Hochverteilte Datenhaltung im Internet

Fachseminar Verteilte Systeme
11. Juni 2002, Nicolas Burri

Betreuung: M. Langheinrich
Seminar-Leitung: Prof. F. Mattern

Um was geht es?

- **Napster?**
- **Gnutella?** **Nur am Rand!**
- **Edonkey2000?**

- **Interessanter:**
  Hochverteilte Datenhaltungs-Systeme

Ablauf

- Kurzvergleich P2P ⟷ Hochverteilte Systeme
- Erwünschte Eigenschaften
- Potentielle Ansätze
- Eine mögliche Zukunft
- Diskussion

P2P vs. Hochverteilte Systeme

- Peer-To-Peer Netzte (P2P)
  - Hierarchisch oder völlig dezentral
  - Passive Verbreitung von Inhalten
  - Kurze Planung, einfach implementiert
  - Heute weit verbreitet
- Hochverteilte Datenhaltungs-Systeme
  - Hierarchisch oder völlig dezentral
  - Aktives Einfügen von Inhalten ins Netz
  - Wissenschaftliche Planung und Analyse
  - Noch nicht im grossen Rahmen eingesetzt

Erwünschte Eigenschaften

- Hohe Verfügbarkeit / Skalierbarkeit
- Einfaches Speichern von Daten
- Datenkonsistenz
- Schnelles, effizientes Auffinden von Daten
- Anonymität von Autor, Server und Client
- Verschlüsselte Kommunikation
- Zensurresistenz

Erwünschte Eigenschaften

- Hohe Verfügbarkeit / Skalierbarkeit
- Einfaches Speichern von Daten
- Datenkonsistenz
- Schnelles, effizientes Auffinden von Daten
- Anonymität von Autor, Server und Client
- Verschlüsselte Kommunikation
- Zensurresistenz

Einfügen und Speichern

- **Primitiver Ansatz:**
  - **Komplette Datenspiegelung auf alle Server**
- **1. Verbesserungsidee:**
  - **Daten geschickt fragmentieren und auf Sub-Set der Server redundant verteilen**

  Aber: Problematisch für Suche

Speichern Forts.

- **2. Verbesserungsidee:**
  - **ID für Daten berechnen z.B. mittels Hash aus**
    - **Inhalt**
    - **AutorenID**
    - **Zugehörigen Suchworten**
  - **Bestimmen eines Sub-Sets von Servern aus dem Hashwert**

Retrieval

- Bei deterministischer Verteilung der Daten einfach
- Ansonsten sehr schwierig, da Routing der Suchanfrage komplex

Deshalb:
Probabilistisch errechnen wo Daten am wahrscheinlichsten zu finden sind

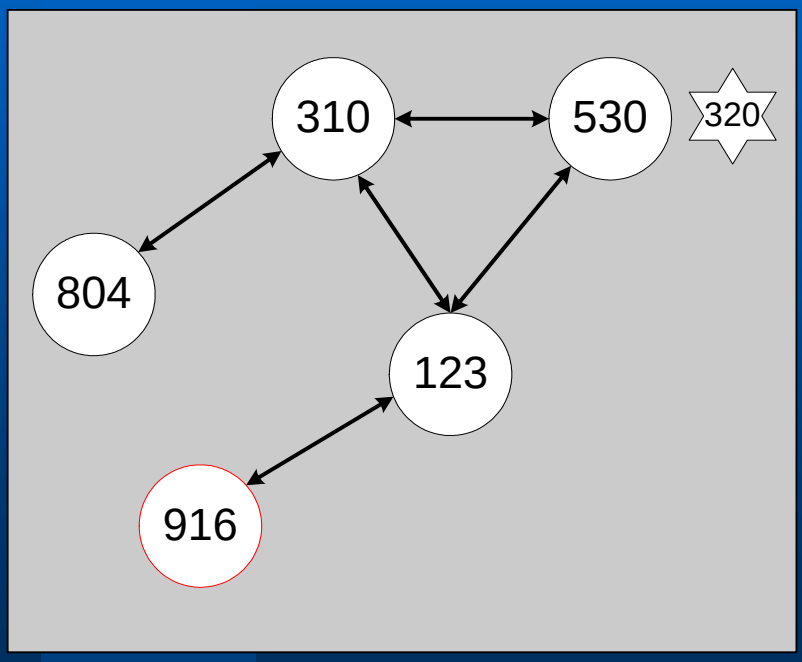
Ansatz - Beispiel

- Alle Knoten sind gleichberechtigt
- Jeder Knoten hat eine eindeutige ID
- Jeder Knoten besitzt eigenen Speicher
- Jeder Knoten führt eine Tabelle über ihm bekannte Knoten (ID) und deren IP-Adressen

Ansatz - Beispiel Forts.

