Indexierung und Suche von Knotenkapazitäten in Strukturierten P2P Overlay-Netzwerken: Ein Ansatz mit Multidimensionalen Bäumen

Bachelor Arbeit

von

Ivan Tolstun

aus

Nabereznie - Tschelni

vorgelegt am

Lehrstuhl für Rechnernetze und Kommunikationssysteme
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

Oktober 2015

Betreuer:
Jun.-Prof. Dr.-Ing. Kalman Graffi
Andreas Disterhöft M.Sc.

Zusammenfassung

Im Rahmen dieser Bachelorarbeit wird ein Verfahren entwickelt, das alle Kapazitäten (z.B. verfügbare Bandbreite, verfügbarer Speicher, Rechenkapazität etc.) von Knoten innerhalb eines einzigen Overlays durchsuchbar macht. Das Verfahren kann die Indexierung und Suche von Knotenkapazitäten mit mehreren Attributen durchführen.

Zunächst werden die theoretischen Grundlagen behandelt, , die das Fundament der Arbeit bilden. Im zentralen Teil der Arbeit wird das entwickelte Verfahren beschrieben.

Letztendlich wird das in dieser Arbeit implementierte theoretische Verfahren anhand eines Simulators geprüft. Hierzu wird PeerfactSim.KOM verwendet. Auf Basis der Ergebnisse der Tests werden die Schlussfolgerungen gezogen.

Danksagungen

Zunächst möchte ich mich an dieser Stelle bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelor-Arbeit unterstützt und motiviert haben.

Ganz besonders gilt dieses Dank Herrn Andreas Disterhöft, der meine Arbeit und somit auch mich betreut hat.

Inhaltsverzeichnis

List of Algorithms

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

In einem reinen Peer-to-Peer-Netz arbeiten alle Teilnehmer dezentral. Dezentral bedeutet, dass auf keine zentrale Kontroll-, Daten- oder Dienstinstanz in Form eines Servers zurückgegriffen wird. Die Komplexität der Organisation von Knoten in einem dezentralen Netz ist ungleich größer als die in zentralen Netzen.

Für die Zusammenarbeit aller Komponenten der Netze werden Informationen gesammelt, um Auskünfte über den aktuellen Zustand des Netzes zu erhalten. So kann man neben Zustandsinformationen über das System auch Informationen über die Knotenkapazitäten der teilnehmenden Knoten erheben. Jedoch sind die Verfahren, die für die Zustandsinformationen über das System verwendet werden, nicht geeignet für die Knotenkapazitäten der teilnehmenden Knoten. Bei der Suche nach Knotenkapazitäten sollen mehrere Suchkriterien unterstützt werden. Anders ausgedrückt spricht man über einen mehrdimensionalen Raum, in der jede Dimension für ein Suchkriterium steht.

Für einen Zugriff auf Key Y1 oder Key X1, wie in Abbildung 1.1 zu sehen ist, könnten zum Beispiel auf Hashing basierende Verfahren benutzt werden. Es stellt sich die Frage was passiert, wenn ein Zugriff auf Key Y1 und Key X1 nötig ist. Mehrdimensionale Datenstrukturen lassen sich mit mehreren Schlüsselwerten indexieren. Im Unterschied zu den linearen Räumen betrachtet man nun nicht mehr Paare f(x1,x2) , sondern Tupel der Form (x1, y2, z1, ...). Hierbei ist (x1, y2, z1, ...) der (Gesamt-)Schlüssel.

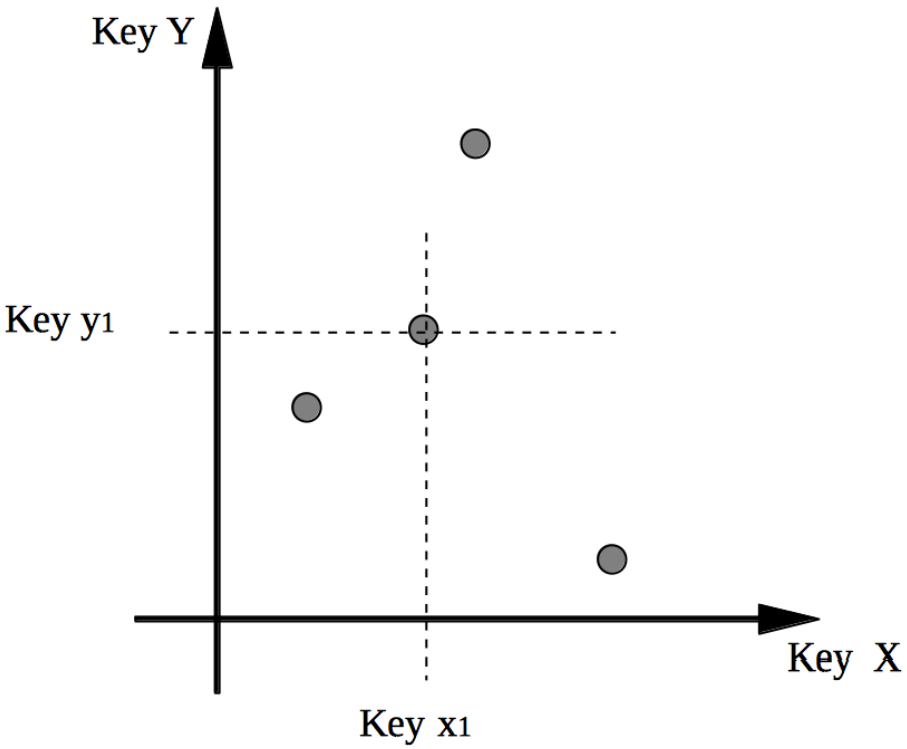


Abbildung 1.1: 2- dimensionale Indexstrukturen

Grundsätzlich stehen für Multikey- Indexing sowohl baumbasierte Verfahren als auch auf Hashing basierende Verfahren zur Verfügung.

- Mehrdimensionales Hashing
- MDB- / KdB-Bäume
- Grid Files
- R- / R*-Bäume

Diese Methoden sind auf lokalen Systemen wirksam, aber ein Peer-to-Peer Netzwerk ist ein verteiltes System und oben genannte Verfahren müssen dafür angepasst werden.

Das Problem liegt demnach darin, ein für Bereichsanfragen hinreichend gutes Verfahren zu finden, das mit mehreren Attributen umgehen kann. Für Indexierung und Suche von Knotenkapazitäten in strukturierten P2P Overlay-Netzwerken ist eine Modifikation von existierenden Verfahren oder eine

neue Methode notwendig.

1.2 Zielsetzung

Zu diesem Zeitpunkt existieren viele gut funktionierende Methoden für den Aufbau eines Peer-to-Peer-Netzes, aber keine effizienten Methoden für die weitere Indexierung und die Suche nach Knotenkapazitäten mit mehreren Attributen. Im ursprünglichen Chord-System sind nur Punktanfragen über einen Attributwert möglich.

Die Zielsetzung dieser Arbeit ist die Entwicklung einer Lösung, die alle Kapazitäten (z.B. verfügbare Bandbreite, verfügbarer Speicher, Rechenkapazität etc.) von Knoten innerhalb eines einzigen Overlays durchsuchbar macht. Hierbei unterscheidet sich die Arbeit von bestehenden Overlays wie beispielsweise Adaptive Chord.

Der entwickelte Algorithmus soll implementiert und getestet werden. Die Tests werden in einem Simulator durchgeführt.

1.3 Aufbau

Hier ist eine kurze Beschreibung der Struktur der Arbeit.

- Im Kapitel 3.1 wird die Theorie beschrieben, die eine Grundlage der vorliegenden Arbeit sind. Anbei sind die Hauptideen der Verwaltung der multidimensionalen Daten zu finden.
- Im Kapitel 3.2 sind die Veränderungen in der Theorie, um die Verfahren zur P2P Netz zu anpassen.
- Das Prinzip der Indexierung der Knoten im Netz wird im Kapitel 3.3 beschrieben. Es wird der Prozess der Konstruktion einer Struktur im Netz beschrieben.
- Das Kapitel 3.4 beschreibt den Suchalgorithmus der Kapazitäten im P2P Netz.
- Testergebnisse dieses Verfahrens werden in Kapitel 4 vorgestellt. Dieser Abschnitt enthält die praktischen Ergebnisse der Implementierun von der in dieser Arbeit entwickelte Verfahren.

