierbarkeit dieses Systems stellte sich als besonders außergewöhnlich heraus.
Ein solches globales Dateisystem ist sicher auch für Firmen als Anwender sehr interessant. Leider ist der Umsetzungsaufwand sehr hoch und eine Implementation wäre ein Großprojekt, weswegen es wohl noch einige Jahre dauern wird, bis ein dezentrales Dateisystem einsatzbereit ist.

Wirtschaftshemmnis

Insgesamt sind die Möglichkeiten gewaltig. Umsetzungen von Firmen werden jedoch wohl noch auf sich warten lassen, denn gerade in der Wirtschaft haftet den Filesharing Netzwerken nach wie vor der Makel des illegalen Musikaustauschs zu sehr an.
Auch sind Patente in Amerika ein Hindernis für weitere Entwicklungen. Dort ist zum Beispiel der essentielle Auffindungsalgorithmus der DHTs patentiert.
Vorstöße in dieser Richtung sind somit wohl eher von der Open-Source-Gemeinschaft zu erwarten, als von der Industrie.

Anmerkung

Informationen über Filesharing Netzwerke sind sehr rar in gedruckter Buchform. Lediglich die Geschichte und über rechtliche Thematiken sind zahlreiche Werke zu finden.

12 Glossar

Achillesferse: Die Achillesferse stammt aus der griechischen Mythologie. Die rechte Ferse war die einzige Stelle, an welcher der Sagenheld Achilleus verwundbar war. Der Begriff wird heute vor allem als Metapher verwendet und bezeichnet eine verwundbare Stelle eines Systems.

ASCII: [æski] Steht für „American Standard Code for Information Interchange“ und ist eine 7-Bit-Zeichenkodierung

Bucket: Als Bucket wird eine abstrakte Datenstruktur bezeichnet, die sich zur Sortierung diskreter Daten eignet

ceil: Ceil (Aus dem Englischen: ceiling, Decke) bezeichnet eine Funktion zum Aufrunden von Zahlen und liefert die nächsthöhere Ganzzahl

Exploit(er): Ein Exploit ist das Ausnutzen einer Schwachstelle eines Systems (Sicherheitslücke). Dementsprechend ist ein Exploiter jemand, der gezielt diese für den eigenen Vorteil nutzt.

Flamewar: Ein Flame (aus dem Englischen: to flame, aufflammen) ist ein ruppiger oder polemischer Kommentar. Flamewar ist es, wenn mehrere Personen „flamen“.

Grids: Unter Grid versteht man eine Infrastruktur, welche eine integrierte, gemeinschaftliche Verwendung von geographisch getrennt liegenden autonomen Ressourcen erlaubt.

GUI: GUI steht für Grafical User Interface.
Eine grafische Benutzeroberfläche erlaubt dem Benutzer eine Interaktion mit dem Rechner über grafisch gestaltete Elemente mittels Zeigegeräten wie eine Maus.

Hashtabelle: Spezielle Indexstruktur, die zum Beispiel dazu geeignet ist, bestimmte Daten in einer großen Datenmenge zu finden. Sie stellt eine Alternative zu Baumstrukturen dar.

Horner Schema: Umformungsverfahren für Polynome, um die Berechnung von Funktionswerten so einfach wie möglich zu machen.

Hub: Ein Hub ist ein Knotenpunkt (innerhalb dieser Arbeit ist damit ein Netzwerkknoten gemeint).

ld(n): Logarithmus Dualis – oder Logarithmus zur Basis 2 – $ld(x) = \ln(x)/\ln(2)$

MIME (-Type): Multipurpose Internet Mail Extensions ist ein Kodierstandard, der den Inhalt von Daten deklariert.

Node: Eine Node bezeichnet einen Netzknoten.

Overhead: Bezeichnet Daten, die nicht direkt angefragt wurden, jedoch zur Erhaltung und Funktionalität des Systems wichtig sind.

pareto-effizient: Beschreibt einen Zustand, in dem es nicht mehr möglich ist, mindestens ein Individuum besser zu stellen, ohne ein anderes schlechter zu stellen.

Ping: Ein Computerprogramm, mit welchem geprüft werden kann, ob ein bestimmter Host in einem IP-Netzwerk erreichbar ist.

Pong: Antwort auf einen Ping; meistens mit Absenderinformationen.

Queue: Warteschlange nach dem „first in, first out“ Prinzip.

Schneeballsystem: Auch Pyramidensystem genannt; charakterisiert sich im Wesentlichen dadurch, dass durch das Anwerben von neuen Teilnehmern, die ihrerseits wiederum eine neue Generation von Teilnehmern anwerben sollen.

Servent: Applikation, die **Serv**er und Cli**ent** in sich vereint. „Umgangssprachlich“ oft als Servant bezeichnet, da Servant so viel bedeuted wie „jemand der arbeitet“.

SHA1: Eine des **s**ecure **h**ash **a**lgorithm. Er stellt eine standardisierte kryptologische Hash-Funktion dar. Secure bedeutet Kollisionsfrei. Nach aktuellem Stand ist dieser Algorithmus nicht mehr sicher, da er geknackt wurde.

Stream: Mit Datenströmen (englisch: data streams) bezeichnet man in der Informatik kontinuierliche Abfolgen von Datensätzen, deren Ende nicht im Voraus abzusehen ist.

Traffic: Fluss von Digitaldaten innerhalb von Computernetzwerken.

tit-for-tat: "wie du mir, so ich dir".

ToolTipText: Ein kurzes Informationsfenster, das meist angezeigt wird, wenn sich der Mauszeiger über einem grafischen Element befindet.

TTL: Time To Live. Gibt an wie oft ein Paket noch weitergeleitet werden soll. Erreicht dieser Wert „0“, so wird es nicht mehr versandt.

Tupel: Ein Tupel bezeichnet eine Zusammenstellung von Objekten, mit einer bestimmten Anzahl von Elementen, sowie eine festgelegte Reihenfolge.

Webwarez: ['webwarɛs] Bezeichnet im Computerjargon illegal beschaffte oder verbreitete Software durch das Internet.

XOR: Eine mathematische exklusive-oder-Verknüpfung. Eine Gesamtaussage ist dann wahr, wenn eine von zwei Aussagen wahr ist, aber nicht beide.

13 Abbildungsverzeichnis

14 Literaturverweise

[ADS] Andersen, David G.: SET
URL: http://www.eurekalert.org/pub_releases/2007-04/cmu-cmp041007.php
Letzter Zugriff am 18.07.2007

[ANT] Ants: Anonymes Filesharing
URL: http://sourceforge.net/projects/antsp2p/
Letzter Zugriff am 28.07.2007

[CCC] Chaos Computer Club: Boycott the music industry
URL: http://www.ccc.de/campaigns/boycott-musicindustry
Letzter Zugriff am 07.09.2007

[CFS] CFS: Wide-area cooperative storage with CFS
URL: http://pdos.csail.mit.edu/papers/cfs:sosp01/
Letzter Zugriff am 17.05.2007

[CRDC] Congress readies broad new digital copyright bill
URL: http://news.com.com/2100-1028_3-6064016.html
Letzter Zugriff am 07.09.2007

[EGEE] EGEE: Enabling Grids for E-sciencE (EGEE)
URL: http://public.eu-egee.org/
Letzter Zugriff am 02.08.2007

[EDK] eDonkey: Filesharing Anwendung
URL: http://www.overnet.org/
Letzter Zugriff am 06.09.2007

[ELL] Ellacoya: Media Alert
URL:http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf
Letzter Zugriff am 07.03.2007

[FNV] FileNavigator: Filesharing Anwendung
URL: http://www.filesharingzone.de/filenavigator.php
Letzter Zugriff am 18.04.2007

[FST] The giFT Project: Internet File Transfer
URL: http://gift.sourceforge.net/
Letzter Zugriff am 06.09.2007

[GTL1] Gnutella: Protocol Development
URL: http://rfc-gnutella.sourceforge.net/developer/index.html
Letzter Zugriff am 04.08.2007

[GTL2] Gnutella: The Gnutella Developer Forum
URL: http://groups.yahoo.com/group/the_gdf/
Letzter Zugriff am 16.07.2007

[GTL3] gnuTellaVision: Real Time Visualization of a Peer to Peer Network
URL: http://people.ischool.berkeley.edu/~rachna/courses/infoviz/gtv/paper.html
Letzter Zugriff am 06.09.2007

[HHE] Hammerle, Hannelore (EGEE Public Relations and Press Office): - EGEE
URL: http://public.eu-egee.org/intro/
Letzter Zugriff am 02.06.2007

[KDC] Kendall, David: Copyright in Cyberspace
URL: http://www.copyrightassembly.org/briefing/DEKWabashSpeech4.htm
Nicht öffentlich zugänglich- alternative unter
URL: http://www.quicktopic.com/25/D/cD8dwc52A3p.html
Letzter Zugriff am 12.05.2007

[KJSE] Krömer, Jan; Sen, Evrim, 2006
NO COPY: Die Welt der digitalen Raubkopie
Leipzig, Tropen Verlag

[KZA] KaZaA: Filesharing Anwendung
URL: http://www.kazaa.com/
Letzter Zugriff am 12.04.2007

[LMW] Limewire: Filesharing Anwendung
URL: http://www.limewire.com/
Letzter Zugriff am 16.04.2007

[MFB] Mohr, Franz (Bearbeitung), 2006
KODEX: Bürgerliches Recht, 33. Auflage
Wien, LexisNexis Verlag

[MMK] Maymounkov, Mazières: Kademlia
URL: http://www.cs.rice.edu/Conferences/IPTPS02/109.pdf
Letzter Zugriff am 12.08.2007

[MPSC] Mahlmann, Peter; Schindelhauer, Christian, 2007
P2P Netzwerke: Algorithmen Und Methoden, 1. Auflage
Berlin Heidelberg, Springer Verlag

[NA1] OpenNap: Open Source Napster-Clone
URL: http://opennap.sourceforge.net/ 18.04.2007
Letzter Zugriff am 18.05.2007

[NA2] JNerve: Java Napster Server Project
URL: http://jnerve.sourceforge.net/
Letzter Zugriff am 12.05.2007

[OPR] Opera: Key Features
URL: http://www.opera.com/products/desktop/?htlanguage=de/
Letzter Zugriff am 20.06.2007

[PST] Pastry: A substrate for peer-to-peer applications
URL: http://research.microsoft.com/~antr/Pastry/
Letzter Zugriff: 12.06.2007

[RJN] Röttgers, Janko: Napster wird stillgelegt
URL: http://www.heise.de/tp/r4/artikel/8/8442/1.html
Letzter Zugriff am 16.04.2007

[SC1] SETI@home: Search for Extraterrestrial Intelligence
URL: http://setiathome.berkeley.edu/
Letzter Zugriff am 14.05.2007

[SC2] Evolution@home: Evolutionary- research
URL: http://evolutionary-research.net/
Letzter Zugriff am 12.07.2007

[SET] SET: Similarity Enhanced Transfers
URL: http://www.eurekalert.org/
pub_releases/2007-04/cmu-cmp041007.php
Letzter Zugriff am 08.10.2007

[SFR] Schmidbauer, Franz: Das Recht im Internet
URL: http://www.internet4jurists.at
Letzter Zugriff am 16.04.2007

[SMKKB] Stoica, Morris, Karger, Kaashoek, Balakrishnan: Chord
URL: http://pdos.csail.mit.edu/chord/
Letzter Zugriff am 12.03.2007

[SRP] Steinmetz, Ralf, 2005
Peer-to-Peer Systems and Applications
Berlin, Springer Verlag

[SYT] Stephens Lighthouse: YouTube
URL: http://stephenslighthouse.sirsi.com/
archives/2006/07/youtube_ruleswh.html
Letzter Zugriff am 16.07.2007

[TIJ] Taylor, Ian J., 2004
From P2P to Web Services and Grids: Peers in a Client/Server World
Berlin, Springer Verlag

[WMX] WinMX: Filesharing Anwendung
URL: http://www.winmxworld.com/
Letzter Zugriff am 19.06.2007

[WST] Waste: Anonymous, secure and encrypted collaboration tool
URL: http://waste.sourceforge.net/
Letzter Zugriff am 14.05.2007

15 Anhang

15.1 Byzantine-fault-tolerance

Den Namen verdankt dieses Problem dem Byzantinischen Generals Problem, welches die Urform davon ist.

Bei dem GCFS nach dem ursprünglichen Sinne sind nicht Betrüger (die das System absichtlich sabotieren wollen), sondern vielmehr fehlfunktionierende Servents gemeint. Diese können zum Beispiel nicht antworten, da eventuell ihr Betriebssystem abgestürzt ist. Falls ein Servent unterschiedliche (protokollkonforme) Ergebnisse liefert, spricht man von einem byzantinischen Fehler.

Eine einfache Mehrheitsentscheidung reicht zur Behandlung eines byzantinischen Fehlers nicht aus, da den verschiedenen Servents unterschiedliche Informationen vorliegen. Drei Teilnehmer reichen damit nicht aus um einen byzantinischen Fehler zu erkennen und zu korrigieren. Bei einem unterschiedlichen Ergebnis müssen damit mindestens vier Servents beteiligt sein, um eine doppelte Mehrheitsentscheidung zu treffen („2-aus-4 Entscheidung“ oder auch „Medianbildung“).

Abbildung 32 erläutert das Problem näher:

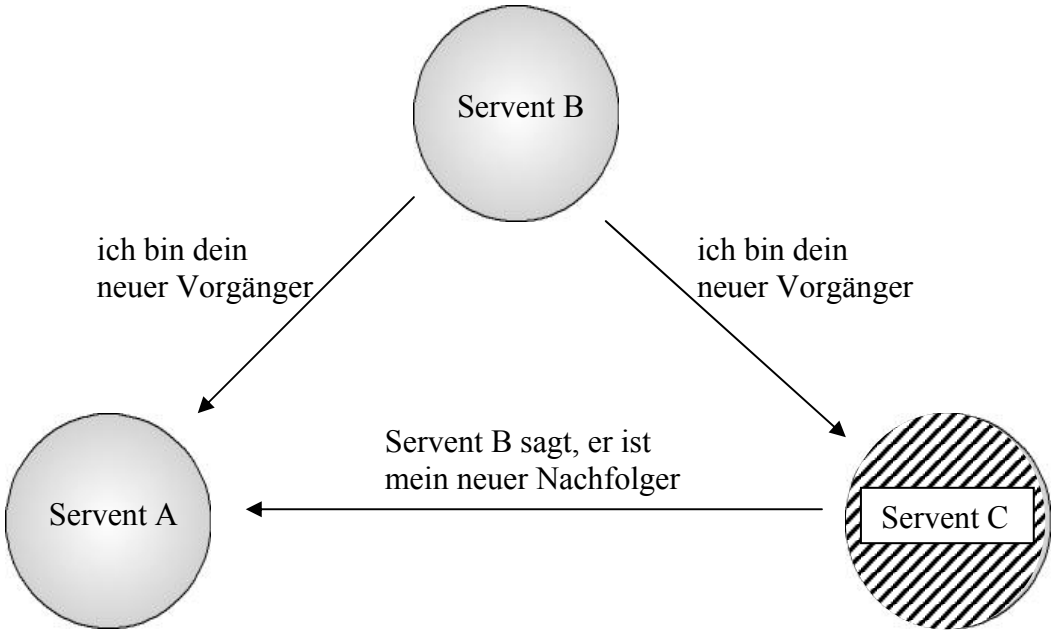

Abbildung 32: Byzantine-fault-tolerance mit Servent C als Falschspieler

Daher gibt es folgende Voraussetzung die erfüllt sein muss:
m… Anzahl der Servents
t… Anzahl der fehlerhaften bzw. sabotierenden Servents

$m \geq 3 \cdot t + 1$

15.2 Chaos Computer Club: Boycott the music industry

April 03, 2004 (tina)
After the legal action the IFPI (International Federation of the Phonographic Industry) took against filesharing users, the CCC calls for a the boycott of IFPI member companies. Blaming the users as a remedy for missing the information age is unacceptable. The industry should have adapted their business strategies to the digital world long ago anyway.

Freedom of Information Is No Crime

The CCC regards those lawsuits filed by the German section of the IFPI as highly questionable, to say the least. We won't accept the music industry trying to accomplish their goals by striking ordinary users with awe as the industry claims immense amounts for indemnification. Such claims are not even enforcable by law in Germany. The IFPI's intentions are rather to intimidate participants of filesharing services. This becomes clear with the recent campaign of the GVU (Society for the Prosecution of Copyright Infringements). Here, too, legal misinformation is spread concerning the culpability of copyright infringements, to shy away users from using p2p-services.

Copyright is no natural right, but rather a compensation for the author as he or she makes her or his work available for the general public. Predominantly, copyright is a part of the personal rights. It should also be noted that German law, unlike its American counterpart, doesn't know the concept of copyright being given away, e.g. an artist can never lose the rights for his songs to a record company. "Copyright" is therefore only partially a translation for the German "Urheberrecht". To secure the author's economical existence, certain rights to distribute his/her work are given to him or her, but they underlie certain limitations. For example, a work may still be copied freely for private use. This right, also called 'fair use', is a part of the freedom of information and a fundamental human right in the eyes of the CCC.

The music industry now is trying to undermine this fact with countless campaigns. It tries to drag down the concept of fair use to the same level with child pornography and Nazi-propaganda. The chairman of the board of the Society for Musical Performance and Mechanical Copyrights (GEMA) demanded on the German music fair Popkomm that p2p users should be pursuited using the same ways and means which have been in use by the police to pursuit child pornographers and neonazis. This is an immense infamy and degradation for filesharing users.

But the economically rather unimportant copyright industry wants to go still one step further. The new 'Guidelines of the European Union for sanctions and procedures to protect the rights of intellectual property' grant them the right to search private homes without first obtaining a search warrant from a judge. Misuse and even industrial espionage are guaranteed to happen.

So it is only natural to ask: *"Will the population be criminalised as a whole, because the industry is not able to deliver the offer for the overwhelming demand? Will the individual's freedom be sacrificed for an industry's demand for market stability? How come you can make more money with cell phone ringtones than with music?"*

Next to the political reasons for a boycott of the music industry, there are also a few very practical reasons:

- with the profits from CD sales the industry finances the legal actions against our children. Why should we, as a society, finance the munition of the enemy?
- with the profits from CD sales, the music industry finances research and production of DRM and other measures of anti-copy mechanisms. Why should we finance technologies that will keep us from exerting our right to freedom of information and fair use?
- we have bought our right to privately copy CDs by paying GEMA dues on every raw CD and DVD and on the CD-burners as well. It is inacceptable that this right is now referred to as 'stealing'.

Why are p2p-networks so popular? There are a few reasons:

- **the quality of publicly available music has gone down.** Music now seems only to exist to guarantee a bigger turnover for the industry. Since most modern bands are only in the charts for a short time now, less people bother to buy expensive CDs of songs and bands which nobody will remember after a few years anyway.
- **the prices for audio CDs are too high.** At least the main target group of teenagers and young adults can't afford the highly priced CDs. Studies prove that popmusic's chief purchasers are adults of 40 years and up.
- **CD anti-copy provisions detains people from playing their CDs in any player except the very newest.** Even many CD-player for cars cannot play DRM-protected CDs. This drives many people to the filesharing services to download and burn their music to run in their players.
- **the variety in music stores is limited.** If you're looking for rare pieces, the filesharing services are your last resort. For people who don't live in big cities or don't have the time to visit countless music stores, the p2p-network offer the chances of finding the favorite song from 30 years ago without much running or waiting.
- **filesharing services are the ideal distribution channel for the new generation** - the only thing missing is an appropriate payment function. The music industry has missed out on the internet movement, while the listeners have found their own way to use current technology to share their music collection and make new friends. Most listeners would be very happy to pay their favorite artist for the music they make. But there are yet ways to be found to get the money to the artist in a more direct way.

The music industry should stop whining now! The CCC is therefore demands: hit them where they it will hurt them the most. Take away their turnover! So they won't be able to use their profits to take on legal actions and advertisement campaigns against their customers.

The CCC has made banner and images for free use to support this campaign. Filesharers may voice their anger using our images and linking to our protest in this way.
We ask people to link to us from as many websites as possible.

Lastly we would like to point you to the words of the German comedian Dirk Bach at the 2004 Echo Awards to the congregated 'Pop Idol'-style clone-bands: *"How dare you wonder at your sales going down?"*

[CCC]

15.3 Congress readies broad new digital copyright bill

By Declan McCullagh
Staff Writer, CNET News.com

Published: April 23, 2006, 6:00 AM PDT
Last modified: April 24, 2006, 10:00 AM PDT

update For the last few years, a coalition of technology companies, academics and computer programmers has been trying to persuade Congress to scale back the Digital Millennium Copyright Act.

Now Congress is preparing to do precisely the opposite. A proposed copyright law seen by CNET News.com would expand the DMCA's restrictions on software that can bypass copy protections and grant federal police more wiretapping and enforcement powers.

High Impact

The draft legislation, created by the Bush administration and backed by Rep. Lamar Smith, already enjoys the support of large copyright holders such as the Recording Industry Association of America. Smith, a Texas Republican, is the chairman of the U.S. House of Representatives subcommittee that oversees intellectual-property law.
A spokesman for the House Judiciary Committee said Friday that the Intellectual Property Protection Act of 2006 is expected to "be introduced in the near future." Beth Frigola, Smith's press secretary, added Monday that Wisconsin Republican F. James Sensenbrenner, chairman of the full House Judiciary Committee, will be leading the effort.
"The bill as a whole does a lot of good things," said Keith Kupferschmid, vice president for intellectual property and enforcement at the Software and Information Industry Association in Washington, D.C. "It gives the (Justice Department) the ability to do things to combat IP crime that they now can't presently do."
During a speech in November, Attorney General Alberto Gonzales endorsed the idea and said at the time that he would send Congress draft legislation. Such changes are necessary because new technology is "encouraging large-scale criminal enterprises to get involved in intellectual-property theft," Gonzales said, adding that proceeds from the illicit businesses are used, "quite frankly, to fund terrorism activities."
The 24-page bill is a far-reaching medley of different proposals cobbled together. One would, for instance, create a new federal crime of just trying to commit copyright infringement. Such willful attempts at piracy, even if they fail, could be punished by up to 10 years in prison.
It also represents a political setback for critics of expanding copyright law, who have been backing federal legislation that veers in the opposite direction and permits bypassing copy

protection for "fair use" purposes. That bill--introduced in 2002 by Rep. Rick Boucher, a Virginia Democrat--has been bottled up in a subcommittee ever since.

A DMCA dispute

But one of the more controversial sections may be the changes to the DMCA. Under current law, Section 1201 of the law generally prohibits distributing or trafficking in any software or hardware that can be used to bypass copy-protection devices. (That section already has been used against a Princeton computer science professor, Russian programmer Dmitry Sklyarov and a toner cartridge remanufacturer.)

Smith's measure would expand those civil and criminal restrictions. Instead of merely targeting distribution, the new language says nobody may "make, import, export, obtain control of, or possess" such anticircumvention tools if they may be redistributed to someone else.

"It's one degree more likely that mere communication about the means of accomplishing a hack would be subject to penalties," said Peter Jaszi, who teaches copyright law at American University and is critical of attempts to expand it.

Even the current wording of the DMCA has alarmed security researchers. Ed Felten, the Princeton professor, told the Copyright Office last month that he and a colleague were the first to uncover the so-called "rootkit" on some Sony BMG Music Entertainment CDs--but delayed publishing their findings for fear of being sued under the DMCA. A report prepared by critics of the DMCA says it quashes free speech and chokes innovation.

The SIIA's Kupferschmid, though, downplayed concerns about the expansion of the DMCA. "We really see this provision as far as any changes to the DMCA go as merely a housekeeping provision, not really a substantive change whatsoever," he said. "They're really to just make the definition of trafficking consistent throughout the DMCA and other provisions within copyright law uniform."

The SIIA's board of directors includes Symantec, Sun Microsystems, Oracle, Intuit and Red Hat.

Jessica Litman, who teaches copyright law at Wayne State University, views the DMCA expansion as more than just a minor change. "If Sony had decided to stand on its rights and either McAfee or Norton Antivirus had tried to remove the rootkit from my hard drive, we'd all be violating this expanded definition," Litman said.

The proposed law scheduled to be introduced by Rep. Smith also does the following:

• Permits wiretaps in investigations of copyright crimes, trade secret theft and economic espionage. It would establish a new copyright unit inside the FBI and budgets $20 million on topics including creating "advanced tools of forensic science to investigate" copyright crimes.

• Amends existing law to permit criminal enforcement of copyright violations even if the work was not registered with the U.S. Copyright Office.

• Boosts criminal penalties for copyright infringement originally created by the No Electronic Theft Act of 1997 from five years to 10 years (and 10 years to 20 years for subsequent offenses). The NET Act targets noncommercial piracy including posting copyrighted photos, videos or news articles on a Web site if the value exceeds $1,000.

• Creates civil asset forfeiture penalties for anything used in copyright piracy. Computers or other equipment seized must be "destroyed" or otherwise disposed of, for instance at a

government auction. Criminal asset forfeiture will be done following the rules established by federal drug laws.
• Says copyright holders can impound "records documenting the manufacture, sale or receipt of items involved in" infringements.
Jason Schultz, a staff attorney at the digital-rights group the Electronic Frontier Foundation, says the recording industry would be delighted to have the right to impound records. In a piracy lawsuit, "they want server logs," Schultz said. "They want to know every single person who's ever downloaded (certain files)--their IP addresses, everything."
CNET News.com's Anne Broache contributed to this report.

[CDRC]