Peer to Peer - Technologien

Chaos-Seminar des CCC Ulm

Stefan Schlott <stefan.schlott@ulm.ccc.de>

Inhalt

- Definition, Beispiel
- Allgemeine Fragestellungen in P2P
- Anwendungen, Ausblick

Einleitung

Was war/nannte sich P2P – Die Geschichte

- Usenet-Server-Verbund; Unix „talk“
- 1996: ICQ – Instant Messenging zw. Clients
- 1999: Napster
- 2000: Gnutella
- 2001: FastTrack, KaZaA, eDonkey

Einleitung

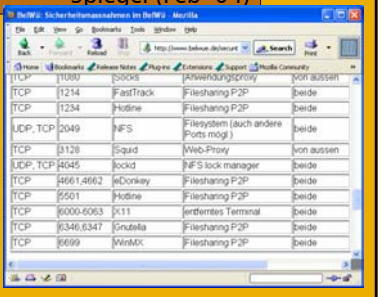
P2P bei Industrie und Presse

„Wer hat Schuld an P2P?“
Spiegel (Feb ’04)

Die Beklagten erlösen Millionen von Dollar über ihre Tauschplattformen, indem sie Werbung verkaufen, die den Anwendern angezeigt wird, während diese Gesetze brechen.

Cary Sherman, RIAA (Aug ’03)

Die Strategie der Industrie:
Nur der Hammer hilft
Spiegel (Feb ’04)



P2P dient quasi nur zum verbreiten Illegalen Materials und Austausch copyright-geschützter Medien.

Einleitung

- Will ich so etwas benutzen?
- „P2P“ an sich wert-neutrale Technik
- Geeignet für spezielle Szenarien

Definition

- Keine eindeutige Definition
- Sehr generisch:
  - Dynamisches An- und Abmelden aller Teilnehmer
  - Peers haben hohen Grad an Autonomie, Gleichberechtigung
- Technisch:
  - Selbstorganisierend
  - Ohne zentrale Serverkomponente

Definition

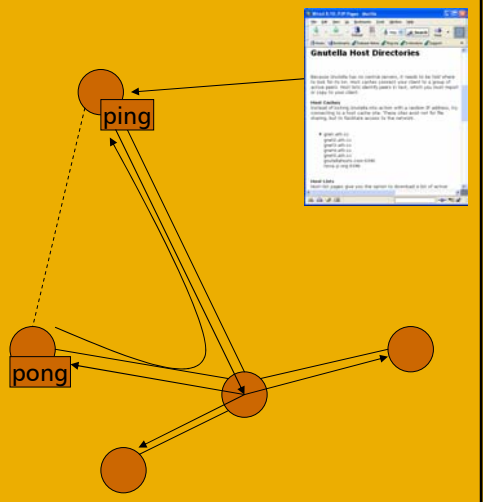
- Semi- / Quasi-P2P-Netze
  - Großteil der eigentlichen Funktionalität liegt bei den Clients
  - Aber: Zentrale Serverkomponente nötig
  - Beispiel: Napster
- „Echte“ P2P-Netze
  - Vollkommen dezentraler Aufbau
  - Beispiel: Gnutella

Beispiel: Gnutella

- Im März 2000 in nur 14 Tagen entwickelt
- Auf Nullsoft-Homepage zum download
- Verbreitung durch Slashdot-Effekt
- AOL/Time Warner (Inhaber von Nullsoft) verbietet Verbreitung
- Brian Mayland entschlüsselt Protokoll, Hilfe durch anonymen IRC-Chatter

Beispiel: Gnutella

- Einstieg: IP aus #gnutella oder Web-seite
- Gewünscht: Ständige Verbindung mit 4 Peers
- Suche weiterer Servents mit ping/pong
- Hinweg: Fluten
- Rückweg: ID des Ping-Pakets merken
- Routing Loops mit IDs erkennen

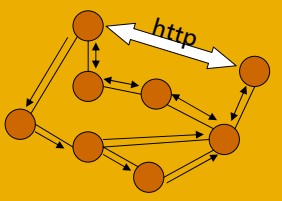


Beispiel: Gnutella

- Jedes Gnutella-Paket („Deskriptor") besitzt
  - Eindeutige ID
  - TTL (hop count)
  - Payload descriptor = Kommando (ping, pong, query, queryhit, push)
- Fluten: ping und query
- Servent merkt sich für jede ID die Verbindung, über die es empfangen wurde
  - Routing loops erkennen
  - Rückweg für Antworten
- Jeder Hop dekrementiert TTL.
  - TTL=0: Paket verwerfen.

