Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.

Ion Stoica, Robert Morris, David Karger,
M. Frans Kaashoek, Hari Balakrishnan

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Präsentation von Christopher Gelbke

Proseminar „Algorithmen für Rechnernetzwerke“
bei Prof. Dr. Christian Schindelhauer

Motivation

UNI FREIBURG

Was ist Chord:

- Einfaches Suchprotokoll für Peer-to-Peer Netzwerke.

Anforderung an Chord:

- Dezentral – alle Knoten sind gleichwertig.
- Skalierbar – gute Performanz in großen Netzwerken.
- Flexibel – Knoten können das Netzwerk jeder Zeit betreten und verlassen.
- Verfügbarkeit – jeder Knoten findet jeden Schlüssel.

Gliederung

UNI FREIBURG

- Einführung „Peer-to-Peer-Netzwerke“
- Funktionsweise von Chord
  - Konsistente Hash-Funktionen
  - Chord-Ring
  - Suche von Knoten
  - Hinzufügen neuer Knoten
  - Entfernen von Knoten
  - Stabilisierung
  - Fehlerbehebung
  - Skalierbarkeit
- Zusammenfassung

Einführung Peer-to-Peer Netzwerk

Eigenschaften eines P2P Netzwerks:

- Jeder Knoten ist gleichberechtigt.
- Datenaustausch zwischen Knoten.
- Es ist dynamisch.
- Verfügbarkeit / Fehlerfreiheit ist nicht gewährleistet.

Peer [1]

Einführung Peer-to-Peer Netzwerk

UNI FREIBURG

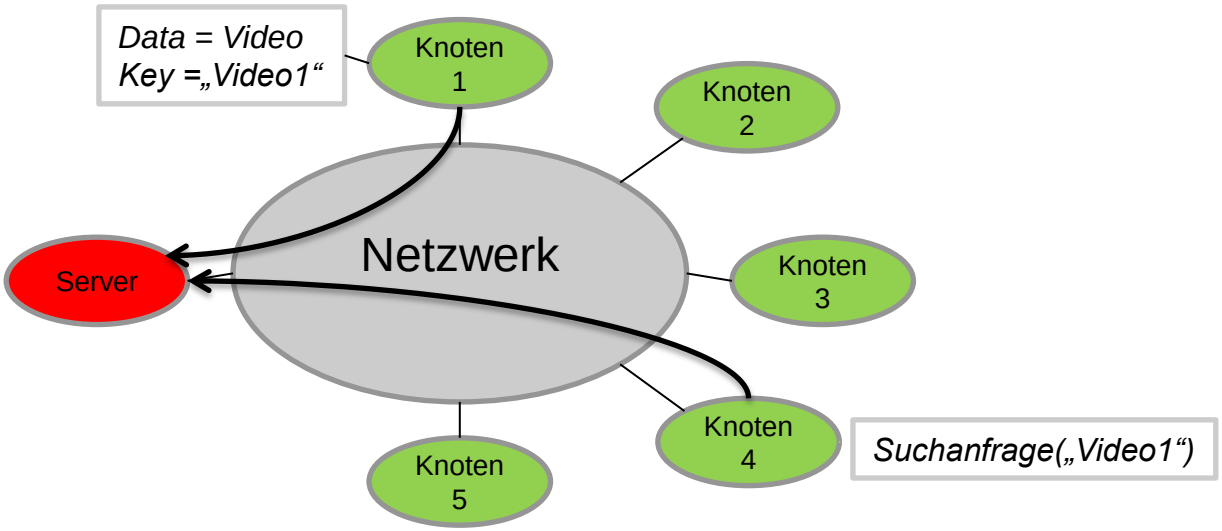
Wie findet man einen gewünschten Knoten im P2P Netzwerk?

Es gibt verschiedene Ansätze:

- Zentralisierter Ansatz
- Flooding Ansatz
- Gerouteter Ansatz

Zentralisierter Ansatz

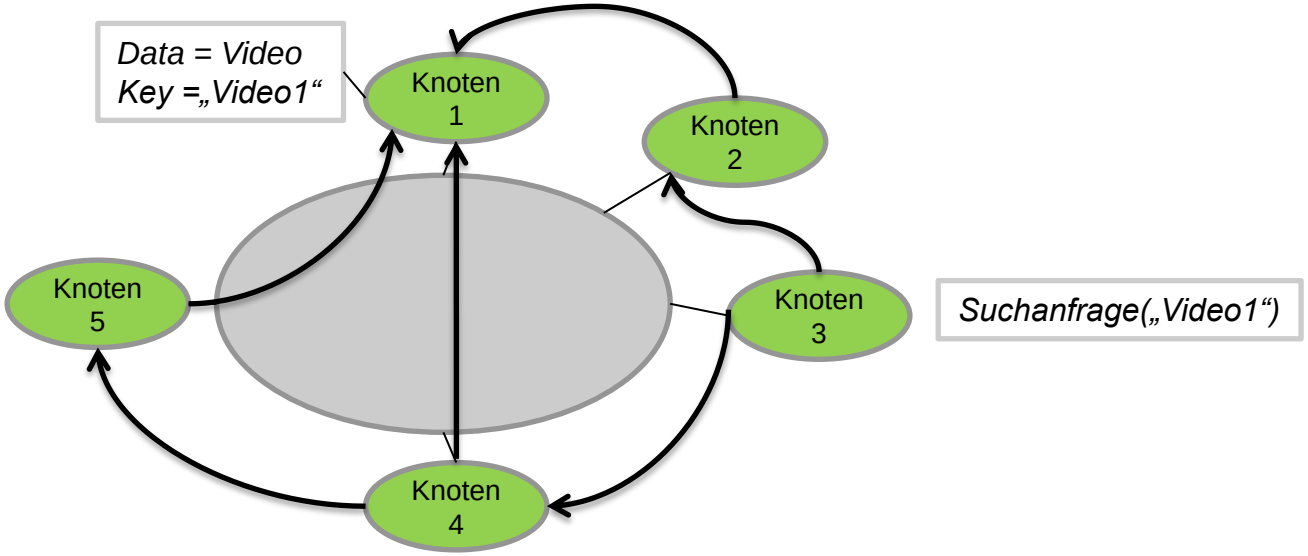
Zentraler Server, der alle Anfragen steuert.



