Peer-to-Peer-Netzwerke

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer

Sommersemester 2006

10. Vorlesung

31.05.2006

schindel@informatik.uni-freiburg.de

Inhalte

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

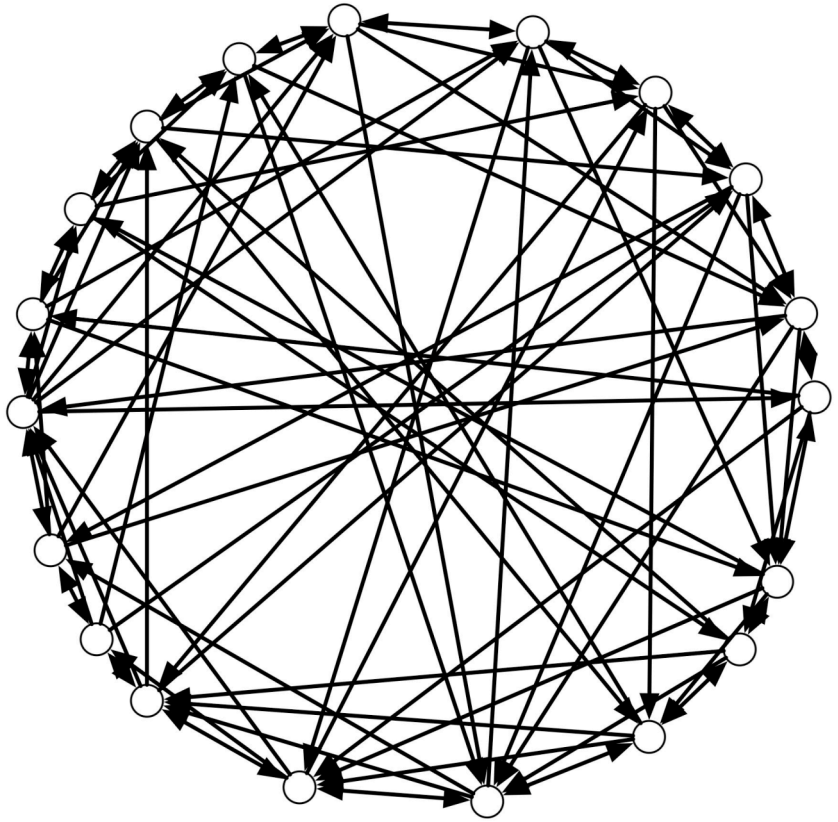
- **Kurze Geschichte der Peer-to-Peer-Netzwerke**
- **Das Internet: Unter dem Overlay**
- **Die ersten Peer-to-Peer-Netzwerke**
  - Napster
  - Gnutella
- **CAN**
- **Chord**
- **Pastry und Tapestry**
- **Gradoptimierte Netzwerke**
  - Viceroy
  - Distance-Halving
  - Koorde
- **Netzwerke mit Suchbäumen**
  - Skipnet und Skip-Graphs
  - P-Grid
- **Selbstorganisation**
  - Pareto-Netzwerke
  - Zufallsnetzwerke
  - Metrikbasierte Netzwerke
- **Sicherheit in Peer-to-Peer-Netzwerken**
- **Anonymität**
- **Datenzugriff: Der schnellere Download**
- **Peer-to-Peer-Netzwerke in der Praxis**
  - eDonkey
  - FastTrack
  - Bittorrent
- **Peer-to-Peer-Verkehr**
- **Juristische Situation**

Chord

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **von Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek und Hari Balakrishnan (2001)**

- **DHT mit Hash-Bildbereich {0,..,2^m-1}**
  - für genügend großes m
- **Ring-Verknüpfung der Peers**
- **Abkürzungen im Ring durch exponentiell gestaffelte Zeiger auf Nachfolger**



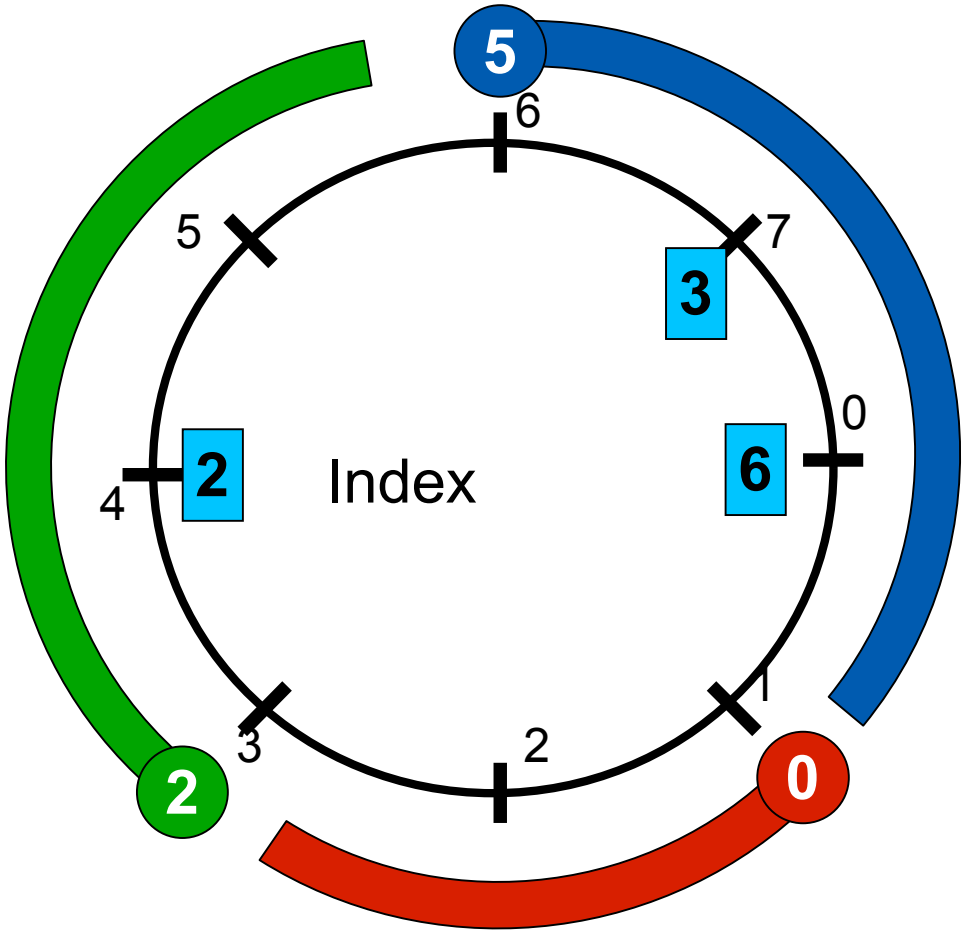
Chord als DHT

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **n: Knotenanzahl, Knotenmenge V**
- **k: Anzahl Schlüssel, Schlüsselmenge K**
- **m: Hashwertlänge: m >> log max{K,N}**

- **Zwei Hash-Funktionen bilden auf {0,..,2^m-1} ab**
  - $r_V(b)$: bildet Peer b zufällig auf {0,..,2^m-1} ab
  - $r_K(i)$: bildet Index i zufällig auf {0,..,2^m-1} ab

