Chord

Peer-to-Peer-Protokoll
http://pdos.csail.mit.edu/chord/

Till Janetzki
(till@literaturport.de)
Seminar "Internet Routing“,
Technische Universität Berlin
WS 2007/2008 - 8. Februar 2008

Was ist Chord ?

- Protokoll für quasi beliebig große P2P-Netzwerke
- strukturiertes Overlay Netzwerk
- implementiert als verteilte Hash Tabelle (DHT)
- bietet nur Basisoperationen
- MIT-Lizenz

Überblick

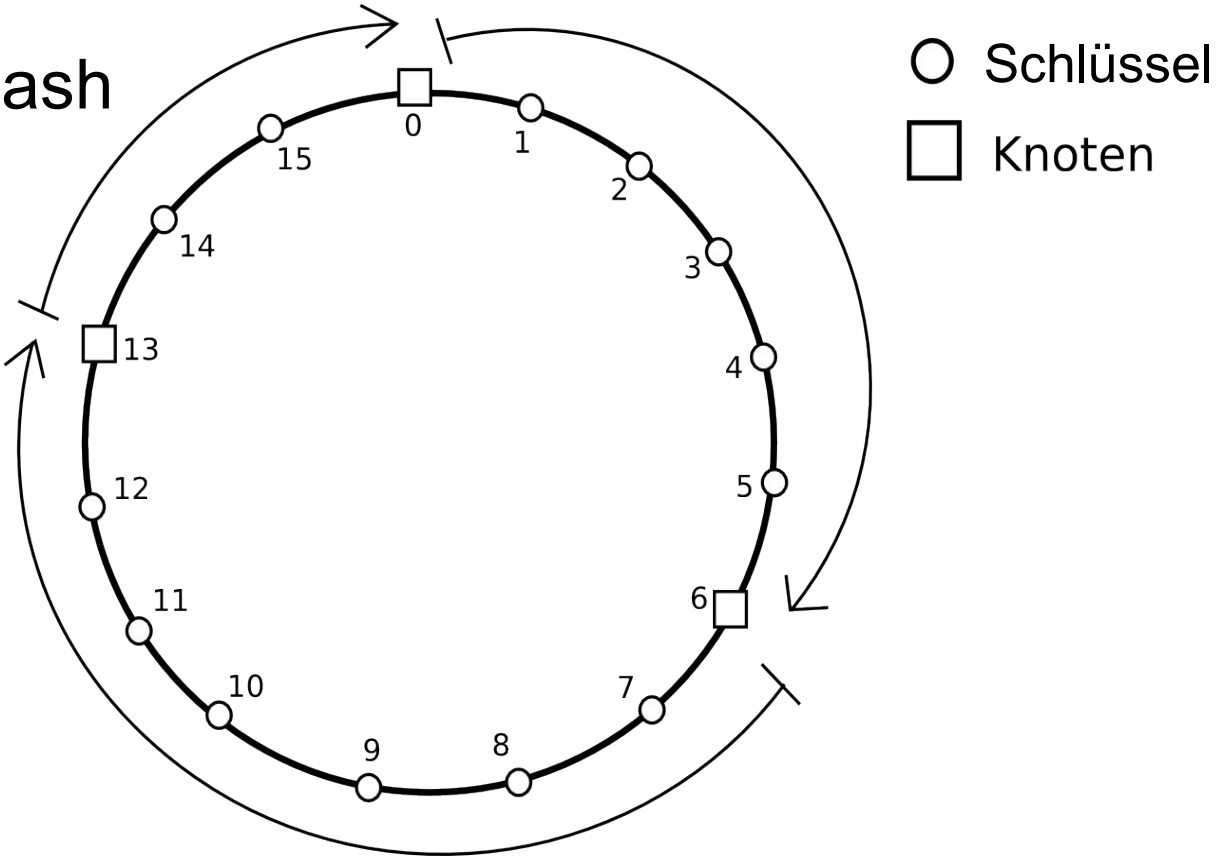
- Was ist Chord ?
- Motivation
- Aufbau
- Suchen und Einfügen
- Eigenschaften
- Simulation
- Zusammenfassung

Motivation

- effizientes Suchen/Routen ist ein zentrales Problem in großen Peer-to-Peer-Netzwerken (P2P)
- Angestrebt: trotz minimalem Wissen über das Netzwerk, schnell Suchen zu können
- garantiert korrekte Suchergebnisse

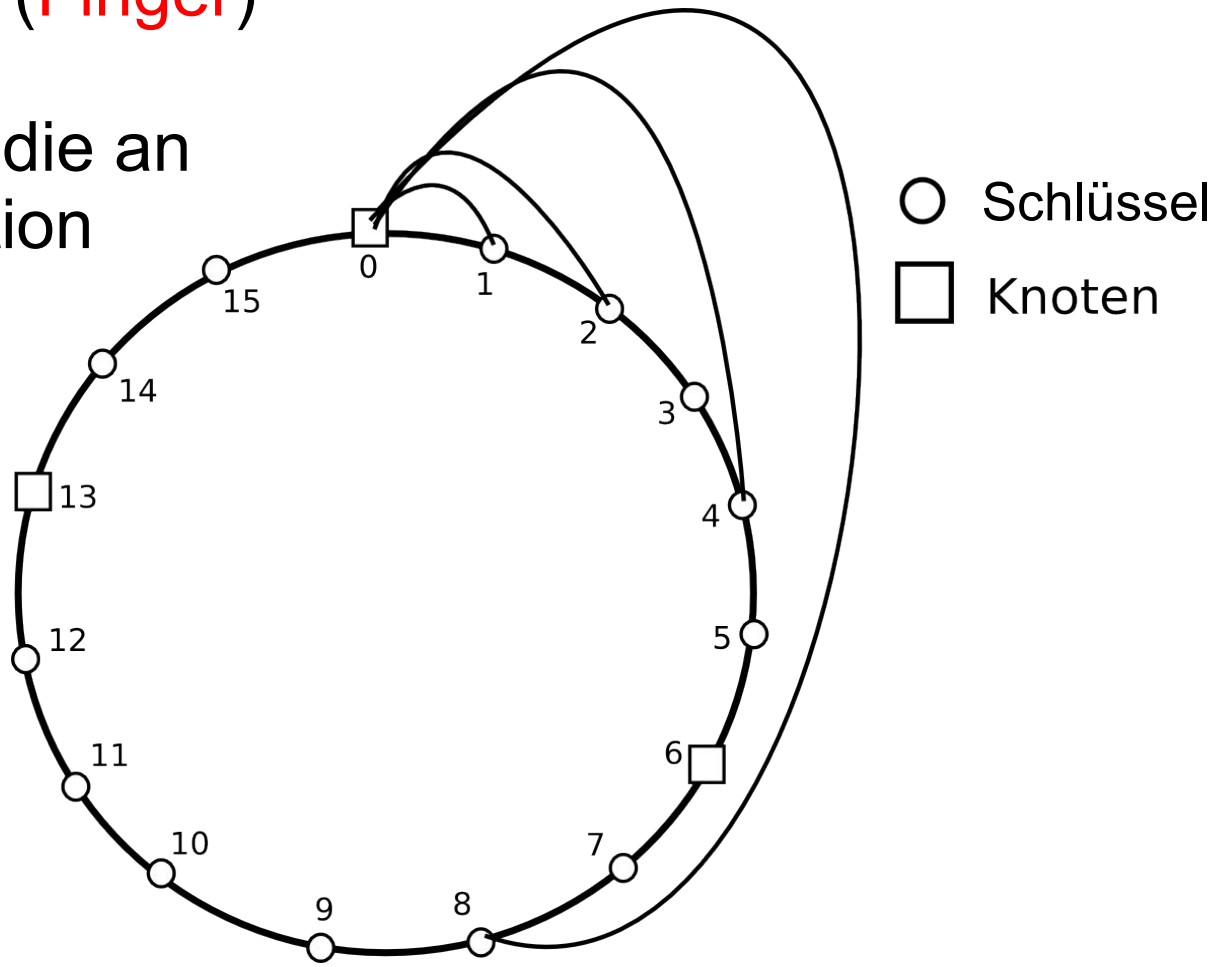
Aufbau

- virtueller Ring der Größe 2^m
- Daten und Knoten im selben Wertebereich
- ID berechnet sich aus Hash der IP-Adresse
- Jeder Knoten ist zuständig für seine Vorgängerschlüssel



Fingertabelle

- jeder Knoten kennt seinen Vorgänger- und Nachfolgerknoten
- $\log(m)$ weitere Knoten (Finger)
- Finger sind die Knoten die an 1,2,4,8,...,2^{m-1}-ter Position im Ring folgen
- ist dort kein Knoten wird der nächst-folgende Knoten verwendet



Einfügen von Knoten

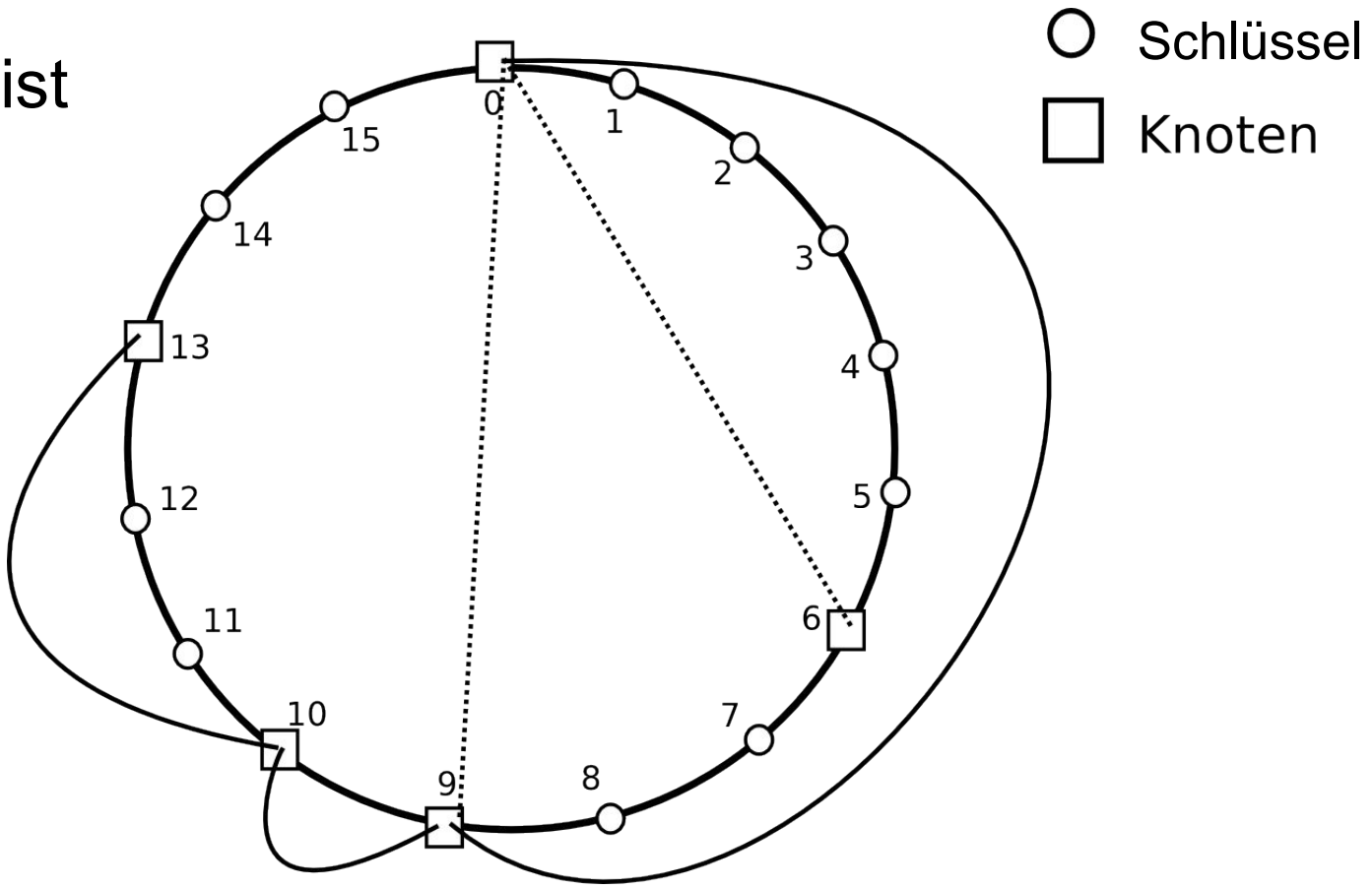
- Verbindung mit direkten Nachbarknoten genügt
- neuer Knoten *n* ermittelt eigene ID und fragt nach Nachfolgerknoten
- *n* informiert diesen
- *n* füllt Fingertabelle mit Hilfe benachbarter Tabellen
- Integration mittels Stabilisierungsfunktion der anderen Knoten

Suchen

- gesucht wird zunächst der Vorgängerknoten
- immer der größtmögliche Eintrag aus der Fingertabelle wird befragt
- pro Weiterleitung wird die Distanz mindestens halbiert
- Suchdistanz liegt im Mittel bei $\frac{1}{2}\log(n)$

Suchen - Beispiel

- Knoten 0 sucht den Nachfolger von Schlüssel 11
- Knoten 0 fragt Knoten 9 und wird zu 10 weitergeleitet
- Knoten 10 weiß das Knoten 13 der Nachfolgerknoten ist



Stabilisierung

- 3 Komponenten: *Stabilisierung*, *Überprüfung der Finger* und des *Vorgängerknotens*
- periodisch aufgerufen
- reicht nicht in allen Fällen, z.B. sind Kreise im Netz möglich
- Reparaturalgorithmus wird nur im Notfall ausgeführt

Eigenschaften - Vorteile

- Lastverteilung
- Dezentralisierung
- Skalierbarkeit
- Flexible Namensgebung
- Robustheit

Eigenschaften - Nachteile

- fehlende Anonymität
- Latenz - $\frac{1}{2}\log(n)$ unter Umständen zu lang
- keine regionale Lastverteilung
- alle Aussagen nur mit „*hoher Wahrscheinlichkeit*“

Anwendungen

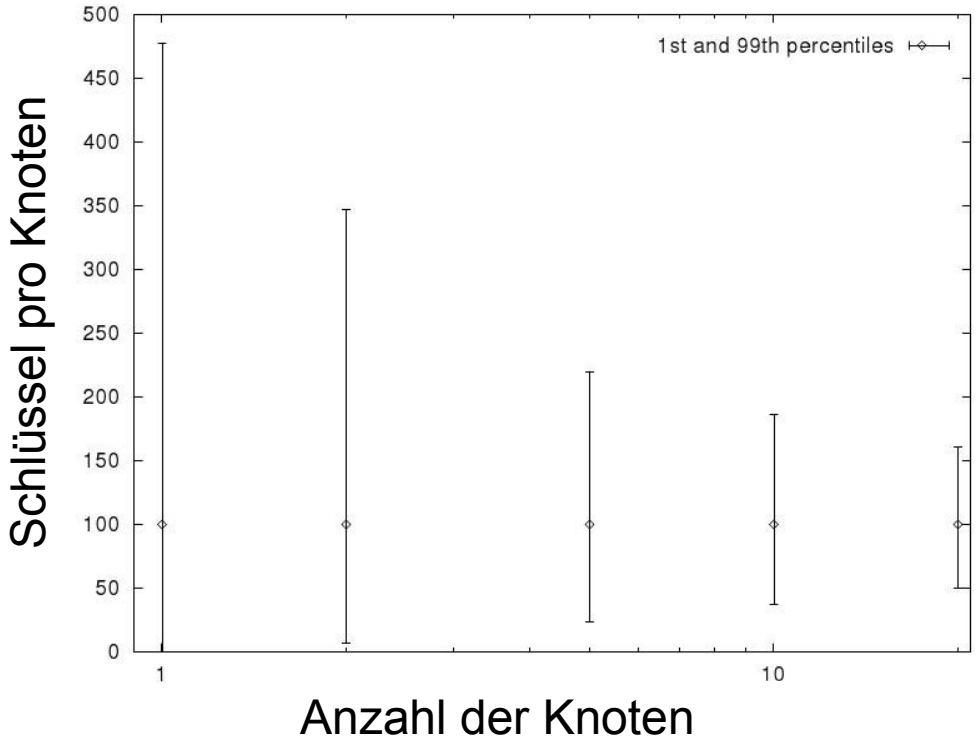
- verteilte Indexsuche
- verteiltes Rechnen
- Kooperatives Spiegeln (z.B. Coral)
- Hosten ohne dauernde Verbindung

