HUMBOLDT-UNIVERSITÄT ZU BERLIN

Peer-to-Peer-Systeme

Teil IV: Erste DHTs – CAN und Chord

Björn Scheuermann

Humboldt-Universität zu Berlin
Wintersemester 2015/16

Suche in Gnutella

- Gnutella sucht durch Fluten im Netzwerk, also mit sehr hohem Aufwand
- Um vorhandene Daten sicher zu finden, muss das *gesamte* Netzwerk durchsucht werden

$\Rightarrow$ Sehr ineffizient, skaliert nicht!

- Grund: Die Peers haben keinerlei Information darüber, wo bzw. in welcher „Richtung“ sich ein gesuchtes Datenelement befindet

Wie können wir Overlay-Netzwerke bauen, in denen vorhandene Daten gezielt und effizient gefunden werden können?

Hashtabellen

Zur Erinnerung: Hashtabellen als Datenstruktur für das schnelle Wiederfinden von Informationen

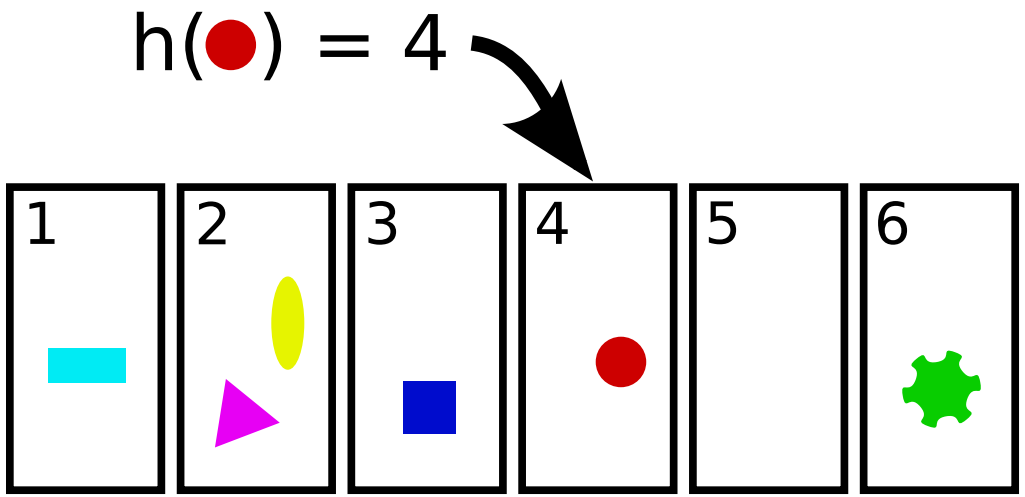
- Datenelemente haben einen *Schlüssel* (Dateiname,. . .)
- Mithilfe einer *Hashfunktion* werden Schlüssel auf Hashwerte fester Länge (i. d. R. Zahlen bzw. Bitstrings) abgebildet

$$h : K \to Q$$

- Normalerweise ist $|K| \gg |Q|$
- Idealerweise ist bei Hashfunktionen die Ausgabe so weit wie möglich unabhängig von der Eingabe
- Normalerweise wird aus Anwendungsperspektive angenommen, dass jedem Schlüssel $x \in K$ ein quasi-zufälliger Hashwert $h(x) \in Q$ zugeordnet wird

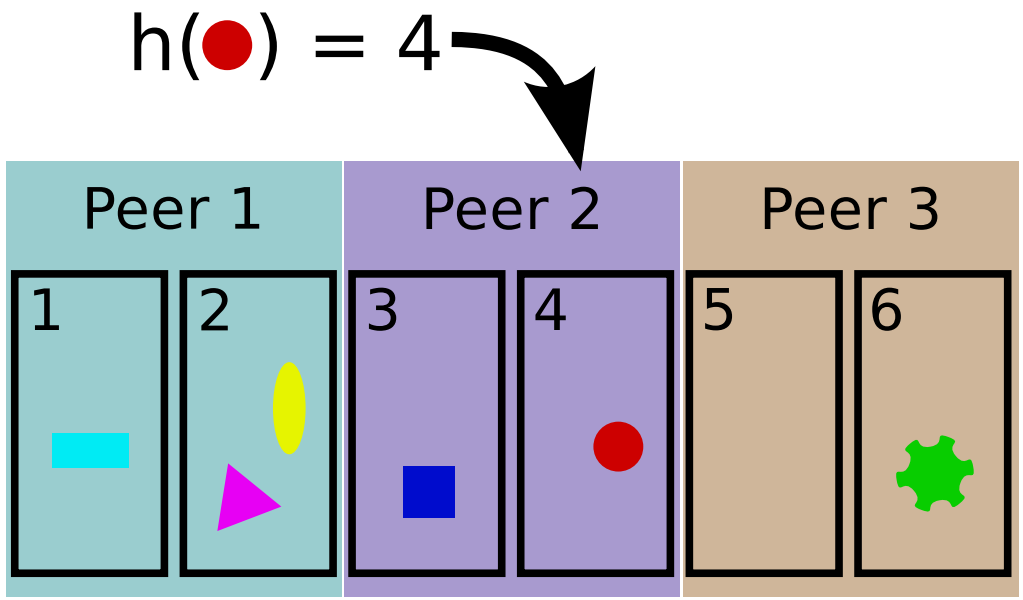
Hashtabellen

- In einer Hashtabelle werden Datenelemente in einem durch den Hashwert ihres Schlüssels festgelegten „Fach“ abgelegt
- Das richtige „Fach“ kann so in einem Schritt ($O(1)$) gefunden werden
- Bietet also ein einfaches Interface, um Informationen über Schlüssel-Wert-Paare zu verwalten



