



Peer 2 Peer Netzwerke

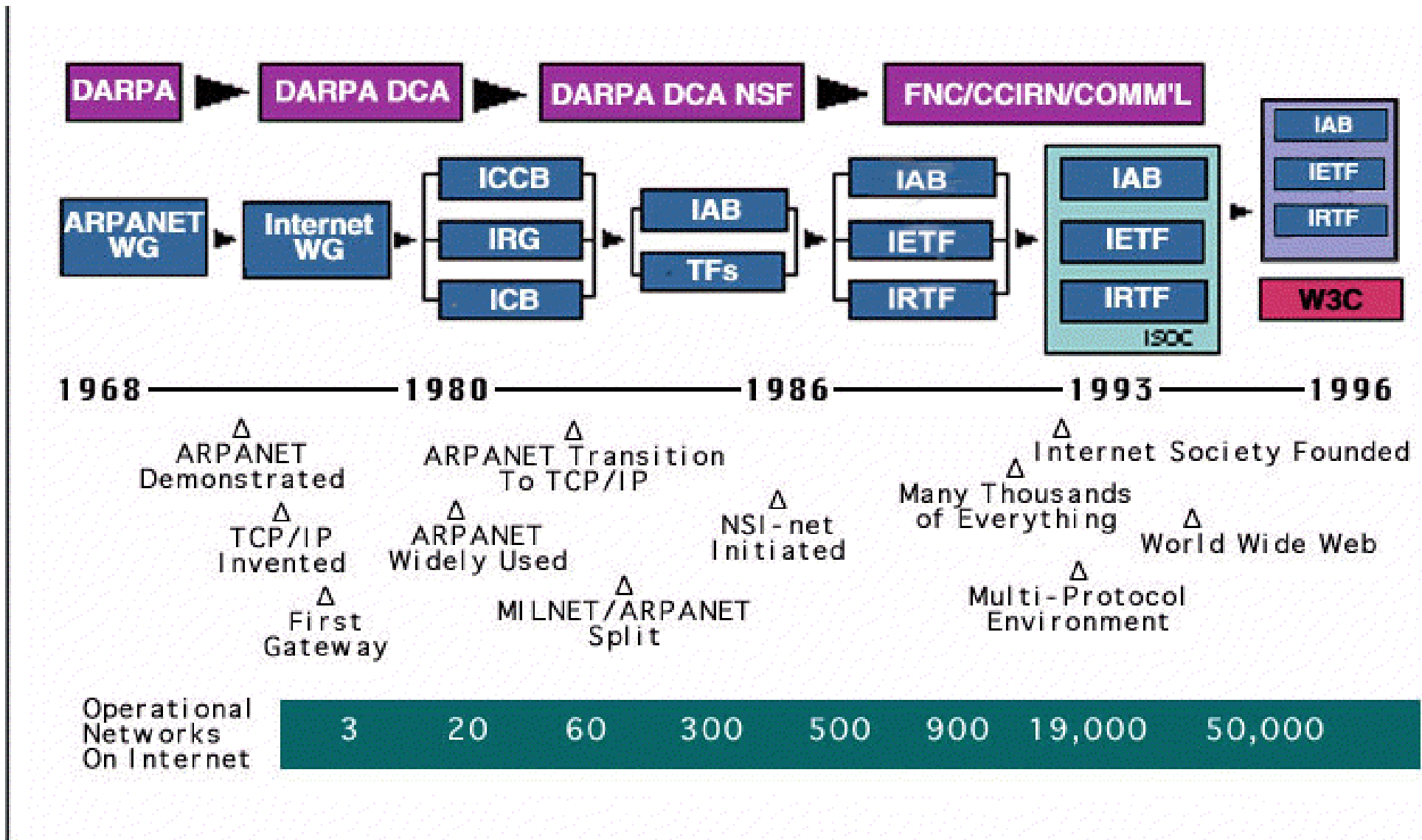
- Liqing Gao, Karsten Groll, Lars Rieche
Institut für Informatik
FU Berlin
- 19.11.2008

- Einführung in P2P Systeme
- Warum Gnutella nicht skaliert
- Freenet

Motivation und Definition

- Einführung in P2P Systeme
 - 1. Geschichte des Peer-to-Peer Modells
 - 2. Warum Peer-to-Peer?
 - 3. Eigenschaften von Peer-to-Peer Systemen
 - 4. Applikationen und Klassifizierung
 - 5. Peer-to-Peer and Overlays
- Warum Gnutella nicht skaliert
- Freenet

Entwicklung des Internets



- **1st Generation Internet**
- Original Internet (ARPANET) war ein Peer-to-Peer Netzwerk
 - Verbindung unabhängiger Knoten, welche auch bei Ausfällen von einzelnen Verbindungen und Knoten funktionierte.
 - Eine dezentralisierte Verbindung verteilter, unabhängiger Systeme -> P2P

Entwicklung des Internets

- **2nd Generation Internet**
- E-mail (1971) und World Wide Web (1991)
- Wechsel in ein Client/Server Modell
 - Web-, Gaming-, DNS-, Datenbank-, DHCP-, ... Server
- Zentrale Verwaltung durch Service Provider

Entwicklung des Internets

- **P2P Generation Internet**
- **3rd Generation Internet?**
 - Günstige Leistungsstarke Hardware
 - Günstige Verbindung ins Internet
 - Frühe P2P Systeme
 - Napster
 - Gnutella
 - Seti@Home

Motivation und Definition

- Einführung in P2P Systeme
 - 1. Geschichte des Peer-to-Peer Modells
 - 2. Warum Peer-to-Peer?
 - 3. Eigenschaften von Peer-to-Peer Systemen
 - 4. Applikationen und Klassifizierung
 - 5. Peer-to-Peer and Overlays
- Warum Gnutella nicht skaliert
- Freenet

Probleme des Internets

- Skalierbarkeit
- Flexibilität / Erweiterbarkeit
- Sicherheit / Erreichbarkeit

- Bietet das Peer-to-Peer Prinzip geeignete Lösungen?

- **Skalierbarkeit**

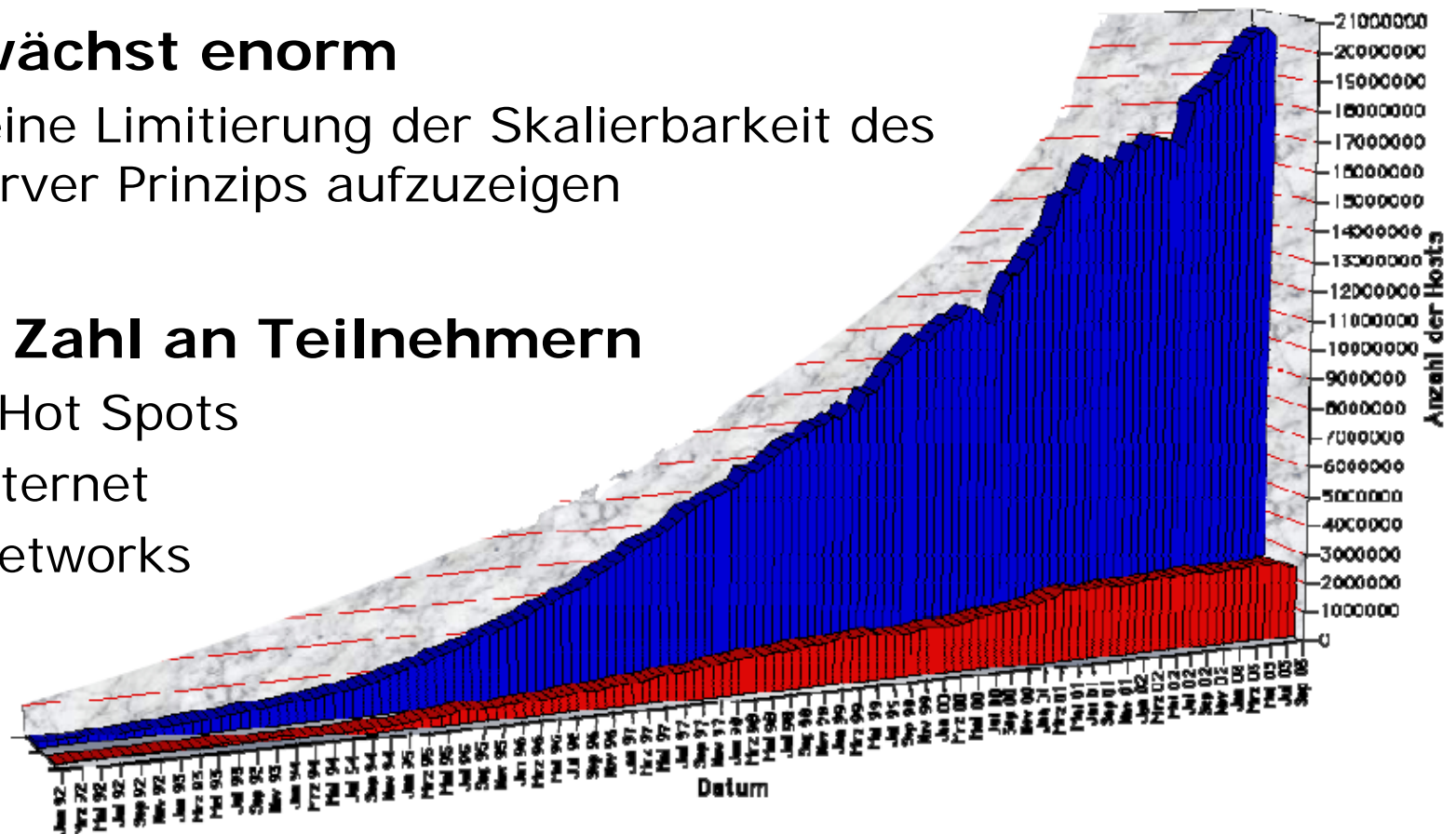
Möglichkeit eines Systems effizient zu arbeiten und funktionieren während es wächst

- **Internet wächst enorm**

- Scheint eine Limitierung der Skalierbarkeit des Client-Server Prinzips aufzuzeigen

- **Steigende Zahl an Teilnehmern**

- Wireless Hot Spots
- Mobile Internet
- Sensor Networks

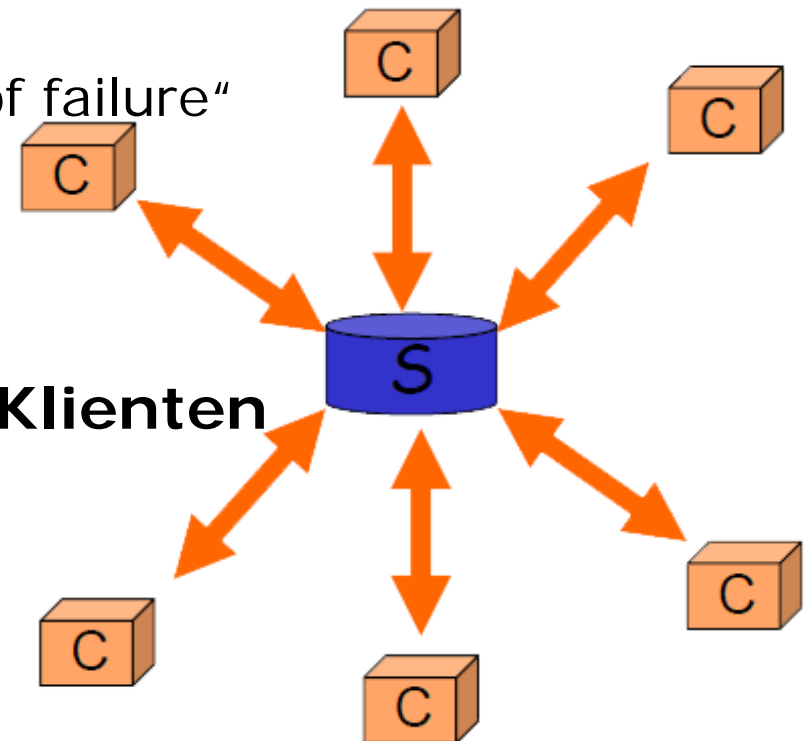


- **Generelle Probleme der Client-Server Architektur**
- **Limitierte Ressourcen eines Servers**