Beispiel: Gnutella

- Suche: Analog zu ping
- Fluten des Netzes, Begrenzung mit TTL, Antworten über Rückweg
- Query: Beliebiger String, Interpretation nicht spezifiziert
- Download: Direkt, über http



Beispiel: Gnutella

Gut für den ersten Versuch, aber:

- Gigantische Verschwendung von Bandbreite (ständiges Fluten, pings)
- Skaliert schlecht
- Verhalten von query nicht standardisiert
- Kein Suchen identischer Dateien
- Weiterentwicklung: Supernodes

Problembereiche in P2P-Netzen

- Einstieg ins Netz
- Fluten
- Skalierende Suchfunktion
- 1:1-Kommunikation

…und Details wie

- Firewalls
- Anonymität
- Freeriders

Einstieg ins Netz

- Mundpropaganda, manuelle Eingabe
- Suche in Broadcast-Domäne
  - Ausreichend für lokale Anwendungen wie z.B. Groove für Abteilungen
- Webseite (statisch/dynamisch)
  - Statisch: Von Hand gepflegt
  - Anmeldung über Webformular
  - Automatisches Registrieren
  - Publizieren von Host caches

Einstieg ins Netz

- IRC-Channel
  - Manuell: Frühzeiten von gnutella
  - Automatisch: à la BotNets
- DynDNS host entries
  - Eine Reihe Adressen reservieren
  - Diese werden als „Erstkontakt“ genutzt
  - Update beim Ausloggen oder nach längerer Uptime

Einstieg ins Netz

…oder weiterer kreativer Mißbrauch anderer Dienste

- „Stiller Briefkasten“: E-Mail-Box(en) mit Mails m. eigener IP
- Eintrag in Gnutella Host Cache mit bestimmtem Port

Fluten

- Nur wenn unbedingt nötig!

- Nachricht an alle Knoten
- Reichweitenbegrenzung mit TTL
- Loops vermeiden
  - Je Nachricht eindeutige ID
  - IDs gesehener Pakete best. Zeit merken
  - Bereits gesehene Pakete verwerfen

Suchen

- Typische Suchanfragen
  - Titel/Dateiname (Substring)
  - Format, Bitrate, Länge (Metainformationen)
  - Identische Exemplare (File hash)
- Triviallösung: Fluten
  - Hoher Netzwerktraffic
  - Lange Antwortzeit
  - Wegen TTL: Evtl. nicht alle Treffer
- Triviallösung: Indexserver
  - Single point of failure

Suchen

Anregung aus normalen Datenbanken

- Lineares Suchen → Flooding
- Sortierte Daten, binäre Suche → ?
- Hash tables → ?

Übertragung auf verteilte Systeme:
Distributed Hash Tables (DHTs)

Suchen

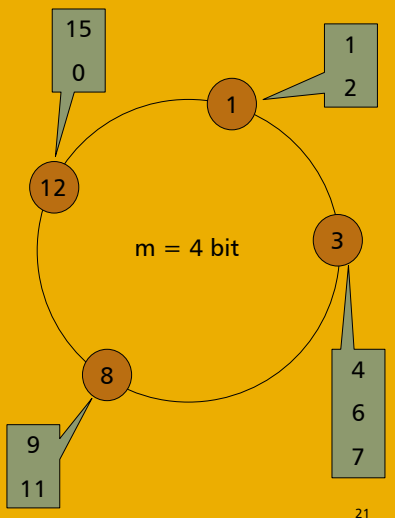
Beispiel: Chord

- Bietet Funktion lookup(key) → nodenr.
- Key und Knotennr.: Schlüssellänge m bit
  - Selber Raum für Schlüssel und Knoten!
  - Typischerweise mit SHA1 erzeugt
  - Kollisionen vermeiden: m groß genug wählen
- Eigene Knotennummer: Aus IP abgeleitet
  - Vorteil: Verfälschen nicht möglich
  - Nachteil: Brute force angreifbar

Suchen

- Bedingung für Funktion: Jeder Knoten kennt seinen Nachfolger → Ring
- Jeder Knoten hält Daten, für die gilt:

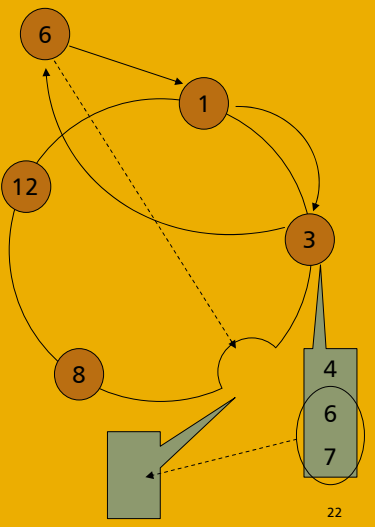
nodenr ≤ key < succ(nodenr)