Kapitel 2

Verwandte Arbeiten

Die grundlegende Idee, Kapazitäten durchsuchbar zu gestalten, ist nicht neu:

[GMHS07] "Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems / SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems"

P^3R^3O.KOM (eingebaut in SkyEye.KOM):

- sammelt Kapazitätsinformationen in einem baumbasierten Over-Overlay ein
- proaktiv
- eigene Kapazität wird an den Vaterknoten gesendet, der die Daten wiederum wiederrum weiter Richtung Wurzel sendet
- push-Verfahren in Richtung Wurzel
- Anfragen (Queries) werden an den eigenen Vater gerichtet

Anfragen haben die Form:

- Suche n Peers, die mindestens X der Kapazität 1, Y der Kapazität 2, ... haben
- Wenn der Vater die Anfrage beantworten kann -> gib Antwort, ansonsten Weiterleitung an den Vater (also Großvater des anfragenden Knotens)
- rekursiver Queryansatz

Dies bedeutet, dass bei einer Anfrage nicht alle verfügbaren Knoten getestet werden, sondern nur eine Teilmenge und zwar die der Vaterknoten kennt.
Wir streben eine Lösung an, wo jede initiierte Anfrage auf Basis aller verfügbaren Knotenkapazitäten beantwortet wird.

[Gra11]Ädding Capacity-Aware Storage Indirection to Homogeneous Distributed Hash Tables"

AH-Chord (Adaptive Heterogeneous Chord):
Erstelle ein Chord-Overlay pro Kapazität (z.B. Bandbreite). Jeder Knoten in diesem Kapazitäts-Overlay erhält eine ID, die anhand seiner aktuellen Kapazität berechnet wird. D.h., dass man anhand der ID sehen kann wie stark ein Knoten ist. Nun kann man innerhalb des Kapazitäts-Overlay suchen, indem man nach einer ID sucht, die der geforderten Kapazitätsklasse entspricht. Durch den Routing-mechanisms von Chord erreicht man einen Knoten, der >= dieser ID ist, somit auch die Anforderung entspricht.

Problematisch ist hier, dass man ein Kapazitäts-Overlay pro Kapazität braucht. Chord ist bekannt für seine hohe Aufrechterhaltungskosten (proaktiv). Außerdem hoher Churn, da eine Änderung der verfügbaren Kapazität sich auch die ID eines Knotens ändert, was sich in ein Aus- und Wiedereintritt in das Kapazitäts-Overlay äußert.
Wir streben eine Struktur an, in der nach mehreren Kapazitäten gesucht werden kann

Kapitel 3

Die Entwickelte Theorie

3.1 Theoretische Fundierung

Für das Verständnis des Prinzips der in der vorliegenden Arbeit entwickelten Methode werden in diesem Kapitel die theoretischen Grundlagen behandelt, welche die Arbeit fundieren.

Für die Grundlage wird ein kd- Baum genommen, der im Prinzip für das P2P Netz nicht verwendet wird. Aber das Verfahren lässt sich an das P2P Netz anpassen. Der Kd-Baum ist die Struktur der Daten mit der Unterteilung des Raums zum Anordnen der Punkte im k-dimensionalen Raum. Kd- Bäume werden für einige Anwendungen, wie zum Beispiel Suchschlüssel in einem mehrdimensionalen Raum (Suchbereich und die Suche nach dem nächsten Nachbarn) verwendet.

Ein kd-Baum ist eine Verallgemeinerung eines binären Suchbaums, der Punkte im k-dimensionalen Raum speichert. Das heißt, dass man einen kd-Baum verwenden kann, um eine Sammlung von Punkten in der kartesischen Ebene zu speichern. Die Abbildung 3.1 zeigt wie ein kd-Baum funktioniert.

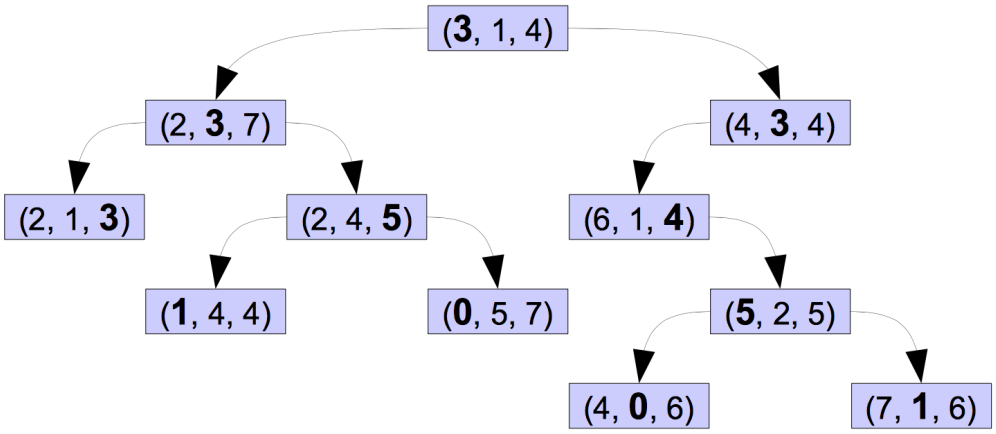


Abbildung 3.1: Ein Beispiel des kd- Baums

Es ist zu beachten, dass die bestimmte Komponente jedes Knotens des kd- Baums fett markiert wird, weil jeder Knoten entlang der fettgedruckten Komponente sortiert wird. Wenn man alle Knoten in ihrem linken Unterbaum anschauen, wird man feststellen, dass die ersten Komponente einen Wert streng weniger als drei hat. Ähnlich wird in dem rechten Teilbaum die erste Komponente jedes Knotens wenigstens drei sein. Dieser Trend setzt sich in dem gesamten Baum fort.

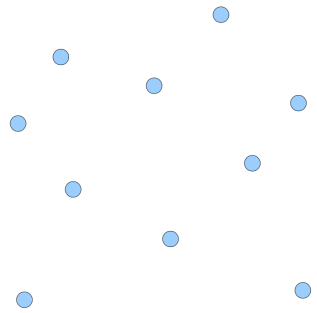


Abbildung 3.2: Die Datenpunkte

Aus diesen Datenpunkten (Abbildung 3.1) wird ein kd- Baum gebaut. Wir wählen einen Knoten, um den Datensatz in zwei Gruppen zu teilen: Eine Gruppe, deren x Komponenten weniger als die zerreißenden Knoten sind und eine zweite Gruppe, deren x Komponenten mindestens so groß wie die zerreißenden Knoten sind. Dann wird der nächste Raum ebenfalls in zwei gleiche Teile geteilt. Die Operationen werden rekursiv wiederholt. Wie in Abbildung 3.3 zu sehen ist.

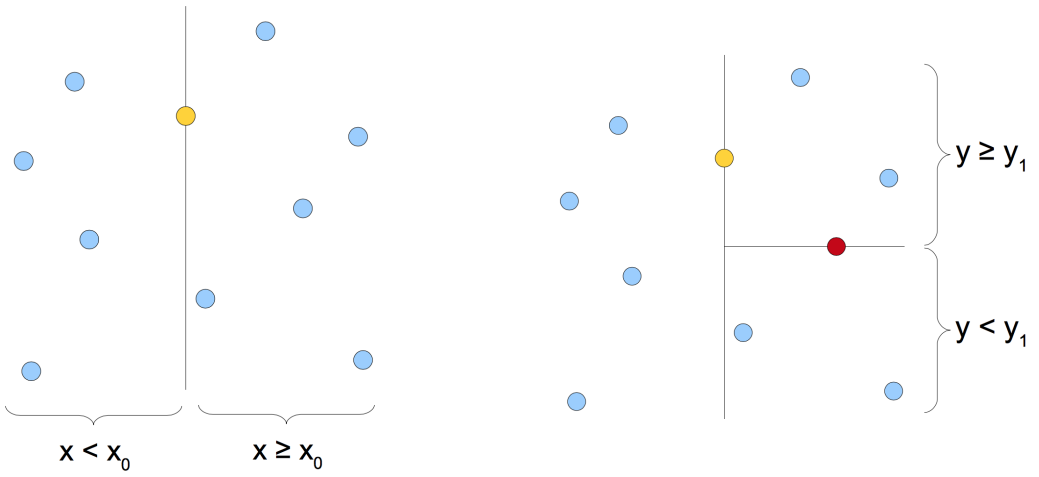


Abbildung 3.3: Zerlegung nach Dimension

Das Prinzip des kd- Baums wird klarer, wenn es mit einem Pseudocode beschrieben wird. Der Pseudocode 1 führt das Prinzip der Konstruktion des Baums vor.