Verteilte Hashtabellen

- Eine verteilte Hashtabelle (Distributed Hash Table, DHT) wendet das Schlüssel-Wert-Prinzip an, um die Informationen systematisch auf die Peers in einem Overlay zu verteilen
- Jeder Peer ist für einen Teil des Hash-Wertebereichs zuständig



Interface einer DHT

Eine verteilte Hashtabelle bietet einer darauf aufsetzenden Anwendung folgende Funktionalität:

- Ablegen von Schlüssel-Wert-Paaren auf einem für den Schlüssel „zuständigen“ Peer im Overlay („put“)
- (Effiziente) Suche nach einem Schlüssel („get“)

Dies wird ermöglicht, indem die Peers im Overlay systematisch so verbunden werden, dass effizientes Routing zu dem für einen gegebenen Schlüssel zuständigen Peer möglich ist.

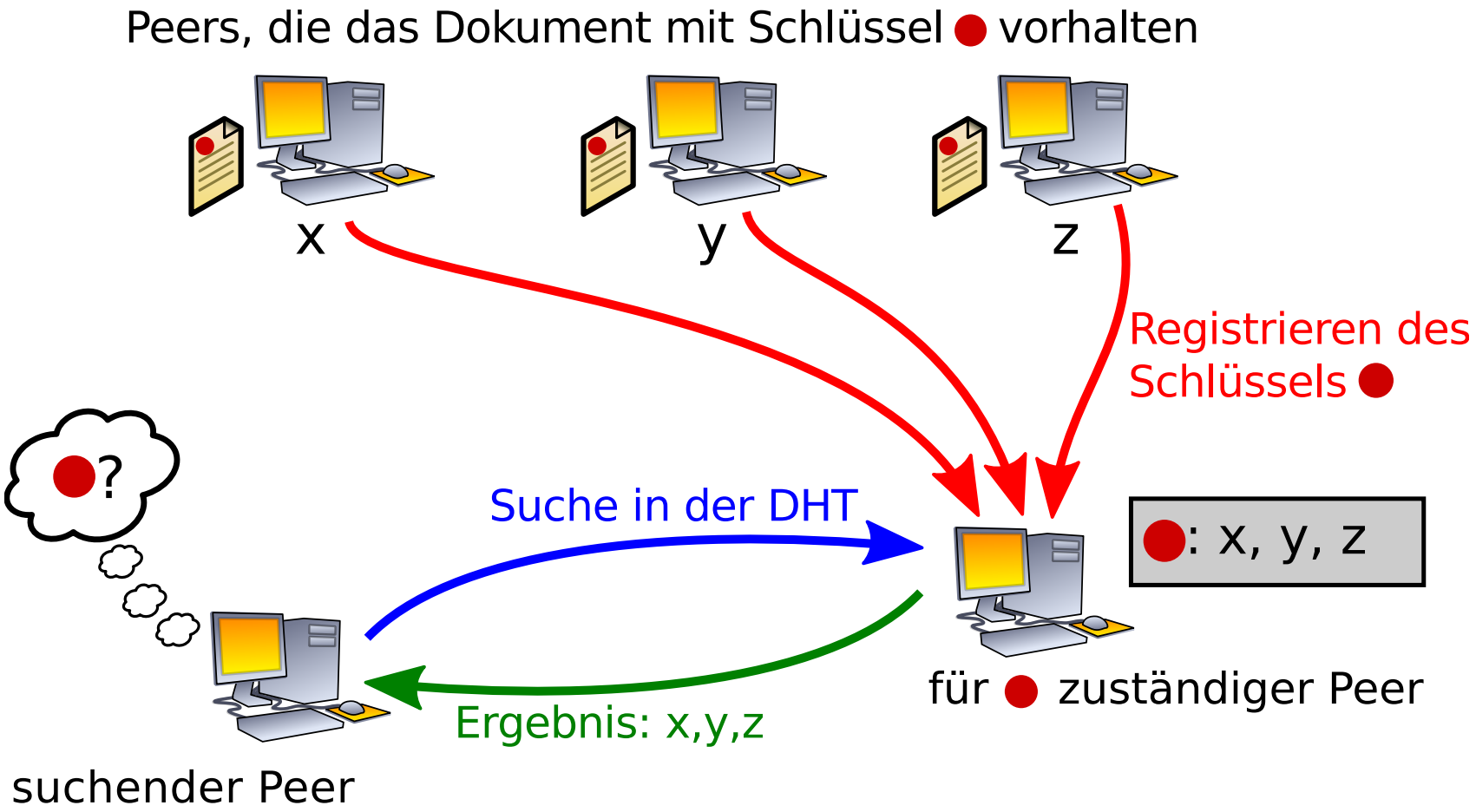
Wir betrachten nun also *strukturierte* Overlays

Welche Vor- und Nachteile einer DHT erwartet ihr im Vergleich zu einem unstrukturierten (Gnutella-artigen) Overlay?

Schlüssel-Indirektion

- Beim Beitritt zu einer DHT wird einem Peer die Zuständigkeit für einen Teil des Schlüsselraums übertragen
- Wichtig: Das bedeutet *nicht* zwangsläufig, dass alle Daten auf diesen Peer übertragen werden müssen
- Bei großen Datenelementen (z. B. Dateien beim Filesharing, Schlüssel = Dateiname, Daten = Dateiinhalt) ist es nicht sinnvoll, die Dateien auf den zuständigen Peer zu kopieren!
- Auf dem für einen Schlüssel zuständigen Peer kann stattdessen die Liste der Peers abgelegt werden, die die Daten bereithalten (das ist analog zum Suchindex in Napster!)