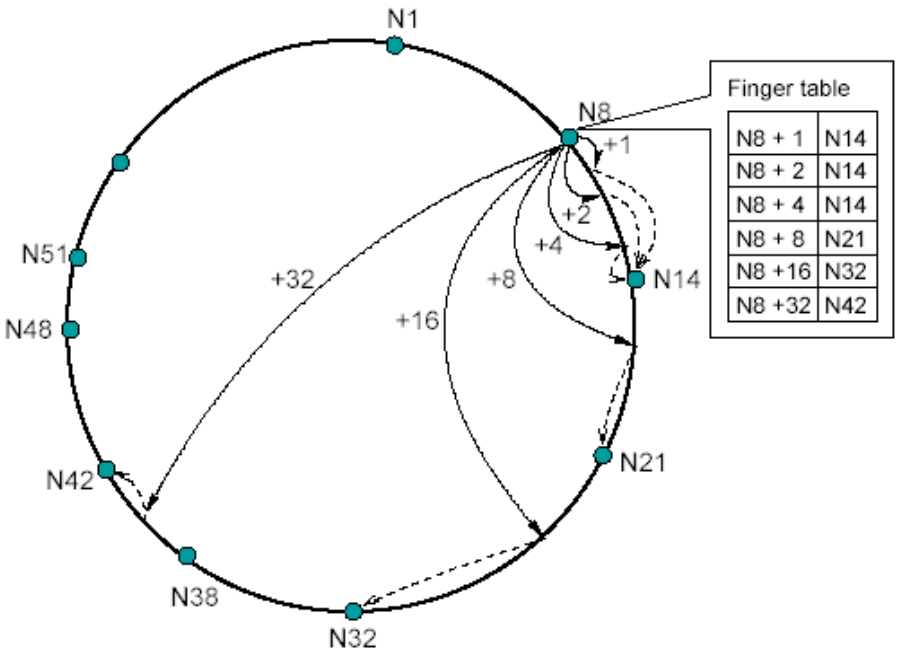
- **Abbildung von i auf einen Peer b = $f_V(i)$**
  - $f_V(i) := \arg\min_{b \in V} (r_B(b)-r_K(i))$



Die Datenstruktur von Chord

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Für jeden Knoten b:**
  - successor: Nachfolger
  - predecessor: Vorgänger
  - Für i ∈ {0,..m-1}
    - Finger[i] := Der Knoten der dem Wert $r_V(b+2^i)$ folgt
- **Für kleine i werden die Finger-Einträge immer gleich**
  - Nur unterschiedliche Fingereinträge werden gespeichert
- **Lemma**
  - Die Anzahl unterschiedlicher Finger-Einträge für Knoten b ist mit hoher Wahrscheinlichkeit O(log n)
- **Hohe Wahrscheinlichkeit = $1-n^{-c}$**



Balance in Chord

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **n: Anzahl der Knoten im P2P-Netzwerk**
- **k: Anzahl der Schlüssel $\geq$ 1**

- **Theorem**
  - Die Datenstruktur von Chord hat folgende Eigenschaften
    - Balance&Load: Mit pol. W‘keit $(1-n^{-c})$ werden in jedem Knoten höchstens O(k/n log n) Schlüssel gespeichert
    - Dynamik: Tritt ein neuer Knoten hinzu oder verlässt ein Knoten das Netzwerk müssen mit pol. W‘keit höchstens O(k/n log n) Schlüssel bewegt werden.
- **Beweis**
  - …

Eigenschaften der Datenstruktur

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Lemma**
  - Der Abstand $|r_V(b.succ) - r_V(b)|$ ist
    - im Erwartungswert $2^m/n$,
    - mit hoher Wahrscheinlichkeit höchstens $O((2^m/n) \log n)$ und
    - mit hoher Wahrscheinlichkeit mindestens $(2^m/n)/ n^c$ für eine Konstante $c>0$
    - In einem Intervall der Länge w $2^m/n$ sind mit hoher Wahrscheinlichkeit
      - $\Theta(w)$ Knoten, falls $w=\Omega(\log n)$
      - höchstens $O(w \log n)$ Knoten, falls $w=O(\log n)$
- **Lemma**
  - Die Anzahl der Knoten, die einen Fingerzeiger auf Knoten b besitzen ist
    - im Erwartungswert $O(\log n)$
    - mit pol. Wahrscheinlichkeit höchstens $O(\log n)$

Suchen in Chord

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

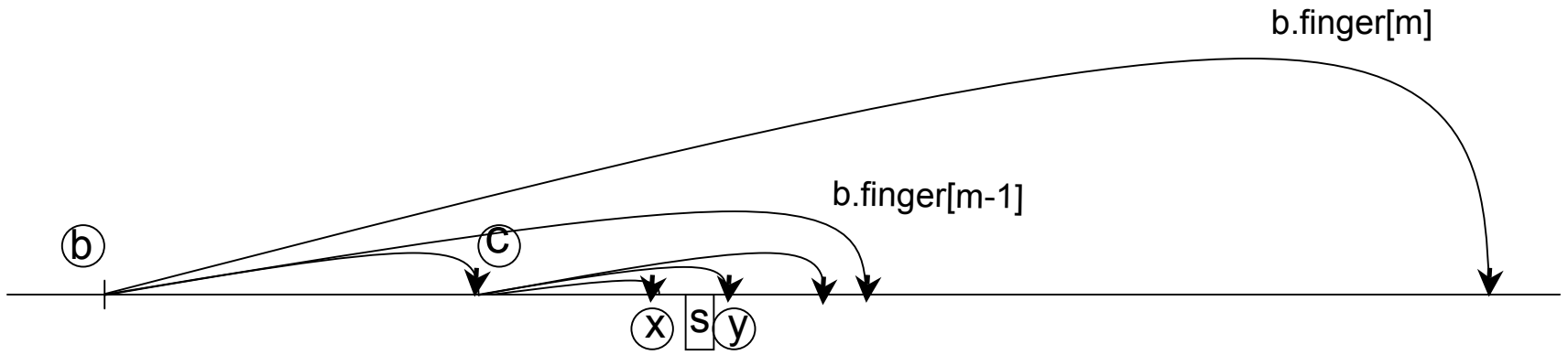
- **Theorem**
  - Die Suche braucht mit hoher W'keit O(log n) Sprünge

- **Suchalgorithmus für Element s:**
  - Abbruch(b,s):
    - Knoten b,b'=b.succ gefunden, mit rK(s) $\in [r_V(b),r_V(b')|$
  - Hauptroutine: Starte mit irgendeinem Knoten b

```
while not Abbruch(b,s) do
    for i=m downto 0 do
        if r_K(s) ∈ [r_V(b.finger[i]),r_V(finger[i+1])] then
            b ← b.finger[i]
        fi
    od
```