Algorithm 1 Die Struktur des 2-d- Baums:

```
1:  Algorithm BuildKdTree(P, depth)
2:  Begin
3:      if <P enthält nur einen Punkt> then
4:          return ein Blatt, das diesen Punkt speichert
5:      else
6:          if <depth gerade ist> then
7:              Split P mit einer vertikalen Linie L durch dem Mittel der X-Koordinate
8:              in P1 ( links von L oder auf L ) und P2 (rechts von L)
9:          else
10:             Split P mit einer horizontalen Linie L durch das mittlere Y-Koordinate
11:             in P1 ( unter L oder auf L ) und P2 ( oben L )
12:         end if
13:         V_left <– BuildKdTree(P1 ,depth + 1)
14:         V_right <– BuildKdTree(P2 ,depth + 1)
15:         Erstellen Sie eine Knoten V, der L speichert.
16:         Stellen V_left der linken Kind der V und V_right der rechten Kind von V.
17:         return V
18:     end if
19: End
```

Bei der inhomogenen Variante enthalten die inneren Knoten je nach Ebene die Schlüsselinformation der zuständigen Dimension sowie Sohnzeiger auf weitere inneren Knoten. Nur die Blätter verweisen auf den Link der Daten, die jeweils mehrere Links aufnehmen können. In Abbildung 3.4 befinden sich z.B. die Links der Daten C, B und D in einem Blatt.

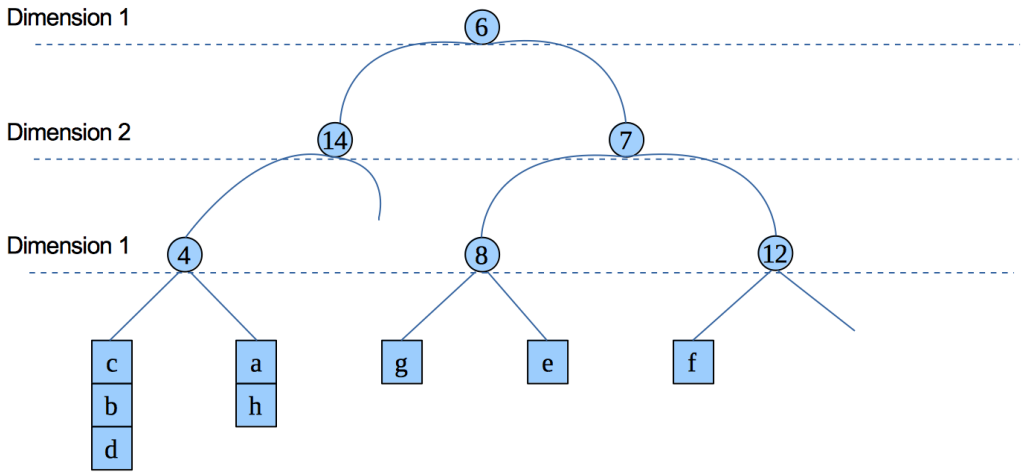


Abbildung 3.4: 2-d-Baum (inhomogen) zu den Datenpunkten a,b,c,d,e,f,g,h

Die Struktur der Knoten ist im Pseudocode 2 zu sehen.

Algorithm 2 Die Struktur eines Knotens:

```
1:  const N=10;                                   ▷ die Zahl der Räume der Schlüssel

2:  sruct Leaf
3:  Begin
4:      int key[N];                               ▷ das Massiv der Schlüssel bestimmt das Element
5:      char *info;                               ▷ die Informationen des Elementes
6:  End

7:  sruct Node
8:  Begin
9:      Leaf i;                                   ▷ nur die Blätter verweisen auf die Datenrecords
10:     int key[N];                               ▷ das Massiv der Schlüssel bestimmt das Element
11:     Node *left;                               ▷ linker Unterbaum
12:     Node *right;                              ▷ rechter Unterbaum
13: End
```

Der Suchalgorithmus ist mit dem Pseudocode 3 zu beschrieben.

Algorithm 3 Kd-Baum Abfrage:

```
1:  Algorithm SearchKdTree(V,R)
2:  Input: Die Wurzel (ein Teilbaum) eines kd-Baums, und eine Reihe R
3:  Output: Alle Punkte auf Blätter unten V, die in dem Bereich liegen
4:  Begin
5:      if <ein Blatt> then
6:          Bericht des bei V gespeicherten Punkt, wenn es in R liegt.
7:      else
8:          if <region(V_right) ist vollständig in R enthalten> then
9:              ReportSubtree(V_right)
10:         else
11:             if <region(V_right) schneidet R> then
12:                 SearchKdTree(V_right,R)
13:             end if
14:         end if
15:         if <region(V_left) ist vollständig in R enthalten> then
16:             ReportSubtree(V_left)
17:         else
18:             if <region(V_left) schneidet R> then
19:                 SearchKdTree(V_left,R)
20:             end if
21:         end if
22:     end if
23: End
```

3.2 Baumstruktur

Für die Klassifikation der Knoten nach ihren Knotenkapazitäten wird der Algorithmus des kd-Baums geändert. Der gesamte mehrdimensionale Raum der Knotenkapazitäten wird durch den kd- Baum unterteilt. Damit könnte jeder Knoten seine Gruppe berechnen, zu der er gehört. Die kleinen Änderungen im Algorithmus des kd-Baums werden mit einem Beispiel in diesem Kapitel beschrieben.

Die Baumstruktur selber verändert sich kaum. Jede Dimension bekommt ein festes Maximum des Werts und jeder Rechner verfügt über die Informationen (über den Maximalwert jeders Raums). Der erste Raum wird laut des Maximalwertes in zwei gleiche Teile geteilt. Der Pseudocode 4 führt das Prinzip vor.

Algorithm 4 Die Zerlegung des zweidimensionale Raums:

```
1:  Algorithm BuildKdTree(Dimension, depth)
2:  Begin
3:      if <Dimension gleich oder weniger 1> then
4:          Ende des Algorithmus
5:      else
6:          if <depth gerade ist> then
7:              Split Dimension mit einer vertikalen Linie L durch dem Mittel der X-Koordinate
8:              in Dimension_1 ( links von L oder auf L ) und Dimension_2 (rechts von L)
9:          else
10:             Split Dimension mit einer horizontalen Linie L durch das mittlere Y-Koordinate
11:             in Dimension_1 ( unter L oder auf L ) und Dimension_2 ( oben L )
12:         end if
13:         V_left <– BuildKdTree(Dimension_1 ,depth + 1)
14:         V_right <– BuildKdTree(Dimension_2 ,depth + 1)
15:         Erstellen Sie eine Knoten V, der L speichert.
16:         Stellen V_left der linken Kind der V und V_right der rechten Kind von V.
17:         return V
18:     end if
19: End
```

Die Abbildung 3.5 zeigt ein Beispiel mit einer Zerlegung des zweidimensionale Raums. Der Algorithmus zum Konstruieren von drei oder mehreren Räume ist identisch.

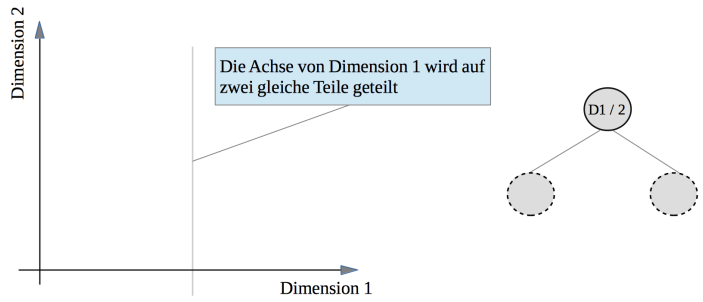


Abbildung 3.5: Der erste Raum wird in zwei gleiche Teile geteilt.