Suchen

Neuer Knoten:

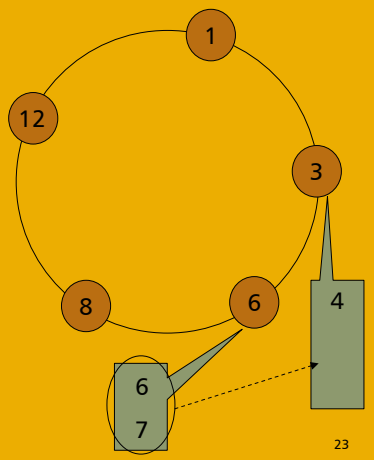
- Sich selbst suchen → eigene Position
- Vom so gefundenen Vorgänger Daten übernehmen
- Ursprünglichen Nachfolger erfragen, sich selbst als Nachfolger eintragen



Suchen

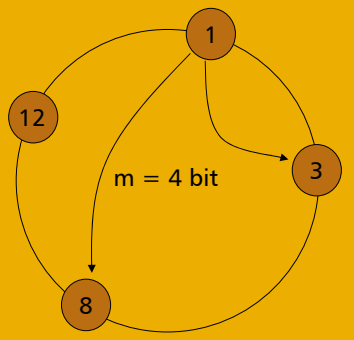
Netz verlassen

- Nach nodeid-1 suchen → Vorgänger
- Daten abgeben
- Eigenen Nachfolger mitteilen
- Netz verlassen
- Erhalten des Rings bei Fehler: Nächsten n Nachfolger speichern



Suchen

- Bis jetzt: Lineare Suche: O(*n*)
- „Finger table“: Adressen der Knoten nodeid+2^i (i=1..m) speichern
- „Abkürzungen“ durch Kreis
- Suche somit wie Binärsuche: O(*log n*)



Finger table von Knoten 1

2	3	5	9
1	3	3	8

Suchen

- Chord bietet effiziente Suche bei geringem Overhead
- Korrektheit mathematisch beweisbar
- Erweiterungen, z.B. …
  - achord (Zensurresistenz)
  - hyperchord (Ring in „beide Richtungen“)
- Wird verwendet z.B. in Tarzan und CFS (cooperative file system)

Suchen

Weitere Varianten (u.a.):

- Kademlia
  - Binärbaum
  - eMule nutzt Kademlia-Netz als „Zweitnetz“ zu eDonkey-Servern
- CAN (content addressable network)
  - d-dimensionales Koordinatensystem auf einem d-Torus (d Löcher)

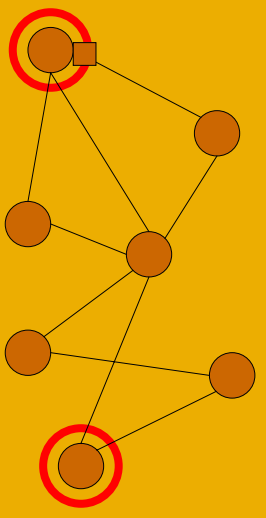
1:1-Kommunikation

Punkt-zu-Punkt-Kommunikation

- Brauchen fast alle Anwendungen :-)
  - File download
  - Chat
  - …
- Triviallösung: Direkte Verbindung
  - Keine Anonymität von Sender u. Empfänger
  - Problematisch wenn beide hinter Firewall

1:1-Kommunikation

- Alternative: Routing durchs P2P-Netz
- Einzelner Knoten ohne Firewall dient als Relay
- Routing durch das P2P-Netz
- Mögliche Optimierung: Nur einige Hops im P2P-Netz



1:1-Kommunikation

Mögliche Routingverfahren:

- In Chord: Adressierung mit Node-IDs, Routing über Finger tables
- Separates Routingprotokoll: AdHoc-Protokolle!
  - Reaktiv, z.B. AODV
  - Proaktiv, z.B. OLSR
- Problem i.d. Praxis: ADSL…

1:1-Kommunikation

Beispiel: Mute

- Relativ neu (ca. 07‘2003)
- Schwerpunkt Anonymität
- Idee: Ant routing

MUTE protects your privacy by avoiding direct connections with your sharing partners in the network. Most other file sharing programs use direct connections to download or upload, making your identity available to spies from the RIAA and other unscrupulous organizations.