- Sehr fehleranfällig, da auf Server angewiesen.
- Vertreter: Napster

Flooding Ansatz

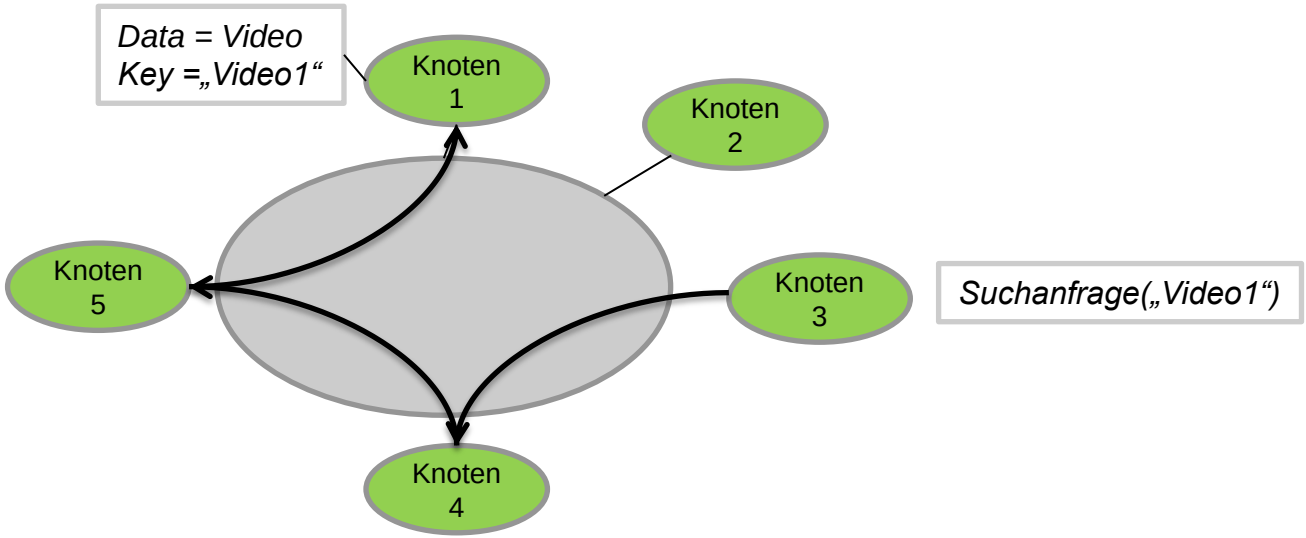
Es muss mindestens ein Knoten bekannt sein.



- Nicht sehr effizient.
- Im schlechtesten Fall O(N) Nachrichten pro Suchanfrage.
- Vertreter: Gnutella

Gerouteter Ansatz

Jeder Knoten besitzt Informationen über andere Knoten. Sogenannte „routing tables“.



- Höhere Effizienz als Flooding Ansatz.
- Vertreter: Chord, Freenet

Chord - Konsistente Hash-Funktion

UNI FREIBURG

Chord verwendet eine Konsistente Hash-Funktion zur Suche.

Anforderung an die Hashfunktion:

- Gleichverteilt
- Kollisionssicher

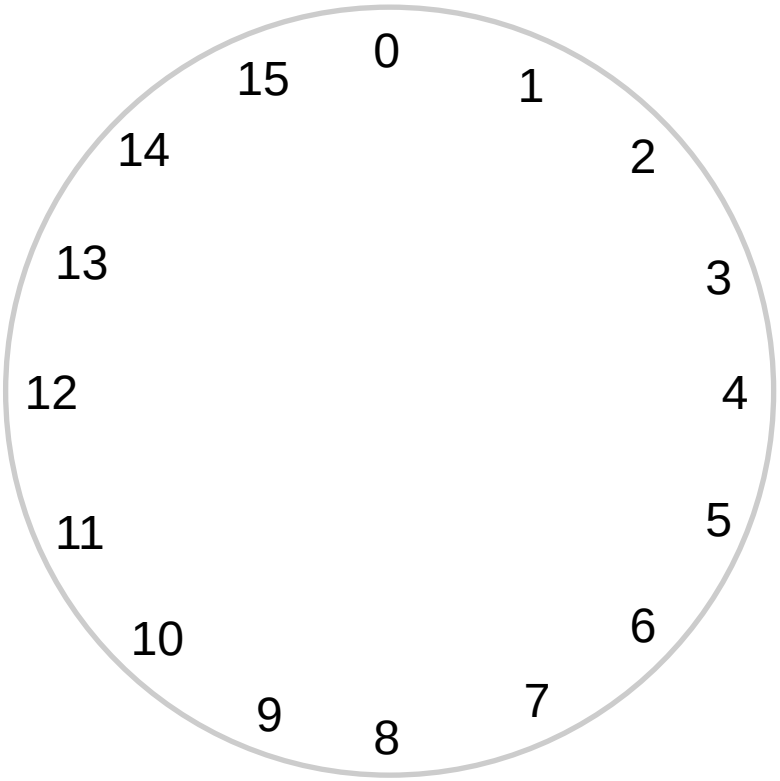
Der Schlüsselraum ist m-bit für Knoten und Schlüssel.

- Knoten ID -> Hash(IP-Adresse)
- Schlüssel ID -> Hash(Schlüssel)