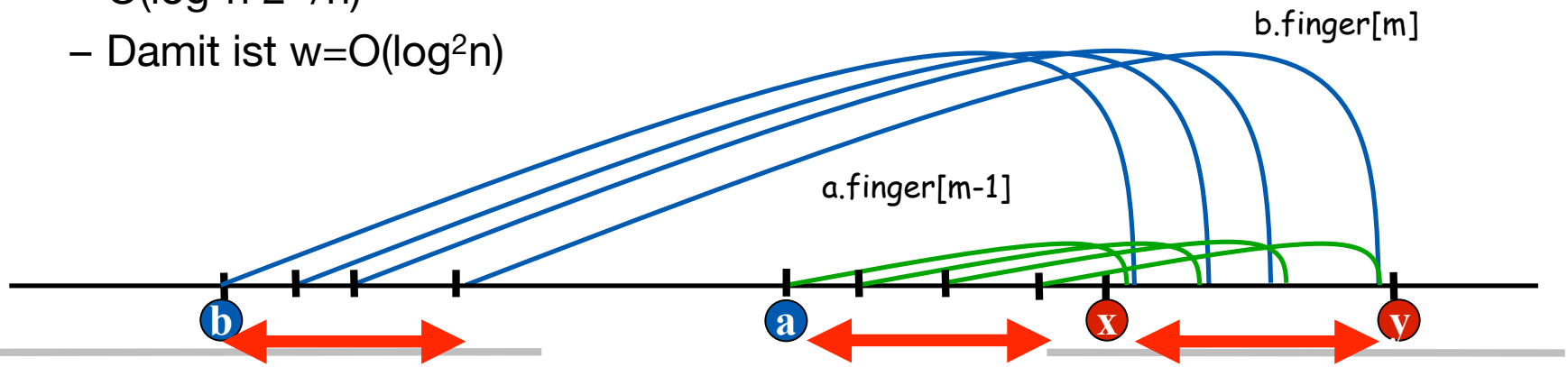
Suchen in Chord

- **Theorem**
  - Die Suche braucht mit hoher W'keit O(log n) Sprünge
- **Beweis:**
  - Mit jedem Sprung wird die Entfernung zum Ziel mindestens halbiert
  - Zu Beginn ist der Abstand höchstens 2^m
  - Der Mindestabstand zweier benachbarter Peers ist $2^m/n^c$ mit hoher W'keit
  - Damit ist die Laufzeit beschränkt durch c log n



Fingeranzahl

- **Lemma**
  - Der Ausgrad im CHORD-Netzwerk ist O(log n) mit hoher W‘keit
  - Der Eingrad im CHORD-Netzwerk ist $O(\log^2 n)$ mit hoher W‘keit
- **Beweis**
- **Der minimale Abstand zweier Peers ist $2^m/n^c$ (mit hoher W‘keit)**
  - Damit ist der Ausgrad beschränkt durch c log n (mit hoher W‘keit)
- **Der maximale Abstand zweier Peers ist $O(\log n\, 2^m/n)$**
  - Jeder Peer, der mit einem seiner Finger auf diese Linie zeigt, erhöht den Eingrad des nachstehenden Peers.
  - Die Gesamtlänge der Streckenabschnitte, wo solche Peers liegen ist $O(\log^2 n\, 2^m/n)$
  - Damit ist $w=O(\log^2 n)$



Einfügen von Peers

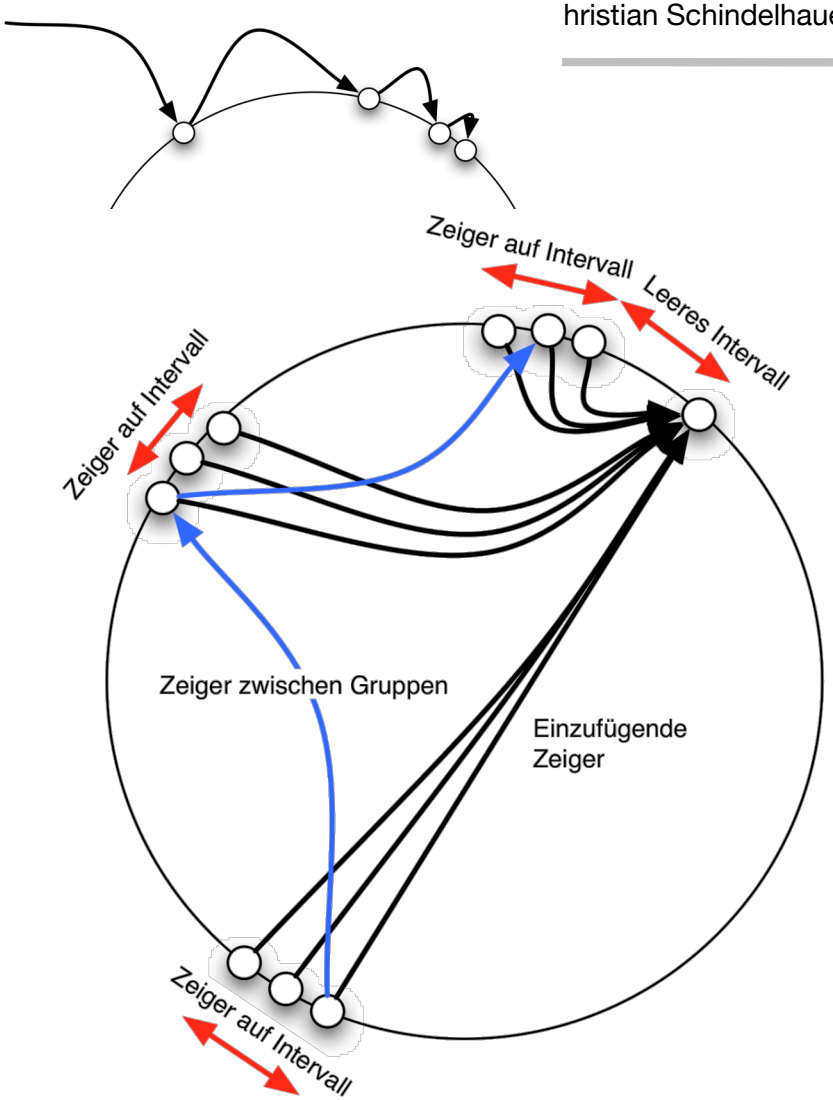
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - $O(\log^2 n)$ Nachrichten genügen mit hoher W'keit, um Peers in CHORD aufzunehmen

Einfügen von Peers

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
hristian Schindelhauer

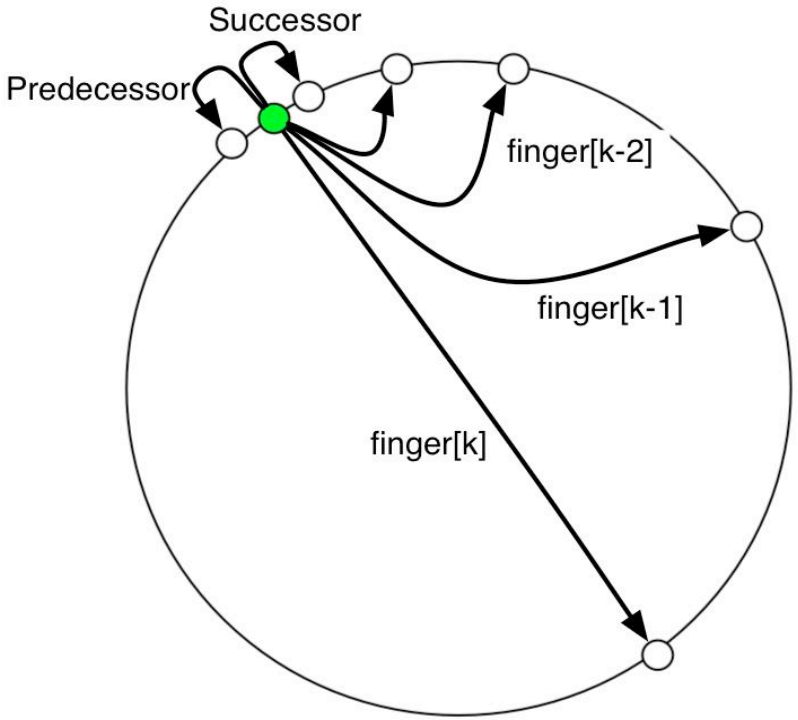
- **Zuerst wird Zielgebiet in O(log n) Schritten gesucht**
- **Die ausgehenden Zeiger werden vom Vorgänger und Nachfolger übernommen und angepasst**
  - Die Zeiger müssen jeweils um bis zu O(log n) Schritte entlang des Rings angepasst werden
- **Der Eingrad des neuen ist mit hoher W'keit $O(\log^2 n)$**
  - Zu suchen kostet jeweils O(log n)
  - Diese sind jeweils in Gruppen von maximal O(log n) benachbart.
  - Damit fallen nur O(log n) Suchen á Kosten O(log n) an
  - Die Aktualisierung hat jeweils konstante Kosten