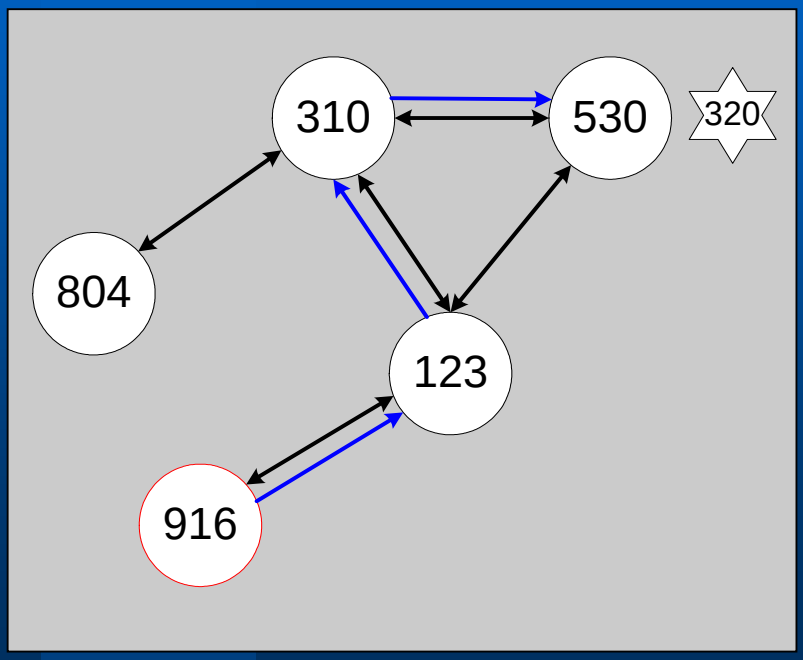
- Daten haben eine ID mit demselben Format wie Knoten
- DatenID berechnet aus Hash von
  - AutorenID
  - Suchstring der Daten

Retrieval



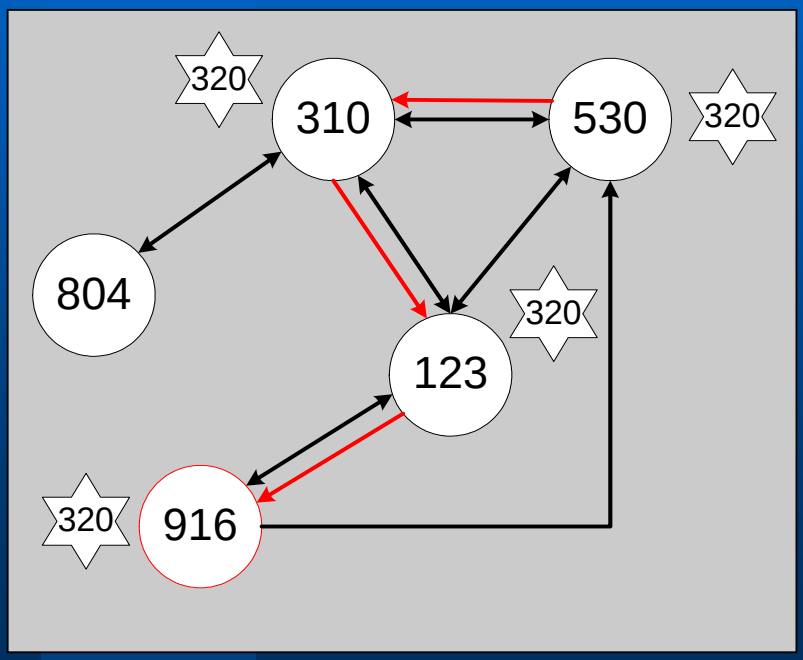
- Knoten 916 will Datensatz 320
- Datensatz 320 gespeichert in Knoten 530

Retrieval - Request



- Jeder Knoten leitet die Frage an den Nachbarn mit der ID die am Nächsten zu 320 ist

Retrieval - Reply

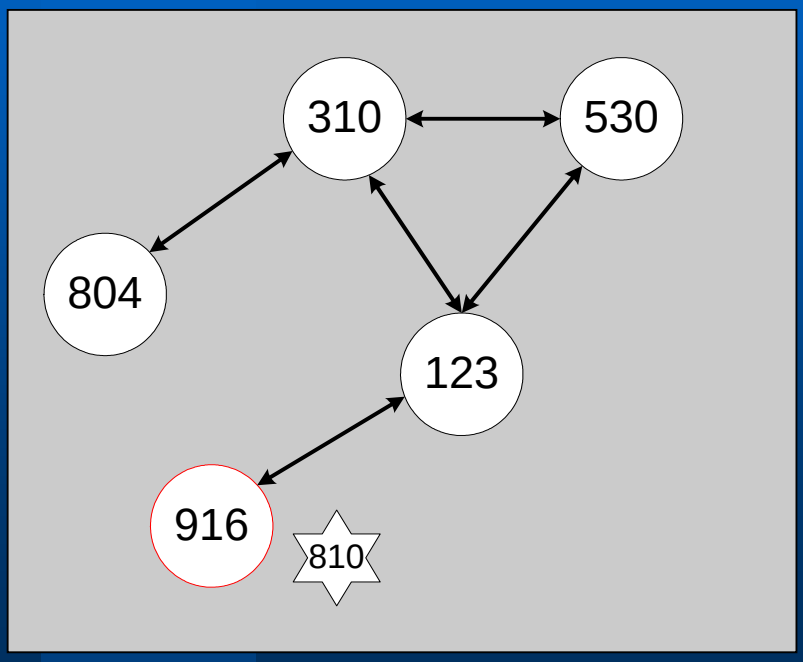


- Datenpaket 320 wird entlang dem Suchpfad zurück-geschickt und in jedem Knoten dupliziert
- Jeder Knoten fügt 530 in die Liste seiner bekannten Knoten ein

Vorteile dieses Systems

- **Routing verbessert sich mit der Zeit**
  - **Knoten werden zu „Spezialisten“ für Sets von IDs die ihrer eigenen ID nahe sind**
    - **Jeder Knoten speichert hauptsächlich Daten mit IDs die ähnlich zur KnotenID sind**
    - **Jeder Knoten kennt hauptsächlich andere Knoten mit ähnlichen IDs**
  - **Populäre Daten werden vermehrt repliziert**

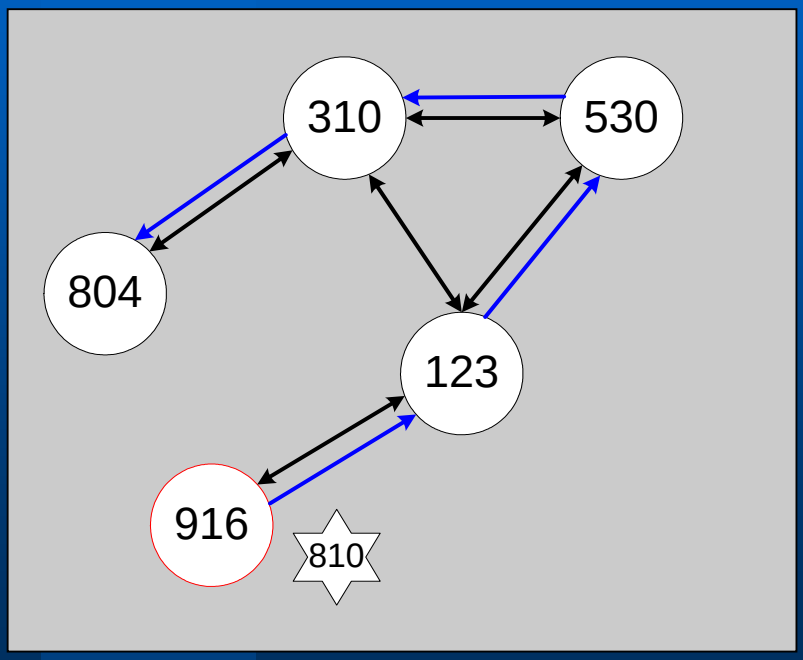
Einfügen und Speichern



- **Knoten 916 will Datenpaket 810 einfügen**
- **Datenpaket 810 soll fünffach repliziert gespeichert werden**

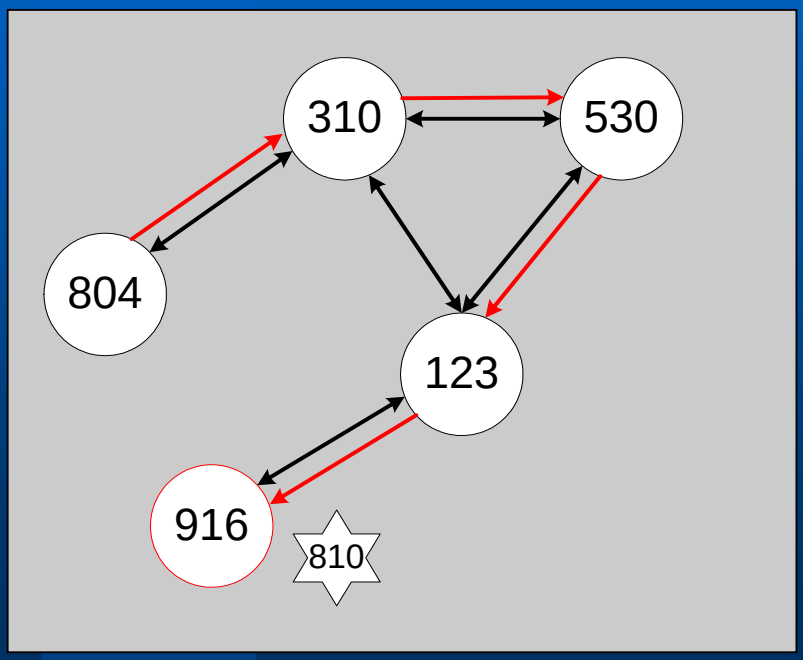
  - Hops-to-live

Einfüge - Request



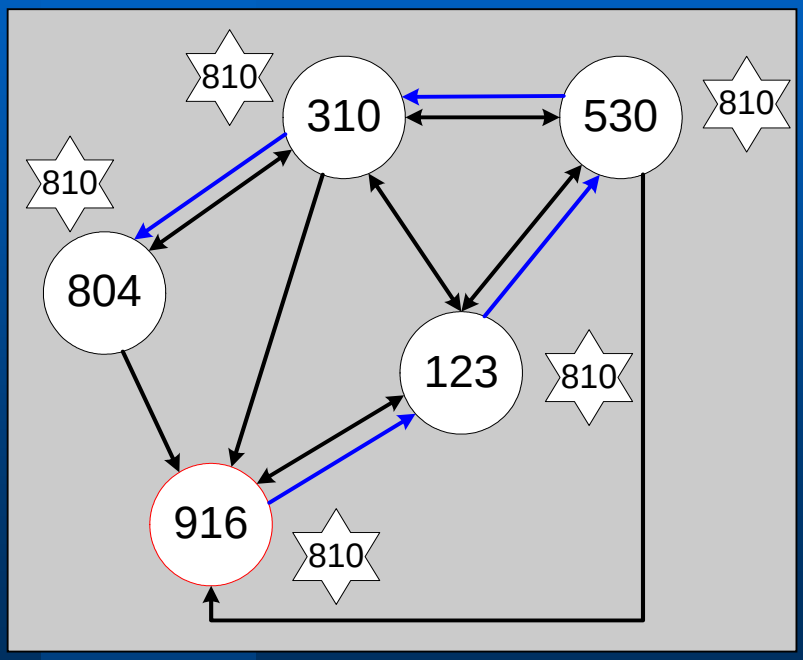
- 916 startet Insert Request für Schlüssel 810 analog zu einer Suchanfrage.
- Bei Kollision wird der existierende Datenblock zurück-geschickt, wie bei einer Suchanfrage

Einfüge - Reply



- Falls keine Kolli-sion auftritt bis Hops-to-live auf Null, dann OK-Message zurückschicken

Daten einfügen



- Falls 916 eine OK-Message bekommt sendet er das Datenpaket auf demselben Weg
- Jeder Knoten fügt 916 in die Liste seiner bekannten Knoten ein

Vorteile dieses Systems

- Neue Daten werden in Knoten gespeichert, die schon auf ähnliche IDs spezialisiert sind
- Durch das Einfügen neuer Daten, wird ein Knoten im Netz bekannter
- Ein Versuch falsche Daten unter einer belegten ID einzufügen, führt zu einer Reproduktion der echten Daten

Wie gut funktioniert das?