Dann wird der nächste Raum ebenfalls in zwei gleiche Teile geteilt. Wie in Abbildung 3.6 zu sehen ist.

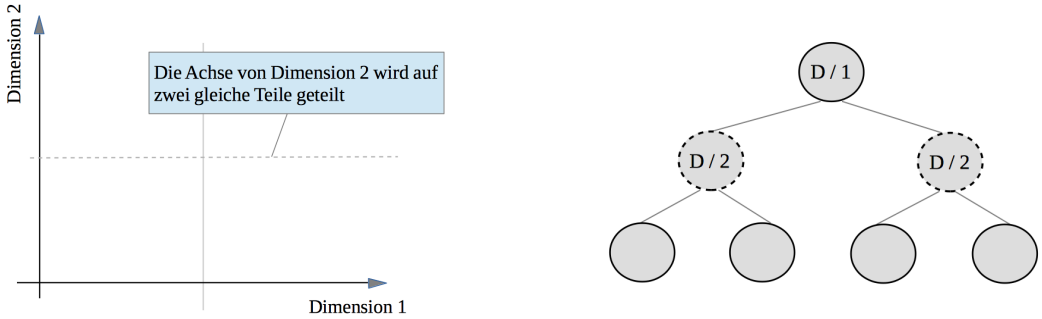


Abbildung 3.6: Der zweite Raum wird in zwei gleiche Teile geteilt.

Die Operation wird rekursiv wiederholt.

3.3 Indexierung

Die Indexierung, die hier verwandt wird, basiert auf einer Ansammlung von Verweisen, die in den Blattknoten und in den Inneren Knoten gespeichert werden. Bedingt durch den Aufbau des Baums sind die Blattknoten für genau eine Kombination von Kapazitäten verantwortlich. Die inneren Knoten hingegen sind folglich für einen Bereich zuständig, wie die Abbildung 3.7 zeigt. Dies können wir zu unserem Vorteil bei der Kapazitätssuche verwenden. Mehr dazu beinhaltet das Kapitel 3.3 SSuche".

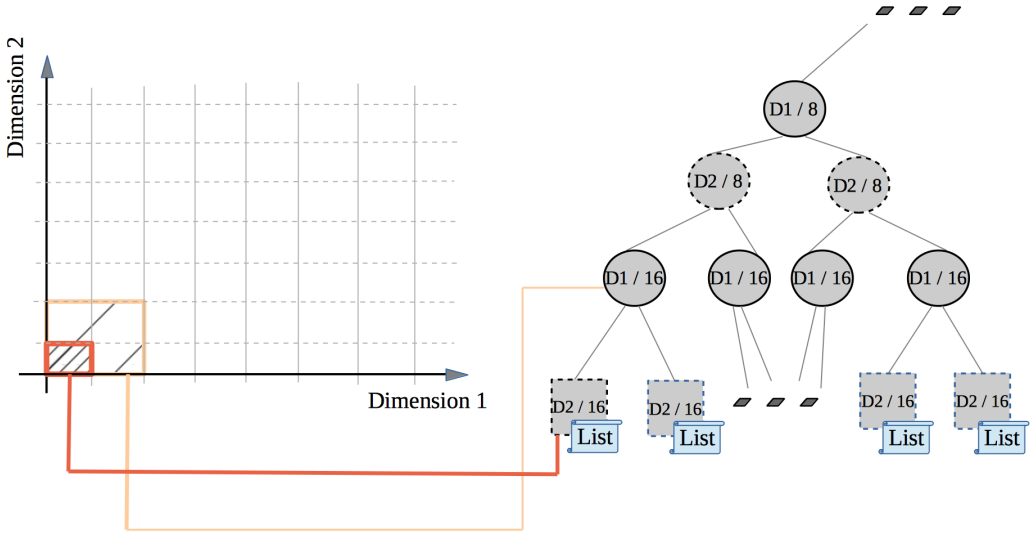


Abbildung 3.7: Verteilung der Verantwortung einzelner Knoten für den Raumbereich

Die Verweise liegen in Form von Bitketten vor und werden weiter "Baumschlüssel"genannt. Der Aufbau des Schlüssels fängt mit der Wurzel an. Deshalb kann jeder Knoten anhand seines Schlüssels die vorherigen Knoten des kd- Baums berechnen. Der Aufbau des Schlüssels wird in der Abbildung 3.8 gezeigt.

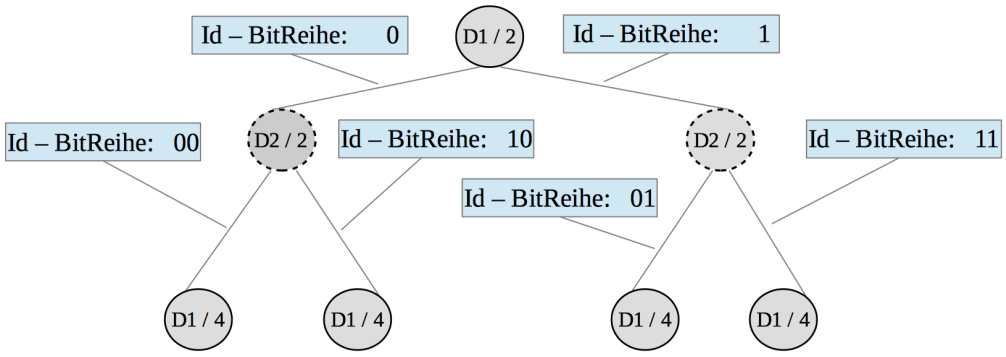


Abbildung 3.8: Verantwortung von Raumbereich

Die Frage ist nämlich wie die Blattknoten von den Kapazitäten der Knoten des P2P Nets erfahren. Jeder Knoten des P2P Nets berechnet anhand des mehrdimensionalen Baums und seiner aktuell verfügbaren Kapazitäten einen Baum-Schlüssel, der auf Blattknoten im Baum zeigt und der für diesen Bereich verantwortlich ist. Dann wird mit der SHA1 Funktion wird ein PeerID im Overlay anhand des

Schlüssels des Baums berechnet. Damit könnte der Knoten des P2P Nets sich bei einem Blattknoten des kd- Baums melden.