Die Datenstruktur von Chord

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Für jeden Knoten b:**
  - successor: Nachfolger
  - predecessor: Vorgänger
  - Für i ∈ {0,..,m-1}
    - Finger[i] := Der Knoten der dem Wert $r_{\mathcal{V}}(b+2^i)$ folgt
- **Für kleine i werden die Finger-Einträge immer gleich**
  - Nur unterschiedliche Fingereinträge werden gespeichert
- **Chord**
  - benötigt O(log n) Hops für Lookup
  - benötigt $O(\log^2 n)$ Hops für das Einfügen oder Entfernen von Peers



Routing-Techniken für CHORD: DHash++

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, Robert Morris (MIT)**
  „Designing a DHT for low latency and high throughput“, 2003

- **Idee:**
  - Betrachte CHORD
- **Optimiere Routing durch**
  - Datenlayout
  - Rekursion (statt Iteration)
  - Nächste Nachbarauswahl
  - Replikation versus Kodierung von Daten
  - Fehlerkodierungs-optimierter Lookup
- **Füge geeignetes Transport-Protokoll hinzu**
  - Striped Transport Protocol (statt STP)

Datenlayout

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

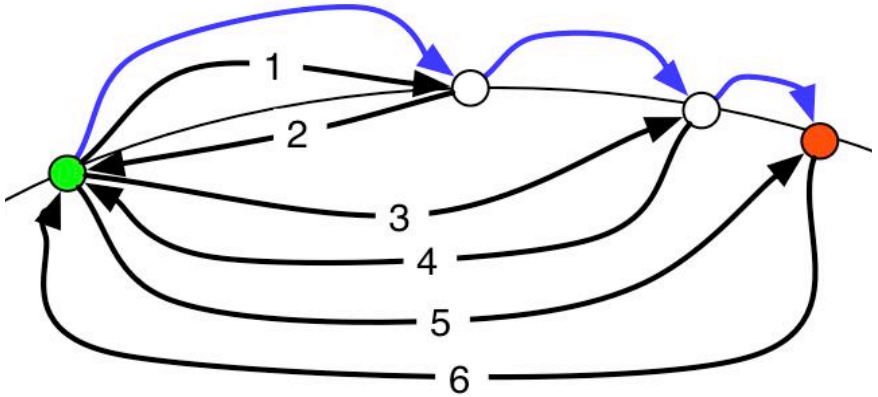
- **Sollen Daten überhaupt verteilt gespeichert werden?**

- **Alternativen:**
  - Key location service
    - Speichern der Referenzinformation auf Dokumente
  - Verteilte Datenspeicherung
    - Verteilung der Dokumente auf die Peers
  - Verteilte Speicherung von Datenblöcken, d.h.
    - Entweder Caching von Datenblöcken
    - oder blockweise Verteilung aller Daten über das Netzwerk
    - Weitere Alternative (hier nicht betrachtet):
      - Semantischs Partitionierung und Speicherung von Daten
- **DHash++ verwendet blockweise (unsemantische) Verteilung aller Daten über das Netzwerk**

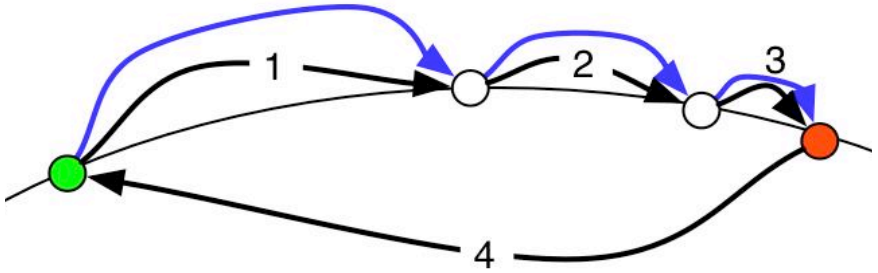
Rekursiver versus Iterativer Lookup

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Iterativer Lookup**
  - Lookup-initiierender Peer erfragt sukzessive die nächsten Nachbarn
  - und führt die weitere Suche selber fort



- **Rekursiver Lookup**
  - Jeder Peer leitet die Suchanfrage sofort selber weiter
  - Dier Zielpeer antwortet dann dem Lookup-Initiator direkt

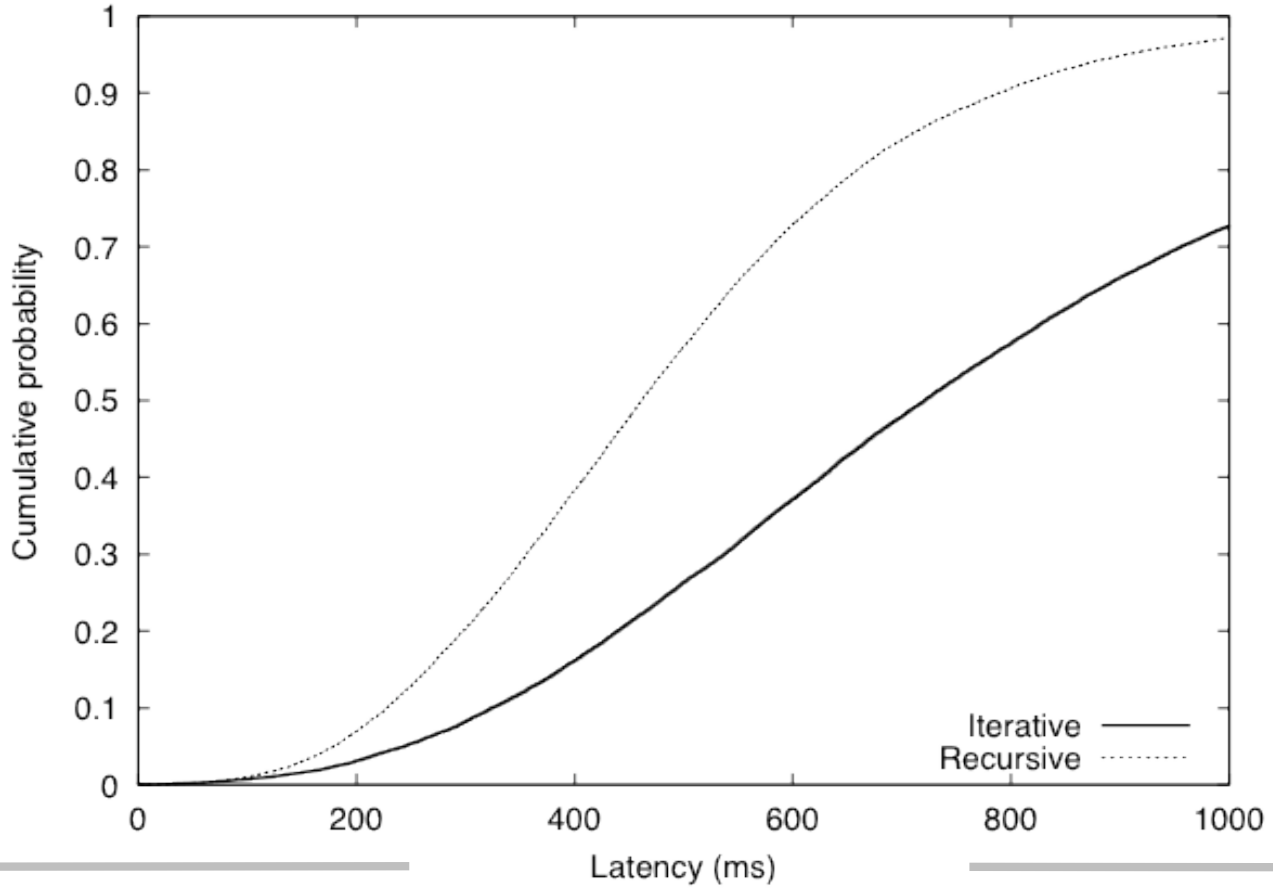


- **DHash++ wählt rekursiven Lookup**
  - da Lookup fast um Faktor zwei schneller