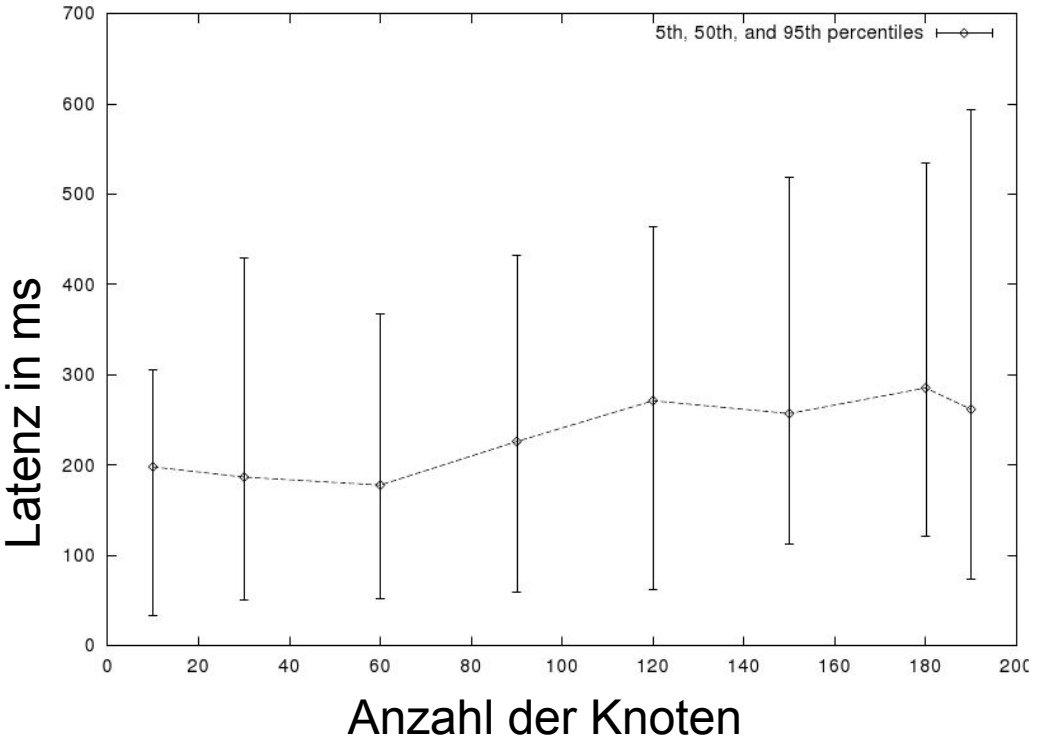
Simulation 1

- in Protokollsimulator und realen Netzen
- grundlegende Bestätigung der Aussagen
- auch Ausfall von mehr als 50% der Knoten kein Problem
- aber: Korrektheit nicht bewiesen

Simulation 2

- Lastverteilung Problematisch, Lösung: virtuelle Knoten
- Latenz skaliert gut aber Anfangslatenz sehr hoch





Zusammenfassung

- hoch skalierbar
- Einfügen: $\log^2(n)$; Suchen: $\frac{1}{2}\log(n)$
- nicht für kleine Netze geeignet
- garantiert korrekte Suchergebnisse
- sehr flexibel
- nicht so verbreitet wie andere DHTs (z.B. Kademlia)
- diverse Umsetzungen z.B. Open Chord, Chord#

Quelle

• Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, pages 149–160. ACM Press, 2001.

Chord

Fragen?