- Immer mehr Traffic
 - Stark konzentriert auf einzelnen Servern
 - Asymmetrischer Traffic-Verlauf
- Problem bei der Skalierbarkeit der Ressourcen
 - Speicher, CPU
- Server birgt Problem des „single point of failure“
- Sterntopologie

- **Ungenutzte Ressourcen auf vielen Klienten**

- CPU
- Memory
- Informationen



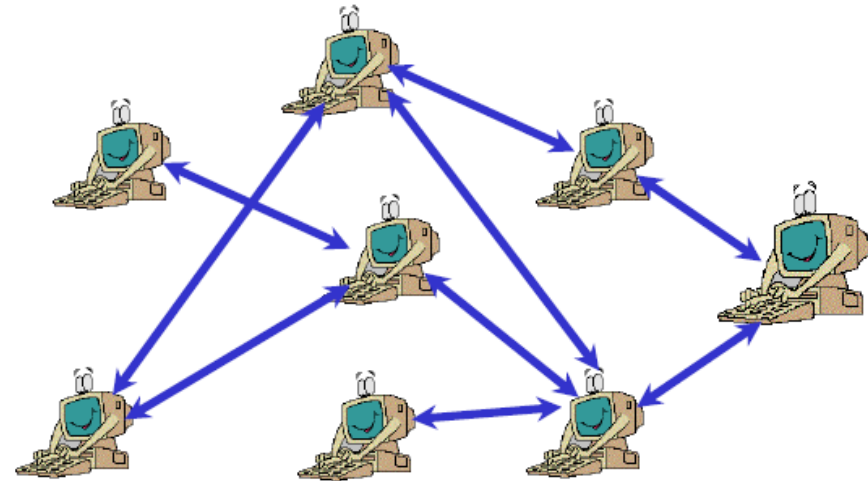
- **Client-Server Architektur skaliert nicht unendlich groß**
 - Limitierte Ressourcen
- **Neue Anwendungen brauchen immer mehr Ressourcen**
 - Bandbreite
 - Speicher
 - CPU
- **Diese Anforderungen können von Zentralen Systemen nicht bewältigt werden**
 - 2002: Kazaa 10^7 Gbyte, Seti@Home 20 TerraFlops
 - 2008: BitTorrent 10^4 TB, Seti@Home 900 TerraFlops
 - IBM Roadrunner: 1.026 TerraFlops, BlueGene/L 478 TFlops

- **Erreichbarkeit wird bei zukünftigen Services immer wichtiger**
 - Cloud Computing (Google Docs, Mail)
 - Ausfall kann hohe Kosten verursachen
- **Steigende Zahl von Angriffen**
 - Distributed denial-of-service attacks (DDoS)
 - Zentrale Server sind gute Ziele
- **Fehlertoleranz**
 - Unmöglich bei Client/Server Model
 - Braucht Verteilung

Motivation und Definition

- Einführung in P2P Systeme
 - 1. Geschichte des Peer-to-Peer Modells
 - 2. Warum Peer-to-Peer?
 - 3. Eigenschaften von Peer-to-Peer Systemen
 - 4. Applikationen und Klassifizierung
 - 5. Peer-to-Peer and Overlays
- Warum Gnutella nicht skaliert
- Freenet

Was sind P2P-Systeme



- Eine erste Definition

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

(Clay Shirky)

- Interaktion von end systems (peers)
- Gemeinsame Nutzung von Ressourcen in end systems
- Keine Zentrale Kontrolle
- Keine Benutzung Zentraler Services
- Teilnehmer sind gleich (equal) und autonom
- Selbstorganisation (self-organization) des Systems

- Dienste werden direkt zwischen Endsystemen genutzt
 - Minimale Anforderungen an die Infrastruktur
 - Keine Nutzung Zentraler Dienste wie DNS, Datenbanken usw.
- Einfach zu Implementieren
 - Zur Installation keine Hilfe eines ISPs notwendig
 - Keine kostspieligen Zentrale Server
 - Realisierung von neuen Diensten (multicast)
 - Keine Standardisierung notwendig

- Vermeidung der Probleme Zentral genutzter Ressourcen
 - Voraussetzung um Skalierbarkeit zu erreichen
 - Große Nachfrage nach Ressourcen von neuen Applikationen
 - SETI@Home: 2,3 Millionen Jahre Rechenzeit seit 1999
 - BitTorrent (10⁴ TB Daten (2008))
 - Würden unglaubliche Kosten verursachen bei zentralem Prinzip
- Redundanz durch Dezentralisierung oder Replikation
 - Weniger Einfluss von DDoS attacks
 - Speicherung von Daten kann nicht verboten werden
 - Speicherung in der Nähe von Peers

Keine Benutzung Zentraler Services

- Fehlertoleranz
- Skalierbarkeit
 - Mehr Peers -> Mehr Ressourcen
- Endverbraucher werden Service Provider
 - Vermeidung von Problemen
 - NAT, firewalls
- Autonomie der Teilnehmer

equal and autonomous

- **All peers are equal**
 - (beinahe)
 - Es gibt Peers mit spezieller Verantwortung
 - Nicht vorhersagbar !!
 - Wechsel bei wechselnden Bedingungen
- **Bessere Verteilung**
- **Dynamische Aufteilung der Verantwortung auf die Peers**
- **Gemeinsam Verteilte und Dezentralisierte Dienste**
- **Warum?**
 - Server sind „single point of failures“
 - Ressourcen in end systems sind oft im idle Modus

Motivation und Definition

- Einführung in P2P Systeme
 - 1. Geschichte des Peer-to-Peer Modells
 - 2. Warum Peer-to-Peer?
 - 3. Eigenschaften von Peer-to-Peer Systemen
 - 4. Applikationen und Klassifizierung
 - 5. Peer-to-Peer and Overlays
- Warum Gnutella nicht skaliert
- Freenet

Peer-to-Peer Systeme

- **Client-Server Systeme**

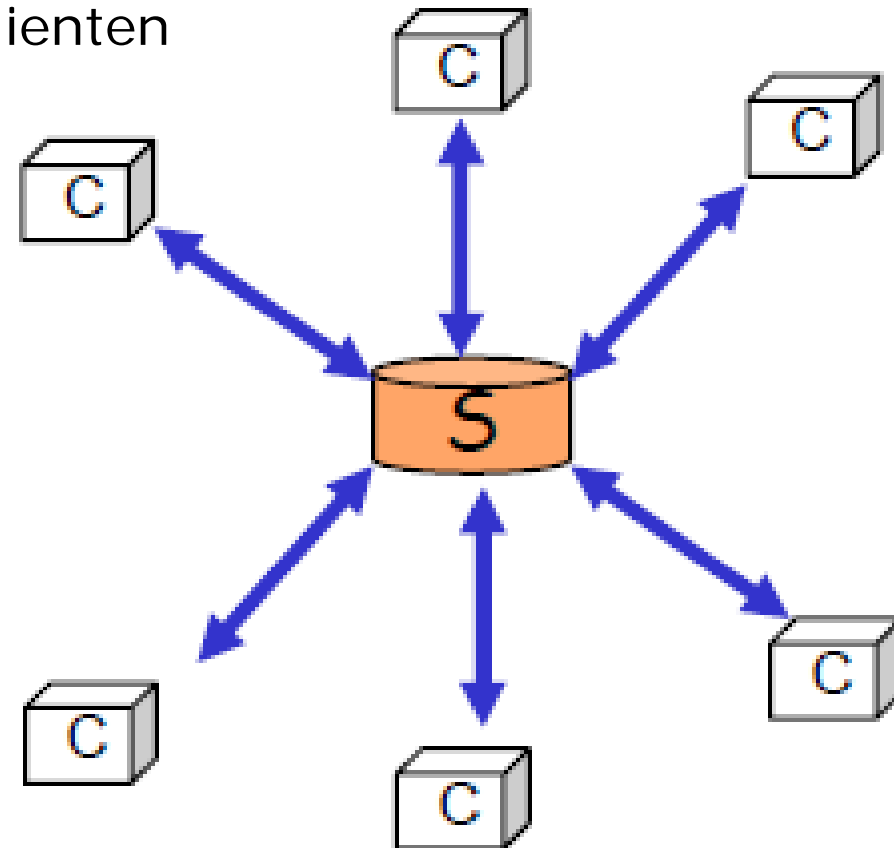
Klassisches System

Keine Interaktion zwischen Klienten

- **Beispiele**

WWW

DNS



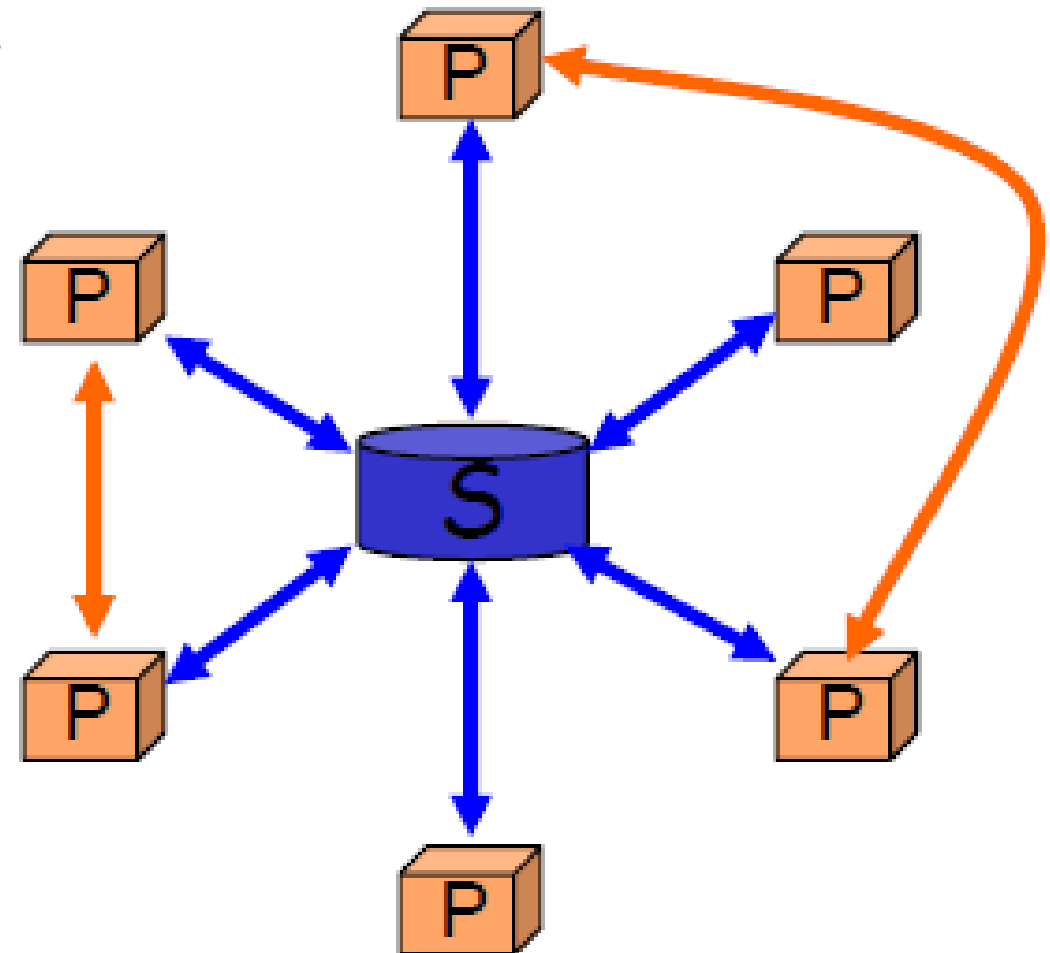
Peer-to-Peer Systeme

- **Hybrid P2P Systeme**

- Gemeinsame Nutzung von Ressourcen
 - Direkte Interaktion von Peers
 - Server zur Koordinierung