Rekursiver versus Iterativer Lookup

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **DHash++ wählt rekursiven Lookup**
  - da Lookup fast um Faktor zwei schneller
- **Abbildung zeigt kumulative Verteilung der Latenzzeiten aus Simulationen**



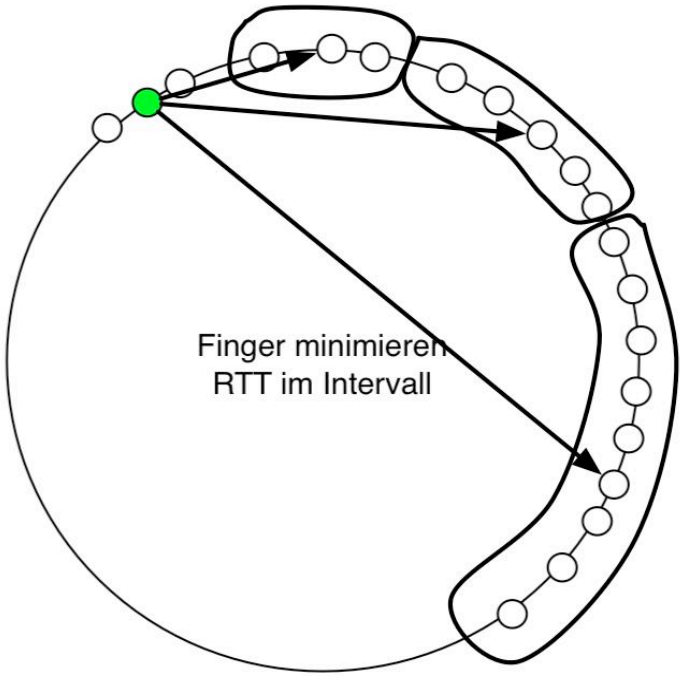
Nächste Nachbarauswahl

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **RTT: Round Trip Time**
  - ist die Zeit, die benötigt wird, eine Nachricht zu einem Peer zu schicken und die Antwort zu erhalten

- **Ansatz von Gummadi, Gummadi, Grippe, Ratnasamy, Shenker, Stoica, 2003, „The impact of DHT routing geometry on resilience and proximity“**
  - Proximity Neighbor Selection(PNS)
    - Die Routingtabelle der Peers wird hinsichtlich der Latenzzeit (RTT) optimiert
  - ProximityRouteSelection(PRS)
    - Nachdem die Routingtabelle erstellt ist, wird der nächste Nachhbar aus der Routing-Tabelle genommen

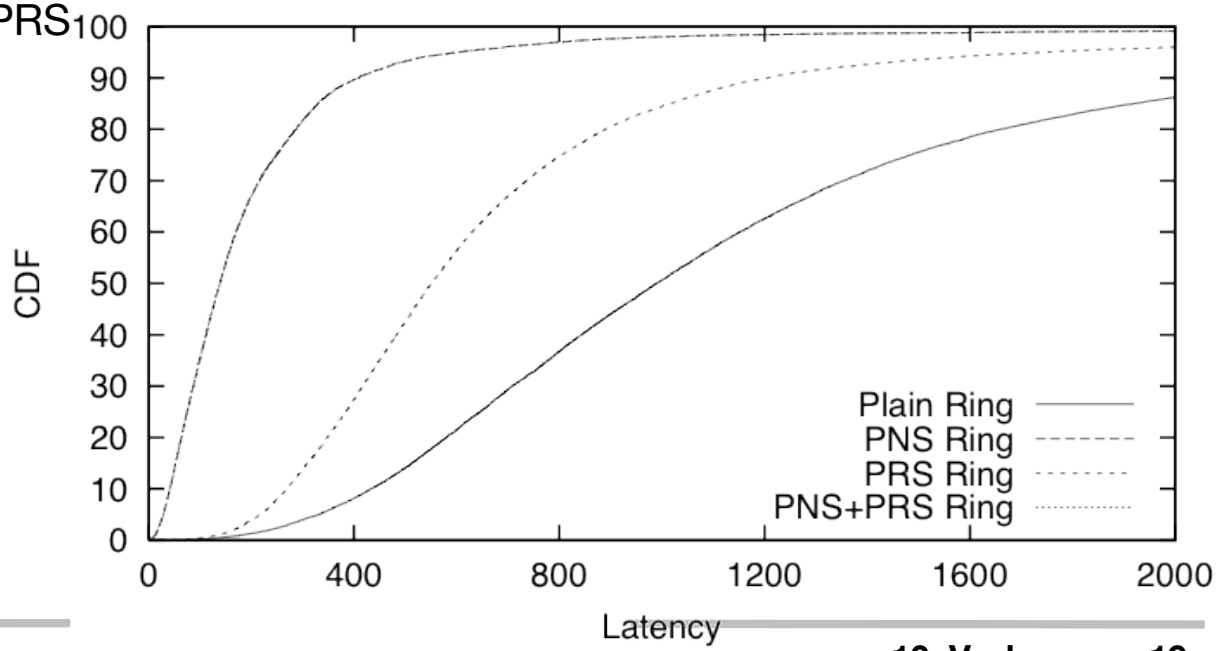
- **Anwendung PNS auf Chord**
  - Wähle für jeden Finger den nächsten Nachbar in einem Intervall



Nächste Nachbarauswahl

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

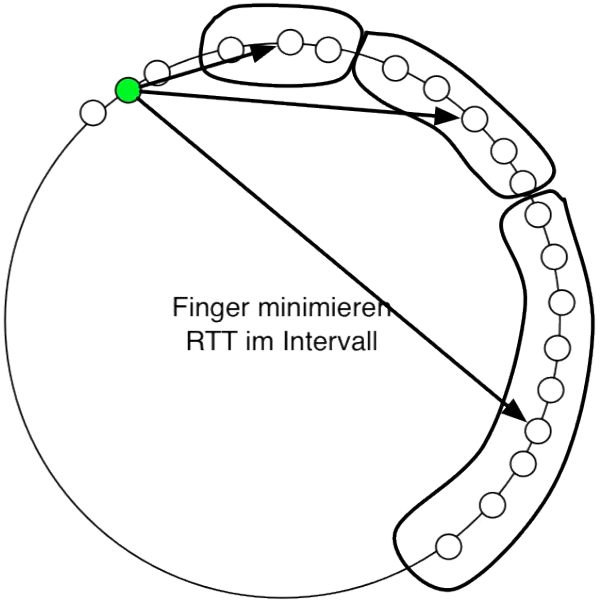
- **Ansatz von Gummadi, Gummadi, Grippe, Ratnasamy, Shenker, Stoica, 2003, „The impact of DHT routing geometry on resilience and proximity“**
  - –Proximity Neighbor Selection(PNS)
    - Die Routingtabelle der Peers wird hinsichtlich der Latenzzeit (RTT) optimiert
  - –ProximityRouteSelection(PRS)
    - Nachdem die Routingtabelle erstellt ist, wird der nächste Nachbarn aus der Routingtabelle genommen
- **Abbildung: Simulation PNS, PRS auf Chord (als Ring bezeichnet)**
  - –PNS so gut wie PNS+PRS
  - –PNS besser als PRS

