



Peer-to-Peer-Systeme

I.1

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

1 Definition (2)



- * Idee
 - ◆ Verwischung der Rollen
 - Dienstnutzer sind gleichzeitig auch Dienstanbieter
 - „Alle sind gleichberechtigt (Peers).“
 - ◆ erwünschte Randeffekte
 - Fehlertoleranz
 - Skalierbarkeit
 - Nutzung der verteilten Ressourcen
 - balancierte Netzlast
 - mit hoher Lokalität

I.3

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

1 Definition

- Was sind Peer-to-Peer-Systeme?
 - ◆ peer = <engl.> der Gleiche, die Gleiche
 - ◆ Gegensatz zu Client-Server-Modell
 - ◆ Peers kommunizieren als Gleiche miteinander
- ▲ Nachteile des Client-Server-Modells
 - ◆ hohes Verkehrsaufkommen
 - Konzentration beim Server, Unterlast in anderen Netzteilen
 - asymmetrischer Verkehr
 - ◆ schlechte Skalierbarkeit
 - ◆ ungenutzte Ressourcen in Clients
 - Speicherplatz, Rechenleistung, Informationen
 - ◆ Server ohne besondere Maßnahmen ist Single-Point-of-Failure

I.2

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

1 Definition (3)



- Ein Peer-to-Peer-System liegt vor wenn,
 - ◆ das System verteilt ist,
 - ◆ der Ausfall eines der Teilsysteme keine schwerwiegenden Folgen hat,
 - ◆ jedes Teilsystem das System nutzt und ...
 - ◆ ... gleichzeitig selbst zum Gesamtsystem beiträgt.
- Beispiel: Informationsdienst
 - ◆ jeder Teilnehmer hält selbst Teilinformationen vor und gibt diese auf Anfrage heraus (Peer ist Client und Server)
 - ◆ es gibt keine zentrale Komponente

I.4

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

1.1 Typische Eigenschaften



- Eigenschaften typischer Peer-to-Peer-Systeme
 - ◆ Selbstorganisation
 - Teilsysteme kommen und gehen
 - Persistenz durch ständig verfügbare kritische Masse
 - ◆ Unabhängigkeit von festen Netzadressen
 - interne Adressierung unabhängig von augenblicklicher IP-Adresse
- Eigenschaften spezieller Peer-to-Peer-Systeme
 - ◆ Verschleierung der Systemaktionen
 - Beispiel: Tauschbörsen
 - verschleierte Verbindung zwischen anbietendem Teilsystem und Informationsinhalt
 - verschleierte Herkunft von Informationen

I.5

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2 Architekturen



- * Herausragendes Architekturmerkmal von Peer-to-Peer-Systemen:
 - ◆ Interaktionsmuster der Komponenten
- Architektur des internen Interaktionsmusters
 - ◆ (Client-Server-Systeme)
 - ◆ hybride Peer-to-Peer-Systeme
 - ◆ Super-Peer-to-Peer-Systeme
 - ◆ reine Peer-to-Peer-Systeme
- Klassifikation der Peer-to-Peer-Kommunikation
 - ◆ strukturiertes Kommunikationsnetzwerk
 - ◆ unstrukturiertes Kommunikationsnetzwerk
- ◆ statisches oder dynamisches Kommunikationsnetzwerk

I.7

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

1.2 Aufgabengebiete



- Unterscheidung nach Aufgaben
 - ◆ datenzentrierte Dienste
 - Speichern und Suchen von bestimmten Datensätzen
 - z.B. Tauschbörsen, Dateisysteme, File-Sharing-Systeme
 - ◆ ausführungorientierte Dienste
 - verteiltes Rechnen
 - z.B. Grid-Systeme
 - Instant-Messaging-Dienste
 - ◆ daten- und ausführungorientierte Dienste
 - rechnergestützte Kooperation
 - z.B. Shared White Board, Groupware

I.6

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.1 Hybride P2P-Systeme



- Zentrale Systemkomponente
 - ◆ Mischung zwischen Client-Server-Interaktion und Peer-to-Peer-Ansätzen
 - ◆ zentrale Komponente übernimmt Serverfunktion
- Beispiele
 - ◆ Napster (Tauschbörse)
 - zentraler Verzeichnisdienst über Orte der Dateispeicherung
 - dezentrales Laden von Dateien
 - ◆ ICQ, AOL Instant Manager (Instant-Messaging-Systeme)
 - zentraler Verzeichnisdienst über eingeloggte Benutzer
 - dezentrale Benutzerinteraktion

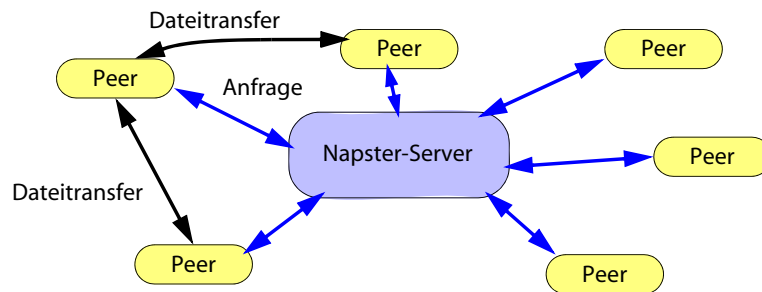
I.8

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.1 Hybride P2P-Systeme (2)



- Beispiel: Napster, erstes populäres P2P-System (1999–2001)
- ◆ Austausch von Musik-Dateien



- ◆ Zentraler Verzeichnisserver
 - Verwaltet Adressen und Dateilisten der Peers
- ◆ dezentraler Dateiaustausch

I.9

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.2 Super-P2P-Systeme



- Zweistufige Struktur
 - ◆ Client-Server-Interaktion zwischen „normalen“ und einer „ausgezeichneten“ Komponenten
 - ◆ Peer-to-Peer-Interaktion zwischen „ausgezeichneten“ Komponenten
- Beispiel: KaZaA (seit 2001)
 - ◆ Tauschbörse für Audio- und Videodateien basierend auf FastTrack-Protokoll
 - ◆ Peer-to-Peer-Interaktion zwischen „normalen“ Teilnehmern
 - ◆ gut angebundene Peers werden zu Super-Peers
 - Zuordnung von „normalen“ Teilnehmern zu einem Super-Peer – kennt peer-lokalen Dateien
 - Peer-to-Peer-Interaktion zwischen „ausgezeichneten“ Teilnehmern (Super-Peers)

I.11

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.1 Hybride P2P-Systeme (3)



- ▲ Probleme
 - ◆ Skalierbarkeit eingeschränkt wegen zentralem Verzeichnisserver
 - ◆ eingeschränkte Fehlertoleranz: Single-Point-of-Failure im Verzeichnisserver
- Erweiterungsmöglichkeit
 - ◆ Ersetzung der zentralen Systemkomponente durch P2P-Netzwerk
 - Super-P2P-Systeme
 - ◆ Beispiel: Napster-Erweiterung
 - Mehrere vernetzte Verzeichnisserver (OpenNap-Netzwerk)

I.10

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.3 Reine P2P-Systeme



- Keine zentrale Komponente
 - ◆ vollständig dezentrale Organisation
- Beispiel: Usenet (seit 1979)
 - ◆ Verteilung von News-Artikeln zwischen Peers
 - ◆ statisches Kommunikationsstruktur
 - konfigurierte News-Feeds
 - ◆ Flooding-Protokoll über alle Verbindungen
 - Austausch von Artikel-IDs
 - anschließende Übertragung noch nicht bekannter Artikel

I.12

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-Avid-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.3 Reine P2P-Systeme (2)