- **Beispiele**

- Napster
 - ICQ
 - Seti@Home



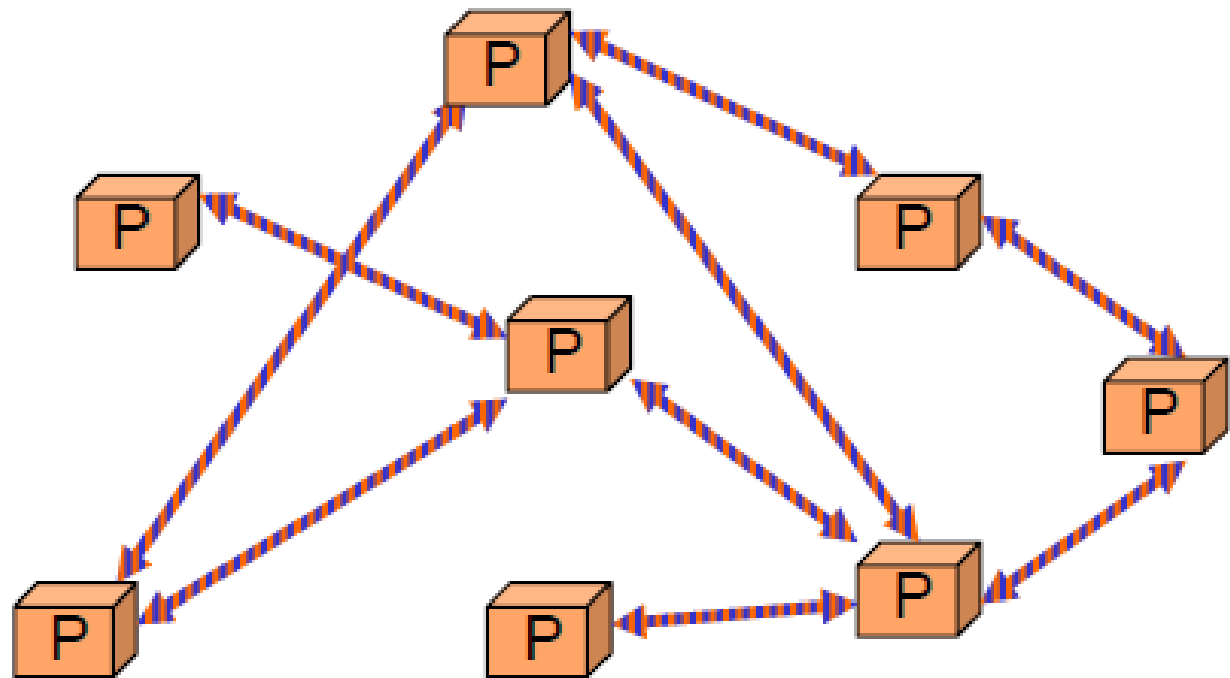
Peer-to-Peer Systeme

• Pure P2P Systeme

Komplett Dezentralisiertes
Management und Nutzung
von Ressourcen

• Beispiele

Gnutella
Freenet
DHT Applikationen
(Past, Ocean-Store)



Motivation und Definition

- Einführung in P2P Systeme
 - 1. Geschichte des Peer-to-Peer Modells
 - 2. Warum Peer-to-Peer?
 - 3. Eigenschaften von Peer-to-Peer Systemen
 - 4. Applikationen und Klassifizierung
 - 5. Peer-to-Peer and Overlays
- Warum Gnutella nicht skaliert
- Freenet

Overlay Netzwerke

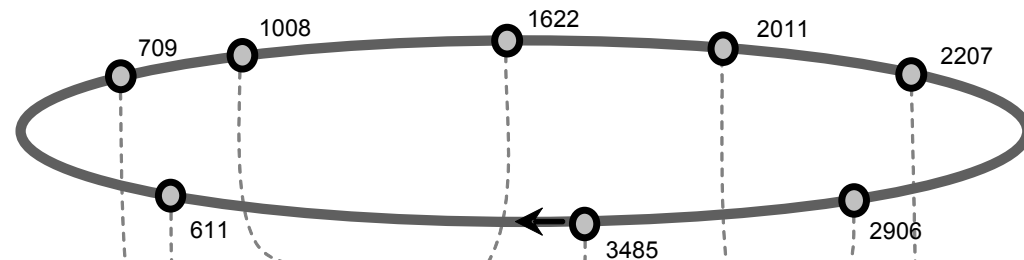
- Das Internet war ein Overlay Network über dem PSTN Telefonnetzwerk
 - Adressierung und Routing durch das PSTN
 - Trotzdem hatte das Internet eigene Adressierungs und Routing Verfahren
- Overlay Networks werden benutzt um Netzwerkfunktionalität zu erweitern
 - Multicast mit dem MBone-Overlay
 - IPv6 im 6Bone
 - Sicherheit durch Virtual Private Networks
- Hauptunterscheid zu non-virtual-networks:

Ein Overlay Network kann seine eigene Netzwerktopologie ausbilden.

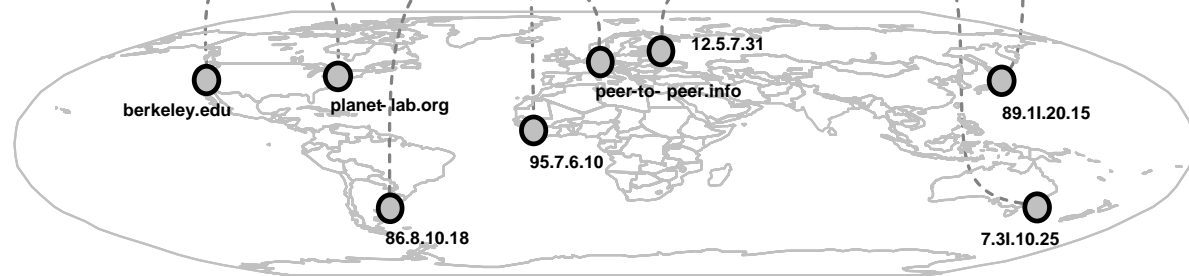
Overlay Netzwerke

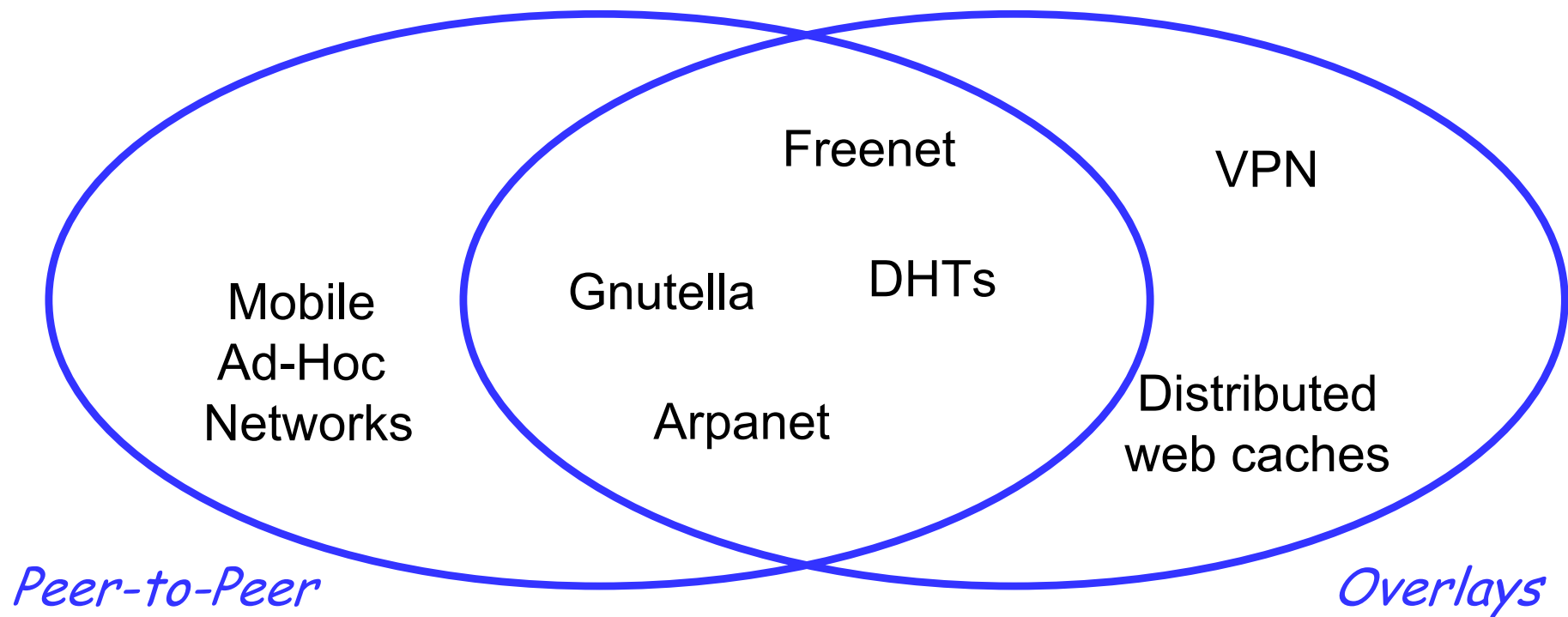
- Peer-to-Peer-Netzwerk \neq Overlay Netzwerk
 - Definition Overlay Netzwerk :
 - "Two application communicate on a different path than the given path by the network layer"

Topologie
des Overlay
Netzwerks



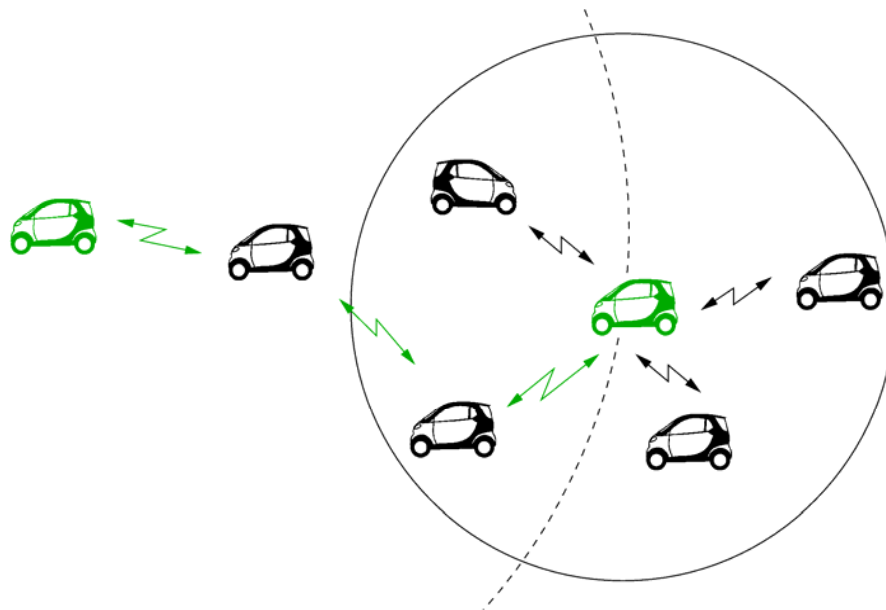
Topologie
des Internets





Overlay Netzwerke

- P2P, aber kein Overlay:
 - Mobile Ad-Hoc Networks
 - Jeder Noder ist Client und Server
 - Keine Infrastruktur
 - Sensor Netzwerke
- Overlay, aber kein P2P
 - Client/Server:
 - Distributed Web-Caches
 - Virtual private networks (VPN)



- P2P und Overlay
 - ARPANET
 - Gnutella
 - Distributed Hash Tables (DHTs)

- Peer-to-Peer networking
 - Dezentralisierte, selbstorganisierte Systeme mit dezentralisierter Nutzung von Ressourcen
 - Prinzip des Original Internets
 - Wurde ersetzt durch Client/Server Modell
- Gründe für heutige Nutzung von P2P Netzwerken
 - Das Heutige Internet hat viele Schwachstellen
 - Skalierbarkeit, Flexibilität, Erweiterbarkeit, Erreichbarkeit und Sicherheit
- Weitere Gründe:
 - Erfolg von P2P Anwendungen
 - Neue Arten von Services (ICQ, file sharing, etc.)