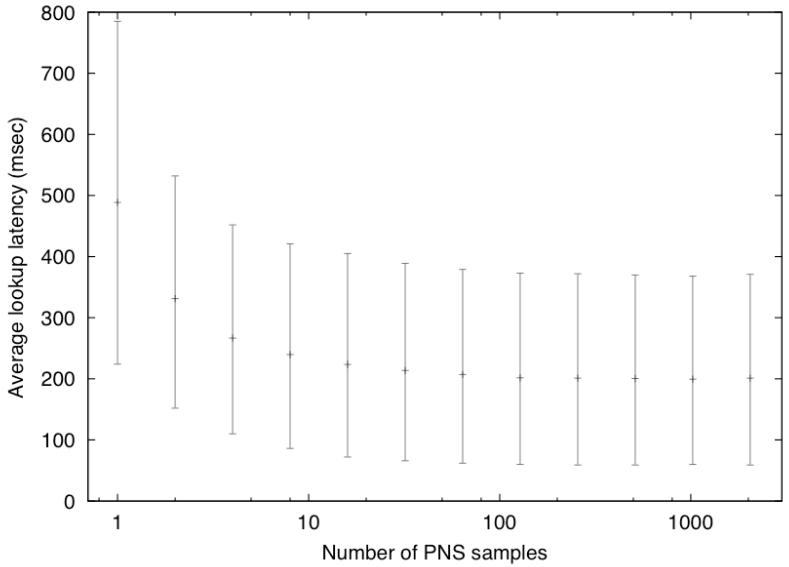


Nächste Nachbarwahl

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **DHash++ verwendet (nur) PNS**
  - Proximity Neighbor Selection
- **Statt im gesamten Intervall nach dem nächsten Nachbarn zu suchen**
  - wird unter einer kleine Auswahl von 16 zufällig gewählten Peers ausgewählt (PNS-Sample
- **Abb. zeigt das (0,1;0,5;0,9)-Perzentil einer solchen PNS-Wahl**





Nächste Nachbarwahl

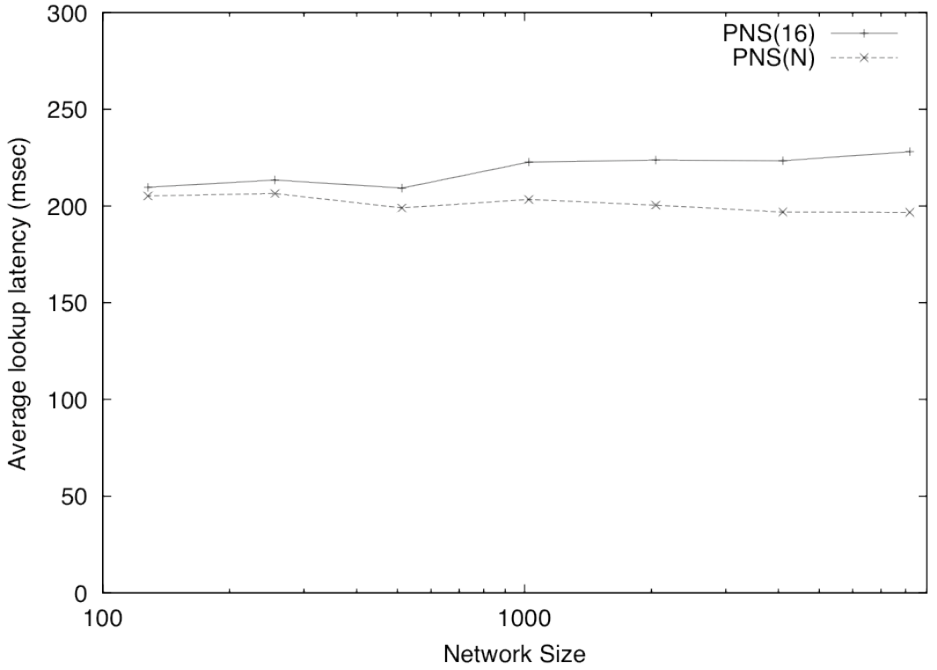
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

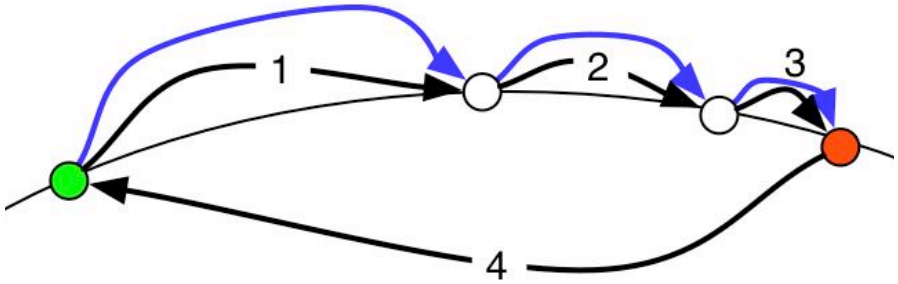
- **DHash++ nimmt als Strategie PNS mit Testmenge von 16 Elementen**
- **Abbildung zeigt die durchschnittliche Latenzzeit bei Lookup, bei Wahl von 16 Elementen im Vergleich mit der kürzesten Wahl**
- **Warum steigt die Laufzeit kaum?**

- **Sei L die durchschnittliche Latenzzeit**
  - Mit 2L sind die beiden letzten Hops am teuersten
  - Der vorletzte erhält im Erwartungswert Latenzzeit L/2, weil zwei Peers zur Auswahl stehen
  - Der davor L/4, dann L/8, ...
- **Damit ist die erwartete Latenzzeit (nach dieser Überschlagsrechnung)**

L + L + L/2 + L/4 + L/8 + L/16 + L/32 ...
= 3 L

- **Simulationen belegen dieses Verhalten**





Kodierung versus Replikation

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Noga Alon, Mike Luby, 1995**
  - führten „Erasure Resilient Codes“ ein
  - Damit kodiert man ein Dokument aus m Fragmenten in n Fragmente, n>m.
  - Aus jeder Kombination der m Fragmente lässt sich das Original-Dokument herstellen
- **Beispiel:**
  - Sei r = 1/4 und ein Dokument bestehe aus 16 Fragmenten
  - Dann vergrößert diese Kodierung das Dokument auf 64 kodierte Fragmente
  - Es wird dabei die vierfache Speichermenge benötigt
  - Um das Dokument abzufragen, können 16 beliebige der 64 kodierten Fragmente genommen werden
- **Hakim Weatherspoon, John Kubiatowicz, 2002,**
  - zeigen (für Oceanstore), dass die Verfügbarkeit von Daten enorm zunimmt, wenn man diese Kodierung verwendet
    - Übungsaufgabe...