Schlüssel-Indirektion



Herausforderungen einer DHT

- Fragen:
  - Wie werden die Schlüssel auf die Peers verteilt?
  - Mit welchen anderen Peers sollte ein Peer direkt verbunden sein?
  - Wie kann im Overlay gezielt zu einem bestimmten Schlüssel bzw. Hashwert geroutet werden?
- Besondere Herausforderungen dabei:
  - Man möchte mit wenig Zustandsinformation (insbes.: wenige Nachbarn) in jedem Peer auskommen
  - Wenn Peers kommen und gehen (oder sogar unerwartet ausfallen!) müssen die notwendigen „Umbauten“ lokal beschränkt bleiben

Die ersten DHTs

Wir betrachten in dieser Vorlesung zunächst die beiden ersten DHTs:

- CAN (Content Addressable Network)

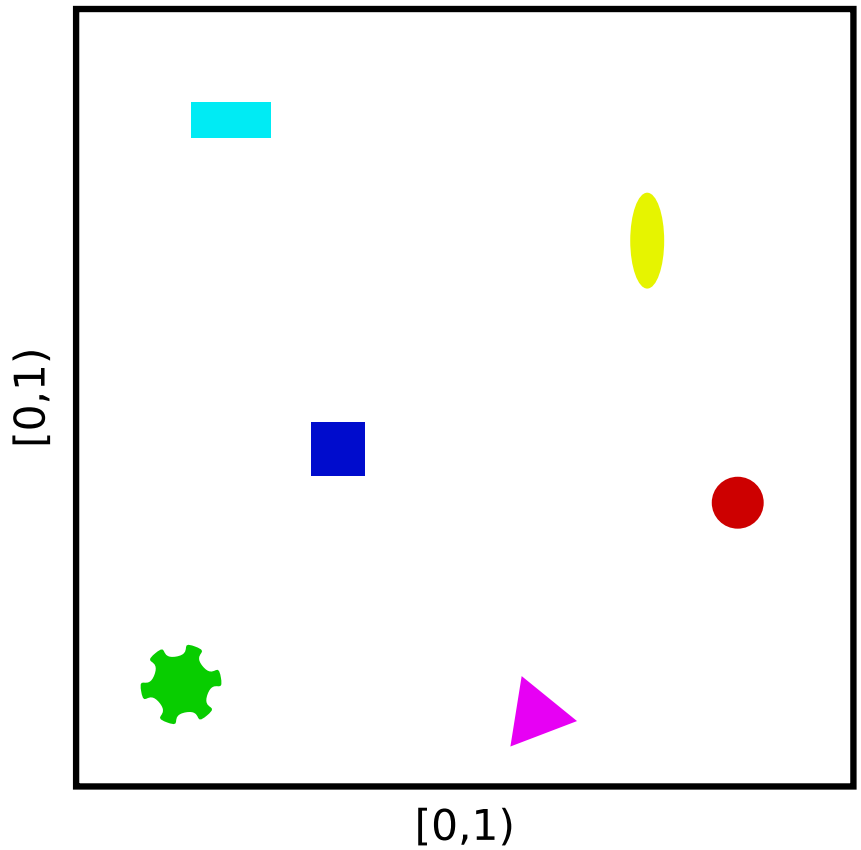
  [Ratnasamy, Francis, Handley, Karp, Shenker: A Scalable Content-Addressable Network, SIGCOMM 2001]

- Chord

  [Stoica, Morris, Liben-Nowell, Karger, Kaashoek, Dabek, Balakrishnan: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, Transactions on Networking, 2003]

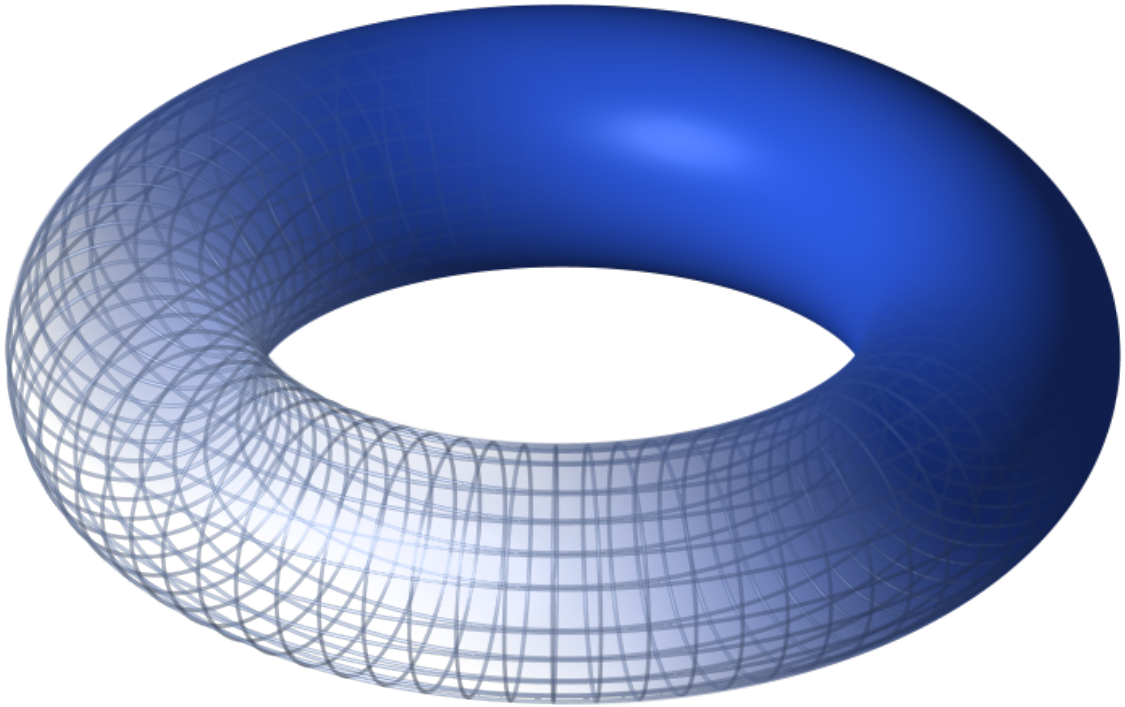
Zweidimensionaler Schlüsselraum in CAN

- In CAN werden die Schlüssel auf Punkte im Quadrat $[0, 1) \times [0, 1)$ abgebildet
- Verwende hierfür zwei Hashfunktionen $K \to [0, 1)$



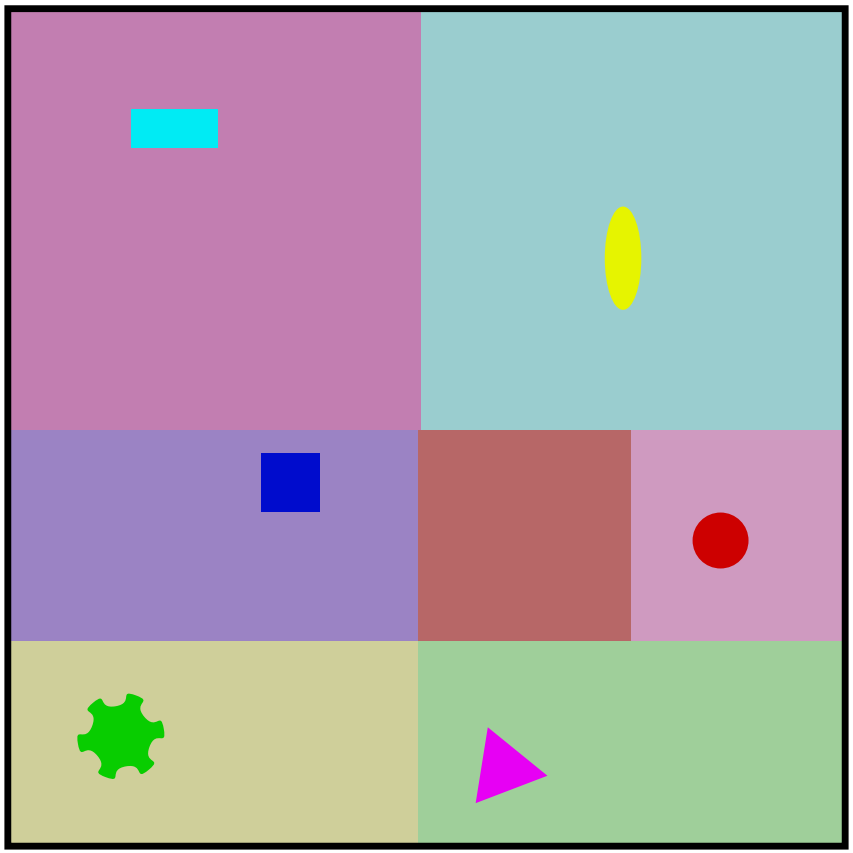
Wraparound: Torus

- Der Schlüsselraum wird als Torus aufgefasst (Wraparound)



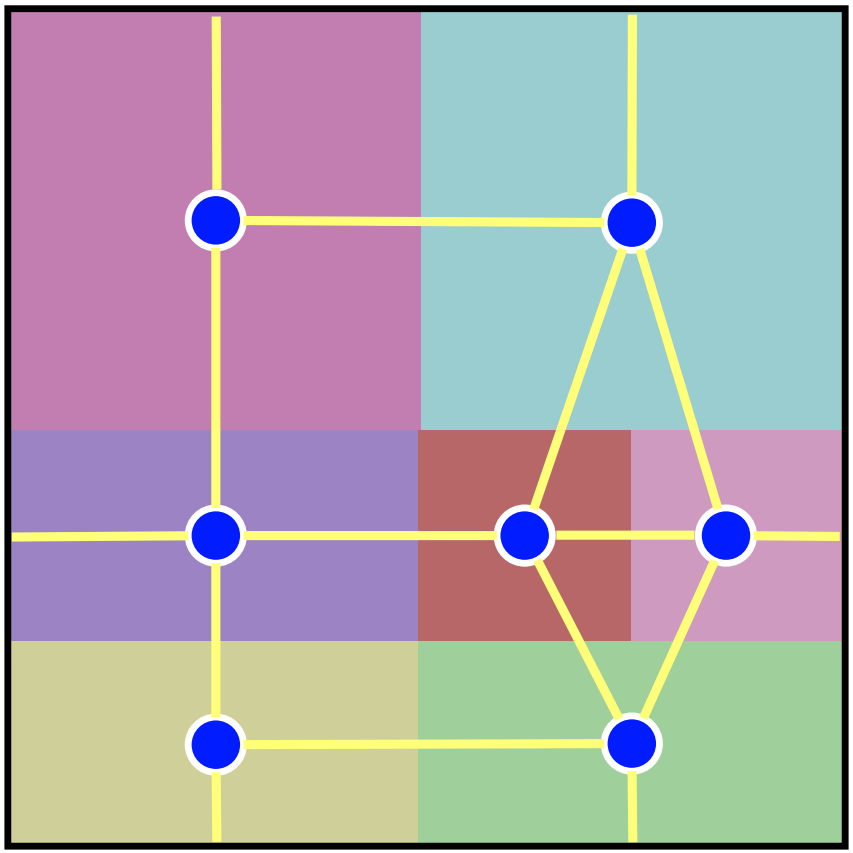
Aufteilung der Schlüssel auf die Peers

- Der Schlüsselraum wird in rechteckige Bereiche aufgeteilt
- Für jeden dieser Bereiche ist jeweils ein Peer zuständig
- Ein Peer verwaltet die Einträge der Schlüssel, die in seinen Zuständigkeitsbereich fallen



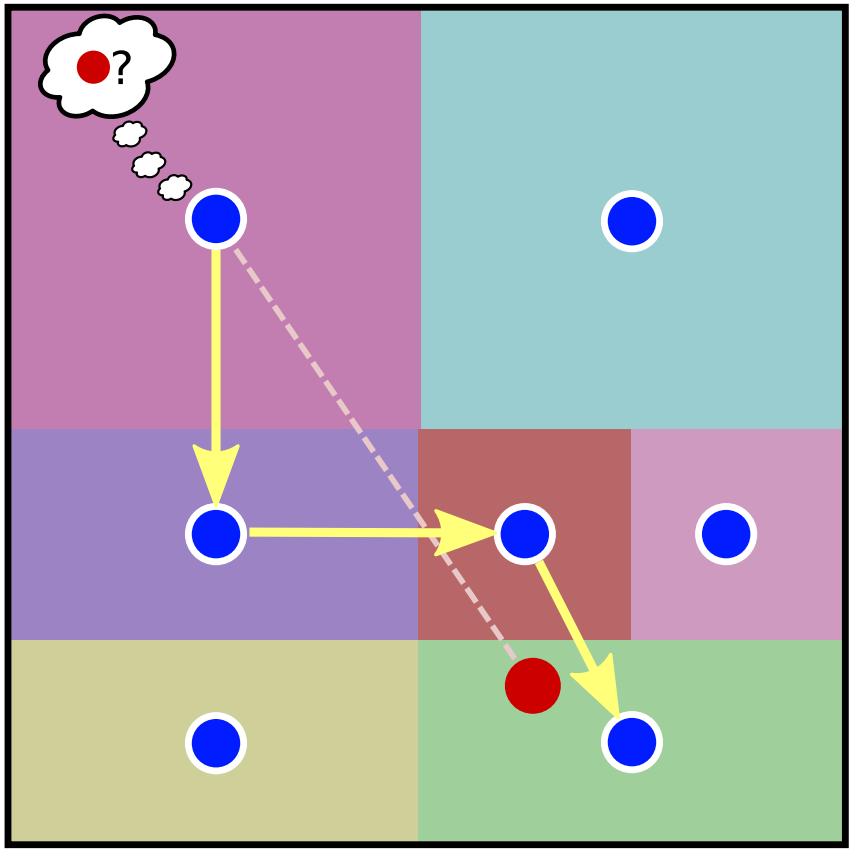
Verbindungen im CAN-Overlay

- Ein Peer hält Verbindungen zu allen Peers mit direkt angrenzendem Zuständigkeitsbereich (gemeinsame Kante)
- Beachte Wraparound-Kanten (Torus!)



Suche im CAN-Overlay

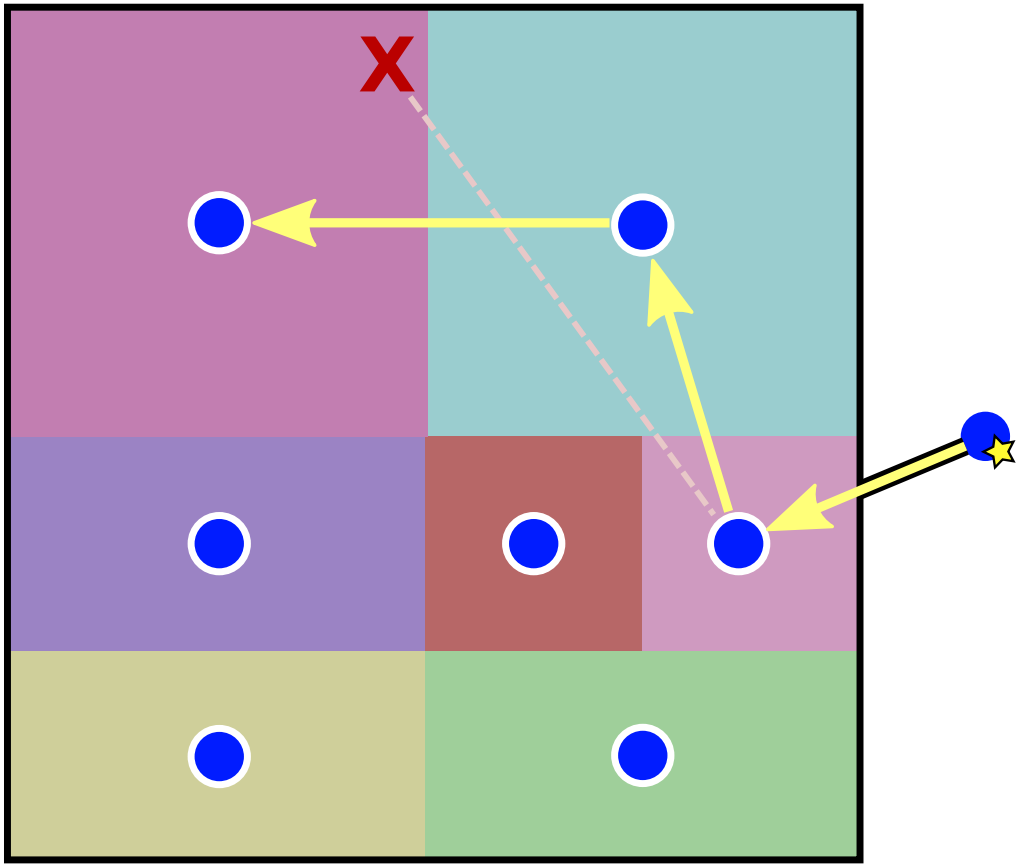
- Zur Suche nach einem Schlüssel wird die Anfrage in Richtung seiner Koordinaten weitergereicht



- Zum Routing genügt also für alle Nachbarn ein Tupel (IP-Adresse, Port, Zuständigkeitsbereich)

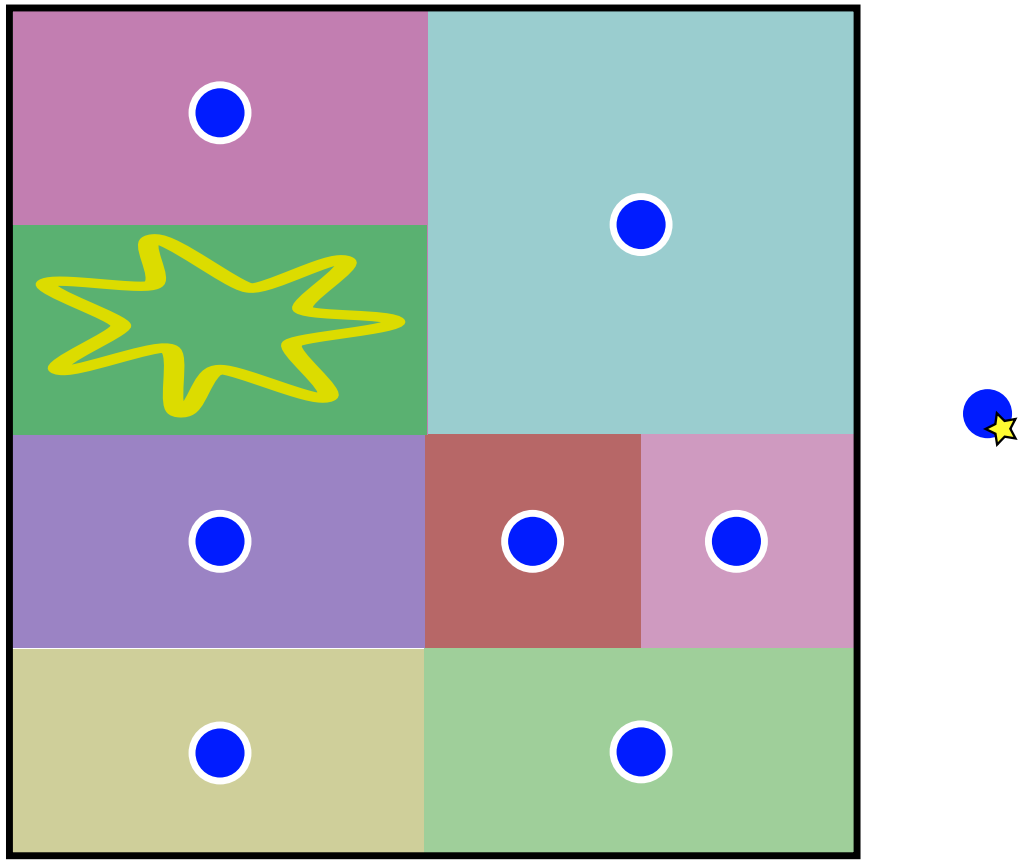
Einfügen eines Peers

1. Neuer Peer kontaktiert einen existierenden Peer
2. Wählt dann einen zufälligen Punkt X aus $[0, 1) \times [0, 1)$
3. Suche nach dem für X zuständigen Peer



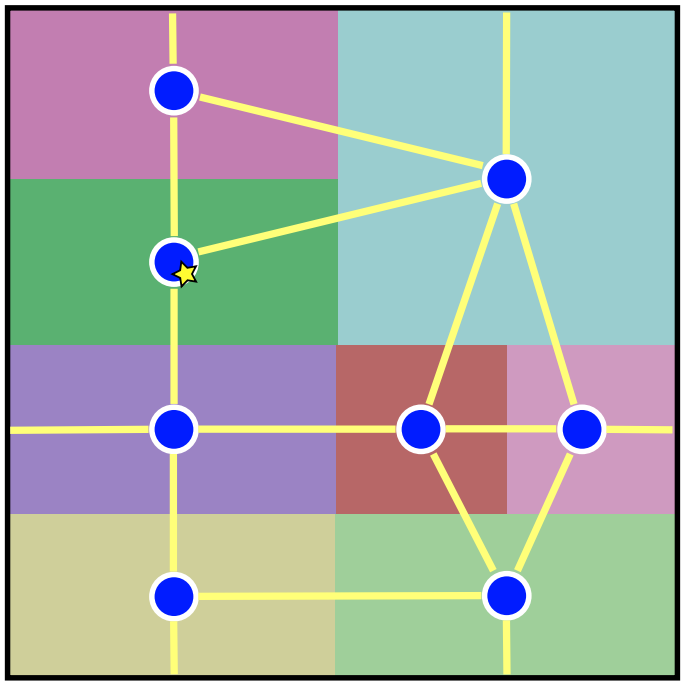
Einfügen eines Peers

4. Zone dieses Peers wird in der Mitte geteilt
5. Für das Teilen gibt es eine festgelegte Reihenfolge der Dimensionen (z. B.: zuerst wird entlang der x-Achse geteilt, dann entlang der y-Achse)



Einfügen eines Peers

6. Einträge aus der übergebenen Hälfte des Zuständigkeitsbereiches werden übertragen
7. Informationen über Nachbarn werden übertragen
8. Den Nachbarn werden die Veränderungen mitgeteilt



Der Peer ist nun vollständig eingebunden, Änderungen im Overlay traten nur lokal auf.

Fairness der Aufteilung

Betrachte die Wahrscheinlichkeit, dass ein Rechteck der Fläche A nach dem Einfügen von n weiteren Peers noch nicht weiter unterteilt wurde.

Diese Wahrscheinlichkeit ist $(1-A)^n \leq e^{-nA}$, also exponentiell klein

Beweis: Beim Einfügen eines Peers ist die Wahrscheinlichkeit, dass ein *anderes* Rechteck gewählt wird, $1-A$. Dass n Peers *alle* ein anderes Rechteck wählen, geschieht mit Wahrscheinlichkeit $(1-A)^n$. Da

$$\forall m > 0 : \left(1 - \frac{1}{m}\right)^m \leq \frac{1}{e},$$

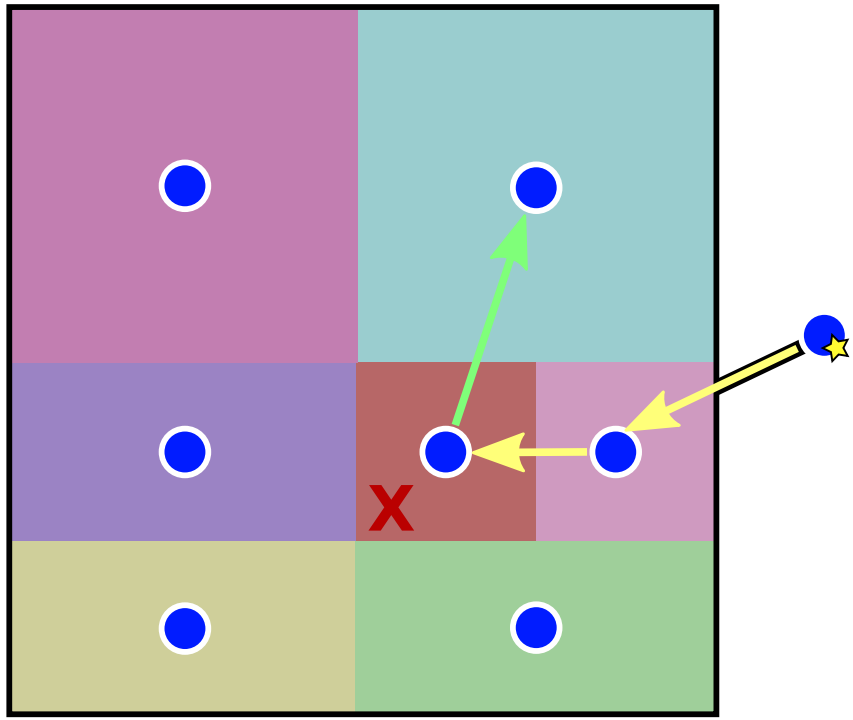
gilt:

$$(1-A)^n = \left((1-A)^{\frac{1}{A}}\right)^{nA} \leq e^{-nA}.$$

Unverhältnismäßig große Gebiete sind also unwahrscheinlich

Volume Balancing

- Die Fairness der Aufteilung kann durch *Volume Balancing* noch weiter verbessert werden
- Beim Einfügen wird der neue Peer wird an Nachbarn mit einer größeren Zone „weitergereicht“, bis ein (lokales) Optimum gefunden wurde



Größe der Zustandsinformation und Routenlänge

- CAN wurde hier im Zweidimensionalen erklärt, lässt sich aber auf einen d-dimensionalen Raum völlig analog erweitern
- Menge der Zustandsinformation für das Routing in jedem Peer (= Anzahl der Nachbarn im Overlay) im Durchschnitt:

$$O(d)$$

- Durchschnittliche Pfadlänge:

$$O(dn^{1/d})$$

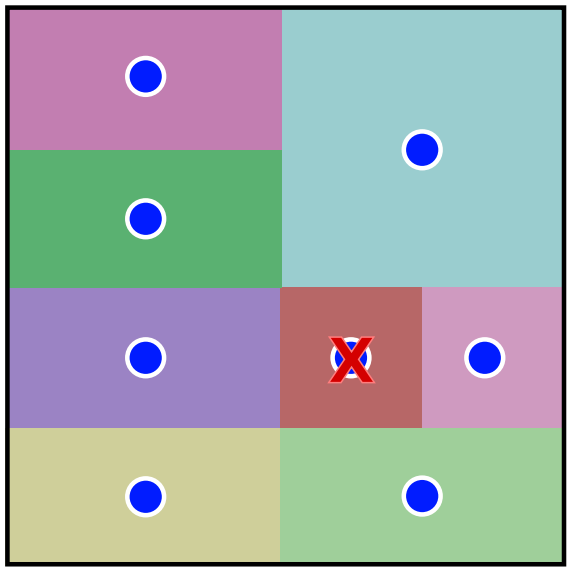
Begründung der durchschnittlichen Zahl von Nachbarn

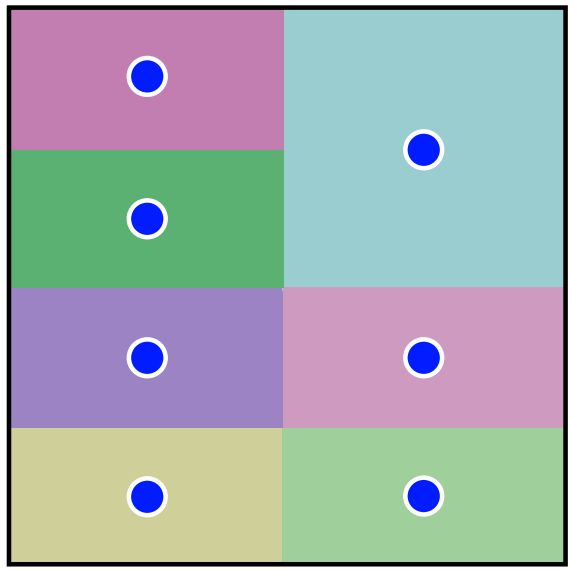
- Dass die durchschnittliche Zahl von Nachbarn in $O(d)$ liegt, lässt sich folgendermaßen begründen:
  - Denken wir die Kanten zwischen benachbarten Zonen als *gerichtete* Kanten (obwohl sie das natürlich nicht wirklich sind)
  - Zwischen unterschiedlich großen benachbarten Rechtecken zeigen die Kanten von der kleineren zur größeren Fläche, sonst ist die Richtung zufällig
  - Nun gehen von jedem Rechteck maximal $2d$ Kanten aus: zu (größeren oder gleich großen) linken und rechten Nachbarn entlang jeder Dimension
  - Bei n Rechtecken gibt es also insgesamt höchstens $2dn$ Kanten
  - Jede Kante verbindet zwei Nachbarn, insgesamt hat ein durchschnittliches Rechteck also höchstens $2 \cdot 2dn/n = 4d$ Nachbarn

Entfernen eines Peers

- Wenn ein Peer das Overlay verlässt, gibt es zwei Möglichkeiten:

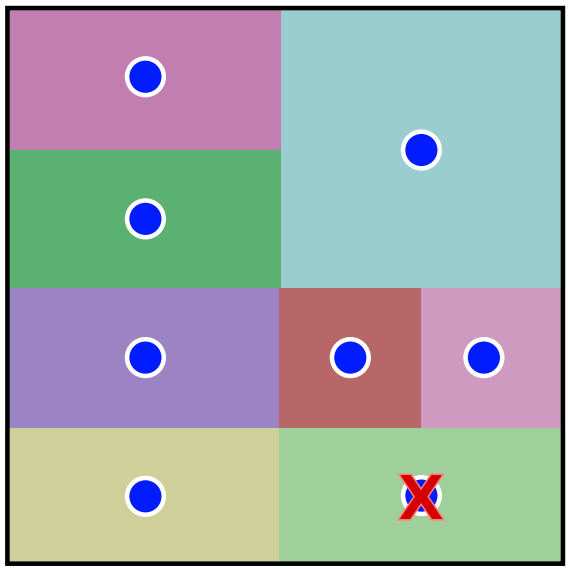
  1. Wenn möglich, sollte er seine Zone mit der passenden anderen Hälfte verschmelzen

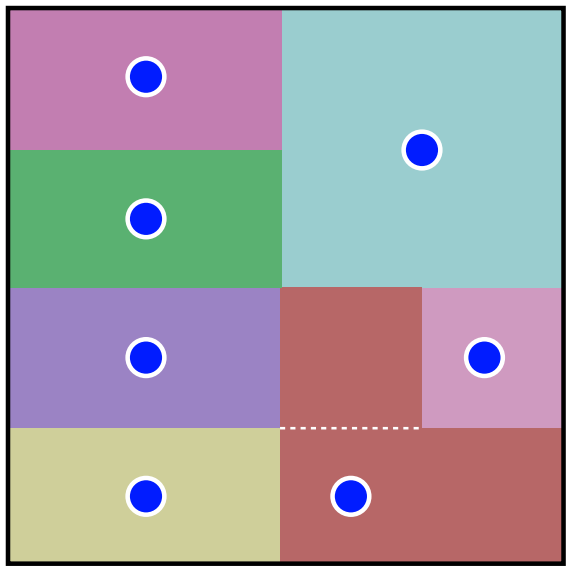




Entfernen eines Peers

- Wenn ein Peer das Overlay verlässt, gibt es zwei Möglichkeiten:
  2. Andernfalls wird das Gebiet an den Nachbarn mit dem kleinsten Zuständigkeitsbereich vergeben, dieser verwaltet dann vorübergehend zwei Zonen





Stand vom 19. November 2015

Kapitel ist noch unvollständig!