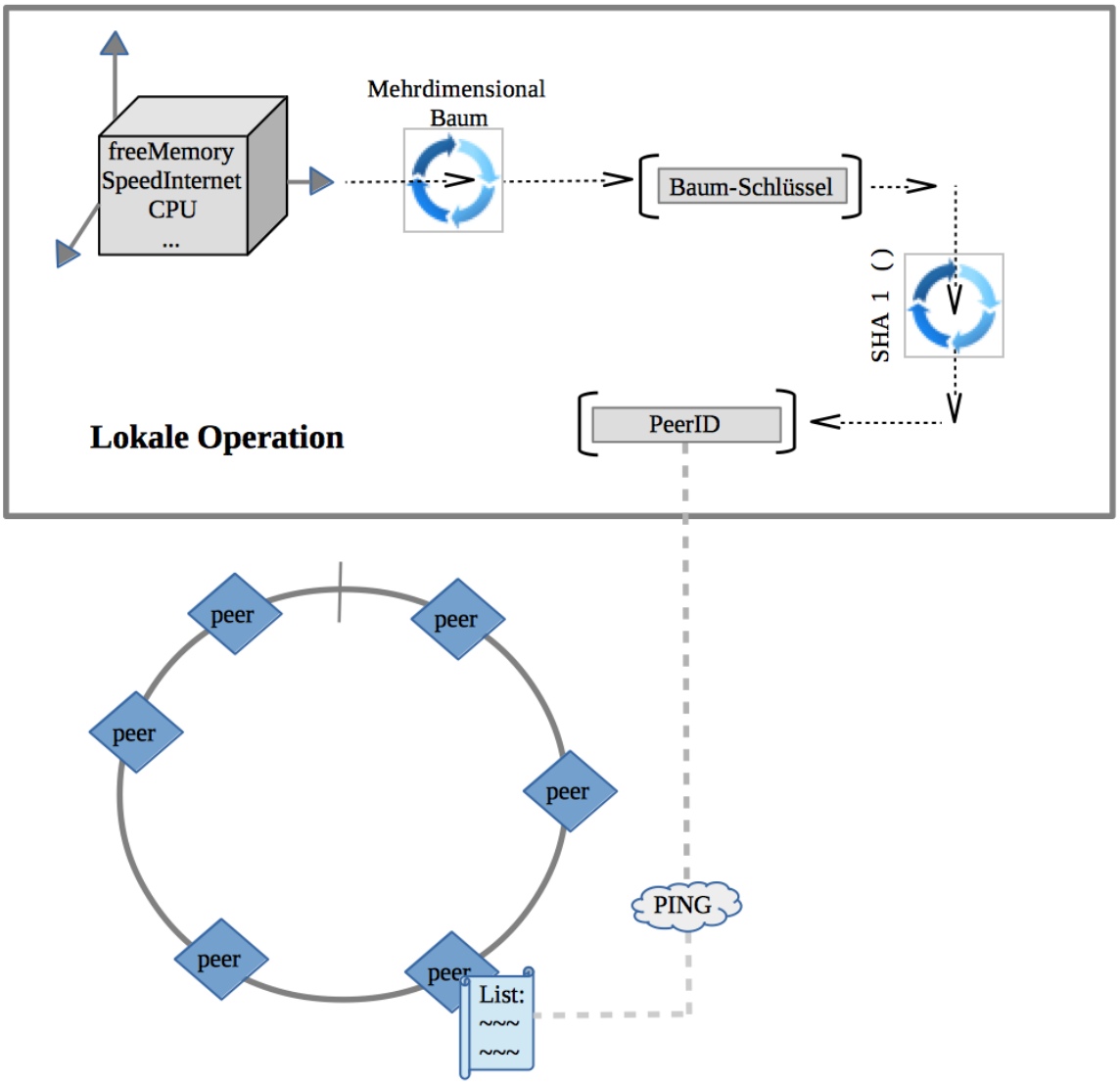


Abbildung 3.9: Ein neuer Rechner online

Der Blattknoten berechnet dann anhand eines mehrdimensionalen Baums einen Schlüssel seines Elternknotens. Der Blattknoten meldet sich bei seinem Elternknoten. Es wird bis einem Kopf vom Baum wiederholt, wie Abbildung 3.10 zu sehen ist.

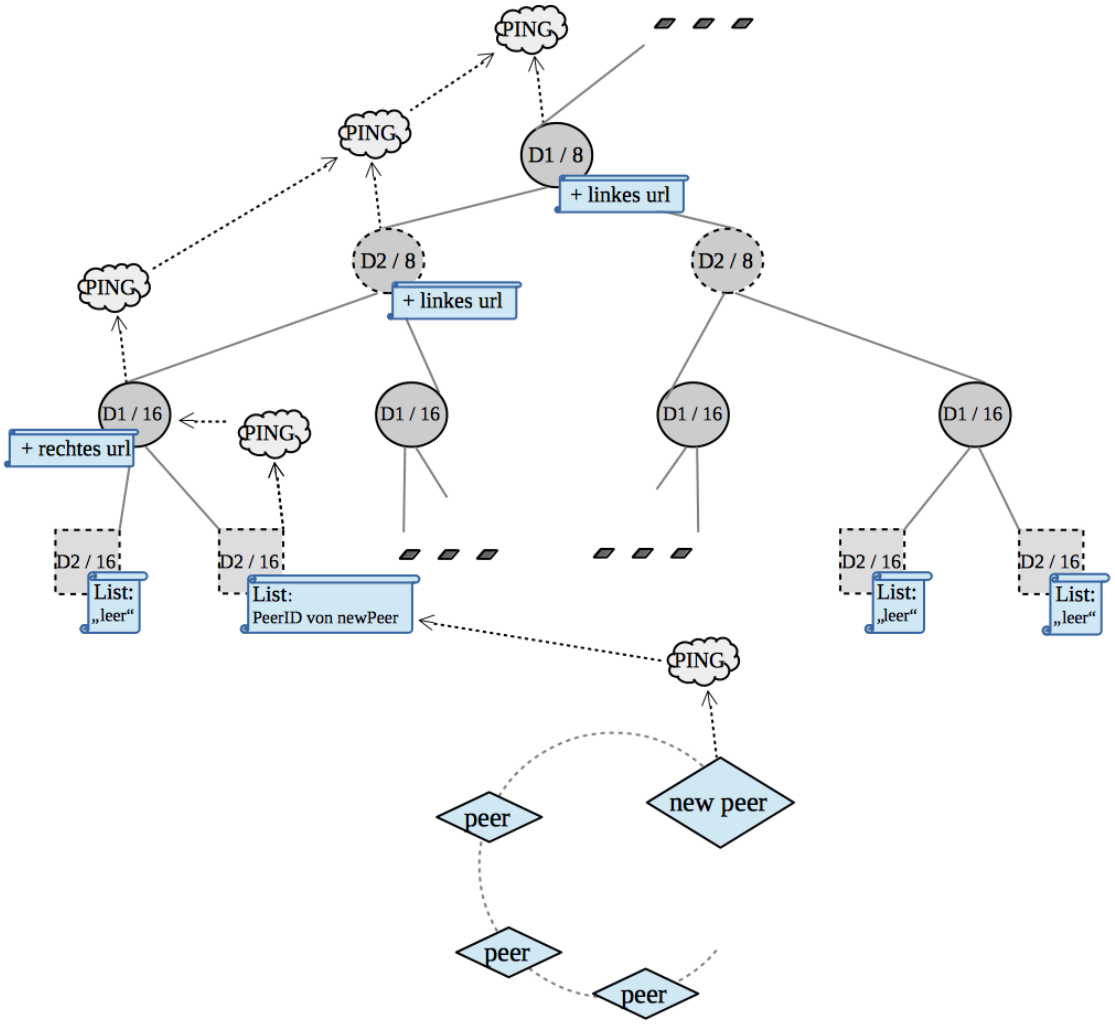


Abbildung 3.10: Ping- Nachrichten

3.4 Suche

Falls erforderlich, die Informationen über die Verfügbarkeit von Knoten im Netz mit den spezifischen Kapazitätsreserven zu Verfügung stellen. Dann kommt der Suchalgorithmus ins Spiel. Ein Knoten, der die Informationen braucht, berechnet mit dem Intervall von Kapazität einen Baum- Schlüssel anhand des mehrdimensionalen Baums, um einen Start- Punkt des Knotens im Baum zu finden. Dies bedeutet, dass die Suche nicht von dem Kopf des Baums beginnt. Falls das Suchintervall klein ist, beginnt die Suche fast von dem Endpunkt zu dem Baum. Wie die Abbildung 3.11 zeigt.

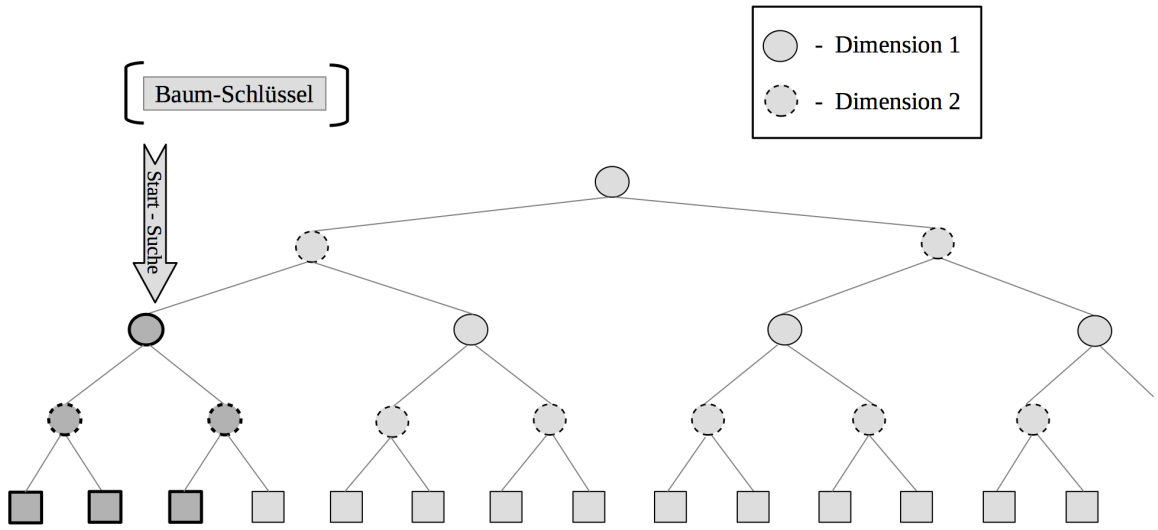


Abbildung 3.11: Ablauf der Suche

Der Schlüssel wird mit Hilfe der SHA1 Funktion im PeerID umgewandelt. Dann wird eine Anforderungsnachricht mit Informationen über einen Suchparameter auf einen Knoten im Overlay mit berechnetem PeerID geschickt. Der Knoten, der die Anforderungsnachricht im Overlaynetz bekommt, prüft, ob ein Kontakt mit dem gewünschten Parameter in Anwesenheit erhaltet, wie ist in der Abbildung 3.12 zu sehen ist.

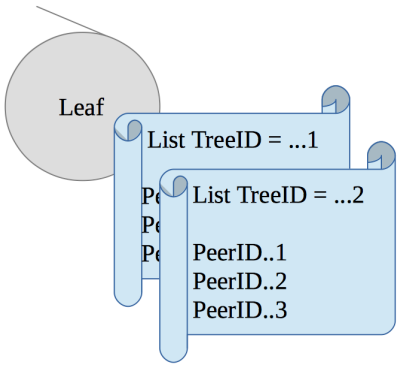


Abbildung 3.12: Anzahl der Lists im Peer

Ein innerer Knoten kann zwei Kontakte, nämlich den rechten und den linken Sohn, haben. Ein Kontakt bedeutet, dass im Netz zumindest ein Knoten mit dem gewünschten Parameter existiert. Ferner wird die Anforderungsnachricht bei der Anwesenheit von den Kontakten zum linken und/oder zum rechten Sohn geschickt. Es ist erwähnenswert, dass diese und weitere Nachrichten mit direkten URL – Adressen geschickt werden, wie die Abbildung 3.13 zeigt.

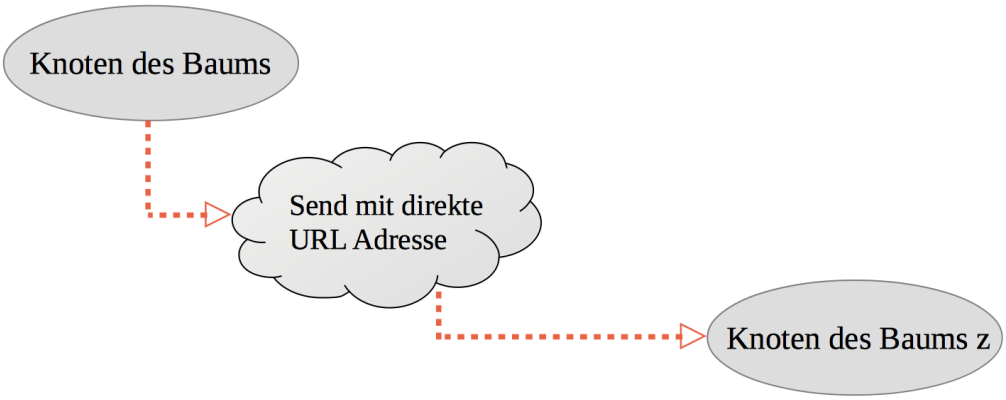


Abbildung 3.13: Nachrichten mit direkte URL – Adressen

Die Anforderungsnachricht wird mehrmals geklont und in verschiedene Richtungen geschickt. Als nächstes wird die Suche von Knoten zu Knoten fortgesetzt, bis die gewünschten Blätter vom Baum erreicht. Wenn eine Nachricht der Suche nach einem Blattknoten kommt, dann wird der Inhalt der Liste überprüft, ob die Liste die gewünschten Kontakte hat. Bei der Verfügung der Kontakte mit dem gewünschten Parameter schickt der Blattknoten die Nachrichten mit der Liste.

Wenn das Netzwerk klein ist, haben viele Blattknoten keine Liste mit PeerID und viele innere Knoten

keine Links. Es bedeutet, dass sich kein Knoten des P2P Nets in dem Raumbereich von der Kapazitäten befindet. Es ist sehr nützlich bei einer Suche, denn wenn ein Innerer Knoten keinen Link hat, dann geht die Suche nicht tiefer in den Baum hinein.

Kapitel 4

Evaluation

4.1 Methodik und Ziele

Im Rahmen dieser Arbeit wird das entwickelte theoretische Verfahren implementiert und bewertet, um zu beweisen, dass das Verfahren arbeitsfähig ist.

Die Ziele sind die Folgenden:

- Beweis durch mehrere Experimente, dass das Verfahren arbeitsfähig ist.
- Die Einschätzung der Effektivität.

Um den ersten Punkt zu beweisen, stellen wir fest:

- Die Zahl der gefundenen Knoten entspricht der Zahl der wirklich existierenden Knoten
- Der Algorithmus arbeitet mit verschiedenem Umfang der Suche
- Die Reservierung der Knotenkapazitäten wurde verwirklicht

Die Realisierung des zweiten Punkts der Ziele wird von den Ergebnissen abhängen.

Zu der Prüfung des implementierten Verfahrens wurde entschieden, den Simulator PeerfactSim zu nehmen, weil der Simulator an der Heinrich-Heine-Universität aktiv weiterentwickelt wird. PeerfactSim.KOM ist ein Simulator für große verteilte P2P-Systeme. Der Simulator wird in Java geschrieben.

Für den Test wurden sieben Szenarien geschrieben. Die Ergebnisse von drei Szenarien werden im

folgenden Paragraph der Auswertung vorgestellt. Jedes Szenario wird 10 mal simuliert. Die Grafiken, die man sieht, sind über alle 10 Simulationen gemittelt. Wir haben hier das arithmetische Mittel verwendet.

Die allgemeinen Parameter der Szenarien:

- 500 Knoten
- Eine Suche mit Wahrscheinlichkeit 5 % wird jeder 300 Sekunden von jeder Knoten initiieren.
- Jede Dimension hat den maximalen Wert 100000.
- Pastry
- kein churn

Die Parameter der Suche werden bei jedem Szenario zufällig generiert und auf drei Typen geteilt:

- A Type: alle Knoten werden gesucht.
- B Type: die Knoten mit einem Abstand zwischen dem maximalen und minimalen Wert der Kapazitäten gleich 1000 werden gesucht.
- C Type: die Knoten mit einem Abstand zwischen dem maximalen und minimalen Wert der Kapazitäten gleich 100 werden gesucht.

Die Szenarien unterscheiden sich auch voneinander durch den Abstand der Zeit zwischen zwei Pings. Die Pings werden für die stabile Arbeit des kd- Baums verwendet.

4.2 Auswertung

In diesem Paragraph sind die Ergebnisse aus der Untersuchung der implementierten Verfahren zu finden.

Die Testergebnisse sind grafisch dargestellt:

- Die erste Grafik zeigt die Ergebnisse von Suchanfragen und zwar die Gesamtzahl aller gefundenen Knoten während der Prüfung. Die Ergebnisse aller Suchanfragen werden zusammengezählt. Die erste Kurve ist die Gesamtzahl aller gefundenen Knoten. Die zweite Kurve ist die Gesamtzahl der Knoten, die gefunden werden sollten.

- Die Ergebnisse der drei Arten von Suchanfragen sind in den folgenden drei Diagrammen dargestellt. Die Diagramme zeigen die durchschnittliche Anzahl von Knoten, die an der Suche beteiligt waren. Die erste Kurve ist die Gesamtzahl der Knoten, die auf der Suche teilgenommen wurden. Aufgrund der parallelen Suche der Knoten zeigt die zweite Kurve die Anzahl der Teilemergruppen, die parallel zueinander sind.

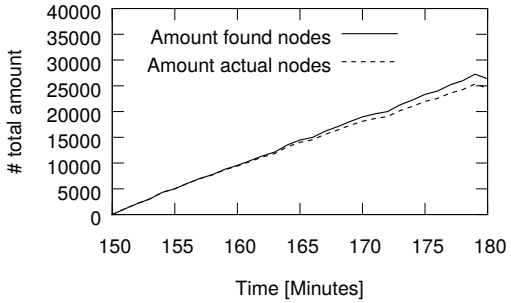
- Das letzte Diagramm zeigt die Anzahl der reservierten Kapazität.

Mit den ersten Ergebnisse ist zu sehen, dass der Algorithmus arbeitsfähig ist:

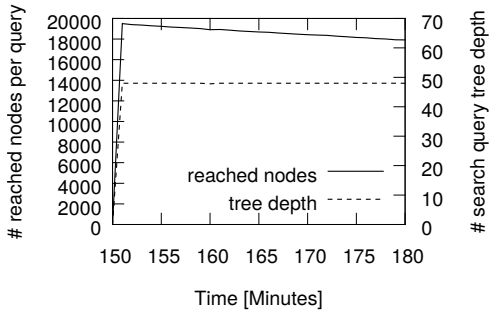
- Abbildung 4.3(a) stellt das Ergebnis für die Unterschiede zwischen den gefundenen Knoten und den wirklich existierenden Knoten dar. Die Unterschiede sind nicht gross und erscheinen wegen des Pings.

- Abbildungen 4.3(b), 4.3(c), 4.3(d) zeigt, dass der Algorithmus bei verschiedenen Parametern der Suche gelaufen wird.

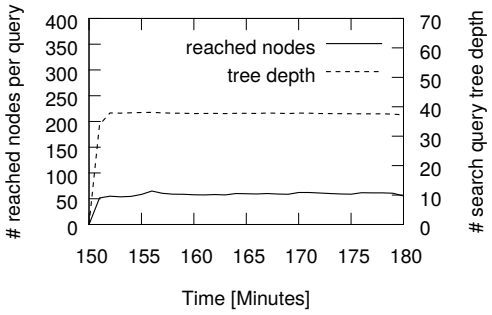
- Das Ergebnis der Reservierung der Knotenkapazitäten wird in der Abbildung 4.3(e) dargestellt. Damit kann man auch kleinere Unterschiede zwischen den reservierten Kapazitäten und den wirklich angeforderten Kapazitäten erkennen..



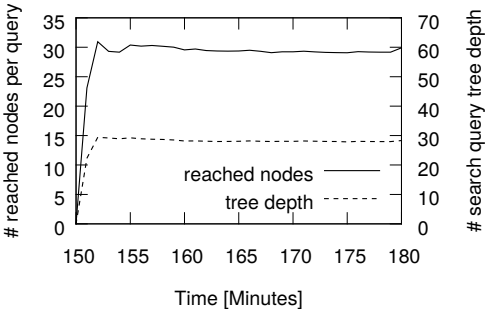
(a) Das Ergebnis der Suche aller Knoten



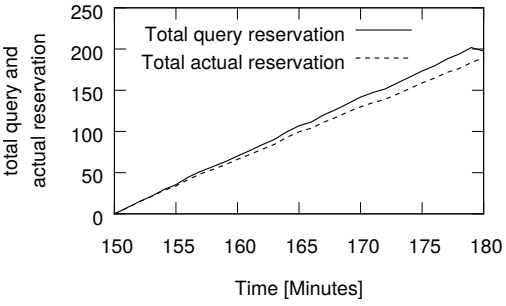
(b) A Type: werden alle Knoten gesucht



(c) B Type der Suche: Abstand zwischen dem max- und min- Wert der Kapazitäten gleich 1000



(d) C Type der Suche: Abstand zwischen dem max- und min- Wert der Kapazitäten gleich 100



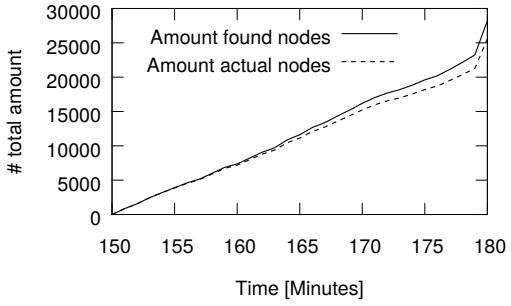
(e) die Reservierung

Abbildung 4.1: Szenario mit 1 min. Ping des Blattknoten und 2 min. Ping des innere Knotens.

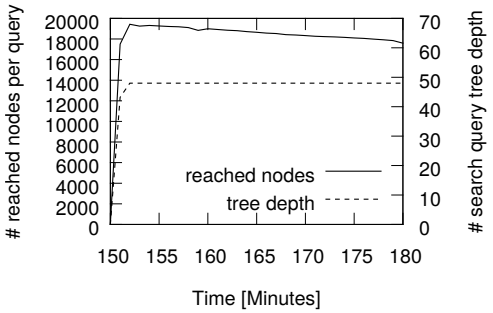
Die Auswertung ergab folgende Ergebnisse:

- Die Abbildungen 4.1, 4.2 und 4.3 zeigt, dass eine Suche mit kleinem Abstand zwischen dem maximalen und minimalen Wert wirksam ist, ganz anders als eine Suche mit grossem Abstand. Das kann man deutlich sehen bei einer Suche mit der A Type, die alle Knoten des P2P Nets sucht.

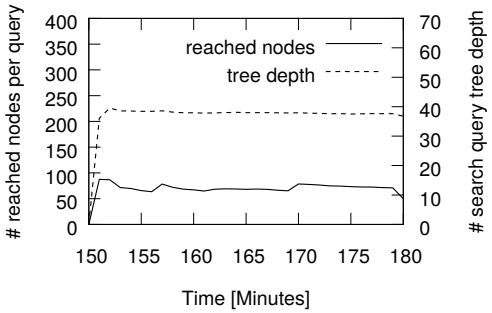
- Wegen der Ping- Nachrichten ist die Suche nicht ganz korrekt, da das Verfahren sich nach der Reservierung der Kapazitäten anhand des Pings aktualisieren soll.



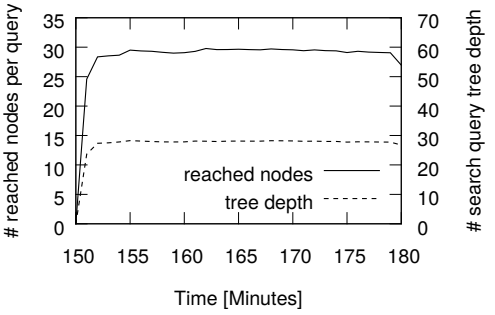
(a) Das Ergebnis der Suche aller Knoten



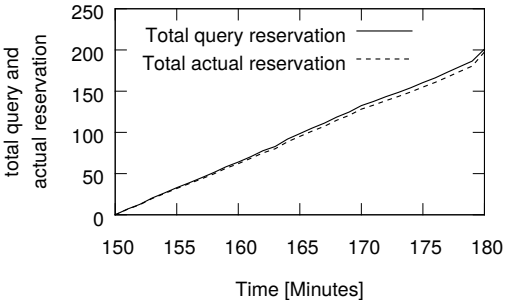
(b) A Type: werden alle Knoten gesucht



(c) B Type der Suche: Abstand zwischen dem max- und min- Wert der Kapazitäten gleich 1000

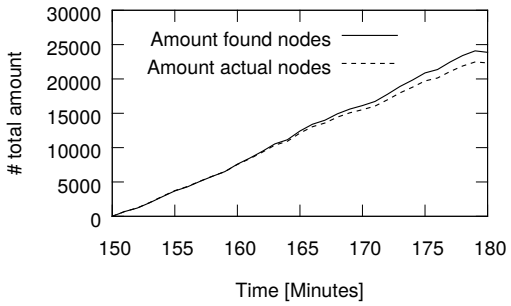


(d) C Type der Suche: Abstand zwischen dem max- und min- Wert der Kapazitäten gleich 100

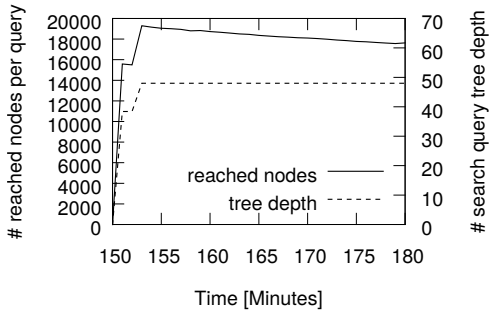


(e) die Reservierung

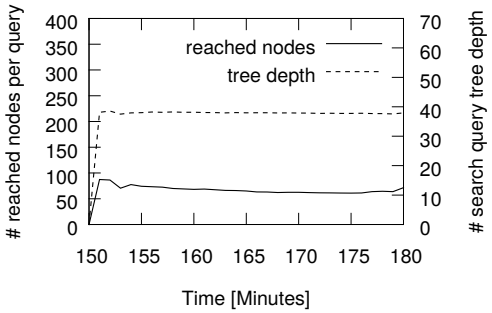
Abbildung 4.2: Szenario mit 30 s. Ping des Blattknoten und 1 min. Ping des innere Knotens.