Kodierung versus Replikation

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Noga Alon, Mike Luby, 1995**
  - führten „Erasure Resilient Codes“ ein
  - Damit kodiert man ein Dokument aus m Fragmenten in n Fragmente, n>m.
  - Aus jeder Kombination der m Fragmente lässt sich das Original-Dokument herstellen

- **DHash++ verwendet einen m=7, n=14, r = 2 Erasure Resilient Code**
  - Damit nimmt die Ausfallwahrscheinlichkeit um zwei Zehnerpotenzen ab
  - Dafür erhöht sich die Fetchzeit im Vergleich zu m=1, n=2, und r=2, d.h. Download-Zeit, um rund 20%

Integration von Kodierung und Lookup

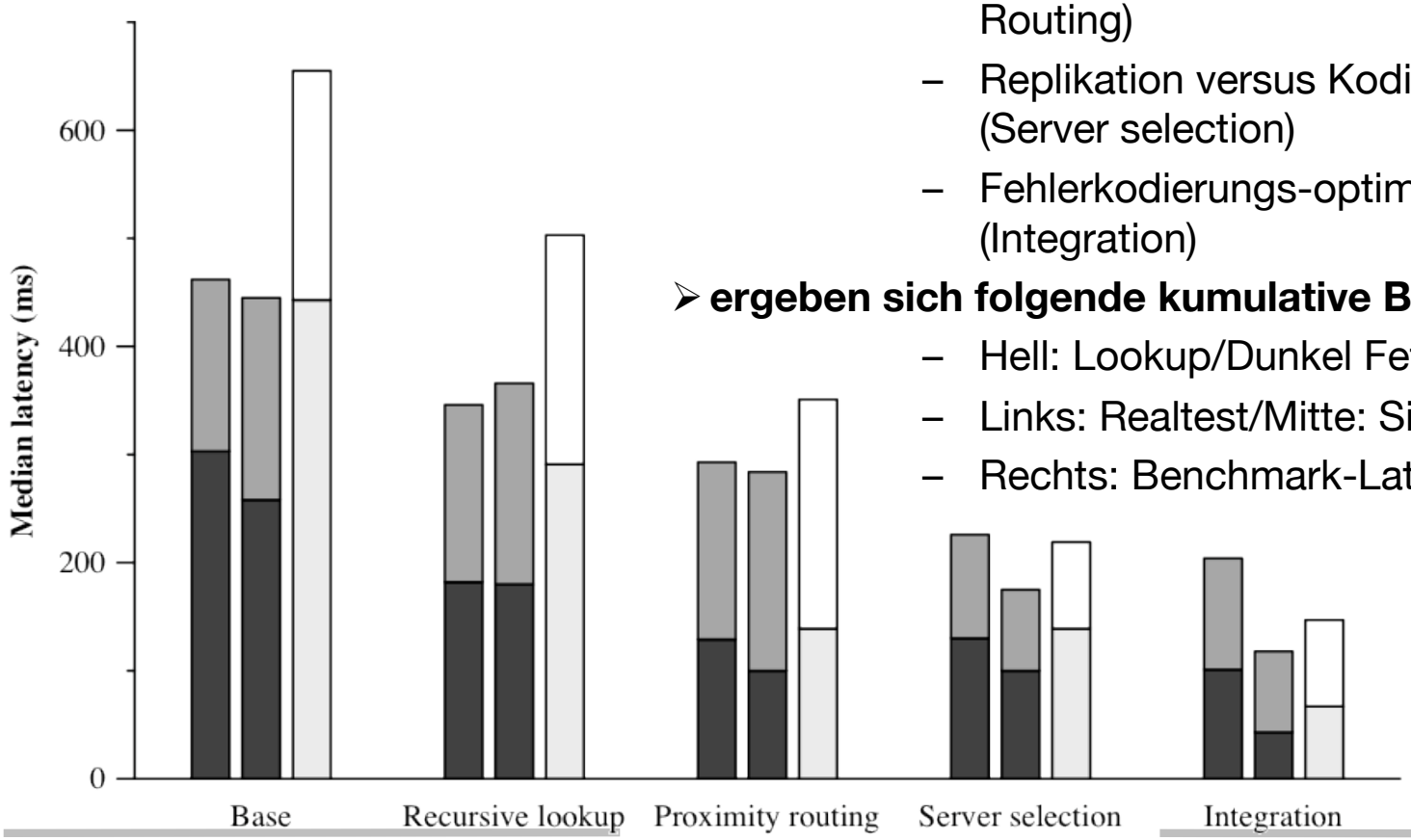
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Die Erasure Resilient kodierten Fragmente werden auf k benachbarten Peers auf den Chord-Ring abgelegt**
- **Problem**
  - Der letzte Hop ist am teuersten
- **Idee:**
  - Es genügen aber k/2 der k benachbarten Peers um das Datum zu erhalten
  - da r = 1/2 gewählt wird
- **Lösung**
  - Die Chord-Suche wird abgebrochen, wenn schon k/2 dieses Intervalls gefunden
  - Dafür wird in der Rekursion die Menge aller Peers, die schon auf den Weg zum Vorgänger des Teilintervalls aufgefunden worden sind, gespeichert
  - Sind k/2 Peers dabei aufgelaufen, wird die Rekursion beendet
  - Die k/2 Peers beantworten die Suchanfrage

Performanzgewinn durch diese Komponenten

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Durch die Komponenten**
  - Datenlayout (Base)
  - Rekursion
  - Nächste Nachbarauswahl (Proximity Routing)
  - Replikation versus Kodierung von Daten (Server selection)
  - Fehlerkodierungs-optimierter Lookup (Integration)
- **ergeben sich folgende kumulative Beschleunigungen**
  - Hell: Lookup/Dunkel Fetch
  - Links: Realtest/Mitte: Simulation
  - Rechts: Benchmark-Latenzmatrix



Das „Underlay“-Netzwerk
Das Internet

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Das Internet (von worldwide inter-networking)**
  - ist das weltweite, offene WAN (wide area network)