Chord - Konsistente Hash-Funktion

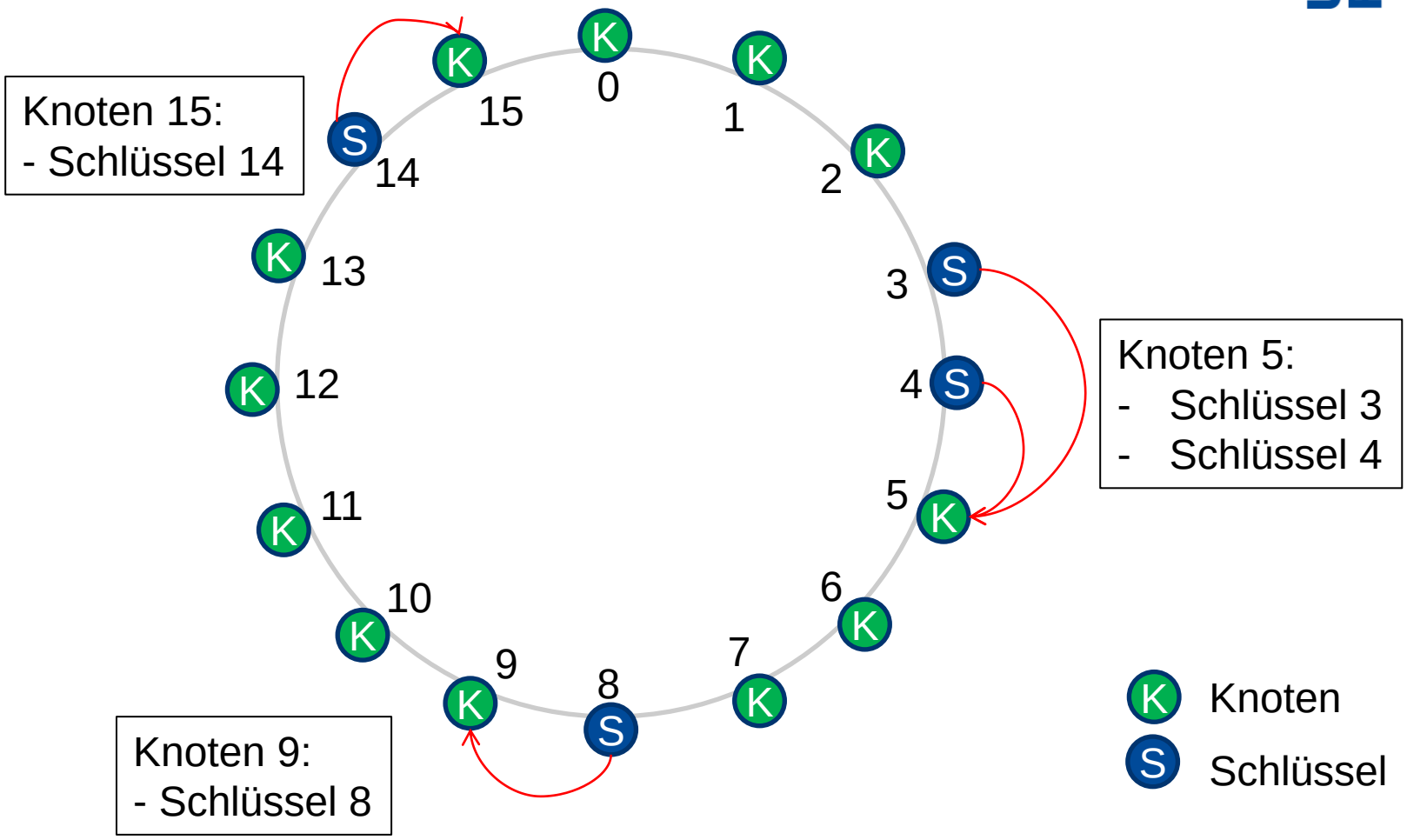
- Als Hash Algorithmus wird SHA-1 verwendet.
- m = 160 Bit. Dadurch sind maximal 2^{160} Knoten, Schlüssel möglich.
- SHA-1 gilt weitgehend als kollisionssicher.
- Weitere Informationen über SHA-1 auf der Webseite [3].

Chord - Ring



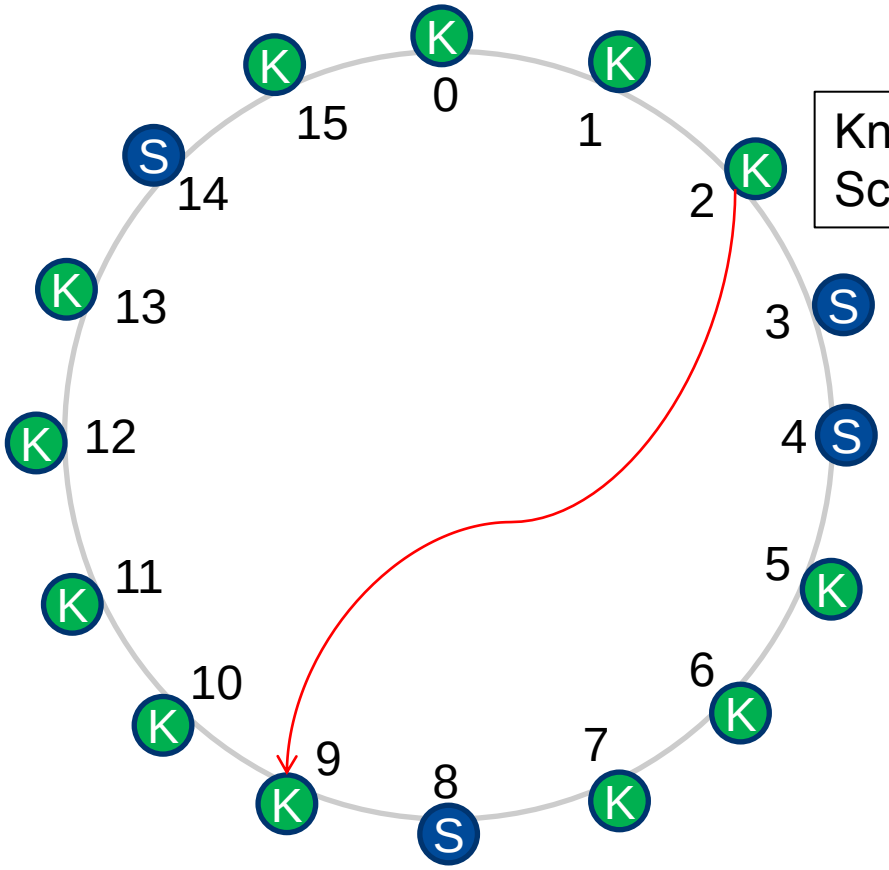
Schlüsselraum m = 4-Bit

Chord - Ring


Knoten 15:
- Schlüssel 14
Knoten 5:
- Schlüssel 3
- Schlüssel 4
Knoten 9:
- Schlüssel 8
K Knoten
S Schlüssel
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Chord - Suche