- Einführung in P2P Systeme
- Warum Gnutella nicht skaliert
- Freenet

- Einführung in P2P Systeme
- Warum Gnutella nicht skaliert
 - Was ist Gnutella?
 - Funktionsweise
 - Protokoll
 - Bandbreiten
 - Free Riding
 - Zusammenfassung
- Freenet

Was ist Gnutella?

- Entwickelt von Justin Frankel und Tom Pepper von Nullsoft (März 2000)
- Ein „File-Sharing“ - Protokoll
- Kein System oder Software
- Open-Source Programm
- Arbeitet ohne zentralen Server.
- Reines P2P-Netz (vollkommen dezentral), da alle Peers gleiche Funktion haben
- Hoher Anonymitätsgrad
- Jeder Client dient gleichzeitig als Server und Suchmaschine =: "servent" (servant?)



Windows

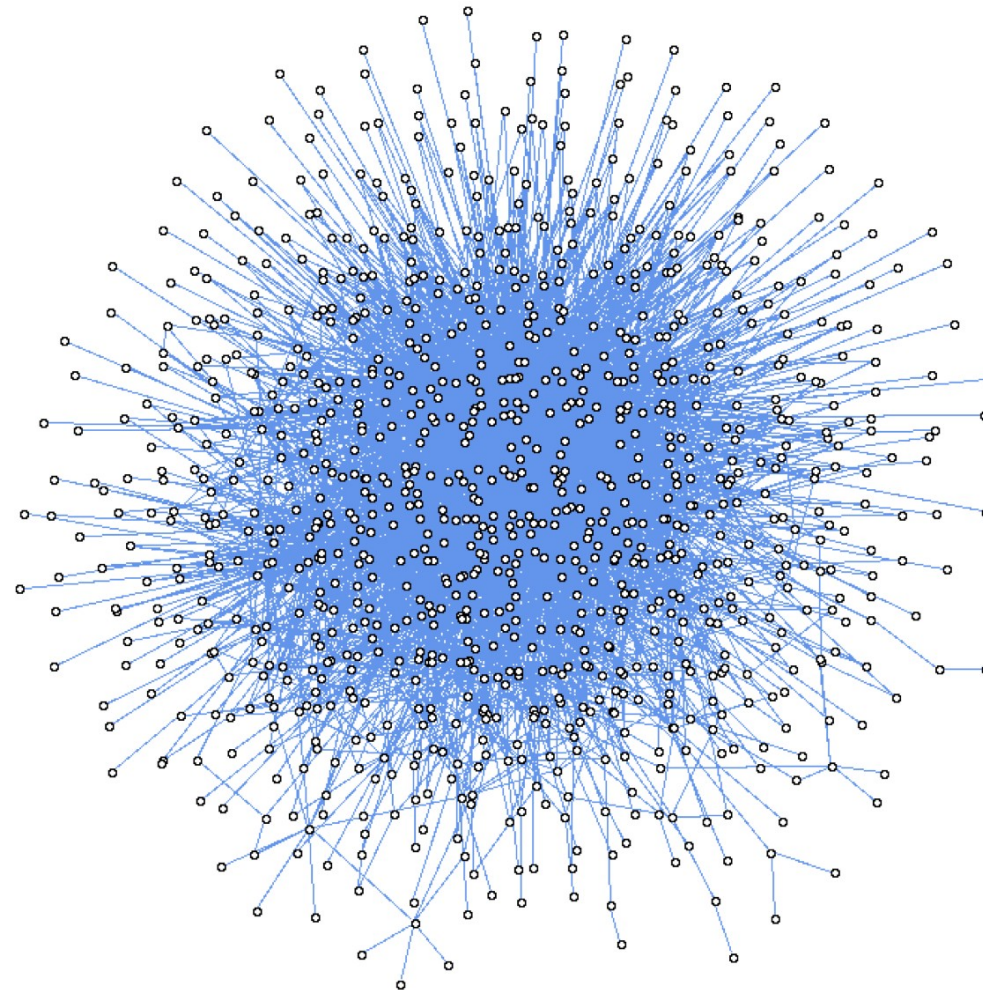
- LimeWire
- Gnotella
- Toadnode
- Gnucleus
- BearShare
- Shareaza
- PheX
- Gnewtella



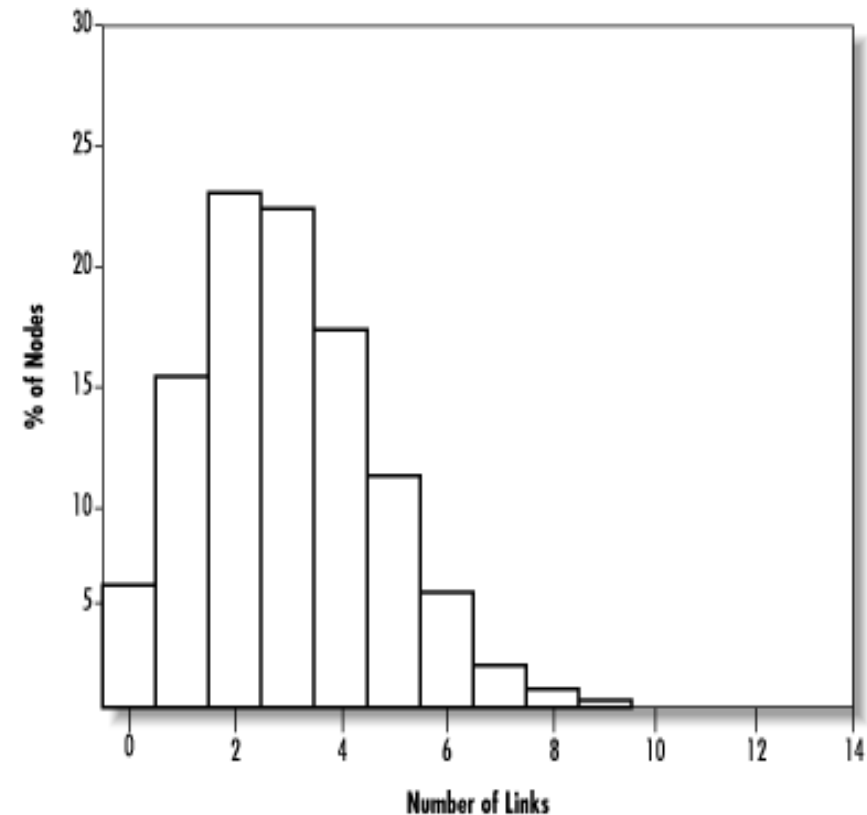
Linux/Unix

- LimeWire
- Gnut
- Gtk-Gnutella
- Mutella
- Phex
- Qtella

Gnutella Schnappschuss im Jahr 2000



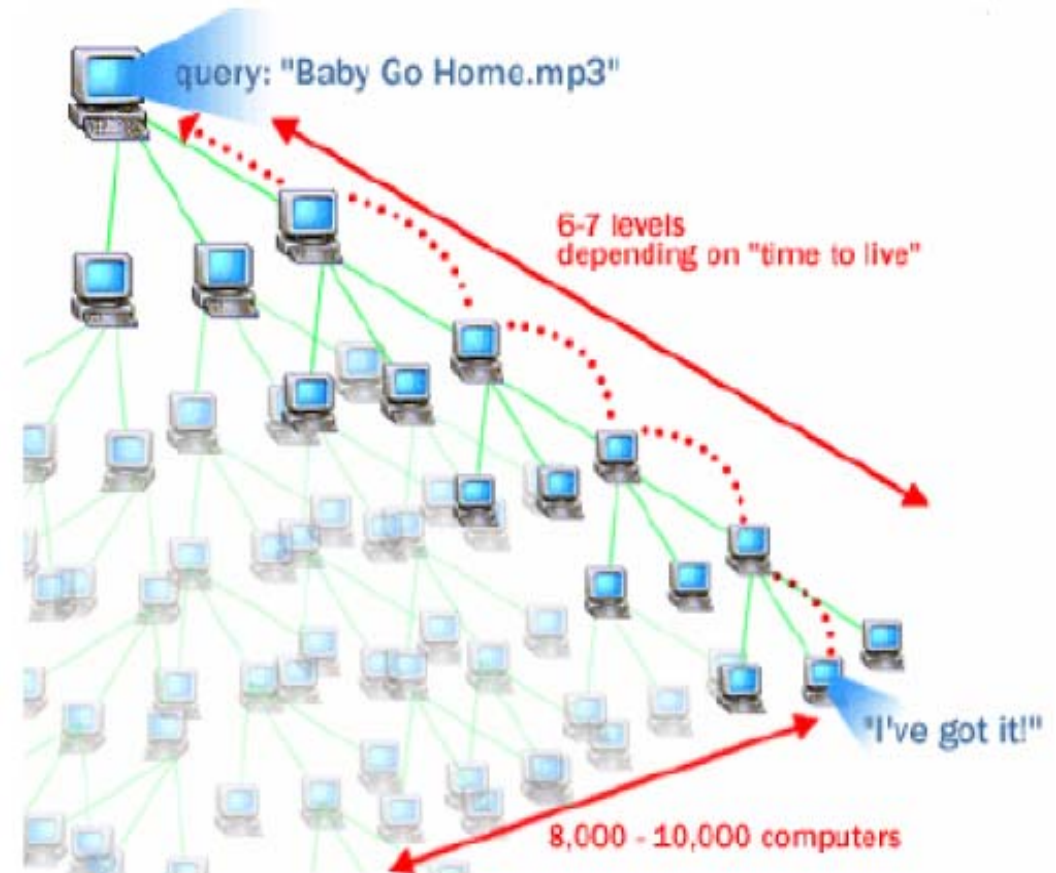
- Einstiegspunkt
 - Gespeicherte Teilnehmerliste (Host-Cache)
 - Öffentlich bekannter Teilnehmer mit fester Adresse:
 - Sendet nur Pong-Pakete und schließt danach die Verbindung.
 - Manuelle Eingeben
- Häufig zwischen 1 und 6 Verbindungen zu anderen



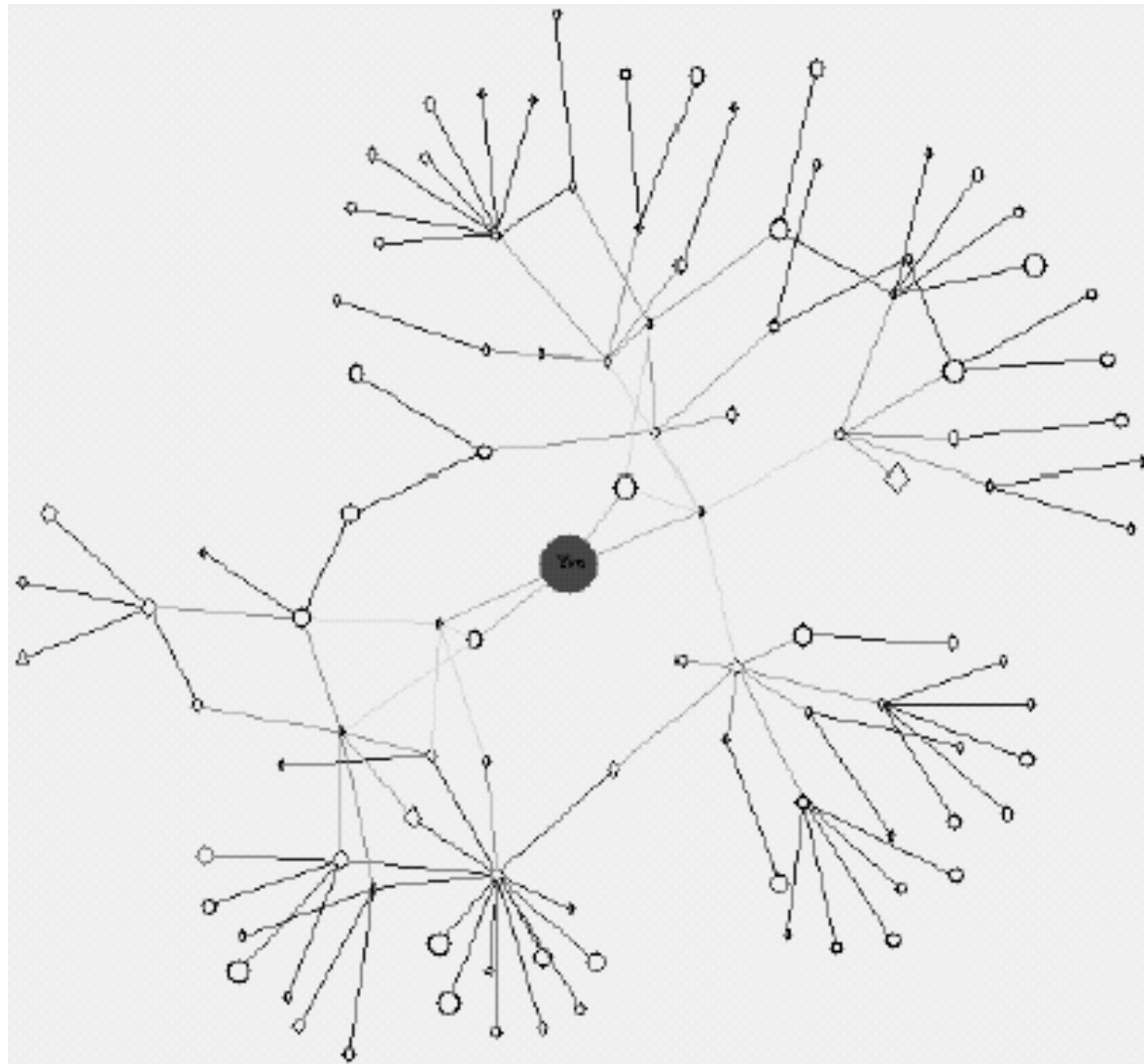
Quelle: [Oram, „Peer-to-Peer“]

Funktionsweise-Flooding Verfahren

- Flooding Verfahren
 - Flooding = "Fluten", ähnlich Breitensuche:
 - Das Netz mit Suchanfragen überfluten
 - An alle Nachbarn broadcastet.
 - Jeder Suche gibt es eine bestimmte Laufzeit, Sog. TTL (Time To Live)



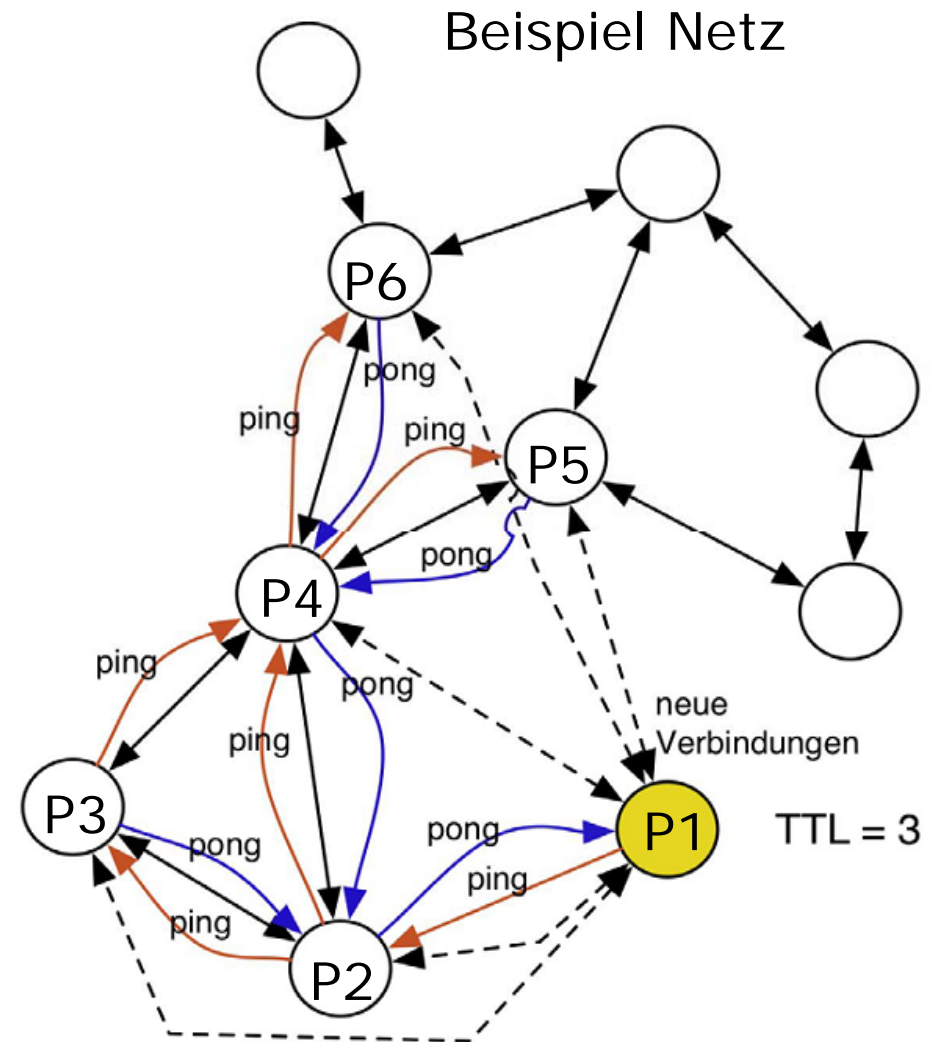
Quelle: [Lutz Küderli, "Peer-to-Peer"]



Verbindungsgraph bei TTL= 4
(GraphViz von Gnucleus)

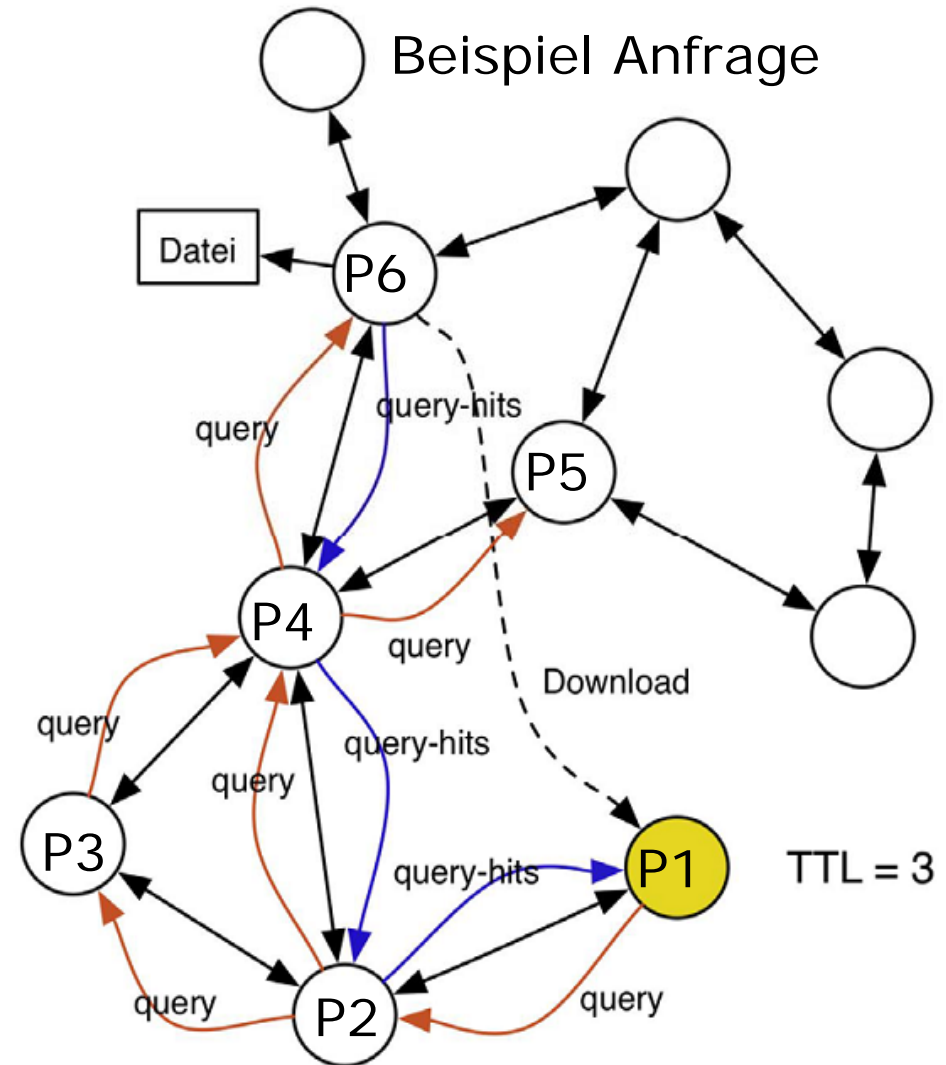
Protokoll - Anbinden

- Ping
 - Anfrage nach anderen Knoten im Netz
- Pong
 - Antwort auf Ping, enthält IP-Adresse und Port usw. von einem anderen Knoten



Protokoll - Anfrage

- Query
 - Anfrage nach Datei wird bis zu TTL-Feld weitergereicht.
- Query-hits
 - Antwort auf umgekehrten Pfad
- Push
 - Sendeaufforderung, zum umgehen von Firewalls



- Analyse von Bandbreite
 - Jede Nachricht besitzt ein einheitlichen Header (23 Byte)
 - Pro Nachricht entsteht mind. 23 Byte Daten (Ping)
 - Bei Pong, Query, Query-Hits Anfrage besitzen noch extra Daten
 - Plus 40 Byte Header-Daten für TCP/IP-Schicht

Header für alle Nachrichten	
Descriptor ID	16 Bytes
Payload Descr.	1 Byte
TTL	1 Byte
Hops	1 Byte
Payload Length	4 Bytes

PING (0x00)
keine weiteren Felder

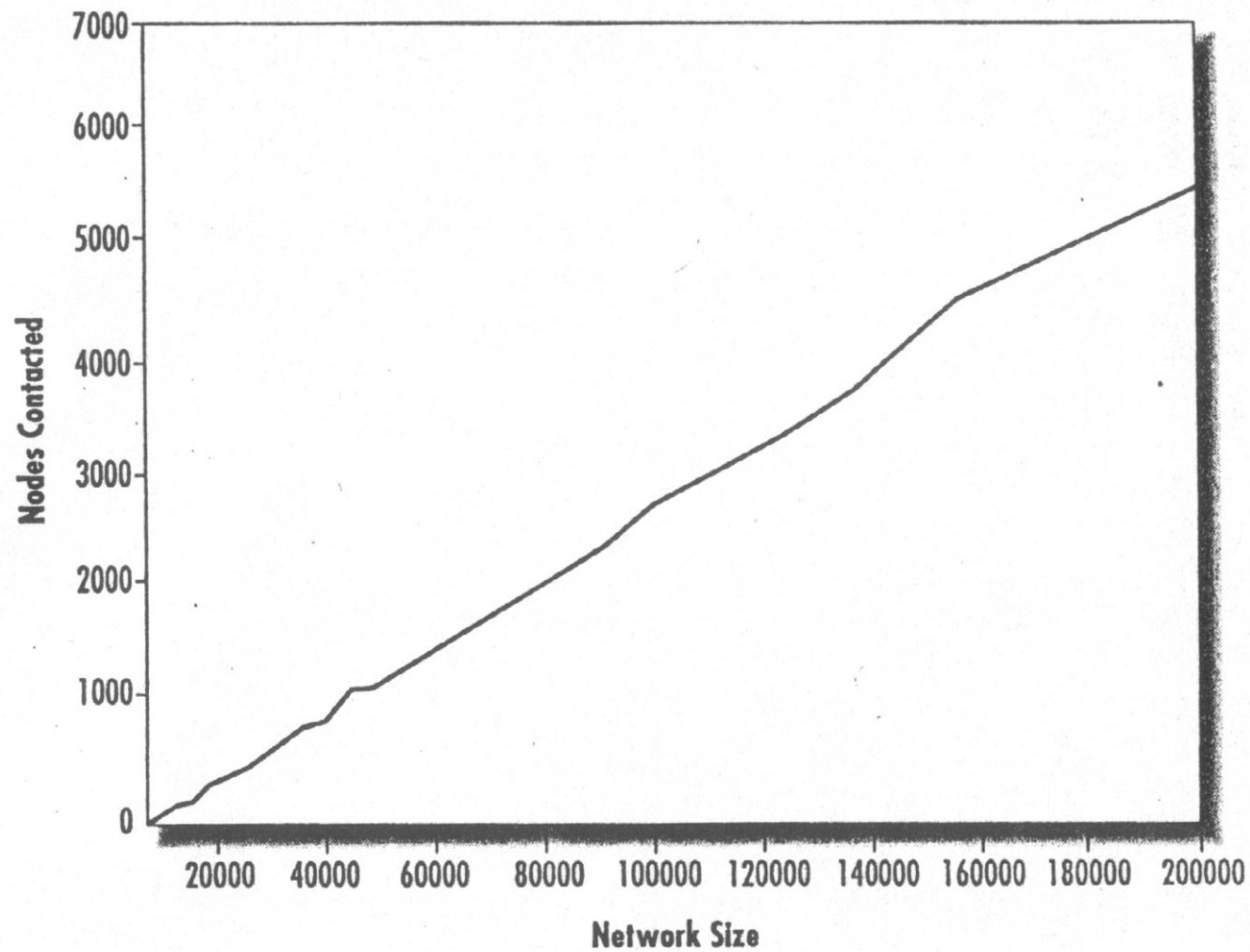
PONG (0x01)	
Port	2 Bytes
IP Address	4 Bytes
Nb. of shared Files	4 Bytes
Nb. of shared KBytes	4 Bytes

QUERY (0x80)	
Min Speed	2 Bytes
Search Criteria	n Bytes

QUERY HIT (0x81)	
Nb. of Hits	1 Byte
Port	2 Bytes
IP Address	4 Bytes
Speed	4 Bytes
Result Set:	
- File Index	4 Bytes
- File Size	4 Bytes
- File Name (+\0\0)	n Bytes
- ... (weitere Result.)	
Servent Identifier	16 Bytes

- Bandbreite für gesamt Netzwerk [www.darkridge.com]
 - N: Verbindungen, T: Laufzeit (TTL)

Bandbreitenauslastung bei 10 qps								
<i>N</i>	<i>T=1</i>	<i>T=2</i>	<i>T=3</i>	<i>T=4</i>	<i>T=5</i>	<i>T=6</i>	<i>T=7</i>	<i>T=8</i>
3	5.4KBps	21.6KBps	65.6KBps	176.2KBps	443.2KBps	1.1MBps	2.4MBps	5.8MBps
4	7.2KBps	39.8KBps	171.8KBps	670KBps	2.4MBps	8.8MBps	30.6MBps	104MBps
5	8.8KBps	63.4KBps	356.6KBps	1.8MBps	9MBps	42.2MBps	194.8MBps	882.6MBps
6	10.6KBps	92.2KBps	642.4KBps	4.2MBps	25MBps	146.8MBps	845.2MBps	4.8GBps
7	12.4KBps	126.8KBps	1.1MBps	8MBps	58.4MBps	412.4MBps	2.8GBps	19.2GBps
8	14.2KBps	166.6KBps	1.6MBps	14.2MBps	121MBps	995.4MBps	8GBps	63.4GBps



Quelle: [Oram, „Peer-to-Peer“]

- Messungen im Gnutella-Netz:
 - Verbindungen: durchschnittlich 4 Peers kontaktiert
 - 80 Byte im Durchschnitt pro Anfrage (plus Kopfdaten der TCP/IP-Schichten, ca. 40 Byte)
 - 10 Anfragen pro Sekunde ankommend
 - 33% Anteil an Eintreffenden Daten [www.cs.ucr.edu]
 - 23% Pings, 40% Pongs, 33% Query, 4% Query-Hits.
- Schlussfolgerung:
 - $10 \text{ Anfragen/Sekunde} * 80 \text{ Byte/Anfrage} * 100/33 * 3$
(Verbindungen) = 9600 Byte/s = 76,8 KBit/s
 - ISDN: 64 Kbit/Sekunde → Bandbreitenlimit

- Bei großer Teilnehmerzahl gibt es auch große Anzahl der Anfragen
- 100 Anfragen pro Sekunde ankommend
 - $4 * 100 * 80 * 3 = 96 \text{ KByte/s} = 768 \text{ KBit/s}$
- 1000 Anfragen pro Sekunde ankommend
 - $5 * 1000 * 80 * 4 = 960 \text{ KByte/s} = 7,68 \text{ MBit/s}$

Free Riding

- Mehr herunterladen, als sie selbst dem Netz zur Verfügung stellen
- Bietet zwar keine Daten an, kostet aber einen Hop
- Stichprobe von 38K Peers[Adar, Huberman, 2000]:
 - 1% der Peers beantworten 50% der Anfragen, und bieten 40% der Dateien an.
 - 70% der Nutzer keinerlei Ressourcen mitanderen teilen

- Nur mittelmäßig großes, komplett dezentrales P2P-Netz
- Bandbreiten wächst linear mit Teilnehmerzahl
- Ineffizienter Suchalgorithmus (Flooding)
- Hohe Netzwerklast

→ Gnutella ist nicht skalierbar!

- Einführung in P2P Systeme
- Warum Gnutella nicht skaliert
- Freenet



FREENET

Inhalt

- Geschichte / Motivation
 - Entstehung
 - Ziele von Freenet
- Funktionsweise
 - Das Netz und Anfragen auf dieses
 - Routing
 - Keys
 - Skalierbarkeit
- Zusammenfassung
 - Gnutella vs Freenet

Geschichte und Ziele

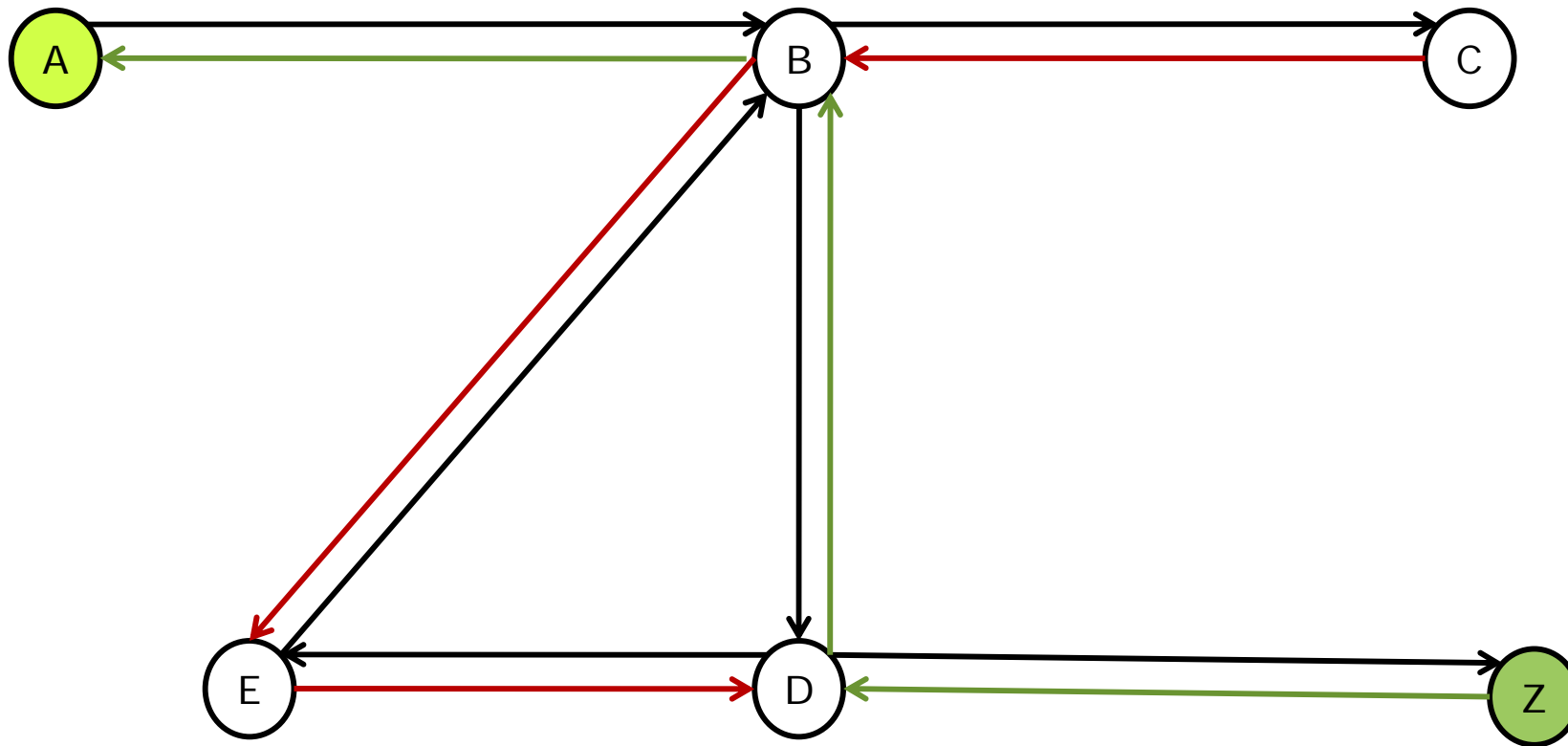
- Geschichte
 - Entwickelt von Ian Clarke zu seiner Studentenzeit (1999)
 - Heute: Community Projekt
 - Aktuelle Version: 0.7 (seit dem 08. Mai 2008)
- Ziele
 - Verhinderung von Zensur
 - Anonymität des Benutzers
 - Ausschluss einzelner Fehlerquellen
 - Effizientes Speichern / Verteilen von Dokumenten (= Informationen)
 - Node-Betreiber sollen Anschuldigungen plausibel abstreiten können

Inhalt (Wo sind wir gerade?)

- Geschichte / Motivation
 - Entstehung
 - Ziele von Freenet
- Funktionsweise
 - Das Netz und Anfragen auf dieses
 - Routing
 - Keys
 - Skalierbarkeit
- Zusammenfassung
 - Gnutella vs Freenet

Anfrage an das Netz

- Client = Server = Node



Anfragen (2)

- Caching
 - Verhinderung des Slashdot Effekts
- Kein Knoten weiß, woher die Ursprüngliche Nachricht kommt
 - Anonymität
- Jede Anfrage hat eine ID
- Einfüge-Anfragen = Sonderform von Suchanfragen

Routing

- „Small World Network“
- In neuen (Sub-)Netzen eher zufällig
- Schlüssel der gewünschten Datei berechnen
- Anfrage an den eigenen Knoten schicken
 - An den „geeignesten“ Knoten weiterleiten
- Kein Multi- / Unicast
- Ähnliche Schlüssel Clustern
- Informationen über die Leistung benachbarter Knoten

Der Freenet Stack

- ... ist eigentlich gar kein Stack
 - sondern eine Hashtabelle
- **Key** verweist auf
 - Ein spezielles Datum
 - Herkunft des Datums
- LRU Cache

KEY	DATA	ADDRESS
8e0109xb87wkhkujhs98k	99usbkjhgd7333khjgs763	tcp/5.34.27.4:6473
uushs89763kjhcx7w732722	yy625423lgsyw4GGcwhgs	tcp/89.34.36.1:24855
kjhks872228x0982876jhd	TTRos384hgygdvuybv1111n	tcp/194.44.62.66:9897
878772kx762776xbv8622		tcp/64.28.67.48:43653
222764kjhx8163wkbkjs77w		tcp/4.18.49.35:65466
57765xkjhd72729jnbck01kj		tcp/55.18.4.1:3895

Keys

- Sichern die Integrität einer Datei
- Repräsentiert durch Hashwerte
 - SHA-1
- 4 Arten von Schlüsseln
 - Keyword Signed Key (KSK)
 - Signed Subspace Key (SSK)
 - Content Hash Key (CHK)
 - Updateable Subspace Key (USK)
- Beispiel:
 - `CHK@fXzNaVTI6ymVnsZzIYWXRPODveIchxN3cUZX4NkSALQ,C9znIft~RgzQFtjNRbP-GWQZfnogJDHRKwZ2W5SD6rI,AAIC--8/80s Rock vol 3 - 05 - Big Country - In a Big Country.flac`
- Aufruf im Browser
 - `http://localhost:8888/[Freenet Key]`

Keyword Signed Key (KSK)

- **KSK@myfile.xyz**
- Vom User festgelegter (sprechender) String
 - Tolles_Video.avi
 - Schlüsselpaar aus diesem String erzeugen
- Integrität der Datei sichern
 - Datei mit privatem Schlüssel signieren
 - Zusätzlich noch mit dem KSK verschlüsseln
- Veröffentlichung des KSK reicht
- KSK sichert nicht die Eindeutigkeit der Datei
- Kann alternativ auch einen Verweis auf einen CHK haben

Signed Subspace Key (SSK)

- Zufälliges Schlüsselpaar
- Funktion eines Namespaces
- Anlegen einer neuen Datei -> eindeutiger Schlüssel
 - $KSK_e = h(KSK)$
 - $PubKey_e = h(PubKey)$
 - $Key = h(KSK_e \text{ XOR } PubKey_e)$
- Datei mit KSK verschlüsseln
- Veröffentlichung
 - Key
 - $PubKey_e$

Content Hash Key (CHK)

- $\text{CHK}@file\text{-hash, decryption-key, crypto-settings}$
- Hashwert der Datei
- Zusätzliche Verschlüsselung mit zufälligem Schlüssel K_f
- Datei wird veröffentlicht mit
 - ihrem Hashwert
 - ihrem Schlüssel K_f
- Anwendung
 - Update einer Datei
 - Splitten einer Datei

Bereitstellen von Daten

- Sonderform von Suchanfragen
- CHK der Datei berechnen
- Anfrage an sich selbst stellen
- Datei schon vorhanden
 - Datei wird als Antwort zurückgeschickt
 - Neuer Schlüssel muss berechnet werden
- Erfolgsfall
 - „Success“-Nachricht
 - Hochladen der Datei

Hinzufügen eines Knotens

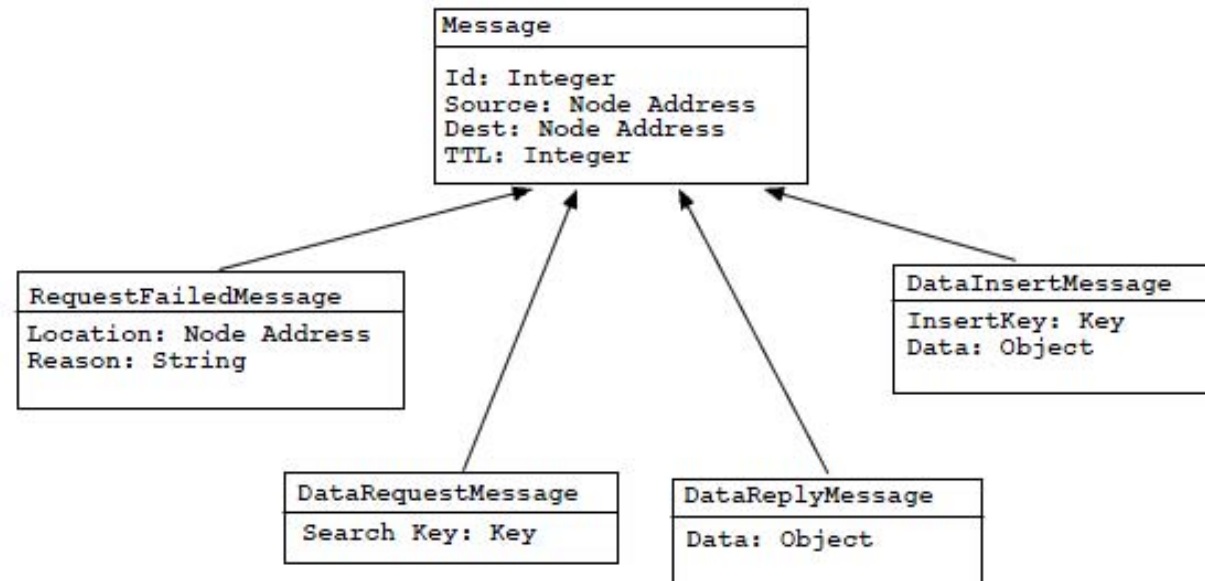
- Knoten N_0 muss mindestens einen anderen Knoten kennen
- k muss sich ankündigen
 - N_0 Wählt zufälligen Seed S
 - $Msg_0 = \text{Adresse} + h(S)$
 - N_0 schickt Nachricht an N_1
- N_1 empfängt die Nachricht
 - Wählt ebenfalls einen zufälligen Seed S_1
 - $Msg_1 = Msg_0 \text{ XOR } h(S_1)$
 - Msg_1 wird an zufälligen Knoten weiterverschickt
- Letzter Knoten erstellt nur einen zufälligen Seed

Protokoll Details (allgemein)

- „Selbstenthaltende Nachrichten“
- Jede Nachricht hat eine 64Bit ID + HTL
 - HTL = 0 --> Nachricht kann weiterverschickt werden
- Freie Wahl des Transportprotokolls
 - TCP, UDP, ...
- Knoten Adresse
 - <Protokoll>/<ip>:<port>
- Knoten haben oft keine feste IP
 - Address-Resolution Keys (ARK)

Protokoll Details (Verbindungen)

- Anfrage beginnt mit Request-Handshake
 - Beinhaltet die Antwortadresse des Senders
- Reply-Handshake
 - Protokollversion
- Handshakes werden gespeichert
 - Einige Stunden lang
 - Keine neuen Handshakes erforderlich
- Node startet Timer anhand des HTL-Wertes
 - Timeouts
 - Reply-Restart Nachrichten



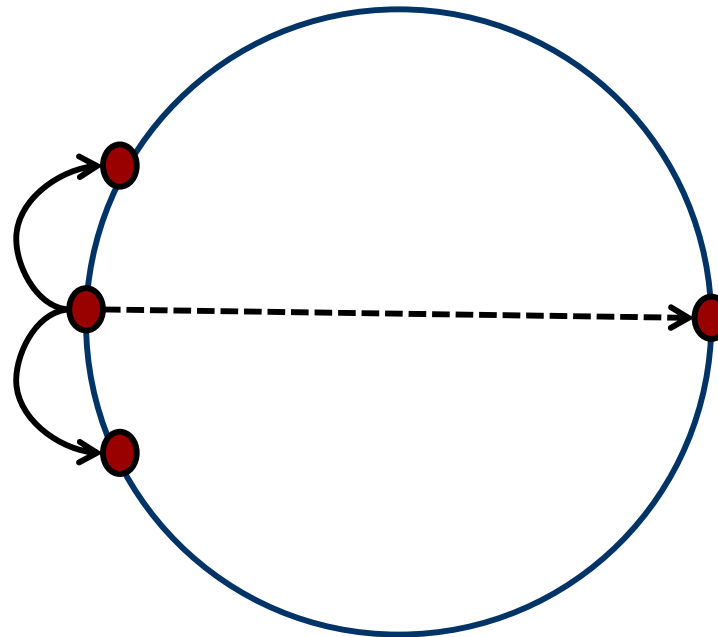
Quelle: [4]

Inhalt (Wo sind wir gerade?)

- Geschichte / Motivation
 - Entstehung
 - Ziele von Freenet
- Funktionsweise
 - Das Netz und Anfragen auf dieses
 - Routing
 - Keys
 - Skalierbarkeit
- Zusammenfassung
 - Gnutella vs Freenet

Leistung / Skalierbarkeit (1)

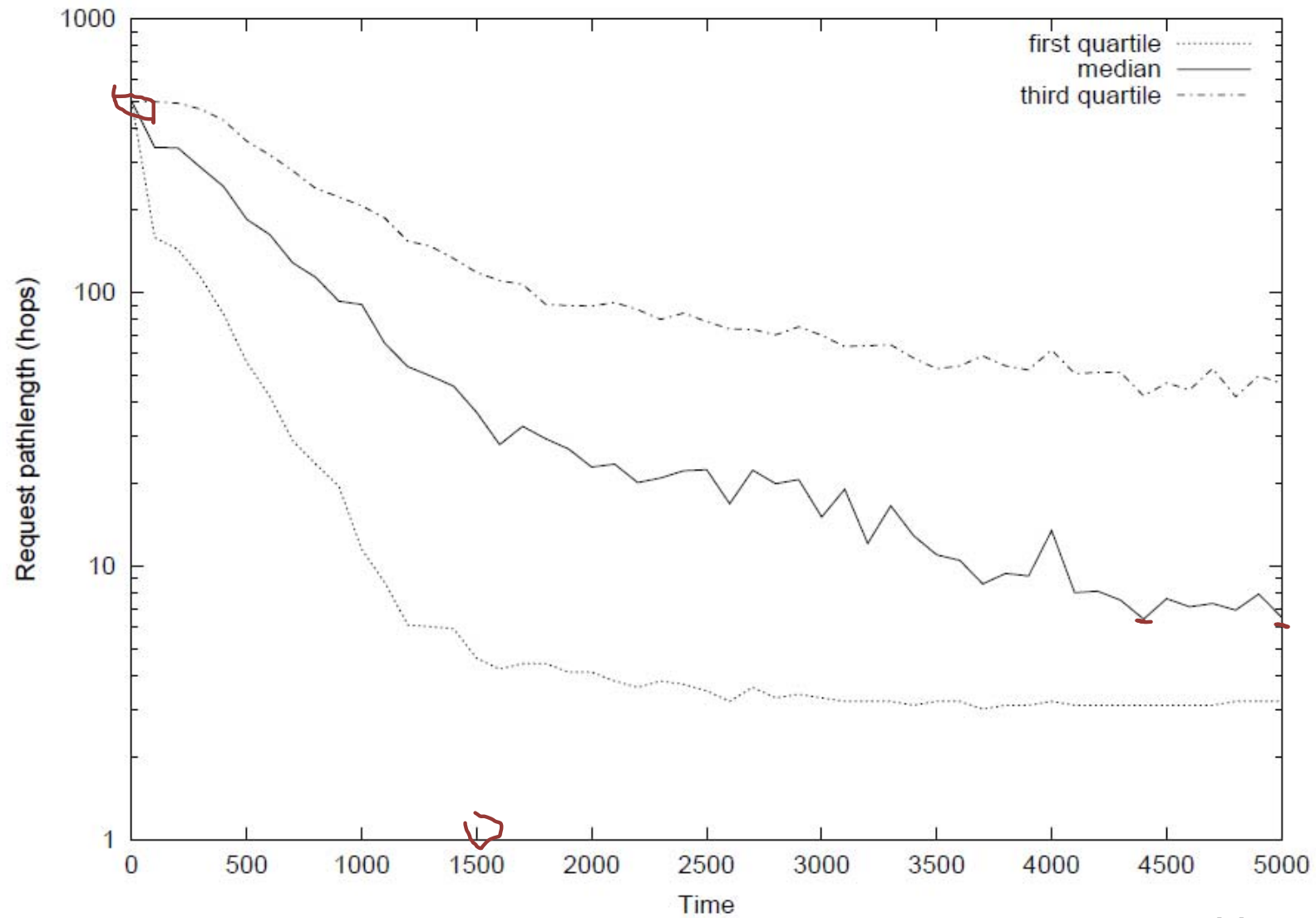
- Ergebnisse von Simulationen
- 1000 Knoten
 - Speichervermögen von 50 Dateien
 - Routingtabelle kann bis zu 250 Einträge speichern
- Ringförmig angeordnet
 - 2 Nachbarn