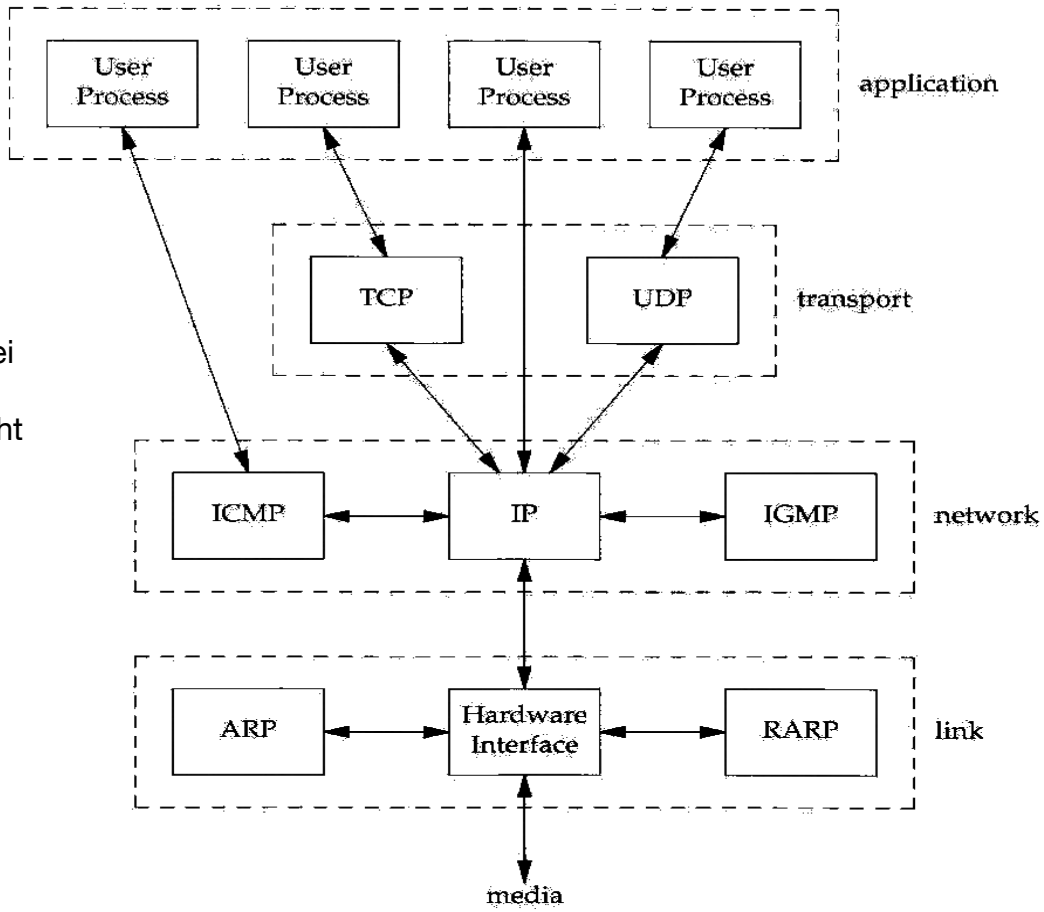
Routing-Schicht

- **IP (internet protocol)**
  + ICMP (internet control message protocol)
  + IGMP (internet group management protocol)

Transportschicht:

Kein Routing: End-to-End-Protokolle

- **TCP (transmission control protocol)**
  - Erzeugt zuverlässigen Datenfluß zwischen zwei Rechnern
  - Unterteilt Datenströme aus Anwendungsschicht in Pakete
  - Gegenseite schickt Empfangsbestätigungen (Acknowledgments)
- **UDP (user datagram protocol)**
  - Einfacher unzuverlässiger Dienst zum Versand von einzelnen Päckchen
  - Wandelt Eingabe in ein Datagramm um
  - Anwendungsschicht bestimmt Paketgröße
  - Versand durch Netzwerkschicht



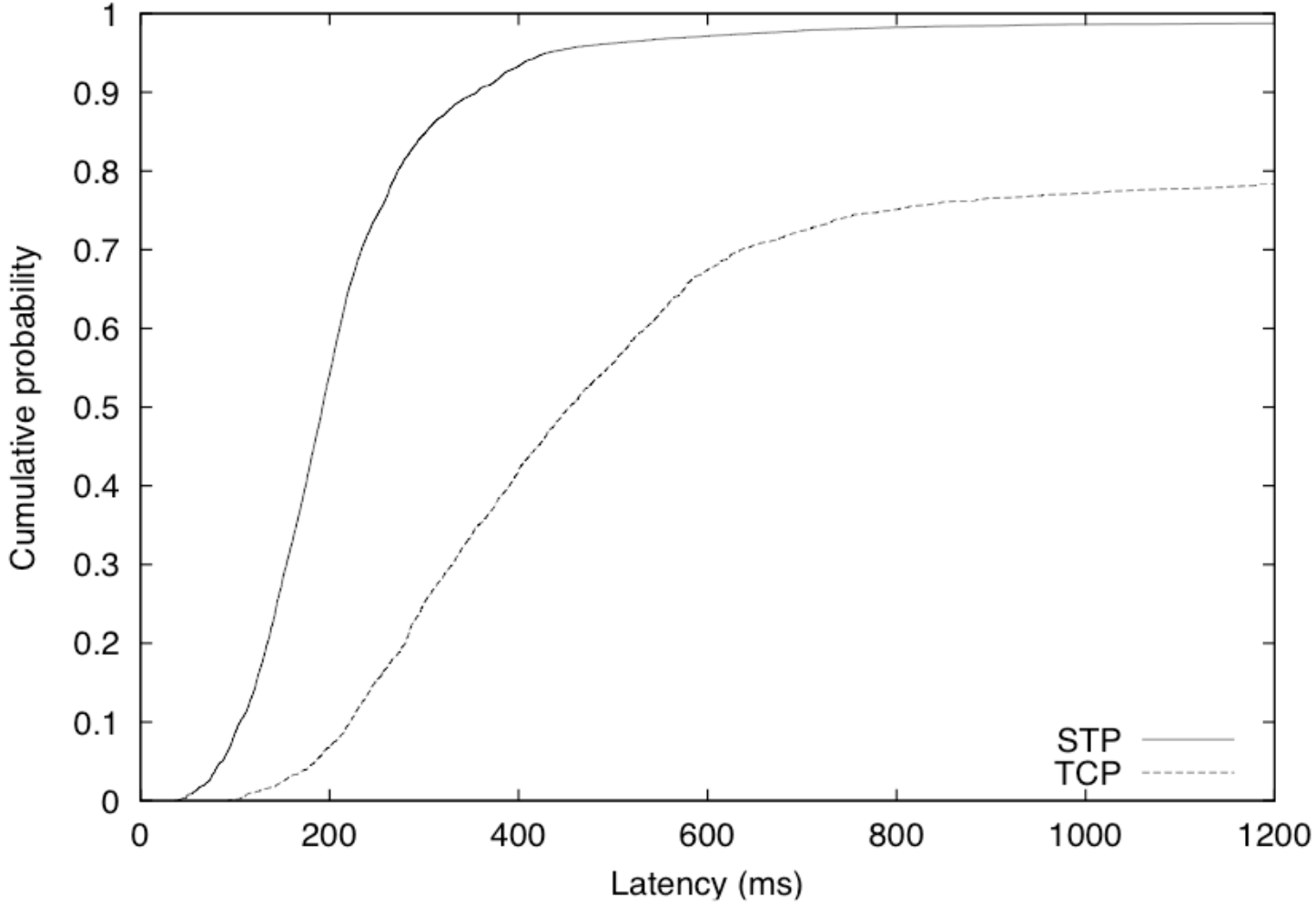
Ein eigenes Transport-Protokoll

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Probleme mit TCP**
  - TCP passt die Bandweite an und startet bei einem Paket pro RTT („slow start“)
  - Dadurch erhöht sich die Latenzzeit
  - TCP arbeitet erst effizient bei einer kleinen Anzahl von langlaufenden Verbindungen
- **Probleme mit UDP**
  - UDP ist unzuverlässig (kein Acknowledgment)
  - Im Routing-Layer werden Pakete absichtlich (!) gelöscht
  - UDP wird als Basis für eigenes Protokoll verwendet werden
- **DHash++ verwendet (eigenes) Transport-Protokoll STP**
  - Striped Transport Protocol (STP)
    - verwendet Window-control mit Datenratenanpassung ähnlich wie TCP
    - hat kein TCP fast-retransmit
      - TCP verwendet implizites Wissen über die Paketordnung, um verlorengegangen Pakete schnell nachzusenden
      - Wegen der stark variierenden Latenzzeiten in CHORD wird die chronologische Ordnung der Pakete zerstört

Vergleich von TCP und SCP in DHash++

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



Diskussion DHash++

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Kombiniert eine Reihe von Techniken zur Verbesserung**
  - der Latenzzeit des Routings
  - der Zuverlässigkeit des Datenzugriffs
- **Dies betrifft die Bereiche**
  - Latenzoptimierte Routing-Tabellen
  - Redundante Datenkodierung
  - Aufbau der Datensuche
  - Transportschicht
  - Integration der Komponenten
- **Alle diese Komponenten können auf andere Peer-to-Peer-Netzwerk angewendet werden**
  - Einige stammen schon daher (z.B. Datenkodierung aus Oceanstore)

- **DHash++ stellt damit eines der modernsten Peer-to-Peer-Netzwerke da**

Ende der 10. Vorlesung

Peer-to-Peer-Netzwerke

Christian Schindelhauer

schindel@informatik.uni-freiburg.de

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer