

**Hochschule für Technik und Wirtschaft HTW Chur**

Telekommunikation/Elektrotechnik

# **Verteilte Systeme**

Peer-to-Peer Netzwerke

**Roman Aebischer**

Dieser Bericht wurde im Rahmen des Bachelor-Studienganges  
Telekommunikation/Elektrotechnik an der Hochschule für Technik und Wirtschaft  
HTW Chur erstellt.

**Studierender**

Roman Aebischer  
Rabengasse 2  
7000 Chur

**Dozent**

Prof. Martin Studer  
HTW Chur  
Ringstrasse/Pulvermühlestrasse 57  
7004 Chur

## Inhaltsverzeichnis

1	Was sind Peer-to-Peer Netzwerke .....	4
1.1	Vor- und Nachteile der Peer-to-Peer Architektur .....	4
2	Anmeldung an einem Netzwerk .....	5
2.1	Zentrale Server .....	5
2.2	Super-Peers .....	5
2.3	Tracker .....	5
3	Aufteilung und Suche nach Dateien .....	6
3.1	Nachbarschaftslisten .....	6
3.2	Hash-Tabellen .....	6
3.3	Distributed Hash-Table (DHT) .....	7
3.4	Chord .....	8
4	Verteilter Download – Split-Stream .....	9
5	Literaturverzeichnis .....	10

# 1 Was sind Peer-to-Peer Netzwerke

Die meisten Netzwerke sind nach dem Client-Server Prinzip aufgebaut. Dabei stellt ein Server einen oder mehrere Dienste zur Verfügung und die Clients können darauf zugreifen. Ein Beispiel dazu ist das Internet. Websites werden von Servern angeboten und die Clients laden diese herunter und stellen sie dar.

Peer-to-Peer (P2P) Netzwerke stellen das Gegenteil zu Client-Server Netzwerken dar. Die einzelnen Computer (Peers) sind einander gleich gestellt. Dabei kann jeder Dienste anbieten und beziehen. Man kann sagen, dass die Client-Server-Architektur hierarchisch und die P2P-Architektur flach ist. [1]

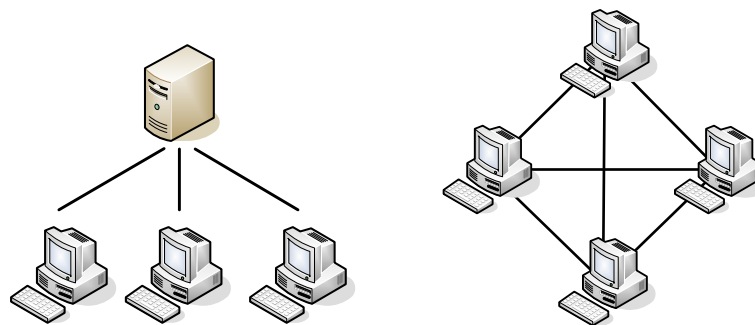


Abbildung 1: Client-Server und Peer-to-Peer Netzwerk

## 1.1 Vor- und Nachteile der Peer-to-Peer Architektur

P2P-Systeme haben gegenüber anderen Netzwerken einige Vorteile. Zum einen lassen sich diese Netzwerke in der Regel sehr gut skalieren. Da jeder Peer sowohl Dienstbenutzer als auch Anbieter ist, halten Angebot und Nachfrage die Waage.

Da die Peers innerhalb eines Netzwerkes nicht zentral gesteuert oder überwacht werden, muss das System den Ausfall eines Peers kompensieren können. Dazu werden die Daten redundant auf mehreren Peers gespeichert, was das P2P-Netzwerk tolerant gegenüber Ausfällen macht. Ein weiterer Vorteil dieser Architektur.

Der Verzicht auf einen Server hat jedoch auch Nachteile. Gerade die Unzuverlässigkeit der einzelnen Peers (Ausfälle) sowie die fehlende oder sehr schwierig zu erreichende Kontrolle über die Inhalte können zu einem Problem werden. Weiter ist die Suche nach Dateien nicht trivial, da kein zentrales Verzeichnis aller Dateien besteht. Welche Ansätze es dazu gibt ist weiter unten ausgeführt.

## 2 Anmeldung an einem Netzwerk

In jedem Netzwerk müssen sich die Teilnehmer erst anmelden, bevor sie darüber kommunizieren können. Bei Client-Server Netzwerken melden sich die Client bei den Servern. Doch wie melden sich Peers in einem P2P-Netzwerk an?

Dazu gibt es zurzeit 3 verschiedene Ansätze:

### 2.1 Zentrale Server

Dieser Ansatz wurde von Napster [2] verwendet. Dabei registriert der Peer seine Liste freigegebener Dateien bei einem zentralen Server und kann auf ihm auch nach Dateien auf anderen Peers suchen. Jedoch widerspricht dieser Ansatz der P2P-Architektur, da er nicht ohne zentrale Stelle auskommt.

Zudem ist diese Lösung nicht gut skalierbar, da der Server nur über begrenzte Kapazität verfügt.

### 2.2 Super-Peers

In die Software integriert befindet sich eine Liste mit hoch verfügbaren Peers. Beim Starten der Software geht diese die Liste durch und verbindet sich mit einem aktiven Peer. Die Liste ist meist auch erweiterbar mit Peers, welche in einer früheren Sitzung gefunden wurden. Bei diesem Verfahren wird komplett auf eine zentrale Struktur verzichtet, womit auch die rechtliche Verfolgung der Betreiber verunmöglicht wird.

### 2.3 Tracker

Bei diesem Verfahren wird nicht ein Netzwerk für sämtliche Dateien aufgebaut. Es wird für jede Datei ein einzelnes Netzwerk aufgebaut. Dabei werden spezielle Webserver (Tracker) eingesetzt, welche ein Verzeichnis mit Dateien sowie eine Liste mit Peers, welche diese Dateien besitzen, führen. Möchte ein Peer eine Datei herunterladen, so fragt er beim Tracker nach der Liste von Peers, welche diese Datei besitzen. Dieser Ansatz ist praktisch gleich zu setzen wie derjenige des zentralen Servers. Die Tracker werden zum Beispiel oft in Zusammenhang mit Bittorrent eingesetzt. [3]

### 3 Aufteilung und Suche nach Dateien

Sämtliche P2P-Netzwerke sehen sich mit denselben zwei Problemen konfrontiert. Zum einen müssen die Daten, welche in einem Netz angeboten werden, untereinander aufgeteilt werden. Zum Anderen müssen die Peers eine Methode haben, um nach den Daten zu suchen.

Die Variante mit einem zentralen Server (Siehe 2.1) wird nicht weiter ausgeführt, da sie keinem P2P-Ansatz entspricht.

#### 3.1 Nachbarschaftslisten

Die Daten werden bei diesem System nicht untereinander aufgeteilt. Das heisst, dass jeder Peer nur die Daten hält, welche er vor dem Eintritt in das Netzwerk hatte.

Sucht ein Peer eine Datei, so fragt er alle seine Nachbarn nach ihr. Sind die Nachbarn nicht im Besitz der Datei, so fragen diese unter ihren Nachbarn weiter. Und so weiter.

Man stellt schnell fest, dass in diesem Netzwerk bei jeder Suche sehr viele Nachrichten generiert werden. Aus diesem Grund wird eine Time to Live (TTL) ähnlich wie bei einem IP-Packet eingesetzt. Jedoch kann es dadurch sein, dass eine Datei aufgrund Ihrer Entfernung zum suchenden Peer nicht gefunden wird.

Nachbarschaftslisten wurden erstmals im Gnutella-Netzwerk verwendet [4].

#### 3.2 Hash-Tabellen

Hierbei werden sowohl die Peers, als auch die Daten innerhalb eines Hash-Bereichs verteilt. Als Beispiel wird hier Modulo 7 als Hash-Funktion gewählt und die Adressen der Peers und Dateien werden aufgrund dieser festgelegt. Somit hält im Beispiel unten der Peer mit der Adresse 1 die Datei mit der Adresse 15 ( $15 \bmod 7 = 1$ ).

Sucht beispielsweise Peer 3 nach der Datei 15, so weiss er aufgrund der errechneten Adresse, dass diese in linker Richtung zu finden ist. Er muss nur noch alle Peers in diese Richtung bis zum Peer mit der Adresse 1 durchgehen.

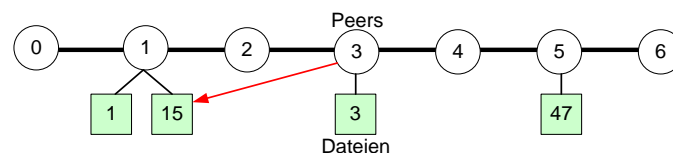


Abbildung 2: Hash-Tabellen

Das Problem dieser Methode ist, dass der gesamte Hash-Bereich beim Hinzufügen oder Entfernen eines Peers neu berechnet werden muss. Ausserdem hat das Netzwerk einen geringen Zusammenhalt, da jeder Peer höchstens 2 Nachbarn hat. Dadurch kann der Ausfall eines Peers zur Trennung des Netzwerks in 2 Teilnetze führen.