Leistung / Skalierbarkeit (2)

- Knoten wurden nach und nach eingefügt
- Zufällige Datei-Inserts; HTL: 20
- Messung der Leistung
 - Snapshots in festen Zeitintervallen
 - 300 Zufällige Anfragen; HTL: 500
 - Messung der mittleren Pfadlänge

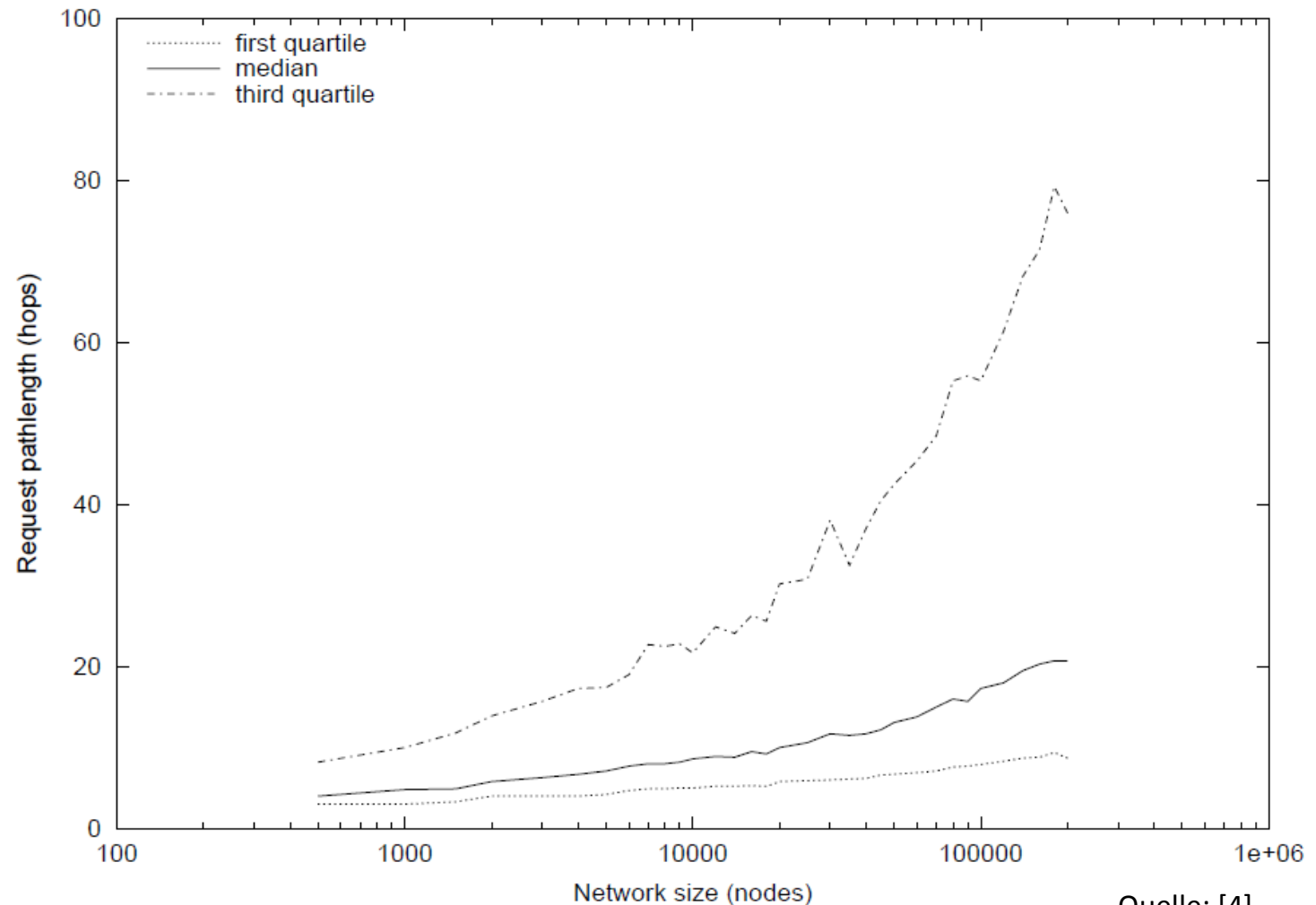
Veränderung der mittleren Pfadlänge



Quelle: [4]

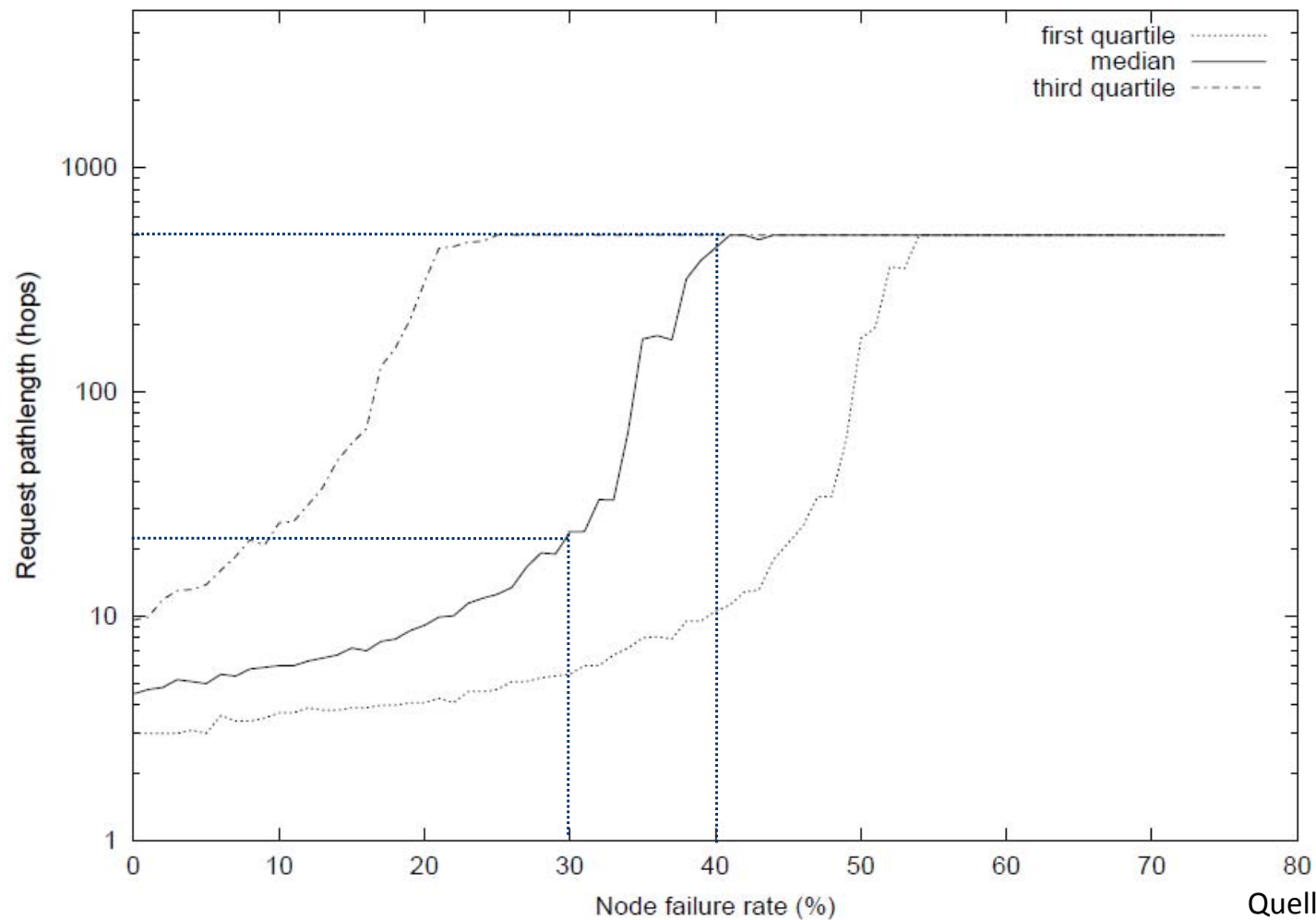
Anzahl der Hops in einem wachsenden Netz

- 20 Knoten
- Alle 5 Zeiteinheiten: neuer Knoten
 - HTL: 10



Quelle: [4]

- 1000 Knoten
 - Zufällige Löschung (= Ausfall) von Knoten



Quelle: [4]

Freenet = Small World

- Small World
 - Viele Knoten mit wenig Verbindungen
 - Wenig Knoten mit vielen Verbindungen
 - Effizientes Erreichen beliebiger Knoten durch Abkürzungen (Milgram)
- Dadurch
 - Gute Skalierbarkeit
 - Hohe Fehlertoleranz

Inhalt (Wo sind wir gerade?)

- Geschichte / Motivation
 - Entstehung
 - Ziele von Freenet
- Funktionsweise
 - Das Netz und Anfragen auf dieses
 - Routing
 - Keys
 - Skalierbarkeit
- Zusammenfassung
 - Gnutella vs Freenet

Gnutella vs Freenet

- Gemeinsamkeit(en)
 - Dezentralisiertes Peer-to-Peer Netzwerk
- Unterschiede

Gnutella	Freenet
Routing durch „Fluten“	Dynamisches Routing
Read-only Netzwerk	Daten werden auf Knoten verteilt
Keine Sicherheitsaspekte	Verschlüsselung von Daten
	Skaliert besser

Quellen (Teil 1: Einführung)

- Oram (Editor), Peer-to-Peer - Harnessing the Power of Disruptive Technologies, Oreilly, 2001
- Ralf Steinmetz, Klaus Wehrle, Peer-to-Peer System and Applications, Springer, 2005
- Cai Ziegler: Smarte Schwärme. Die Technik hinter modernen Peer-To-Peer-Netzen. In: c't. Heise-Verlag, Hannover 16.2005,21, S.160–164
- Detlef Schoder, Kai Fischbach, Rene Teichmann: Peer-to-Peer. Springer, Berlin 2002

Quellen (Teil 2: Gnutella)

- Oram (Editor), Peer-to-Peer - Harnessing the Power of Disruptive Technologies, Oreilly, 2001
- Ralf Steinmetz, Klaus Wehrle, Peer-to-Peer System and Applications, Springer, 2005
- <http://www.darkridge.com/~jpr5/doc/gnutella.html>, gesehen 07.11.2004
- <http://de.wikipedia.org/wiki/Gnutella>, gesehen 07.11.2004
- <http://ntrg.cs.tcd.ie/undergrad/4ba2.02/p2p/>, gesehen 07.11.2004
- <http://www.dfn.de/fileadmin/3Beratung/Betriebstagungen/bt40/winforum-bt40.pdf>
- <http://www.cs.ucr.edu/~csyiazti/courses/cs204/project/gnuDC.pdf>

Quellen (Teil 3: Freenet)

- [1] Andy Oram, Peer to peer: *Harnessing the Power of Disruptive Technologies*, First Edition March 2001, ISBN: 0-596-00110-X
- [2] Ian Clarke, Oskar Sandberg, Brandon Wiley, Theodore W. Hong: *Freenet: A Distributed Anonymous Information Storage and Retrieval System*, Paper, 1999
- [3] freenetproject.org, Stand: 07. November 2008
- [4] Ian Clarke: *A Distributed Decentralized Information Storage and Retrieval System*, Report, 1999, Division of Informatics University of Edinburgh



Vielen Dank!