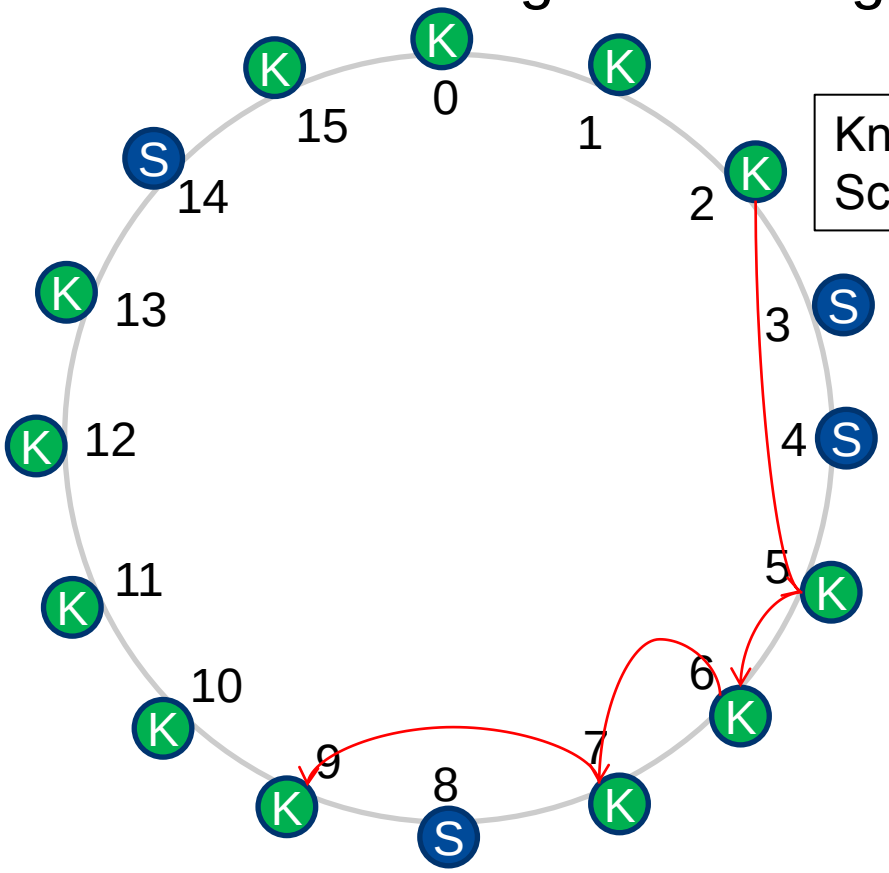
Alle Knoten kennen alle Knoten.



Maximale Suchzeit ist O(1).

Chord - Suche

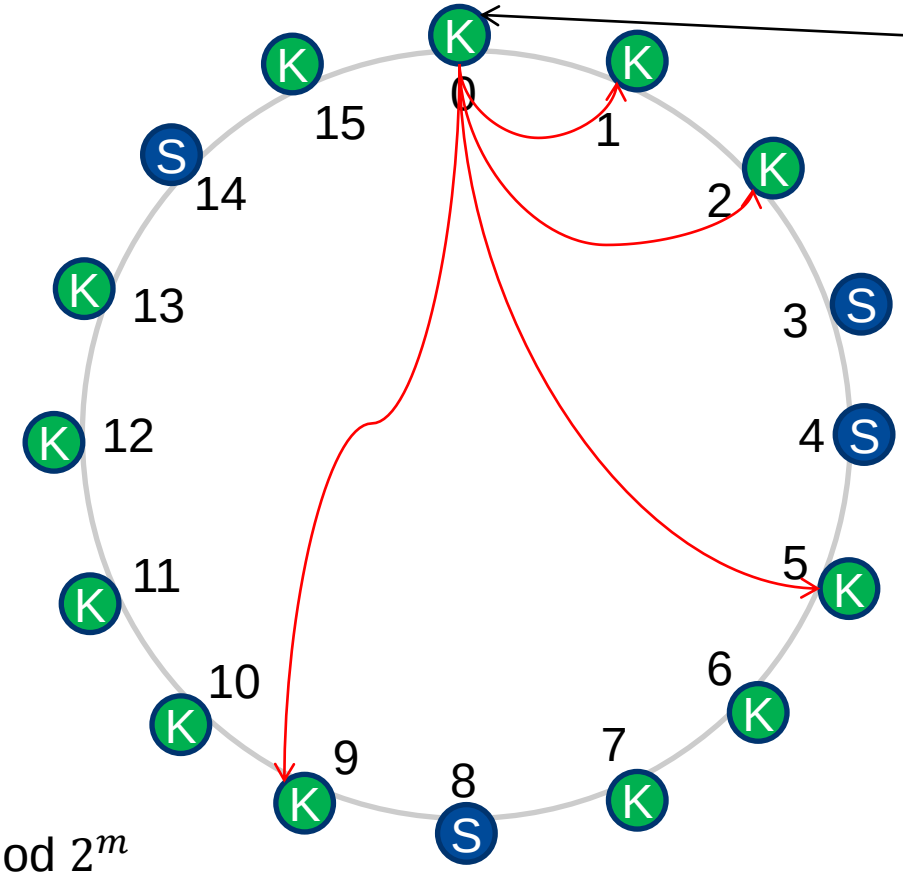
Jeder Knoten kennt sein Nachfolger und Vorgänger.



Maximale Suchzeit ist O(N).

Chord - Suche

Jeder Knoten hat einen sogenannten „finger table“.



i	K(0)
0	1
1	2
2	(4) 5
3	(8) 9

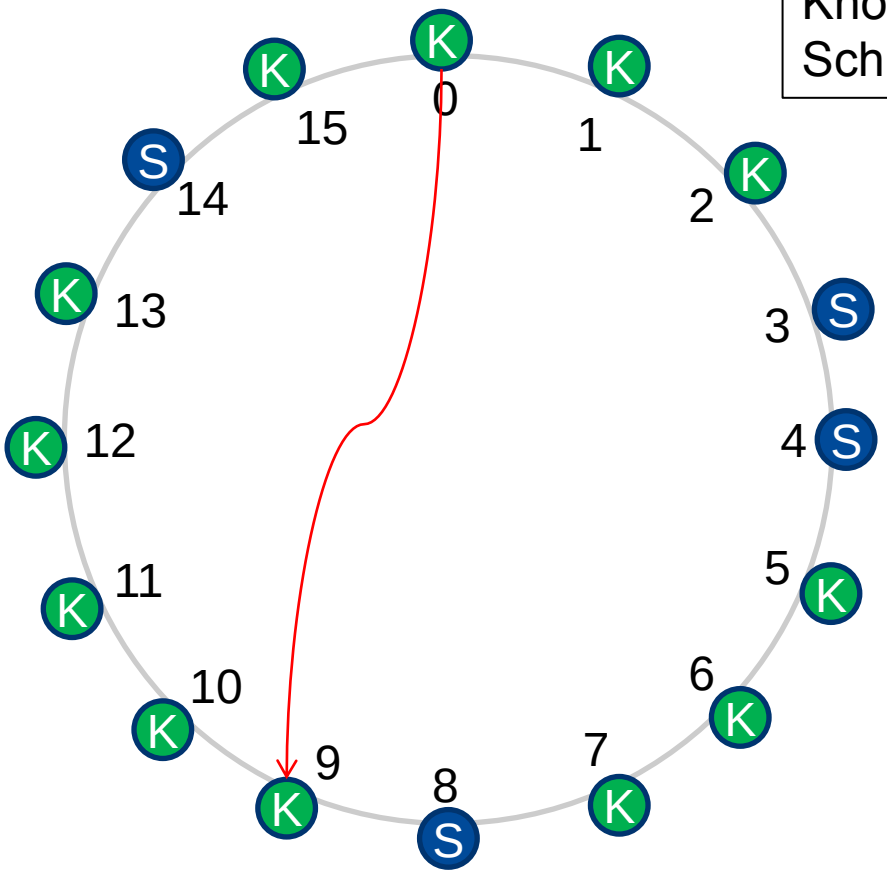
K Knoten

S Schlüssel

- K(n) = (n + 2^i) mod 2^m
- Jeder Knoten hat log(n) Finger

Chord - Suche

Die Suchkosten betragen nun O(log(N))



Knoten „0“ sucht Schlüssel „8“

i	K(0)
0	1
1	2
2	(4) 5
3	(8) 9

K Knoten

S Schlüssel

Chord - Hinzufügen

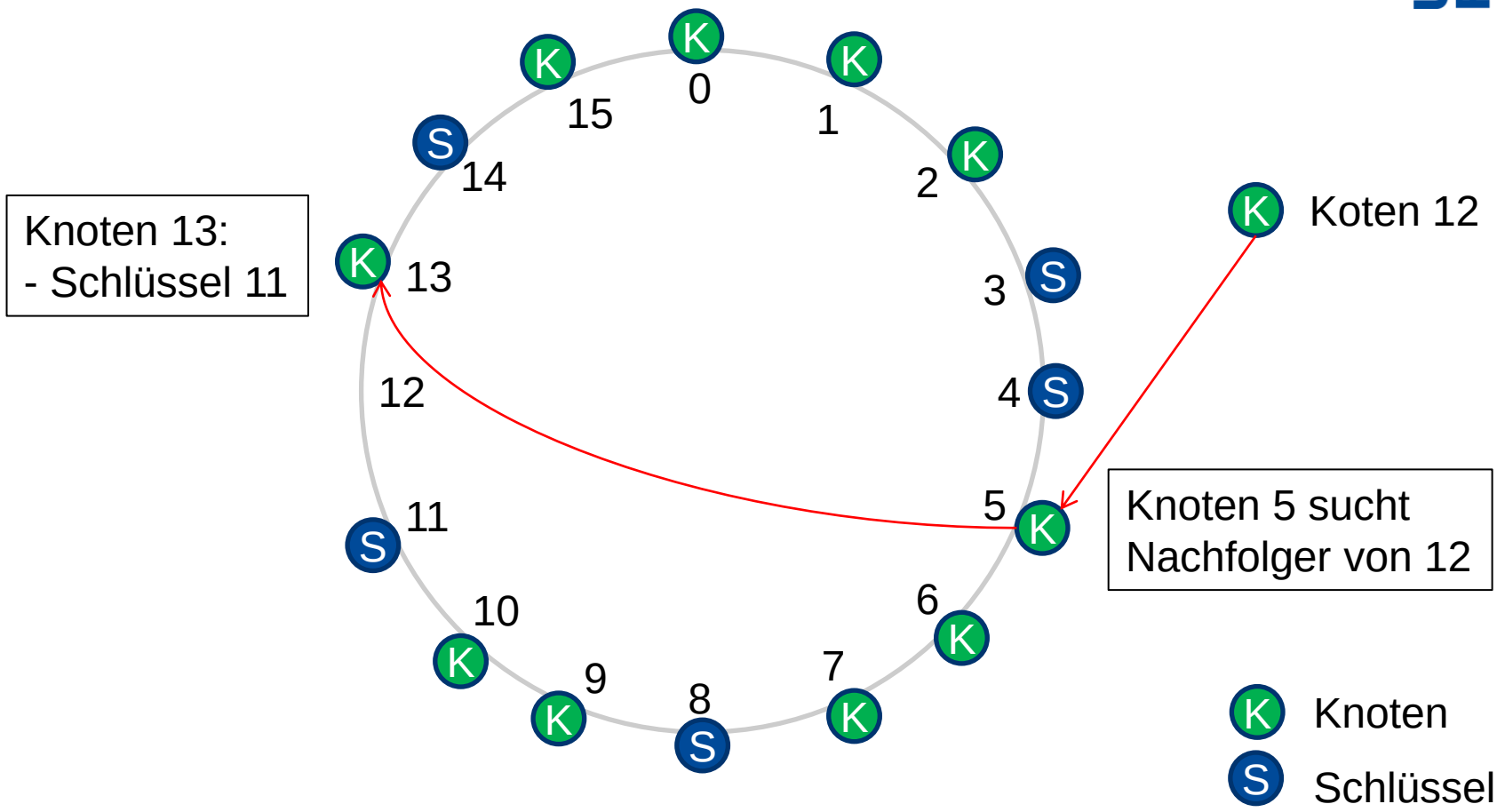
Das Hinzufügen eines neues Knoten in das Netzwerk wird in drei Schritten erledigt:

1. Platz durch Suchanfrage finden.
2. Alte Knoten updaten (Stabilisieren).
3. Schlüssel zum neuen Knoten kopieren, falls vorhanden.

Diese Schritte kosten zusammen $O(\log(N)^2)$ Nachrichten.

