Theorie 3

Fachgruppe Telekommunikationsnetze (TKN)

24. November 2023

Einleitung

Die folgenden Aufgaben werden gemeinsam im Tutorium bearbeitet. In der Veranstaltung Rechnernetze wird die SI-Notation verwendet. Beispiele für Präfixe: m = 10^{-3}, k = 10^{3}, M = 10^{6}, ki = 2^{10}, Mi = 2^{20}. „B“ bezeichnet Bytes, „bit“ Bits.

Übung 1 *DHT Basics*

In der Vorlesung haben Sie bereits verteilte Hashtabellen (*distributed hash tables*, DHTs) kennen gelernt. Beantworten Sie bitte die folgenden Fragen:

1. Welches minimale Interface bieten DHTs mindestens an (3 Funktionen) und was tun diese Funktionen?
2. Wieso ist es einfach, verschiedene Anwendungen mit der selben DHT Software zu betreiben? Funktioniert das auch gleichzeitig?
3. Welche Probleme gibt es mit der Dynamizität und der Größe solcher DHTs? Wie sind jeweils die Lösungen, die in der Vorlesung vorgestellt werden?
4. Erinnern Sie sich an die Struktur von DNS. Welche Strukturen haben DHTs im Vergleich zu DNS?
5. Wie funktioniert der *Chord Lookup*?
6. Wie funktioniert die *Chord Joining Operation*?
7. Was versteht man unter *latency stretch* (Formel und Erklärung)?

Lösung

1.
   - `lookup(key)` → `value` bzw. `get(key)` → `value`
   - `insert(key)` bzw. `set(key)`
   - `delete(key)`

   Um Verwirrung zu vermeiden, verwenden wir den Begriff *lookup* ausschließ-

lich für die Zuständigkeitsabfrage für einen gewissen Schlüssel und *nicht* für die Abfrage des Wertes. Aus historischen Gründen ist dieser Begriff leider mehrfach belegt.

2. Das Interface einer DHT ist sehr generisch und kann im Prinzip überall dort verwendet werden, wo eine global verfügbare Schlüssel-Wert-Zuordnung sinnvoll ist. Mehrere Anwendungen können auch gleichzeitig die selbe DHT benutzen, wenn ihre Schlüsselmengen disjunkt sind (z.B. durch ein anwendungsspezifisches Präfix).

3. *Dynamizität:* In einer verteilten Hashtabelle soll die Last möglichst gleichmäßig auf den Knoten verteilt werden. Gleichzeitig ändert sich die Anzahl der Knoten aber ständig. Der Schlüsselraum (hash space) darf also nicht von der Anzahl der Knoten abhängig sein, da sonst ständig alle Werte den Ort wechseln müssten.

   Daher benutzt man einen einen fixen Schlüsselraum sog. *konsistentes Hashing* und ordnet die Zuständigkeit anhand einer Metrik dem "nähesten"Knoten zu. Wenn die IDs der Knoten dann zufällig gewählt werden, sollte die Zuständigkeit annähernd gleich verteilt sein.

   Größe/Skalierbarkeit: In einer DHT müssen alle Anfragen zu dem jeweils zuständigen Knoten gelangen. In einem naiven Ansatz müsste dazu jeder Knoten jeden anderen Knoten im Netzwerk kennen. Aber so ein vollvermaschtes Netzwerk ist sehr schwierig aufrecht zu erhalten, ab einer gewissen Größe.

   Stattdessen muss eine DHT eine Art Routing-Mechanismus anwenden um die Nachrichten über mehrere Zwischenstationen zuverlässig ans Ziel zu bringen. In Chord dient dazu die Ring-Struktur. Dieses Routing zu optimieren ist Gegenstand der aktuellen Forschung.

4. DNS und DHT erfüllen prinzipiell ähnliche Aufgaben: Zuordnung von Schlüssel (bzw. Name) zu einem Wert. DNS ist dabei in einem strikt hierarchischen Baum organisiert mit den Root-DNS-Servern an der Wurzel, wohingegen DHTs typischerweise nicht hierarchisch aufgebaut sind und Knoten ein *peer* des anderen ist.