(mute.sf.net)

1:1-Kommunikation

Verbindung zwischen zwei Nodes

- Jede Node: RSA-Key
- Beim Verbindungsaufbau
  - Eigenen Public Key senden
  - Auf Key warten
  - AES-Key generieren, mit Public Key des Gegenüber verschlüsseln und senden
  - Auf AES-Key für Empfang warten, diesen entschlüsseln
  - → Verschiedene Keys für jede Richtung

1:1-Kommunikation

…nach dem Verbindungsaufbau:

- Austausch der Host List:
  - Eigene „unique ID" (virtual address), eigener Listen Port
  - Liste bekannter Mute-Hosts (IP+Port)
  - Connection: Accepted/Rejected
- Danach: Mute messages
  - Message ID, Flags
  - From, To (virtual addresses)
  - Utility counter: Gibt grobe Anzahl an, wieviel Last (Nachrichten) dieses Paket erzeugt hat

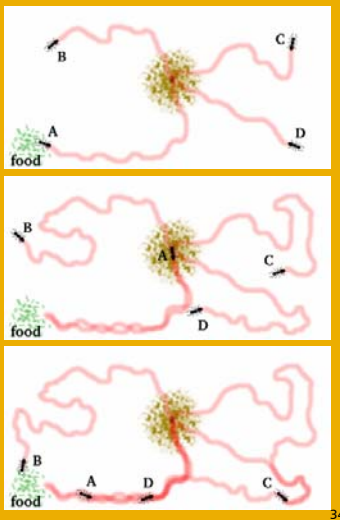
1:1-Kommunikation

Ant routing

- Ursprüngliche Idee Mitte der 90er
  - "Ant-based load balancing in telecommunications networks"
  - "Routing With Swarm Intelligence" (1997)
- Ameisen benutzen Pheromone zur Orientierung
- Pheromone „verduften“ langsam

1:1-Kommunikation

- Ameisen hinterlassen Duftspur → Rückweg
- Futter finden: Zurück zum Nest, Spur wird stärker
- Beim Treffen auf andere Spur: Stärkerem Duft folgen
- Kurze Wege → Schnell → Stärkerer Duft



1:1-Kommunikation

Ant Routing in Mute - Routingtabelle

- Jede Node hält eine duplikatfreie Queue mit den N zuletzt gesehenen (Absender-)adressen
  - Absenderadresse: Virtual address
  - Adresse benutzen → top of queue
- Pro Adresse: Queue (nicht duplikatfrei) mit M zuletzt benutzten Channels (ergibt Mapping: Zieladresse → Next Hop)
  - Channel: IP, Port, Socket, Schlüssel
  - Channel benutzen → Oben erneut einfügen
- Empfang einer Nachricht: Channel in Liste der entspr. Virtual Address speichern

1:1-Kommunikation

Ant Routing in Mute - Routingvorgang

- Ähnlich zur Suche in Gnutella
- Weiterleiten
  - Zieladresse bekannt → Zufällig einen der vorh. Channels wählen

    Mit kl. Wahrscheinlichkeit: Irgendeinen der bek. Channels wählen
  - Sonst: Broadcast der Nachricht an alle Nachbarn
- Gegen Abhören: Hop-by-Hop encryption

1:1-Kommunikation

Mögliche Angriffe auf Mute:

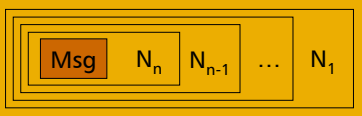
- Man in the middle beim Key exchange
- Host list poisoning
  - Liste d. Einstiegsknoten manipulieren
  - Rechner um observierte Node plazieren
- Timing attack

→ „Mute protects your privacy“, nicht „Mute guarantees your privacy“

1:1-Kommunikation

Beispiel: Tarzan (gaaanz kurz)

- Generisches System für IP forwarding
  - Sender bestimmt Route, baut Tunnel auf
  - Tunnelendpunkt: NAT
- Anonymität mittels Onion Routing:
  - Voraussetzung: Schlüssel für alle Hops
  - Für alle Nodes $N_{n-1}..N_1$: Nachricht=Enc_i(Nachricht,N_{i+1})



Firewalls

- Grenzen überwinden – wie?
- „Push“-Requests
  - Verbindungen von innen aufbauen und offenlassen
- „Relay“-Requests
  - Knoten ohne Firewall dient als Vermittler
  - Vermittler sendet „Push“ an beide Knoten, leitet dann Daten weiter

