P2P - Theorie Aufgabe 3

Arne Link (1582381), Lars Fritche (1691285)
Gruppe 14 (S217/103)

December 16, 2013

1. Aufgabe 4.1 Distributed Hash Tables

1.1. a) ID space partitioning

Es kann deshalb vorteilhaft sein, da man so verhindern kann, dass die Zuteilung von Nodes auf Objectidentifier clustern kann und manche Nodes sehr viele Identifier verwalten müssen und andere sehr wenige. Man versucht die Verteilung also möglichst gleichmäßig zu gestalten, jedoch gilt das natürlich nur für die Annahme, dass die benutze Hashfunktion zur Erzeugung der Identifier gleichmäßig verteilte Werte erzeugt.

Es macht also keinen Sinn, den Wertebereich der Identifier aufzuteilen, wenn sich die Werte der einzelnen Einträge in Untermengen des Wertebereichs clustern.

1.2. b) Content distribution over DHT's

a). Chord

Vorteil: Einfacher Aufbau, Klare verteilung des Contents.

Nachteil: Ohne finger Trees recht anfällig gegen Ausfälle, auch mit noch anfällig gegen (gezielte) Angriffe.

Use case: Filesharing, load balancing

b). Tapestry

Vorteil: Sehr effizientes Routing, besonders von häufig abgefragten Content / zu häufig angesprochenen Nodes.

Nachteil: Fehleranfälliger als beispielsweise KAD oder CAN (mit muliple realities), relativ komplizierte verwaltung des States pro Node.

Use case: Filesharing, Effizienter unicast (P2P Messaging)

c). KAD

Vorteil: Sehr unanfällig gegen zufällige Ausfälle, schnelles Routing.

Nachteil: Potentiell leicht ungleiche Verteilung des Netzwerkes, demzufolge längere Routing wege und ungleiche Verteilung des Contents. Macht es auch Anfällig gegen gezielte Angfriffe.

Use case: Filesharing Netzwerke.

1.3. c) Churn in DHT's

Durch das schnelle Hinzukommen und Verlassen von Knoten wird die konsistente Adressierung erschwert, da alte Adressbereiche in die Obhut neuer Knoten übergehen, während parallel dazu Anfragen zu diesen Adressen gesendet werden. Darüber hinaus erschwert es Features, wie die "finger tables" von Chord, da sich die "successor" zu verschiedenen "fingers" schnell ändern können, sich

diese Änderung jedoch schnell propagieren muss. Weiterhin können Hash buckets verloren gehen, wenn sich ein Knoten abmeldet, ohne seine Buckets in die Verantwortung eines anderen Knotens zu übertragen.

Um das Problem der verlorenen Hash Buckets zu umgehen, könnten Adressbereiche doppelt vergeben werden, um sicherzustellen, dass das plötzliche Verlassen eines Knotens, ohne DHT-konformes Abmelden, zu einem Informationsverlust führt.

Um das Problem mit den "finger tables" zu umgehen, könnte man Quality-of-Service-Knoten einführen, die ein fester Bestandteil des Netzes sind und mit einer extrem geringen Warscheinlichkeit ausfallen. Diese Knoten werden so verteilt, dass sie mit hoher Warscheinlichkeit als Eintrag der "finger tables" gewählt werden, womit sichergestellt ist, dass sich die Einträge auch bei einer hohen Join-Leave-Rate nicht oft ändern.

2. Aufgabe 4.2 Chord

2.1. a) Routing path length

Wenn man ohne "finger tables" rechnet, so muss im Durchschnitt der halbe Kreis abgewandert werden, was in dem Fall 2^n / 2 Hops wären. Im worst-case benötigt man einen kompletten Kreisumlauf mit 2^n Hops. Verwendet man "finger tables" so verändert sich die Komplexitätsklasse von O(n) nach O(log n). Dies bedeutet dass man maximal n * log(2) Hops benötigt und im Durchschnitt n * log (2) /2.

Wenn man nun nicht mehr von einer vollständigen Besetzung des Rings ausgeht, so ändert sich für die "finger tables" nichts, da die Intervalle fest gewählt werden. Für den normalen Weg über die einzelnen Successverbindungen ändert sich der maximale Weg zu N und der durchschnittliche Weg zu N/2.

2.2. b) Finger table size

Die maximale Anzahl der "fingers" kann direkt aus der Bitzahl des Wertebereichs der Identifier abgelesen werden, deshalb kann es bis zu n "fingers" geben. In der Realität wird über die IP via SHA-1 gehasht, wobei der Wertebereich der Hashfunktion = [0, 2^{160} - 1]. Das bedeutet, dass es bis zu 160 "fingers" geben kann, während wir für die maximale Anzahl an Nodeidentifier 2^{32} Hashwerte erhalten (zusammen mit dem Port wären es sogar 2^{48}). Da diese Hashwerte gleichmäßig verteilt sind, ist es auch in der Realität wohl nicht unüblich 160 Fingers zu haben, sofern viele Knoten an der DHT mitwirken.

2.3. c) Using finger tables

Node 0:

i	target ID	successor
1	[1,2)	3
2	[2,4)	3
3	[4,8)	3
4	[8,16)	10
5	[16,32)	28
6	[32,0)	33

Node 33:

i	target ID	successor
1	[34,35)	34
2	[35,37)	34
3	[37,41)	34
4	[41,49)	34
5	[49, 1)	50
6	[1,33)	3

Nodes 50:

i	target ID	successor
1	[51,52)	58
2	[52,54)	58
3	[54,58)	58
4	[58,2)	58
5	[2,18)	3
6	[18,50)	28