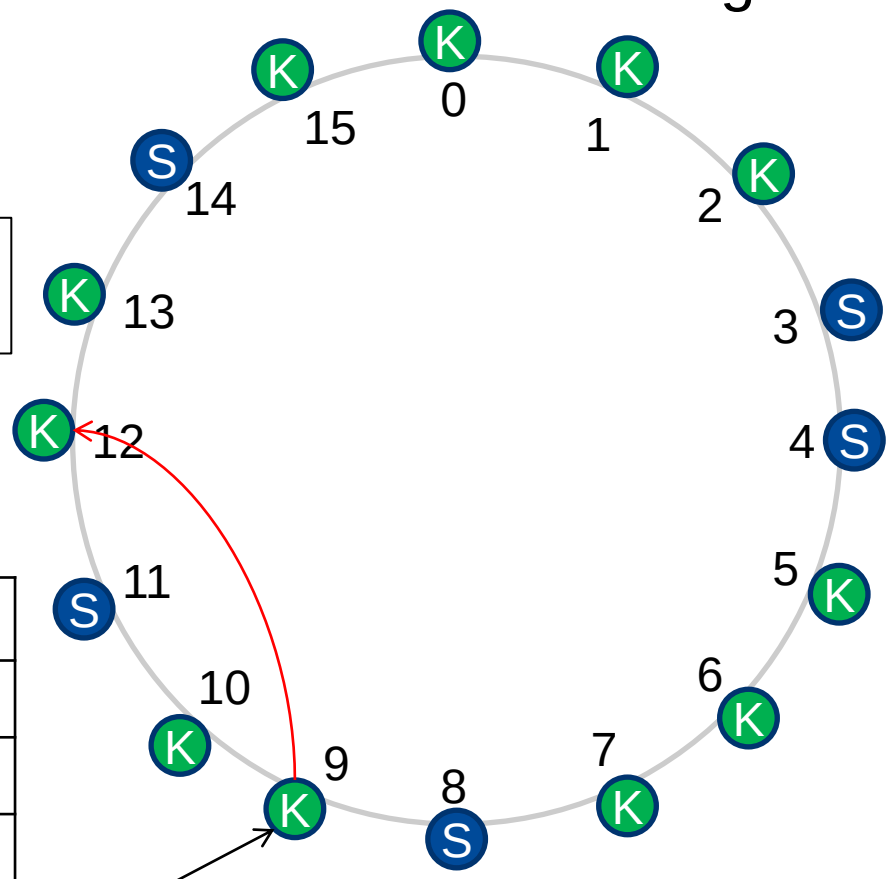
Chord - Hinzufügen

Platz des Knoten durch Suche finden:



Chord - Hinzufügen

Vorgänger Knoten aktualisieren ihre Finger (Stabilisieren):



Knoten 13:
- Schlüssel 11

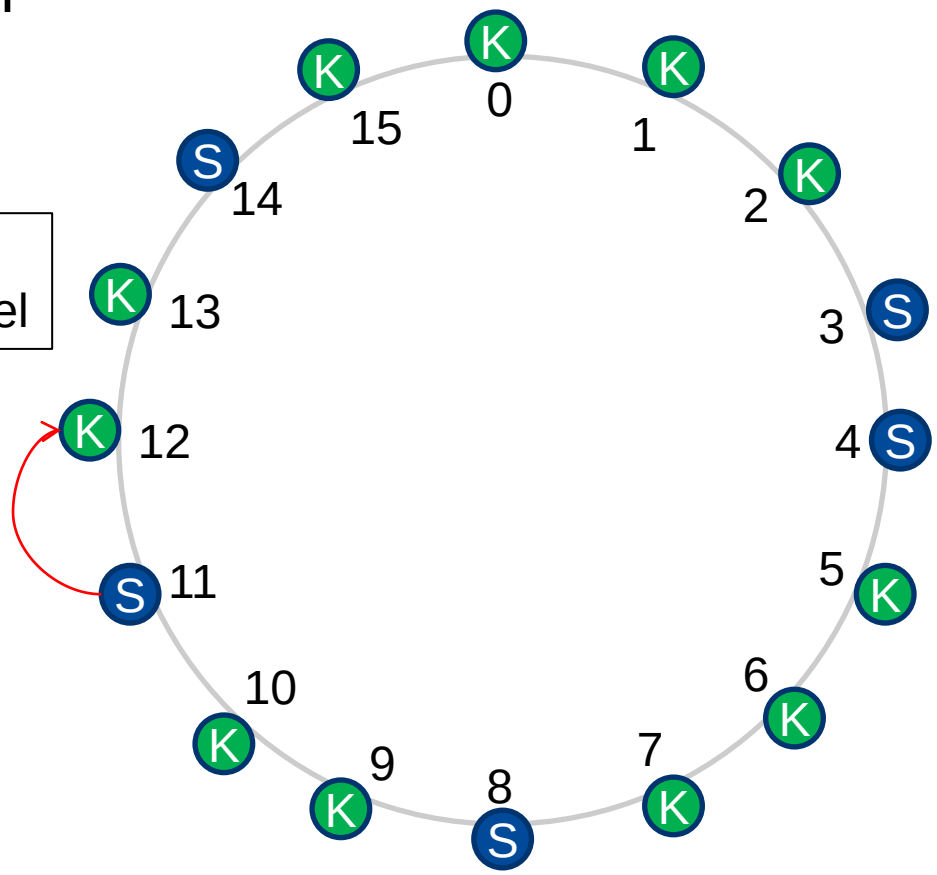
i	K(9)
0	10
1	(11) 12
2	13
3	2

K Knoten

S Schlüssel

Chord - Hinzufügen

Schlüssel kopieren:



Knoten 13:
- Keine Schlüssel

Knoten 11:
- Schlüssel 11

K Knoten

S Schlüssel

Chord - Entfernen

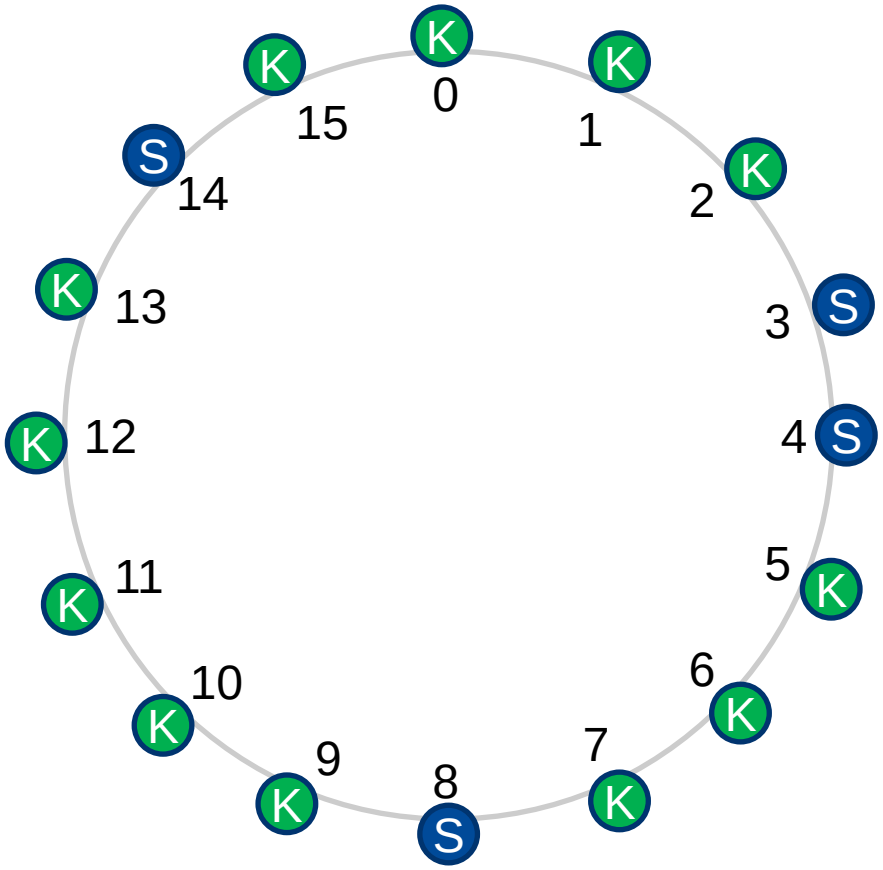
UNI FREIBURG

Das Entfernen eines Knoten im Netzwerk wird in erfolgt in den Schritten:

1. Knoten verlässt Netzwerk
2. Schlüssel Nachfolger zuordnen
3. Stabilisiere Netzwerk (entfernt inaktive Knoten)

Chord - Entfernen

Netzwerk vor Verlassen eines Knotens:

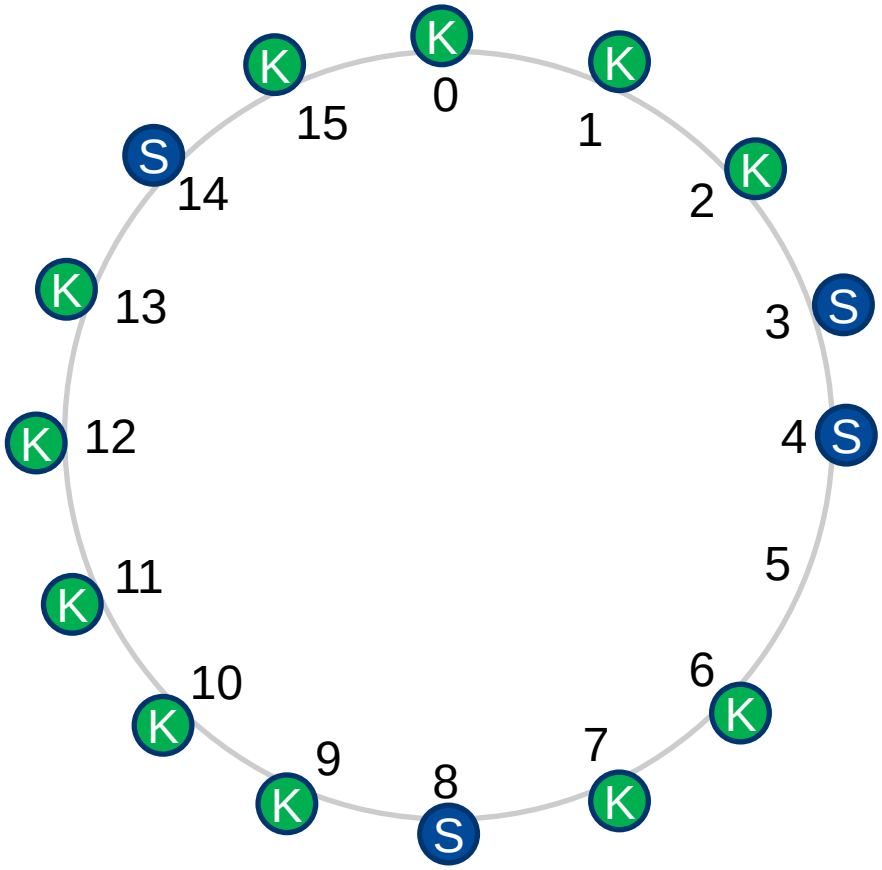


K Knoten

S Schlüssel

Chord - Entfernen

Knoten 5 verlässt Netzwerk:

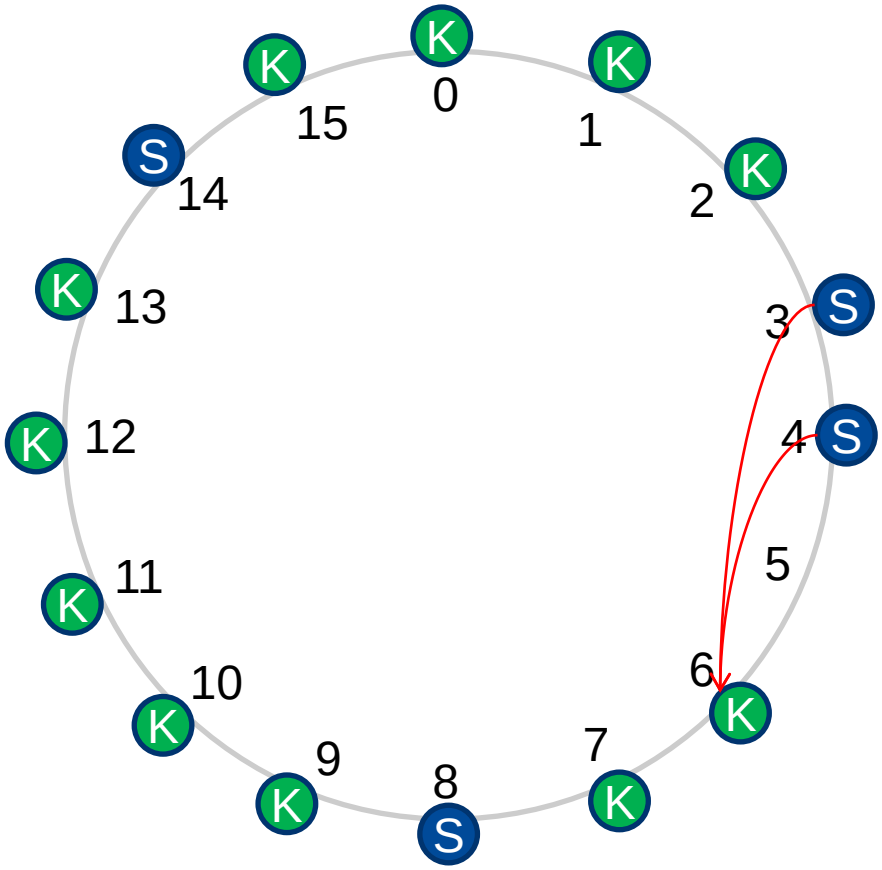


K Knoten

S Schlüssel

Chord - Entfernen

Schlüssel Nachfolger zuordnen:

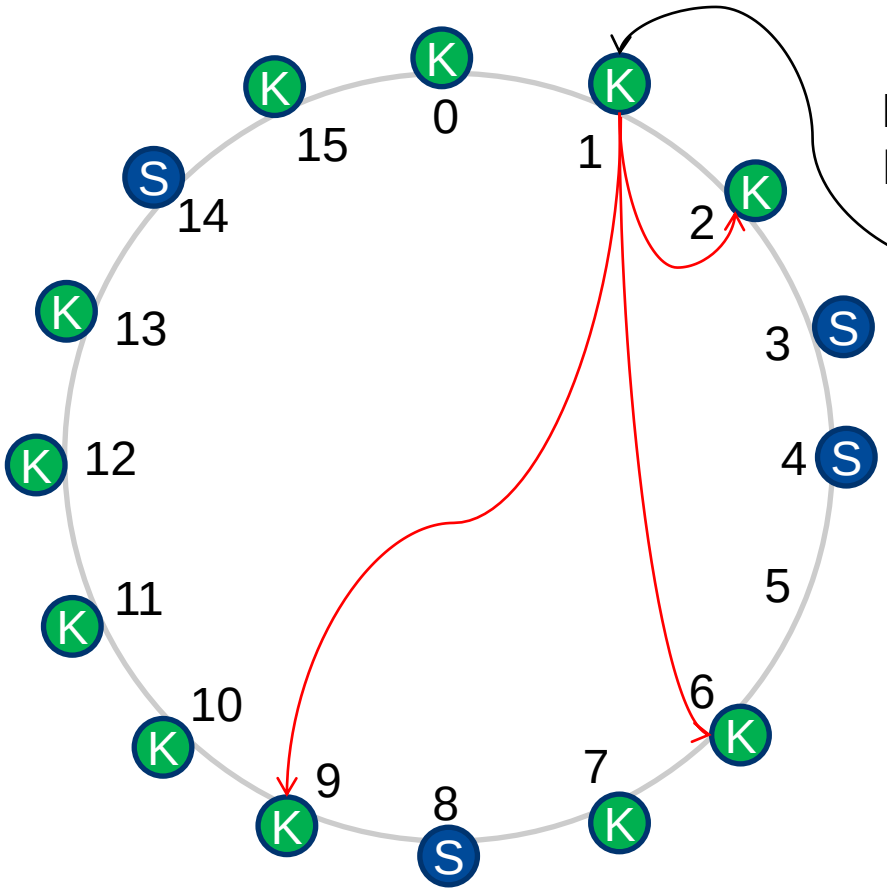


K Knoten

S Schlüssel

Chord - Entfernen

Stabilisieren:



Knoten 0, 1, 2 passen Ihre „finger tables“ an.

i	K(1)
0	2
1	(3) 6
2	(5) 6
3	9

K Knoten

S Schlüssel

Chord - Stabilisierung

UNI FREIBURG

Jeder Knoten ruft periodisch den Stabilisierungsalgorithmus auf:

- fix_fingers(): Behebt Fehler in den „finger tables“
- stabilize(): Holt sich Vorgänger-Knoten des Nachfolge-Knoten.

- Behebt isolierte und unzugängliche Knoten
- Behebt Zerteilungen des Rings.

Chord - Fehlerbehandlung

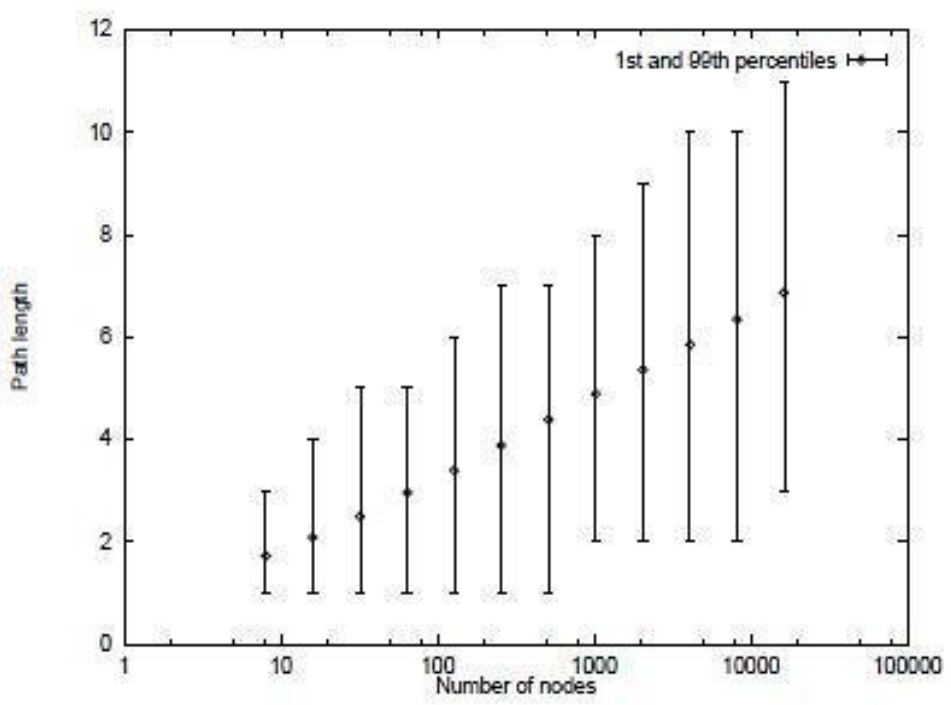
UNI FREIBURG

- Netzwerk bleibt intakt solang jeder seinen richtigen Nachfolger/Vorgänger kennt.

- Im schlimmsten Fall ergibt dies eine einfache Suche mit einer Laufzeit von O(N), da jeder Knoten seinen Nachfolger kennt.

Chord - Skalierbarkeit

- Die Kosten betragen wie erwartet O(log(N)).
- Das bedeutet gute Skalierbarkeit bei großen Netzwerken.



[2]

Zusammenfassung

UNI FREIBURG

- Einfache Implementation
- Beweisbare, effiziente Suche
- Gute Skalierbarkeit log(n)
- Funktioniert auch noch mit defekten Knoten

Ende

UNI FREIBURG

Vielen Dank für ihre Aufmerksamkeit

Referenzen

[1] Mauro Bieg, Wikipedia, P2P-network.svg, *http://de.wikipedia.org/wiki/Peer-to-Peer*, 12. August 2001.

[2] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Application.* ACM SIGCOMM, San Deigo, CA, August 2001, Seite 8.

[3] National Institute of Standards and Technology. *SECURE HASH STANDARD. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, 1 August 2002.*