Weiter ist die Dateisuche sehr langsam, da im schlimmsten Fall jeder Peer des Netzwerks mit einbezogen werden muss. In diesem Beispiel bei einem Zugriff von Peer 0 auf eine Datei auf Peer 6.

### 3.3 Distributed Hash-Table (DHT)

Bei DHT wird ein sehr viel grösserer Hash-Bereich gewählt als bei den normalen Hash-Tabellen und jeder Peer erhält aufgrund seiner errechneten Adresse einen Teil dieses Bereichs zugesprochen.

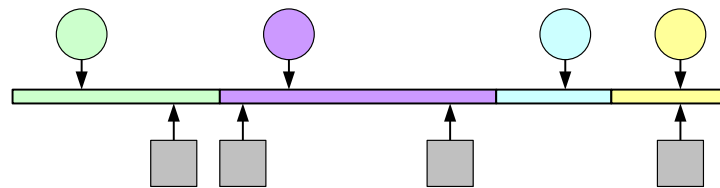


Abbildung 3: Distributed Hash-Table

Dadurch erreicht man, dass beim Hinzufügen eines Peers nur die beiden Nachbarn eine Änderung machen müssen. Diese geben dem neuen Peer einen Teil ihres Bereichs ab.

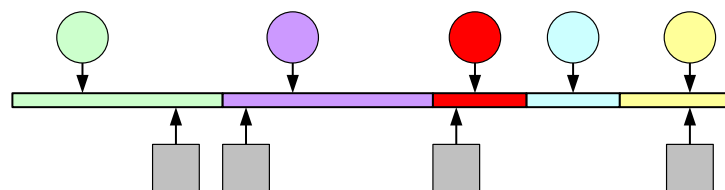


Abbildung 4: Distributed Hash-Table 2

Ähnlich verhält es sich beim entfernen eines Peers. Dabei übernehmen die beiden Nachbarn dessen Hash-Bereich.

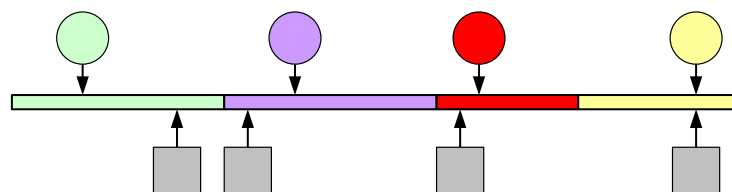


Abbildung 5: Distributed Hash-Table 3

Durch diese Änderungen gegenüber herkömmlicher Hash-Tabellen wurde erreicht, dass ein Hinzufügen oder Entfernen eines Peers keine Auswirkungen auf das Netzwerk hat (mit Ausnahme der Nachbarn). Jedoch bleiben die anderen Schwächen, wie die geringe Verkettung untereinander, bestehen.

### 3.4 Chord

Bei Chord werden die Daten und Peers wieder mittels DHT angeordnet. Jedoch werden die beiden Enden miteinander verbunden, so dass ein Ring entsteht. Dadurch wird eine höhere Ausfallsicherheit erreicht, da jeder Peer über zwei Wege erreichbar ist. Zudem hält jeder Peer genau den Bereich zwischen ihm und seinem höheren Nachbarn (höhere Adresse).

Um die Suche zu beschleunigen verwendet man Zeiger, welche das Netzwerk so unterteilen, dass eine binäre Suche möglich wird. [5] Sucht man eine Datei, so folgt man dem letzten Zeiger, dessen Adresse noch unter der Gesuchten ist.

Sucht man im Beispiel unten von Peer N8 nach einer Datei mit der Adresse 23, so folgt man dem Zeiger (+8). Dieser verweist auf den Peer N21, welcher die Datei hält.

Zudem stellen die Zeiger zusätzliche Verbindungen zwischen Peers dar, was die Ausfallsicherheit erhöht.

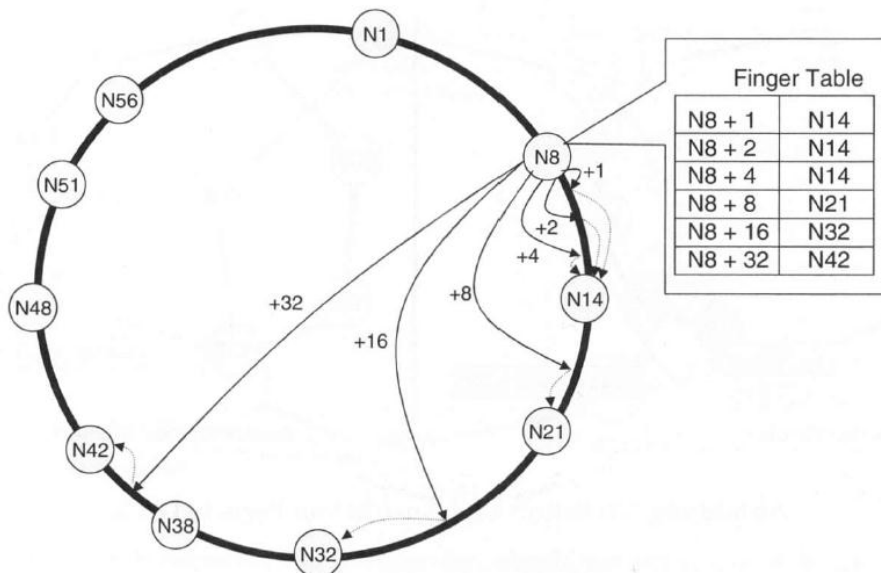


Abbildung 6: Finger Table in Chord [1]

## 4 Verteilter Download – Split-Stream

Werden Dateien von einem Server geladen, so teilen sich sämtliche Benutzer die Netzanbindung des Servers. Gerade bei grossen Datenmengen, wie sie bei Filesharing oder Videoübertragungen entstehen, stellt dies ein enormes Problem dar. Um den Server zu entlasten, werden so genannte Split-Stream-Protokolle eingesetzt.

Dabei wird die Datei in viele Teile zertrennt, welche zwischen 64kB und 1MB gross sind. Diese werden dann an unterschiedliche Clients weitergegeben. Zudem wird den Clients eine Liste anderer Clients mitgegeben, welche die restlichen Teile dieser Datei besitzen. Da sich die Clients nun untereinander die fehlenden Teile zur Verfügung stellen, muss der Server die Datei theoretisch nur einmal versenden.

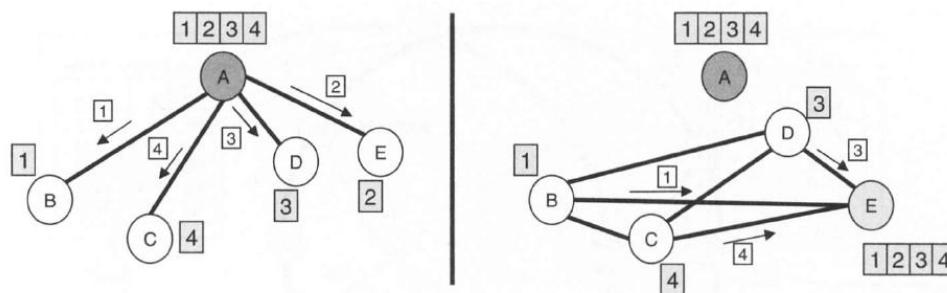


Abbildung 7: Split-Stream Protokoll

## 5 Literaturverzeichnis

- [1] Andreas Eberhart, Stefan Fischer, Carsten Kleiner, Arne Koschel Jürgen Dunkel: Systemarchitekturen für verteilte Anwendungen. Hanser, 2009.
- [2] Wikimedia.: Napster - Wikipedia [<http://de.wikipedia.org/wiki/Napster>], May 2010.
- [3] Bittorrent.: BitTorrent [<http://www.bittorrent.com/>], May 2010.
- [4] Wikimedia.: Gnutella - Wikipedia [<http://de.wikipedia.org/wiki/Gnutella>], May 2010.
- [5] Wikimedia.: Binäre Suche - Wikipedia [[http://de.wikipedia.org/wiki/Bin%C3%A4re\\_Suche](http://de.wikipedia.org/wiki/Bin%C3%A4re_Suche)], May 2010.
- [6] Christian Schindelhauer.: Algorithmen für Peer-to-Peer-Netzwerke [<http://www2.cs.uni-paderborn.de/cs/ag-madh/WWW/Teaching/2004SS/AlgoP2P/skript.html>], Apr. 2010.
- [7] Christian Schindelhauer.: Peer-to-Peer-Netzwerke [<http://electures.informatik.uni-freiburg.de/portal/web/guest/detail/-/modulnavigation/view/84/3296/?jsessionid=A4B789788E99FF7F238038028D89DDAE.liferay>], Apr. 2010.