- Beispiel: Gnutella (seit 2000)
 - ◆ verteiltes Dateisystem
 - ◆ dynamische Kommunikationsstruktur
 - mit einem bekannten Knoten kann man in das System integriert werden
 - dynamischer Aufbau von Verbindungen
 - ◆ unstrukturiertes Kommunikationsnetz
 - zufällig entstehende Netzstruktur

I.13

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

3 Verteilte Streuspeicherung



- Engl. Distributed Hashing
 - ◆ dynamische Aufteilung von Daten, Dateien oder ähnlichem auf die Peers
 - Berücksichtigung der Knotenfluktuation
 - ◆ surjektive Abbildung von Hashwert auf Datum
 - für jedes Datum gibt es einen Hashwert
- Idee: Zuständigkeit von Peers für bestimmte Hashwerte
 - ◆ effizientes Routing von Anfragen an den zuständigen Peer
- Fallbeispiel: Chord (seit 2001)
 - ◆ P2P-Protokoll
 - ◆ dynamische Kommunikationsstruktur
 - ◆ strukturiertes Kommunikationsnetz
 - Ring

I.15

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

2.3 Reine P2P-Systeme (3)



- Beispiel: Pastry (seit 2001)
 - ◆ P2P-Protokoll für Vielzahl von P2P-Diensten
 - z.B. verteiltes Dateisystem (*Past*)
 - ◆ dynamische Kommunikationsstruktur
 - ◆ strukturiertes Kommunikationsnetz
 - Baumstruktur entsteht anhand zufällig verteilter Knoten-IDs
 - effizientes Routing von Nachrichten an Knoten mit bestimmter ID
 - ◆ geeignet für verteilte Streuspeicherung (*Distributed Hashing, Distributed Hash Table, DHT*)

I.14

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

3.1 Chord: ID-Zuordnung



- Knoten-IDs (*Node-Keys*)
 - ◆ 160bit-Werte
 - ermittelt über Secure-Hash-Funktion aus IP-Adresse (SHA-1)
 - „zufällige“ Verteilung der Node-Keys über alle Chord-Teilnehmer
- Daten-IDs (*Keys*)
 - ◆ 160bit-Werte
 - ermittelt über SHA-1 vom Datum (oder einem entscheidenden Teil davon)
- Zuordnung von Daten auf Knoten mit gleichem Key
 - ◆ oder dem Knoten mit dem nächst höheren Key (modulo 2^{160})
 - ◆ Anordnung der Knoten nach Key im Kreis (Ringstruktur)

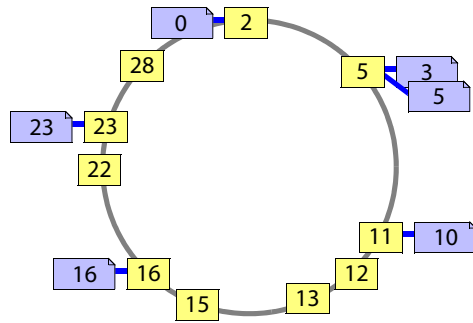
I.16

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

3.2 Chord: Ringstruktur



■ Knotenring und gespeicherte Daten



◆ jeder Knoten kennt Vorgänger- und Nachfolgerknoten

I.17

3.3 Chord: Routing



■ Primitives Routing

- ◆ falls lokaler Knoten kleineren Key hat als gesuchter Key:
 - Weitergabe an Nachfolger

■ Effizientes Routing

- ◆ Fingertabelle im Knoten k mit z.B. b Einträgen
 - Eintrag i enthält Knoten für dessen Key m gilt: $m \geq (k + 2^{i-1}) \bmod 2^b$
 - z.B. $b = 5, k = 22$
 - $i = 1: m \geq (22 + 2^0) \bmod 2^5 = 23 \bmod 32 = 23$ (Nachfolger)
 - $i = 2: m \geq (22 + 2^1) \bmod 2^5 = 24$
 - $i = 3: m \geq (22 + 2^2) \bmod 2^5 = 26$
 - $i = 4: m \geq (22 + 2^3) \bmod 2^5 = 30$
 - $i = 5: m \geq (22 + 2^4) \bmod 2^5 = 12$
- Tabelle durchmisst Ring in verschiedenen Schrittstufen (Hälfte, Vierte, Achtel etc.)

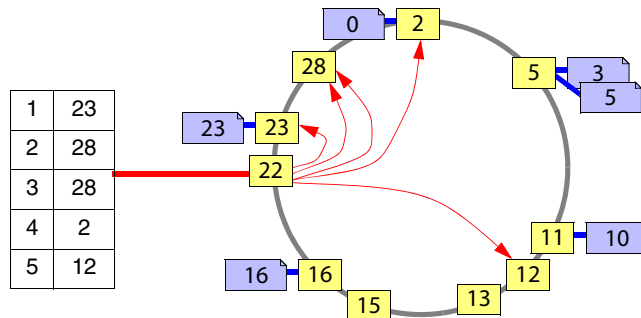
I.18

3.3 Chord: Routing (2)



■ Effizientes Routing (fortges.)

◆ Beispiel für Finger-Tabelle:



- ◆ Routing: Weiterleitung an Knoten mit höchstem Node-Key kleiner als Daten-Key
 - erster Schritt: großer Abstand; folgende Schritte: Abstände kleiner

I.19

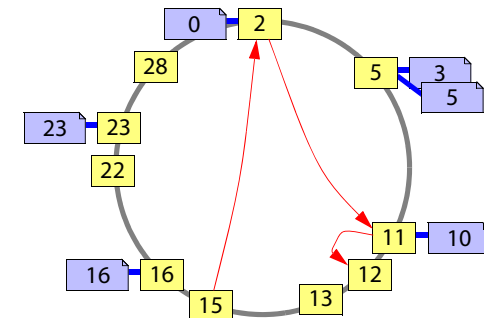
3.3 Chord: Routing (3)



■ Effizientes Routing (fortges.)

◆ Beispiel für Routing:

Suche nach **12**
vom Knoten **15**



◆ hier: nur drei Schritte notwendig

I.20

3.4 Chord: Dynamische Ringstruktur



- Knoten verlässt Ring kontrolliert
 - ◆ Knoten macht Vorgänger mit Nachfolger bekannt
- Knoten verlässt Ring unkontrolliert
 - ◆ Vorgänger kennt für Fehlerfall mehrere Nachfolger
- Knoten mit Key k betritt Ring
 - ◆ irgendein Knoten bekannt
 - ◆ Suche des Knotens der für k verantwortlich ist
 - dieser wird Nachfolger von Knoten k
 - Protokoll für Anpassung der Vorgänger- und Nachfolger-Angaben
 - ◆ Suche von Knoten für die Finger-Tabelle

I.21

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

4 Literatur



- Peer-to-Peer-Systeme
 - ◆ *Peer-to-Peer*. Wikipedia, The Free Encyclopedia, 21.07.2004.
<<http://en.wikipedia.org/wiki/Peer-to-peer>>
 - ◆ H. Reiser, R. Kapitza. *Verteilte Algorithmen*. Unterlagen zur Vorlesung im WS2003/2004, Univ. Erlangen-Nürnberg.
<http://www4.informatik.uni-erlangen.de/Lehre/WS03/V_VA/>
 - ◆ *The Chord Project*. MIT, 21.07.2004.
<<http://www.pdos.lcs.mit.edu/chord/>>

I.23

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

3.5 Verwendung der Schlüssel



- Beispiele
 - ◆ Hashing der Daten selbst
 - ◆ Hashing von Dateinamen
 - ◆ Hashing von anderen Schlüsseldaten des eigentlichen Datums
 - z.B. Name eines Personendatensatzes
- Fehlertolerante Speicherung
 - ◆ redundantes Speichern des Datums in Nachfolgerknoten des für das Datum verantwortlichen Knotens

I.22

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-I-P2P.fm, 2006-07-07 15.56] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>