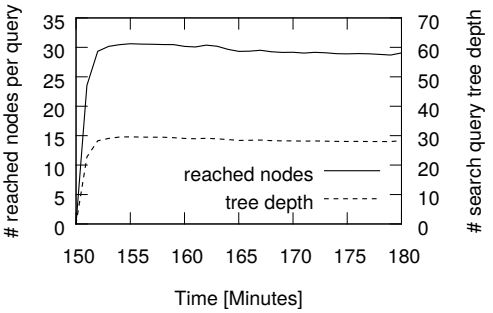
(a) Das Ergebnis der Suche aller Knoten



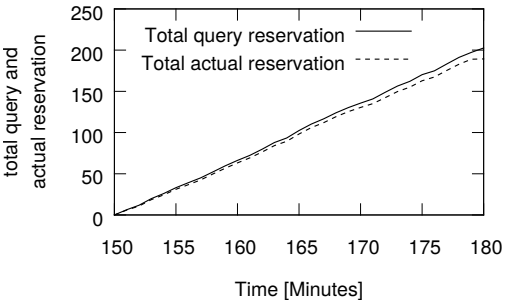
(b) A Type: werden alle Knoten gesucht



(c) B Type der Suche: Abstand zwischen dem max- und min- Wert der Kapazitäten gleich 1000



(d) C Type der Suche: Abstand zwischen dem max- und min- Wert der Kapazitäten gleich 100



(e) die Reservierung

Abbildung 4.3: Szenario mit 2 min. Ping des Blattknoten und 5 min. Ping des innere Knotens.

Kapitel 5

Fazit

5.1 Ergebnis

Im Rahmen dieser Bachelorarbeit wurde ein Verfahren entwickelt, das alle Kapazitäten (z.B. verfügbare Bandbreite, verfügbarer Speicher, Rechenkapazität etc.) von Knoten innerhalb eines einzigen Overlays durchsuchbar macht. Mit Hilfe des Verfahrens ist die Indexierung und Suche von Knotenkapazitäten mit mehreren Attributen möglich.

Aufgrund der gemachten Tests lässt sich schlussfolgern:

Der Algorithmus ist arbeitsfähig, weil:

- die Unterschiede zwischen den gefundenen Knoten und wirklich existierenden Knoten nicht gross sind und wegen des Pings der Blattknoten und der innere Knoten erscheinen.
- der Algorithmus bei verschiedenen Parametern der Suche ausgeführt wird.
- es nur kleinere Unterschiede zwischen den reservierten Kapazitäten und den wirklich angeforderten Kapazitäten gibt.

Die Auswertung ergab folgende Ergebnisse:

- Die Suche mit kleinem Abstand zwischen dem maximalen und minimalen Wert ist wirksam.
- Aufgrund der Ping- Nachrichten ist die Suche nicht ganz korrekt, da das Verfahren sich nach der Reservierung der Kapazitäten anhand des Pings aktualisieren soll.

Das vorliegende Verfahren organisiert die Indexierung und Suche von Knotenkapazitäten in Struktu-

rierten P2P Overlay-Netzwerken mit den zulässigen Abweichungen.

In dieser Arbeit wurde eine wirksame Methode, die jedoch noch einer weiteren Optimierung bzw, Verbesserung bedarf, zur Indexierung und Suche von Knotenkapazitäten in P2P-Netzen entwickelt.

Literaturverzeichnis

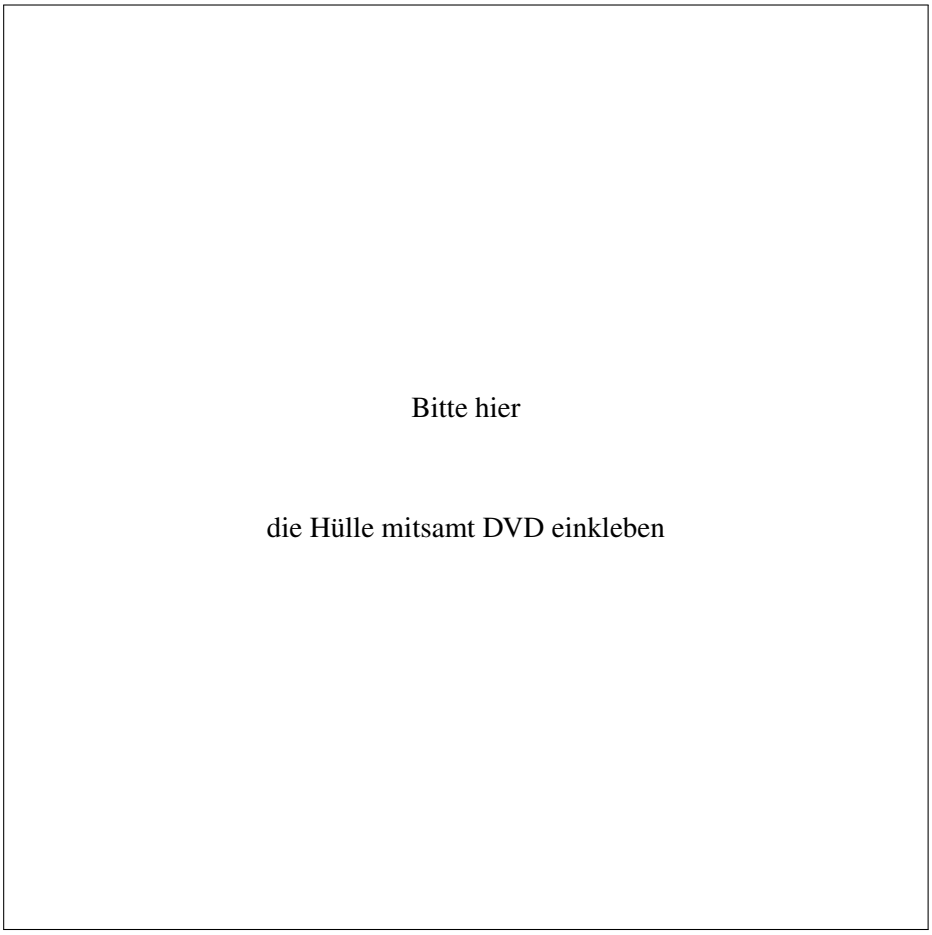
[GMHS07] GRAFFI, Kalman; MOGRE, Parag S.; HOLLICK, Matthias; SCHWINGENSCHLÖGL, Christian: *Detection of Colluding Misbehaving Nodes in Mobile Ad-Hoc and Wireless Mesh Networks, European Patent Office, Application No. /Patent No. 07022624.6 – 1249*. 2007.

[Gra11] GRAFFI, Kalman: PeerfactSim.KOM: A P2P System Simulator – Experiences and Lessons Learned. In: *IEEE P2P '11: Proc. of the Int. Conf. on Peer-to-Peer Computing*, 2011.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 12.Oktober 2015 Ivan Tolstun



Diese DVD enthält:

- Eine *pdf* Version der Bachelorarbeit
- Alle LaTeXund Grafik Dateien mitsamt dazugehörigen Skripten, die verwendet wurden
- Der Quellcode, der während der Bachelorarbeit erarbeitet wurde
- Alle während der Evaluation angefallenen Messdaten
- Alle referenzierten Webseiten und wissenschaftlichen Arbeiten