Hauptseminar Telematik

SS2002

Filesharing Systeme

Betreuer: Thorsten Strufe

Name: Ronny Heidenreich
Matrikelnummer: 27146
Studiengang: Informatik
Fakultät: Fakultät für Informatik und Automatisierung
Universität: Technische Universität - Ilmenau

Inhaltsverzeichnis

1. Aufgabenstellung - Motivation

Analyse und Vergleich existierender Filesharing Systeme

Schwerpunkte:

Topologie, Protokoll, Datenhandling

Beschreibung:

Filesharingsysteme gibt es mittlerweile wie Sand am Meer. Von ***Napster*** über ***Gnutella*** bis zu ***Morpheus*** (***FastTrack***) sind viele davon in der Presse gewesen. Dazu kommen diverse akademische wie der ***Eternity-Service***, ***Freenet*** sowie einfache Varianten a la ***eDonkey*** etc. Da diese Systeme häufig fälschlicherweise unter dem Begriff des Peer-to-Peer-Computing zusammengefasst werden, was dem Begriff kaum gerecht wird, sollen diese auf ihre Topologie, genutzte Protokolle und die Art und Weise des Datenhandling untersucht und klassifiziert werden.

2. Peer-to-Peer (P2P) - Begriffsklärung - Beispiele

2.1 Begriffsklärung

Zu Beginn einige Erläuterungen zu Begriffen, die im Zusammenhang mit Peer-to-Peer häufig genannt werden.

- ***Peer(s):***
  Alle Geräte in einem geschichteten Kommunikationsnetzwerk, die auf der gleichen Protokollebene arbeiten.
  [URL: http://www.kefk.net/P2P/Grundlagen/Begriffe/index.asp]

- ***Peer-to-Peer:***
  Bedeutet etwa soviel wie: von „gleich“ zu „gleich“.

- ***Peer-to-Peer-Kommunikation:***
  Der Informationsaustausch zwischen Geräten, die in einer geschichteten Netzwerkarchitektur auf der gleichen Protokollebene arbeiten.
  [URL: http://www.kefk.net/P2P/Grundlagen/Begriffe/index.asp]

- ***Peer-to-Peer-Architektur:***
  Ein Netzwerk aus mehreren Computern, die das gleiche Programm oder den gleichen Programmtyp nutzen, mit dem Daten kommunizieren und gemeinsam genutzt werden. Jeder Computer bzw. jeder Peer ist, hierarchisch betrachtet, gleichwertig. Außerdem übt jeder Computer gegenüber den anderen Computern des Netzwerks eine Serverfunktion aus. Im Gegensatz zur Client-Server-Architektur ist ein dedizierter (zugehöriger) Dateiserver nicht erforderlich.
  [URL: http://www.kefk.net/P2P/Grundlagen/Begriffe/index.asp]

- ***Peer-to-Peer-Computing:***
  Technologien, die den direkten Austausch von Diensten oder Daten zwischen Computern ermöglichen. Heißt übersetzt etwa "von gleich zu gleich" und bezeichnet ein Netzwerk, in dem die Daten nicht wie im World Wide Web auf zentralen Servern, sondern auf den Rechnern der Netzwerkteilnehmer gespeichert und verarbeitet werden. Alles was der User braucht, ist eine Software, die ihn mit anderen Usern verbindet und einen direkten Zugriff auf dessen Daten ermöglicht.

- ***Filesharing:***
  Filesharing = Dateiaustausch, gemeinsame Datennutzung

2.2 Beispiele

- **Peer-to-Peer-Strukturen im frühen Internet:**

  - Beispiel: ***Usenet***
    Das Usenet basiert darauf, dass Dateien zwischen Computern kopiert werden. Es wurde als System mit dezentralisierten Kontrollstrukturen implementiert. Das Usenet basierte ursprünglich auf dem Protokoll ***UUCP*** (**U**nix to **U**nix **C**o**P**y). (UUCP stellt eine Verbindung zwischen zwei Computern her, tauscht Dateien aus und unterbricht die Verbindung.)
    [http://www.kefk.net/P2P/Grundlagen/Definition/index.asp]

  - Beispiel: ***DNS***
    Das 1983 eingeführte DNS (Domain Name System) ist ein Beispiel für ein System, das Peer-to-Peer-Strukturen mit hierarchischen Kontrollstrukturen mischt. Nameserver arbeiten grundsätzlich sowohl als Client als auch als Server.
    [http://www.kefk.net/P2P/Grundlagen/Definition/index.asp]

    Die Domainnamen (also z.B.: www.tu-ilmenau.de) werden alle durch sogenannte Domain Name Server verwaltet. Ein Domain Name Server macht dabei nichts anderes, als die "lesbaren" Domainnamen (URLs) in ihre jeweiligen numerischen Werte (z.B.: 141.24.190.23) umzusetzen.

- **Peer-to-Peer-Strukturen im heutigen Internet:**
  P2P ist nicht nur auf die bekannten Tauschbörsen wie ***Napster***, ***Gnutella*** oder ***Freenet*** beschränkt, die ihr Haupteinsatzgebiet im Filesharing Bereich haben.

  Es gibt noch weitere Anwendungen in anderen Bereichen, so z.B.

  - ***Collaboration*** und ***Groupware*** (gemeinsames arbeiten)
  - ***Distributed Computation*** (verteiltes rechnen, z.B. ***SETI@Home***)
  - ***Instant Messaging*** (z.B. ***ICQ***, ***Chat***)

  die teilweise zu den P2P-Technologien gezählt werden.
  [http://www.kefk.net/P2P/Grundlagen/Definition/index.asp]

3. Allgemeine Betrachtung von Netzwerktopologien

Der dritte Teil dieser Arbeit wird einige wesentlich Netwerktopologien in ihren Grundzügen vorstellen und Einsatzgebiete in aktuellen Filesharing Netzwerken aufzeigen.

Anmerkung:

- Die Topologien sind unterteilt bzw. bezogen auf den Informationsfluss.
- Nodes sind eigenständige Computer oder Programme.
- Verbindungen zeigen an, dass die Nodes Informationen teilen oder austauschen.
- Es herrscht ein ungerichteter Informationsfluss zwischen den Nodes.

Es können 4 wesentliche Topologien unterschieden werden:

- ***Zentralisiert*** (centralized)
- ***Ring*** (ring)
- ***Hierarchisch*** (hierarchical)
- ***Dezentralisiert*** (decentralized)

3.1 Zentralisiert

Funktioniert im Wesentlichen nach dem Client-Server Prinzip, d.h. alle Funktionen & Informationen sind in einem Server vereint. Die einzelnen Clienten verbinden sich direkt mit dem Server und senden bzw. empfangen Informationen.

Einsatz: bei Datenbanken, Webservern, einfachen verteilten Systemen

Beispiel 1: ***SETI@Home*** ist voll zentralisiert mit einem „***job dispatcher***“ als Server

Beispiel 2: Die original ***Napster Suche*** war zentralisiert, aber das Filesharing an sich nicht!

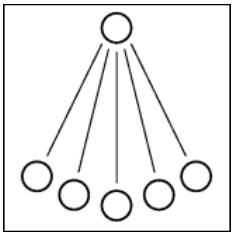

Abbildung 1: Zentralisiert

3.2 Ring

Oft ist die hohe Anzahl an Clienten zuviel für einen zentralisierten Server. Aus diesem Grund ordnet man mehrere Rechner als Ring an, um das Verhalten eines verteilten Servers zu bekommen. Der Ring funktioniert dann so, dass einige der Nodes zum Teil die gleichen Funktionen erledigen aber durch ihre Anordnung noch zusätzliche Vorteile bieten:

so zum Beispiel:
- größere Fehlertoleranz (***failover***)
- bessere Lastverteilung (***load-balancing***)

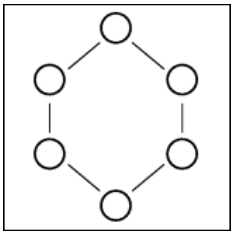
Abbildung 2: Ring

3.3 Hierarchisch

Diese Topologie hat eine lange Geschichte im Internet. So kommt es auch, dass einige sehr bekannte Dienste in dieser Weise organisiert sind.

Beispiel 1: ***DNS*** (**D**omain **N**ame **S**ervice)
Ein typischer Anfragefluss sieht etwa so aus:
Root-Server -> Server mit dem registrierten Namen->
-> evtl. noch weitere Server

Beispiel 2: ***NTP*** (**N**etwork **T**ime **P**rotocol)
Andere Computer synchronisieren sich mit den Root-Time-Servern in einem „***self-organizing Tree***“.

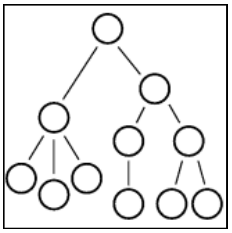
Abbildung 3: Hierarchisch

Beispiel 3: ***Usenet***
Benutzt eine baumartige Struktur, um die Nachrichten von einem zum nächsten Server zu kopieren. Eine kleine Anzahl von Rechnern agiert dabei als Backbone.

3.4 Dezentralisiert

Die Nodes kommunizieren symmetrisch und haben die gleichen Rollen im Netzwerk.

Beispiel 1: Das ***Gnutella-Netzwerk*** ist voll dezentralisiert, es gibt dort nur eine kleine zentralisierte Funktion, um einen neuen Host ins Netzwerk aufzunehmen.

Beispiel 2: ***Freenet***, ein weiteres Filesharing System

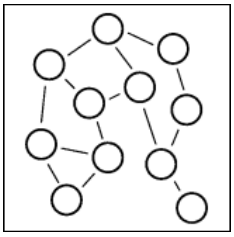
Abbildung 4: Dezentralisiert

=> Wie man bereits an den Beispielen erkennen kann, bildet diese Netzwerktopologie die Grundlage für einige Filesharing Systeme, die im 4. Teil dieser Arbeit noch genauer betrachtet werden.

3.5 Hybride Topologien

ABER die aktuell existierenden Systeme sind meist nicht nur in einer Topologie realisiert, sondern bilden eine Kombination aus den vier vorher genannten Topologien, d.h.:

=> Die Nodes spielen verschiedene Rollen in solchen Systemen und so kann es sein, dass Nodes in einem Teil des Netzwerks eine zentrale Rolle spielen und gleichzeitig Teil einer Hierarchie in einem anderen Teil des Netzwerks sind.

=> man nennt diese kombinierten Topologien ***Hybride Topologien***

Häufige Kombinationen sind:

- ***Zentralisiert*** & ***Ring***
- ***Zentralisiert*** & ***Zentralisiert***
- ***Zentralisiert*** & ***Dezentralisiert***

3.5.1 Zentralisiert & Ring

Aus Sicht der Clienten handelt es sich um ein zentralisiertes System, das aber die Robustheit eines Rings aufweist.

Beispiel: Leistungsstarke Webserver haben meist eine Ringstruktur und können somit die Vorteile der besseren Lastverteilung und größerer Fehlertoleranz nutzen.

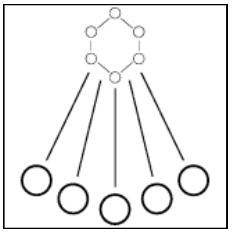

Abbildung 5:
Zentralisiert & Ring

3.5.2 Zentralisiert & Zentralisiert

Ein Server in einem Zentralisierten System ist oftmals ein Client von einem oder mehreren anderen Servern. Diese Struktur wird genutzt um verschiedene Funktionalitäten zu kombinieren.

Beispiel: Webserver, der Informationen aus Datenbanken beschafft und dann zur Verfügung stellt.

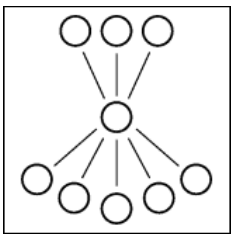

Abbildung 6:
Zentralisiert & Zentralisiert

3.5.3 Zentralisiert & Dezentralisiert

Zentralisierte Systeme eingebettet in dezentralisierten Systemen => ***Supernode Konzept***

Beispiel 1: ***FastTrack*** (benutzt von ***KaZaA***) mit vielen Tausenden von Nodes. Die einzelnen Nodes haben dabei eine zentralisierte Beziehung zu einem Supernode, zu dem sie alle Dateianfragen weiterleiten (ähnlich der Napstersuche). Die Supernodes verbinden sich untereinander zu einem dezentralisierten Netzwerk (wie Gnutella) und verbreiten so die Suchanfragen.

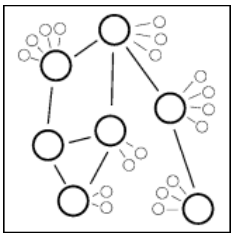

Abbildung 7: Zentralisiert & Dezentralisiert

Beispiel 2: ***Mailserver***
Mailclientes haben eine zentralisierte Beziehung zu einem speziellen Mailserver und diese verteilen und teilen sich die Mails in einer dezentralisierten Weise.

[URL: http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html]

4. Einordnung und Vorstellung einzelner Filesharing Systeme

Man kann die existierenden Filesharing Systeme im Wesentlichen in 2 verschiedene Gruppen unterteilen:

Zentrale Systeme	**Dezentrale Systeme**
Napster / OpenNap	Gnutella (z.B. Morpheus, BearShare, LimeWire)
AudioGalaxy	FastTrack (z.B. KaZaA, Grokster)
iMesh	eDonkey2000
Aimster (Madster)	Freenet
Filetopia	Direct Connect
Songspy	MojoNation

4.1 Zentrale Systeme - Allgemeine Betrachtung

Bei diesen Systemen erfassen ***Indexserver*** die Daten der einzelnen Teilnehmer, die dann als Grundlage für die Bearbeitung der Suchanfragen dienen. Nachdem eine Suchanfrage positiv von einem Indexserver beantwortet wurde, findet der Datenaustausch direkt zwischen den Tauschpartnern statt - ohne extra Server und Vermittler. Peer-to-Peer ist also nur der tatsächliche Download und für Such- und Verwaltungsfunktionen wird auf einen zentralen Server zurückgegriffen. Nachteilig an solchen Systemen ist, dass sie sich leicht durch gezielte Angriffe (z.B. ***DDOS***) auf den Indexserver lahm legen lassen. Jedoch ist es genauso leicht möglich, durch einfaches Abschalten des Indexservers das Netzwerk still zu legen, falls es Bedenken von juristischer Seite gibt.

4.2 Zentrale Systeme - Vertreter

4.2.1 Napster

Suche/Indexierung:
Die Napster-Server bieten Indexierungsdienste, bei denen der Benutzer erfahren kann, welche Inhalte die einzelnen Teilnehmer anbieten und realisieren somit die Suchfunktion im Napster-Netzwerk. Nachteilig wirkt sich in diesem Zusammenhang aus, dass die einzelnen existierenden Indexserver nicht synchronisiert sind und somit die Suche um einiges erschwert wird.

Datenhaltung:
Jeder Nutzer stellt freigegebene Dateien auf seinem Rechner anderen Nutzern zur Verfügung und erhält Zugriff auf die Dateien anderer Nutzer.

Kontaktaufnahme:
Der Client verbindet sich mit einem fixen ***Napster-Redirect-Server***, der ihn auf einen Indexserver weiterleitet. Dort wird dann eine Liste mit den eignen freigegebenen Daten abgelegt.

Nachteile:
- Sämtliche Daten sowie Login und Passwort werden im Klartext übertragen.
- Es gibt keine Anonymität, d.h. die IP-Adressen der beiden Parteien sind in jedem Fall bekannt.
- Es ist keine Resume-Funktion vorhanden.

Weiterentwicklung:
Nachdem das Protokoll von Napster analysiert war, erschienen sehr schnell erfolgreiche Open-Source Implementierungen, z.B. ***OpenNap*** (**O**pen **S**ource **N**apster **S**erver).

Vorteile von OpenNap gegenüber dem ursprünglichen Napster:
- Der Index ist nicht mehr nur auf MP3 beschränkt.
- Es erfolgt ein automatische Synchronisation der Dateibestände.
- Neue (Index)Server die teilnehmen wollen, müssen bestimmte Kriterien erfüllen. (Erreichbarkeit, Bandbreite etc.)

Weitere Napster-Clients, die in der Folgezeit erschienen: ***Napigator***, ***WinMX***, ***FileNavigator***, ***Xnap*** und viele andere für verschiedenste Betriebssysteme.

4.2.2 AudioGalaxy

Ein Großteil der notwendigen Aktionen wird über ein Webinterface abgewickelt - so z.B. die Suche. Der Client dient hier nur als Download-Manager. Indexserver sind nicht nur temporär (wie bei Napster) - sondern Informationen über alle jemals verfügbaren Titel werden längerfristig gespeichert. Durch diese Erweiterung lassen sich bei AudioGalaxy Songs von Nutzern, die momentan nicht online sind, vormerken. Der endgültige Dateitausch findet wieder direkt von Rechner zu Rechner statt. Zum Download wird immer der geographisch nächste Server ausgewählt.

4.2.3 iMesh

Es gibt wieder einen zentraler Indexserver, wodurch Suchanfragen schnell beantwortet werden können. Verwendet wird ein nicht offen gelegtes Protokoll.

Erweiterung:
Verschiedene Teile der gleichen Datei können parallel von mehreren Quellen bezogen werden.

4.2.4 Aimster (seit Januar 2002 → **Madster)**

Auch hier gibt es wieder einen zentralen Server, der die Suchanfragen bearbeitet. Eine Erweiterung bei diesem System ist jedoch, dass die Kommunikation im Aimster-Netz und mit dem zentralen Server verschlüsselt stattfindet.

4.2.5 Filetopia

Dieses System unterscheidet sich ein wenig von den bisherigen zentralen Systemen. Ursprünglich wurde es als eine reine Katalogverwaltung für Dateien entwickelt. In der Folgezeit aber um Dateiaustauschfunktionen (Chatfunktion usw.) erweitert. Kern des Systems ist wieder ein zentraler Server - der ***Filetopia-Index-Server***.

Der Dateiaustausch funktioniert jedoch etwas anders wie bisher. Man kontaktiert zuerst einmal den Gegenüber und bittet um Downloadberechtigung. Dabei bietet man normalerweise gleichzeitig Zugriff auf die eigene Dateisammlung - ähnlich dem FTP Prinzip. Wenn die Berechtigung erteilt wurde, kann der Dateiaustausch starten.

Die Clienten authentifizieren sich per Public-Key-Verfahren und sämtliche Kommunikation mit dem Server und anderen Peers ist verschlüsselt. Bisher gibt es noch keine Suchfunktion über alle Dateien der teilnehmenden Benutzer.

Aktuelle *Weiterentwicklungen* - die in die neueste Version integriert werden sollen:
- ***dezentralisiertes Supernodenetzwerk***
- segmentierte Downloads, d.h. Download des gleichen Files von mehreren Servern gleichzeitig möglich => höherer Speed
- verbesserte Channeloptionen
- Multichats
- weitgehende Serverunabhängigkeit

4.3 Zentrale Systeme - Kurze Bewertung

Vorteile:
- Einfachheit in der Struktur
- Leicht zu verwalten, da alle wesentlichen Funktionen auf einen Ort konzentriert sind.
- Relativ einfach zu sichern, da nur ein wichtiger Host.

Nachteile:
- Keine Fehlertoleranz bei Ausfall des zentralen Servers.
- Anfällig für DOS-Angriffe auf den Indexserver .

→ Skalierbarkeit, hauptsächlich abhängig von den Indexservern

4.4 Dezentrale Systeme - Allgemeine Betrachtung

Im Idealfall ist jeder einzelne Teilnehmer in einem solchen System Client und Server gleichzeitig - man spricht dann von so genannten Servents. Theoretisch sollte es auch für jeden Teilnehmer möglich sein, gesuchte Dateien im Netzwerk zu ermitteln. Aber in der Realität gibt es damit einige Probleme, so dass man diverse Zugeständnisse machen muss. Von der rechtlichen Seite her gesehen ist in es schwieriger mit solchen Systemen umzugehen, da es keine zentralen Server gibt, die man einfach abschalten oder überwachen könnte.

Man kann existierende Systeme in drei große Gruppen unterscheiden:

I. Schnelle Implementierungen, die als Folge des Napster-Erfolgs ohne langatmige Hintergrundanalyse entwickelt wurden und somit schnell zum Einsatz gebracht werden konnten.
→ ***Gnutella***

II. Andererseits gibt es komplizierte & durchdachte Systeme, die von Anfang an sehr genau geplant worden sind, jedoch bisher noch keine sonderliche Verbreitung gefunden haben. Das liegt zum einen in der aufwendigen Implementierung und zum anderen an der schlechten Bedienbarkeit der existierenden Systeme.
→ ***Freenet***

III. Die letzte Gruppe bilden kommerzielle Implementierungen, die auch am erfolgreichsten sind. Sie bieten eine verbesserte Lastverteilung, Maßnahmen gegen das unkontrollierte Wachstum des Netzwerks und eine bessere Einbindung von Dial-up-Verbindungen. Diese Art von Systemen hat eine verbesserte Sabotageresistenz und es sind Zugriffs- und Bandbreitenbeschränkungen möglich. Weiterhin gibt es einen ausgeklügelten Downloadmechanismus und eine schnelle Suche.
→ ***FastTrack*** und ***eDonkey2000***

4.5 Dezentraler Systeme - Vertreter

4.5.1 Gnutella

4.5.1.1 Gnutella - Protokoll

Gnutella ist ein Peer-to-Peer Protokoll und ist somit die Basis des Gnutella Netzwerkes. Es ist der Vorläufer vieler heute eingesetzter Protokolle.

Clienten, die das Protokoll nutzen, sind: z.B. ***BearShare***, ***LimeWire***, ***Phex***.

Suche/Indexierung:
Es gibt keinen zentralen Server zur Indexierung aller verfügbaren Dateien, d.h. Suchanfragen werden solange durch das Netz geleitet, bis ein Servent den gewünschten Inhalt liefern kann.

Die Suche funktioniert nun derart, dass der Urheber einer Suche bei allen mit ihm direkt verbundenen Hosts nachfragt, ob Dateien vorhanden sind, die die gewählten Stichworte enthalten. Die Empfänger der Anfrage geben dann eine entsprechende Rückmeldung, falls die Suche erfolgreich war und leiten unabhängig davon ihrerseits die Suche zu allen weiteren Knoten, mit denen sie jeweils direkt verbunden sind, weiter. Die Knotenrechner wissen selber nicht, ob der Host, der die Anfrage an sie geleitet hat, auch ihr Urheber ist oder selbst nur als Relay fungiert. Damit gleiche Suchanfragen nicht mehrfach ausgewertet werden müssen, werden sie mit den zuletzt bearbeiteten Anfragen verglichen. Zusätzlich verhindert ein TTL Mechanismus das Kreisen von Suchanfragen im Netzwerk. Es wird nie das gesamte Netzwerk durchsucht, d.h. abhängig vom Einstiegspunkt und Suchhorizont wird man immer unterschiedliche Ergebnisse erhalten.

Gnutella benutzt das ***HTTP***-Protokoll, so dass man eine URL als Ergebnis einer Suchanfrage bekommt. Um die IP-Adresse des Zielrechners zu übermitteln, nimmt die Antwort den gleichen Weg zurück durch das Netzwerk über den die Anfrage gekommen ist. Der Datentransfer kann dann direkt zwischen den beiden Teilnehmern stattfinden. Die Kommunikation findet mittels TCP/IP statt.

Spezialfall „***Passive Suche***“: dabei werden alle über den eigenen Knotenpunkt eingehenden Suchanfragen durchsucht. Vorteil dieser Art von Suche ist, dass sie das Netzwerk nicht belastet.

Kontaktaufnahme:
Beim ersten Einstieg holt man sich vom so genannten Host-Cache (dessen Adresse bekannt sein muss) eine Liste mit zufällig ausgewählten IP-Adressen, die als erste Andockpunkte an das Gnutella Netzwerk dienen.

Probleme/Nachteile/Mängel:
- Die dezentralisierte verteilte Suche ist das größte Problem im Gnutella-Netzwerk.
- Die Skalierbarkeit ist bei Gnutella eng an die verfügbare Bandbreite gekoppelt.
  Dieser Zusammenhang ergibt sich aus der Tatsache, dass zu den eigentlich zu übertragenden Dateien noch viel zusätzlicher Netzwerkverkehr kommt. Dieser setzt sich zusammen aus den Suchanfragen und den so genannten Ping & Pong Paketen, die das Netzwerk zusätzlich in einem nicht unerheblichen Maß belasten.
- Theoretisch sollte das Gnutella-Netz beliebig viele oder zumindest sehr viele Benutzer verkraften können; da durch das TTL Prinzip der Sichtbarkeitshorizont eines einzelnen Users beschränkt wird und somit auch die Anzahl der Nachrichten, die er verarbeiten muss.

  Aber:
  - Das Netz wurde mit steigender Zahl der Benutzer zunehmend unbenutzbarer.
    Das lag zum einen an dem erzeugten Netzverkehr und zum anderen schien die Zahl der Treffer einer Suchanfrage reine Glückssache zu sein.
  - Ein weiteres großes Problem im Gnutella-Netz sind Dial-up-Verbindungen. So kam es häufiger vor, dass beim Wegfall von solchen Verbindungen das Netzwerk in einzelne Teilnetze fragmentiert wurde. Das konnte z.B. dann passieren, wenn über diese Station eine wichtige Gnutella-Verbindung lief.

- Der eventuelle Ausweg, dass jeder Servent mit möglichst vielen andern Teilnehmern kommunizieren sollte, ist nicht sinnvoll. Die Netzlast würde schnell jede Bandbreite überschreiten.

- weiterhin fehlen Sicherheitsmodelle und der Datenfluss ist nicht verschlüsselt

Nachdem man einige Studien im Gnutella-Netzwerk durchführte, kam man zu der Einsicht, das Gnutella nicht skalieren kann. Der Hauptgrund ist der immer größer werdende und somit bandbreiten sprengende Netzwerkverkehr.

=> Abhilfe bringen könnten eventuell Caching Techniken und ein Redesign von Grund auf. Diese Einsicht führte zu einigen wesentlichen Weiterentwicklungen.

Weiterentwicklungen:
Nach dem zwischenzeitlichen Zusammenbruch von Gnutella ist es zur Zeit wieder erstaunlich stabil und verkraftet mehr Nutzer den je. Der Grund dafür ist ein neuer Gnutella-Client (LimeWire2.0), der so genannte ***Ultrapeer Features*** einführt. Ultrapeers sind Rechner mit guter Netzanbindung, die temporär als Server für 50 Nutzer fungieren. Damit soll der notwendige Netzwerkverkehr deutlich verringert werden. Das Netz wird in der Folge auch für schmalbandig angebundene Nutzer brauchbar und bremst sich nicht mehr selbst aus. Theoretisch sollte das Gnutella Netz jetzt wesentlich besser skalieren können.

Eines der nächsten Ziele ist die „Intelligente Suche“.

4.5.1.2 BearShare

Setzt auf dem Gnutella-Protokoll auf und ist ein Client für Windows. Integriert ist ein eigener Webserver, der alle freigegebenen Dateien als HTML-Tabelle ins Web stellen kann, so dass auch Nicht-Gnutella-Nutzer per Webbrowser Zugriff auf die Daten erhalten können. Der Gnutella-Horizont kann manipuliert werden, d.h. die Werte der Lebensdauer und die Reichweite der Suchanfragen können verändert werden.

4.5.1.3 LimeWire

Ist ein weiterer Gnutella Servent, der das Gnutella-Protokoll benutzt. Dieser Client ist in Java implementiert. Eine Erweiterung dieses Clients ist die Unterstützung des gleichzeitigen Downloads von mehreren Hosts.

4.5.1.4 Allgemeine Verbesserungen der Clienten

Sie trennen Verbindungen zu „toten“ Systemen.

=> Dies verbessert die Skalierbarkeit des Netzes deutlich, da so eine Fragmentierung verhindert wird.

Verlässt also ein Client (z.B. Modemnutzer) das Netzwerk, suchen die anderen Clients automatisch aus ihrem Host-Cache einen geeigneten Ersatz und stellen die Verbindung wieder her.

4.5.2 FastTrack - Protokoll

FastTrack ist eine Servertechnologie.

Clienten, die FastTrack nutzen, sind: z.B. ***Morpheus***, ***KaZaA***, ***Grokster.***

Aktuelles:
Morpheus arbeitet in der neusten Version auf Grundlage des Gnutella-Netzwerks. Der Grund dafür waren Angriffe auf den Morpheus Client, so dass dieser nicht mehr funktionierte. Es wurden dabei Werte in der Registry von Windows verändert - dies geschah aber aus dem FastTrack-Protokoll heraus, so dass man keinen Einfluss darauf hatte => Umstieg

KaZaA steht kurz vor dem Ende (Klagen in den Niederlanden) - aber nicht unbedingt wegen den teilweise fraglichen Inhalten, sondern man kann sich einfach das Verfahren nicht leisten.

Struktur:
Das Netzwerk ist eine Mischung aus serverbasierendem und serverlosem Aufbau. Dies erreicht man, indem Aspekte des Gnutella-Netzes - mit einer flexiblen, semizentralen Struktur von SuperPeers kombiniert werden.

Suche:
Im Netzwerk selbst soll gar nicht jeder Client als Server für Suchanfragen dienen. Aus diesem Grund werden automatisch und dynamisch ja nach Last bestimmte Rechner zu SuperPeers ernannt und übernehmen dann das Verwalten der Suchanfragen und der Dateilisten. Die normalen Peers im Netzwerk müssen immer an einem solchen SuperPeer angemeldet sein, um am Netzwerk teilhaben zu können. Die Auswahl, welcher Rechner zum SuperPeer ernannt wird, hängt z.B. von der Performance und Anbindung ab. SuperPeers leiten Anfragen auch an andere SuperPeers weiter, d.h. die Suchanfragen breiten sich über die SuperPeers aus.
Somit bilden die untereinander kommunizierenden SuperPeers das eigentliche FastTrack-Netzwerk - anders gesagt, man hat prinzipiell eine Vielzahl von Subnetzen, die über SuperPeers miteinander kommunizieren

Kontaktaufnahme:
Nach dem Login erhält der Client vom Server IPs von einem oder mehreren SuperPeers, die nach erfolgreichem Connect ständig aktualisiert werden. Im Programm selbst ist eine Liste mit SuperPeers fest integriert. Der Client lädt seine Liste der angebotenen Dateien auf einen der SuperPeers.

Die interne Kommunikation zwischen den einzelnen Teilnehmern findet verschlüsselt statt und ist bisher nur teilweise analysiert. Für Downloads wird das ***HTTP***-Protokoll verwendet.
Es wird immer der schnellste Server für den Download gesucht.

Vorteile:
Der große Vorteil dieses Netzes gegenüber anderen beruht auf dem gleichzeitigen Download einer Datei von mehreren Usern, wobei stets unterschiedliche Teile heruntergeladen werden, die dann später wieder zusammengesetzt werden.
Resume ist auch von anderen Rechnern möglich, d.h. wenn jemand offline geht, wird automatisch ein neuer Server für den weitern Download gesucht.
Durch diese Mechanismen ergibt sich insgesamt eine höhere Ausnutzung der Bandbreite, was die Geschwindigkeit der Downloads mitunter enorm steigern kann.

Ein weiterer Vorteil ist die schnelle Suchgeschwindigkeit, die hier nur wenige Sekunden beträgt.

Die verschiedenen Clienten, die mit FastTrack als Grundlage arbeiten, unterscheiden sich in ihrer Funktionalität nur in wenigen Details. (hauptsächlich in ihrer Spyware/Adware ;-))

4.5.3 eDonkey2000

eDonkey2000 ist ein kommerzielles Projekt.

Das Netzwerk basiert auf dem **M**ultisource **F**ile **T**ransfer **P**rotocol (***MFTP***) und stellt eine dezentrale Client-Server Architektur dar. Die zur Nutzung notwendigen Programme sind getrennt als Client und Server realisiert und können unabhängig voneinander betrieben werden.

Der Server übernimmt das Anmelden am Netzwerk, die Verwaltung der Suchanfragen und dient zum Auffinden von anderen Usern. Die Clients erledigen die eigentliche Arbeit beim Downloaden der Dateien

Die erste Anmeldung erfolgt über eine im Client fest integrierte Serverliste. Der Server, mit dem sich der Client verbindet, ist sein Hauptserver. Der Client sendet seine Dateiliste immer an seinen Hauptserver. Der Hauptserver kennt somit alle zum Tauschen freigegeben Daten der bei ihm angemeldeten Clienten. Vom Server erhält der Client eine Liste aller ihm bekannten Server - die so genannte Serverlist. Die Suchanfrage wird normalerweise an den aktuellen Hauptserver gerichtet oder aber mittels „***Extend Search***“ (Serverlist) an weitere Hauptserver, um die Anfrage auf andere Bereiche des eDonkey-Netzes zu erweitern.

Die Server selbst kommunizieren nur wenig, tauschen aber regelmäßig ihre Serverlisten aus. Über diesen Mechanismus werden neue Server im Netzwerk bekannt gemacht.

Vorteile:
Es kommt die so genannte ***FastStream*** Technologie zum Einsatz.

Diese funktioniert derart, dass zuerst alle freigegebenen Dateien mittels eines Hash-Werts indiziert und anschließend mit einer eindeutigen Identifikationsnummer versehen werden. Dadurch wird es möglich, gleiche Dateien zu erkennen und so das Herunterladen einer Datei oder auch nur kleinster Teile davon von mehreren verschiedenen Nutzern zu koordinieren. Die Fragmentierung der Dateien ist ein wesentlicher Bestandteil des eDonkey-Netzwerks. Das hat zur Folge, dass der Download insgesamt schneller und unabhängiger wird. Gleichzeitig wird die Lastverteilung und Bandbreiten-Ausnutzung verbessert.

4.5.4 Freenet

Clienten, die Freenet nutzen, sind: z.B. ***Espra***, ***FreeWeb***.

Freenet funktioniert ähnlich dem World Wide Web, implementiert aber einen eigenen Namensraum in dem „***Keys***“ die URLs ersetzen. Die Servents heißen hier Nodes.

Bei der Entwicklung dieser Anwendung spielten folgende Punke eine wesentliche Rolle:
- Anonymität
- Dezentralisiertheit
- Redundanz
- Verschlüsselung
- Dynamisches Routing

Ein zentraler Bestandteil des Konzeptes ist es, das Material anonym angeboten und empfangen werden kann. Die Daten werden im Netzwerk verschlüsselt und in kleinen Teilen gespeichert. Jeder Freenet-Peer hat einen lokalen Speicherplatz (Cache), in dem jeder andere Peer Daten ablegen darf. Der Betreiber eines Peers weiß selbst nicht, was er speichert. Da sich so auch nicht feststellen lässt, wer eine Datei speichert, ist es nicht möglich, Inhalte gezielt aus dem Freenet zu entfernen.

Es gibt noch einen zweiten Cache in dem Metadaten abgelegt werden. Metadaten enthalten Informationen über lokalen Speicher und den Cache einiger angrenzender Peers.

Wenn eine Datei geladen werden soll, wird der am nächsten liegende Node lokalisiert, der die Datei vorrätig hat. Dann wird sie entlang dazwischen liegender Nodes transportiert, wobei jeder Node die Datei in seinem einstellbaren Cache zwischenspeichert.

=> Inhalte verteilen sich automatisch dahin, wo die größte Nachfrage besteht

Eine Suchfunktion gibt es nicht!

Dateien werden über Keys angesprochen, die mittels einer Hash-Funktion aus dem Dateiinhalt berechnet werden und keine Rückschlüsse auf den Inhalt der Datei zulassen.
=> man muss den Key kennen um eine bestimmte Datei zu finden.
Dafür gibt es thematisch sortierte Keylisten z.B. auf der Freenet-Hompage.

Damit das Netzwerk bei steigender Nutzeranzahl vernünftig skalieren kann setzt Freenet auf einen selbst optimierenden Routingalgorithmus.

Dieser funktioniert derart, dass neue Anfragen nach einem Key von der Software an den Host gerichtet werden, der bereits ähnliche Keys liefern konnte. Da jeder Host die transportierten Daten zwischenspeichert, führt das zu einer Spezialisierung einzelner Hosts für Gruppen von Keys. Diese Spezialisierung ist den benachbarten Rechnern bekannt und somit wird das Routing immer effizienter.
=> es muss sich erst noch zeigen, ob dies bei einer wirklich großen Nutzerzahl auch funktioniert

(Dabei sollte man auch beachten, dass ähnliche Keys keinesfalls ähnliche Inhalte bedeuten.)

Probleme:
- Bandbreitenunterschiede (z.B. bei Dial-up-Verbindungen) werden nicht berücksichtigt.
- Es gibt keine richtige Suchfunktion.
- Informationen können aussterben, d.h. sie verschwinden nach und nach aus dem Netzwerk.
  Grund: Der Cache der einzelnen Nodes ist limitiert. Das führt natürlich dazu, dass alte Daten gelöscht werden, wenn neue Daten hinzukommen. (LRU-Strategie). Und wenn dann in der Folgezeit bestimmte Daten nicht mehr abgerufen werden, sind sie auf immer weniger Rechnern vorhanden und verschwinden irgendwann völlig aus dem Netzwerk.

4.5.5 Eternity Service

Ein System, das unter dem Gesichtspunkt entwickelt wurde, welches es möglich machen sollte, Daten über eine lange Zeitspanne zu speichern, unabhängig davon, ob diese nützlich sind oder nicht.

Um das zu gewährleisten, sollten die Administratoren eines Servers kein Wissen und Einfluss über die gespeicherten Inhalte haben. Gleichzeitig sollten so auch juristische Schwierigkeiten umgangen werden.

Die Daten werden redundant auf verschiedenen Servern im ganzen Netz gespeichert. Es kommen Public-Key Verschlüsselungstechniken zum Einsatz, so dass es extrem schwierig wird, Daten aus dem System zu entfernen.

Dieses Konzept ist nie verwirklicht worden, aber einige Grundideen wurden von Freenet wieder aufgegriffen.

4.5.6 Weitere Dezentrale Systeme

- Meist im Umfeld von Universitäten entstanden, und sind somit durchdachter aber nicht sehr weit verbreitet z.B. ***Jungle Monkey***.

- ***MojoNation*** - von Hackern ins Leben gerufen - noch in der Entwicklung und deshalb noch nicht sehr bekannt.

- Andere (erfolgreichere) konventionelle Systeme sind ***Direct Connect*** und ***Hotline Connect***.

4.5.6.1 Hotline Connect

Nimmt eine Sonderstellung ein, da es keine P2P Lösung ist.

Zum Download oder Upload einzelner Dateien reicht der Hotline Connect - Client. Um aber etwas anbieten zu können muss ein separater „Chat & News & Fileserver“ installiert werden.
Die einzelnen Clients können untereinander nicht tauschen.

Wenn man etwas von einem Server downloaden will, muss man sich zuerst einmal einen Account holen. Die Entscheidung über die Zulassung liegt beim Administrator eines Servers.
(Häufig wird mit Ratios gearbeitet.)

4.5.6.2 Direct Connect

Besitzt eine dezentrale Client-Server Architektur, die zwischen Hotline Connect und einer „normalen“ P2P-Börsen einzuordnen ist.

Die Software ist auch hier wieder in 2 Teile gegliedert:
- Servervariante (Hub)
- Clientsoftware mit Zugang zu den Hubs des Direct Connect-Netzes

Um an dem Netzwerk teilhaben zu können, wählt man sich in einem der öffentlichen Hubs ein und erfährt dort, wie viele User sich an diesem Knoten aufhalten. Man erhält eine Kurzbeschreibung zu jedem Hub, sowie eine Auskunft über die Art der dort getauschten Inhalte.

Hat man den richtigen Server gefunden, kann man seine Suchanfrage an die hier angemeldeten Nutzer stellen. Auch eine „Multi-Hub Suche“ ist möglich bei Hubs, die sich zu einem Tausch-Cluster zusammengeschlossen haben.

4.5.6.3 MojoNation

Es handelt sich hier um einen so genannten ***„P2P-Micro-Payment“***-Dienst mit einem integrierten ***„Reputation System“***, dessen Hauptfunktion darin besteht, Daten zu tauchen.
Die Grundstruktur bildet ein Peer-to-Peer Netzwerk in dem es keine Server gibt, so dass selbst die zum Betrieb notwendigen Datenbanken im Netz verteilt gespeichert werden. Der Benutzer des Netwerks bleibt anonym.

Die Daten, die ins Netzwerk gelangen sollen, werden vom Clienten verschlüsselt und in kleine Teile zerlegt und dann über das Netzwerk verteilt. Die Besitzer der einzelnen Rechner wissen also nichts über die Daten, die auf ihrem Rechner gespeichert sind.

Vorteil:
Durch die Aufteilung in viele kleine Teile (1000 und mehr) können selbst die langsamsten Teilnehmer mithelfen, die Datei zu übermitteln (wie Ameisen).

*Das **Micro-Payment-System**:*

Es gibt so genannte Mojos - das ist die „Währung“ in diesem System. Man kann diese verdienen, indem man den eigenen Speicherplatz „vermietet“, Rechenleistung zu Verfügung stellt und natürlich Dateien freigibt. Auch die Internetanbindung ist für das System interessant. Es gelten die natürlichen Marktgesetzte (die Nachfrage bestimmt den Preis...). Die Mojos, die ein Surfer durch diese angebotenen Dienstleistungen verdient, kann er dann für das Herunterladen von Dateien wieder verwenden. Damit wird verhindert, dass einige der Nutzer nur Daten beziehen und keine liefern. Der „Handel“ findet direkt unter den Benutzern statt, d.h. wenn jemand eine Datei herunterlädt, so bekommen diejenigen die bezahlten Mojos, die die Datei auf ihrem physischen Speicher haben, diejenigen die Datei weiterleiten und der jenige das meiste Mojo, der die Datei ins Netz gestellt hat.

Sinn diese Systems:
Kommerzielle, zentral verwaltete Angebote werden umso langsamer, je mehr Leute dort gleichzeitig Daten abrufen. Bei MojoNation soll das nicht passieren, d.h. je mehr der Mojo-"Markt" an Teilnehmern und vernetzten Computern wächst, desto mehr kommt die Effizienz des Marktes zum Tragen => das Netz wird schneller, nicht langsamer.

*Das **Reputation System**:*
Andere Nutzer können bewertet und die Summe der verdienten Mojos kann eingesehen werden. Die von unabhängigen Dritten gesammelten Bewertungen sollen über gesonderte "Reputation-Server" zugänglich gemacht werden.

=> Dieses Reputationssystem hängt direkt mit dem Zahlungssystem zusammen:
  => die Bewertung eines Nutzers spiegelt sich in seinem Kreditrahmen wieder, den er bei anderen Nutzern hat

4.6 Dezentrale Systeme - Kurze Bewertung

Vorteil:
- können relativ einfach erweitert werden
- sehr fehlertolerant beim Ausfall einzelner Rechner

Nachteil:
- schwierig zu handhaben & zu verwalten
- unsichere Netze, da sehr einfach neue Nodes in das System integrierbar sind, die dann einen beträchtlichen Schaden anrichten können

→ Skalierbarkeit, deutlich besser als bei zentralen Systemen

4.7 Kommerzielle Entwicklungen im P2P Bereich

Abseits von Tauschbörsen hat der Peer-to-Peer Boom auch Unternehmen erfasst, die ihrerseits mit einigen P2P Anwendungen auf den Markt drängen.

4.7.1 Suns JXTA

Läuft unter dem ***Juxtapose-Projekt*** ("nebeneinander stellen") von SUN.

Ziel:
Schaffung eines offenen Peer-to-Peer-Standards, auf dem beliebige Dienste aufsetzen können.

Es ist ein sehr allgemeiner und weitgefasster Ansatz, der alle notwendigen Protokolle definiert.
Es wird z.B. zwischen 3 wesentlichen Ebenen (Layer) unterschieden:

- ***Core:*** Aufbau von Kommunikationskanälen, um kommunikationswillige Partner aufzuspüren oder um den Datenaustausch abzusichern.

- ***Service:*** Definiert Grundlegende P2P-Dienste wie Suchen, Dateien austauschen, Computerressourcen gemeinsam nutzen, Protokolle übersetzen, Benutzer authentifizieren und Dateien verschlüsseln.

- ***Application:*** = Benutzerschnittstelle => Tauschbörsen, Instant-Messaging-Systeme und andere, nutzen die Funktionen der darunter liegenden Schichten

Der Datenaustausch erfolgt in ***XML***.
Grundlage von „***Sune ONE***“ einem P2P-orientierten Gegenentwurf zu Microsofts „***dotNET***“.

4.7.2 Groove

Microsoft ist mit einigen Millionen an diesem Projekt beteiligt, mit dem Hintergrund, dass Groove Teil der ***dotNET***-Strategie werden soll. ***GroveNetworks*** bietet Peer-to-Peer-Groupware mit Funktionen ähnlich dem Lotus-Produkt ***Notes***, ***Quickplace***, ***Sametime*** und wird somit also für Unternehmenszwecke entwickelt. Die Groove-Anwendung enthält unter anderem Module für Diskussionsforen, Brainstorming, Dateiverwaltung, Dokumentbearbeitung, Chat, Kalender und To-do-Listen.
Es gibt keine zentrale Datenhaltung, dafür ist es aber notwendig, das zentrale Directory-Server (Verwalten der Benutzer usw.) und Relay-Server (zum e-Mail versenden) vorhanden sind.
Alle Daten werden auf den Client-Rechnern selbst abgelegt und jeweils zwischen den Clients, die sich zu einer Gruppe zusammengeschlossen haben, synchronisiert. Offline-Arbeit ist ebenfalls möglich, da Änderungen weitergegeben werden, sobald wieder eine Netzverbindung besteht.
Bisher ist das System aber noch recht schwerfällig und benutzerunfreundlich.

5. Abschluss - Fazit

Durch die bisherigen Entwicklungen ist deutlich geworden, welche Punkte bei der Weiterentwicklung von File-Sharing Systemen zu beachten sind:

- Skalierbarkeit (auch bei einer sehr großen Teilnehmerzahl)
- zeitweilige Verbindungen müssen besser berücksichtigt werden
- Suchmechanismen weiterentwickeln
- Sabotageschutz verbessern
- Anonymität
- Stabilität und gute Benutzbarkeit

Dezentrale Systeme sind bisher am erfolgreichsten, da sie schon viele wesentliche Funktionen unterstützen.

Das ***Supernode/Ultrapeer*** Konzept ist ein wichtiger Schritt für die Weiterentwicklung der dezentralen Netze.

Aber bisher sieht es mit der Zuverlässigkeit von Downloads in den Peer2Peer Netzen eher schlecht aus:

→ man hat nicht annähernd den Komfort von HTTP/FTP/Mail und ähnlichen konventionellen Systemen

Es wird in Zukunft viele weitere (kommerzielle) Anwendungen auch außerhalb des Filesharing-Bereichs geben.

6. Literaturverzeichnis

1. Kefk Network
   URL: http://www.kefk.net/P2P/Grundlagen/Begriffe/index.asp
   URL: http://www.kefk.net/P2P/Grundlagen/Definition/index.asp

2. OpenP2P
   URL: http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html

3. Möller, Erik(2001): Kopieren ohne Grenzen
   URL: http://www.heise.de/ct/01/06/links/150.shtml

4. Hansen, Sven / Zota, Volker (2001): Tauschrausch
   URL: http://www.heise.de/ct/01/26/links/158.shtml

5. Zoier, Markus (2002): Peer-to-Peer Tauschbörsen.
   URL: http://www.iicm.edu/research/seminars/ws_01/zoier-peer2peer.pdf

6. Ritter, Jordan (2001): Why Gnutella Can't Scale. No.Really.
   URL: http://www.darkridge.com/~jpr5/doc/gnutella.html

7. Cave, Damien (2000): The Mojo Solution
   URL: http://www.salon.com/tech/view/2000/10/09/mojo_nation/print.html

8. Oram, Andy (2000): Gnutella and Freenet
   URL: http://www.oreillynet.com/lpt/a/208

9. Röttgers, Janko (2002): Das Gnutella-Revival
   URL: http://www.heise.de/tp/deutsch/inhalt/12025/1.html

10. URL: http://www.edonkey2000.com

11. URL: http://www.mojonation.net/docs/technical_overview.shtml

12. Clarke, Ian (1999): A Distributed Decentralised Information Storage and Retrieval System