Freeriding

- „Trittbrettfahrer“ – Problem für kooperative Dienste
- „Gnutella-Studie“
  - 15% der Teilnehmer bieten 94% des Contents
  - 63% halten keine (interessanten) Daten
- Bewertungsmöglichkeiten:
  - Zentrale Kontrolle
  - Gegenseitiges Rating
- Problem: Identifizerbarkeit
  - Nur innerhalb einer Session
    → Gutes „Karma“ nicht erhaltbar
  - Übertragbar → Benutzer identifizierbar

Freeriding

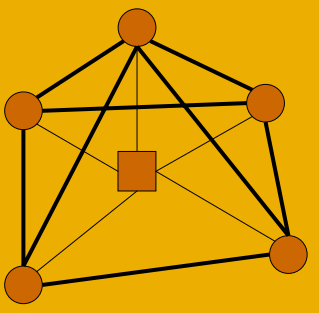
Beispiel: eDonkey

- Gutes Sharing-Verhalten erlaubt mehr parallele Downloads
- Client-side security
  - Gepatchte Clients: Immer bestes Rating

Freeriding

Beispiel: bittorrent

- Verbreitung von Dateien
- Lastverteilung
- enforced fairness: „Tit for tat“
  - Zielt auf pareto-effizientes Gleichgewicht
  - Zentrale Kontrolle (tracker)
  - Rating nur für 1 Sitzung



Freeriding

bittorrent - Up/Download

- Verbindung zu tracker über torrent-File
  - Adresse und Port von Tracker
  - Länge der Datei
  - Hashes für Blöcke der Datei
- Melden, welche Blöcke bereits vorhanden
- Tracker meldet Liste von Peers
- Download der Blöcke („pieces“) direkt

Freeriding

- Piece selection
  - Erstes piece: Zufällig
  - → Möglichst schnelle Streuung des Orginals
  - Weitere pieces: „rarest first“
  - → Möglichst lang eine (verteilte) Kopie erhalten
- Rückmeldung über up/downloads an Tracker
- Drosselung des Uploads entsprechend Infos vom Tracker („choking“)
  - Wenig upload → gedrosselter Download
  - „Optimistic unchoking“: Einige Verbindungen probehalber öffnen

Freeriding

- Gute Lastverteilung gleichzeitiger Downloads
- Aktuelles Protokoll nicht 100%ig „wasserdicht“ (gepatchte Clients)
- Anreize für gleichzeitigen Zugriff:
  - Initialer Seed nur für best. Zeit verfügbar (ankündigen)
  - Torrents über RDF-Feeds ankündigen
  - → Spezielle Clients starten automatisch download

Anwendungen

- Filesharing
- Zensurresistentes Publishing
- Organizer, dezentrale Groupware
- Overlay-Netz, dynam. Brücke zw. Inselanwendungen (z.B. IPv6-Overlay mittels P2P)
- Kommunikation: Chat, Messaging, Telephonie, Streaming

Betrachtung als Overlay-Netz

- Broadcast = Fluten
- Multicast = Eigentlich auch Fluten: Alle Interessenten („Multicast-Gruppe“) entsprechen den Netzteilnehmern
- Unicast = Routing durchs Netz
- Service discovery = Suchen z.B. mit DHT

Positives Szenario

- Gemischtes Netzprotokoll: Native wo möglich, sonst mittels P2P überbrücken
- Ein Netz für verschiedene Anwendungen
- Z.B. IPv6: Netzbereich fürs P2P-Netz
- Anwendungen nutzen normale IPv6-Sockets
- Vorteil: Transparenz, keine „Neuimplementierung“ von TCP
- Im Kernel: Entscheidung, ob native weitergeleitet wird – oder „umverpackt“ wird (dann weiter z.B. in bel. Tunnelprotokoll).

Negatives Szenario

- Wurm nutzt P2P zur Kommunikation
- Multiple Infektionswege, z.B. Mailwurm und Webserver-Exploit
- Neuinfektion bekommt Peerlist des „Vaters“ mit
- Gehackte Webserver dienen als Publikationsmedium für andere Peers
- Über P2P-Netz werden alternative Einstiegspunkte publiziert
- „Master of desaster“ ist einer von vielen Infizierten; in der Masse kaum auszumachen

Peer to Peer - Technologien

- „P2P“ an sich wertneutrale Technik
- Geeignet für spezielle Szenarien
- …die Benutzung entscheidet, ob P2P „gut“ oder „schlecht“ ist

Schönen Abend noch!