- Freenet basiert auf diesen Ansätzen
  - Open Source Implementierung eines Systems, das 1999 von Ian Clark in einem Paper an der Universität von Edinburgh beschrieben wurde
- Resultat einer Freenet Test-Implementierung
  - Eine Routing-Tabelle mit Max 250 Einträgen in jedem Knoten erlaubt das Suchen in einem Netz mit 1Mio Knoten, mit einer durchschnittlichen Pfadlänge von 30

Erwünschte Eigenschaften

- Hohe Verfügbarkeit / Skalierbarkeit
- Einfaches Speichern von Daten
- Datenkonsistenz
- Schnelles, effizientes Auffinden von Daten
- Anonymität von Autor, Server und Client
- Verschlüsselte Kommunikation
- Zensurresistenz

Zensurresistenz

- **Primitivansatz: WWW**
  - **Aber: ein Server kann leicht „entfernt“ werden**
- **Deshalb besser:**
  - **Daten verteilen auf mehrere Server**
  - **Server verteilt über die ganze Welt**
- **Ansatz-Möglichkeiten:**
  - **Verteiltes Read-Only System (vgl. Freenet)**
  - **Komplexer: Verteiltes System mit Möglichkeit zur Datenänderung durch Autor**

Beispiel Publishing - System

- **Netzaufbau**
  - **Client - Server System**
  - **Statische Anzahl Server**
  - **Jeder Client kennt alle Server**
- **Datenspeicherung**
  - **Daten werden mit symmetrischem Schlüssel K verschlüsselt**
  - **K mit „Shamir‘s secret sharing“ fragmentiert**
  - **Aus den Fragmenten von K Datenspeicherorte berechnen**

Publishing - System Forts.

- **Datenspeicherung Forts.**
  - Auf den berechneten Servern werden jeweils Drei Objekte gespeichert
    - Mit Schlüssel K verschlüsselte Daten
    - Entsprechendes Fragment von K
    - Hash aus Serveradresse und geheimem Lösch-Passwort
  - Server kennt den Inhalt der Daten nicht, die er speichert

Publishing – System Forts.

- **Daten-Retrieval**
  - Dokument identifiziert über URL
  - URL enthält alle Indizes der Server die Key-Fragmente enthalten
  - Verschlüsselte Daten von einem der Server holen
  - Genügend Key-Shares sammeln um K zu rekonstruieren

Daten löschen

- Nur Autor kennt das geheime Lösch-Passwort
- Autor sendet Löschanfrage an alle Server die Key-Fragmente enthalten
  - Hash aus Lösch-Passwort und Serveradresse

Daten aktualisieren

- Einfügen der neuen Daten als neues Dokument, mit neuen Passwörtern
- Update Request analog zu Löschanfrage an alle beteiligten Server schicken
- Jeder betroffene Server löscht die alten Daten und speichert stattdessen ein Updatefile mit der neuen URL

In der Praxis

- **Publius:**
  - Seit dem Jahr 2000 entwickelt von zwei AT&T Research Labs Angestellten und einem Abgänger der Universität von New York.
  - Clevere Berechnung der Server
  - Zusätzliche Verschlüsslung als Schutz gegen Angreifer
  - Grösstes Problem ist die URL, die sehr lang wird bei vielen Shares

Eine mögliche Zukunft

- **Datenhaltung in einer „ubiquitären“ Welt**
  - Benutzer speichert seine Daten nicht mehr auf seinem Gerät, sondern im Netz
  - Daten auf lokalen Speichermedien sind Teil des Netzes und gehören nicht unbedingt dem Besitzer des Gerätes (und können von diesem auch nicht gelesen werden)
  - Dadurch alle Daten „überall“ verfügbar

Referenzen

- **Kurs-Homepage**
  http://www.inf.ethz.ch/vs/edu/SS2002/DS/abstracts/Datenhaltung_abstr.html
- **Freenet**
  http://freenetproject.org/cgi-bin/twiki/view/Main/ICSI
- **Publius**
  http://publius.cdt.org/publius.pdf