5. siehe Unit 6 (DHT), Folie 11

6. siehe Unit 6 (DHT), Folie 12

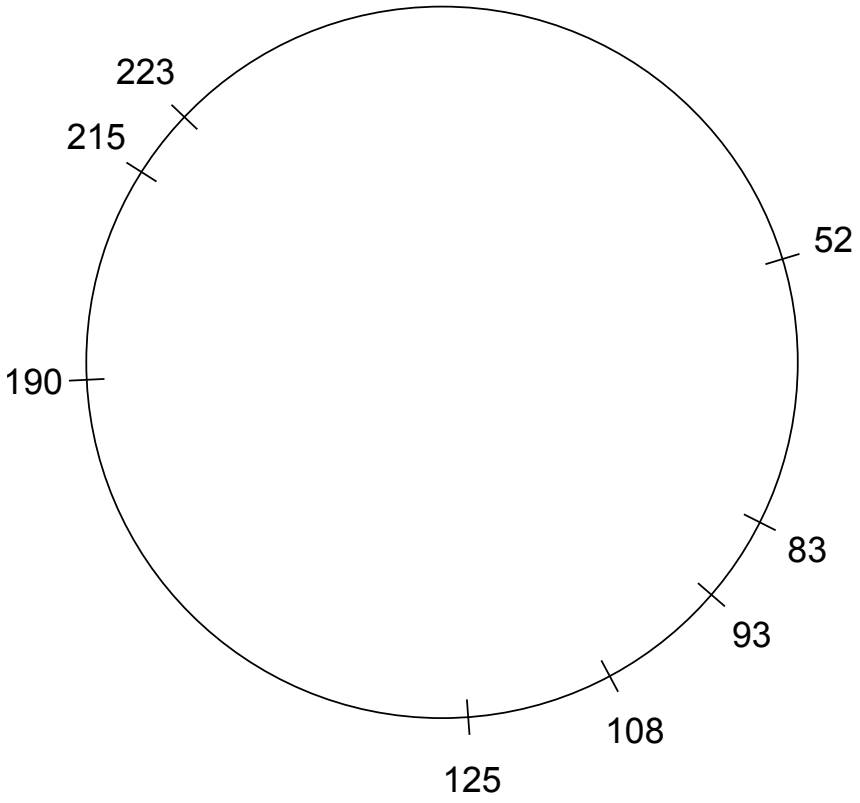
7. Latency Stretch ist ein Faktor welcher beschreibt wie die Overlay-Topologie (also bei Chord die Ring- Struktur) durchschnittlich die Latenzzeiten vergrößert im Vergleich zu einem vollvermaschten Netwerk, bei dem jede Anfrage den direkten Weg nehmen würde.

$$\text{stretch} = \frac{\overline{\delta_{overlay}}}{\overline{\delta_{underlying}}}$$

Übung 2

[subtitle=**Finger-Tables**]

Eine Distributed Hash Table (DHT) benutzt Chord als Implementierung. Die Keys haben eine Länge von 8 bit. Es sind acht Knoten vorhanden. Die IDs der Knoten sind in der Grafik verzeichnet:



1. Was ist die allgemeine Formel zum Ausrechnen des i-ten Wertes in der Finger Table des Knotens mit der ID n bei Chord?
2. Stellen Sie die Finger Table des Knoten mit der ID $n = 52$ auf.
3. Wie vereinfacht eine Finger Table den Chord Lookup?

Lösung

1. $ft(i) = \text{successor}(n + 2^i \bmod 2^m)$
2. $ft(i) = \text{successor}(52 + 2^i \bmod 2^8)$

i	start	ft(i)
0	53	83
1	54	83
2	56	83
3	60	83
4	68	83
5	84	93
6	116	125
7	180	190

3. Der durchschnittliche Aufwand eine beliebige ID im Ring zu erreichen verringert sich von $\mathcal{O}(n)$ Schritten zu $\mathcal{O}(\log n)$ Schritten, wobei jeder Knoten nur m Tabelleneinträge aktuell halten muss.

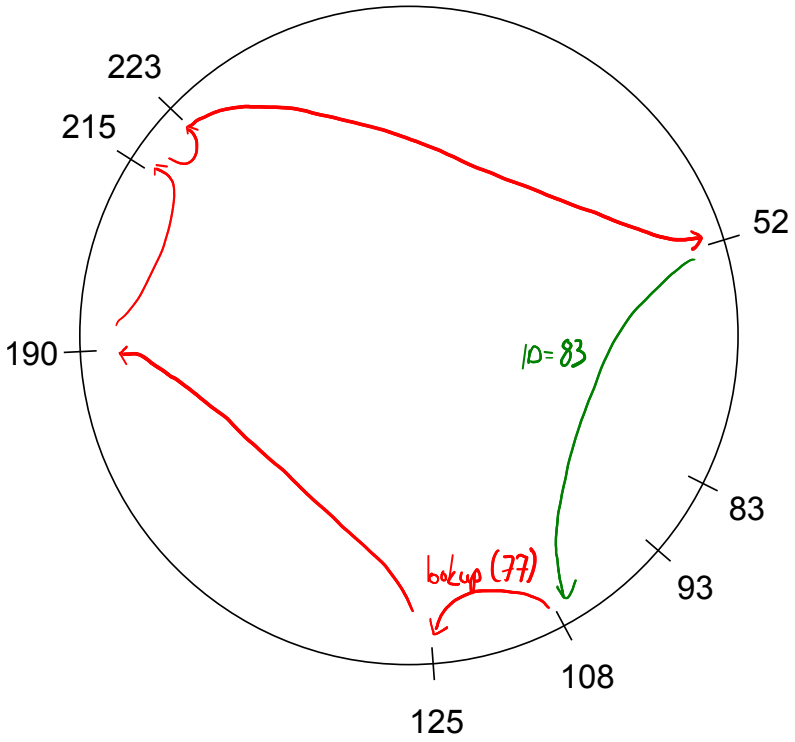
Übung 3 *Finger-Tables*

Eine Distributed Hash Table benutzt Chord als Implementierung identisch mit Aufgabe 2. Die Keys haben eine Länge von 8 bit. Es sind acht Knoten vorhanden.

1. In welchen Knoten sind die Werte mit den Keys 99 bzw. 240 jeweils gespeichert?
2. Knoten 108 möchte erfahren, welcher Knoten für den Key 77 zuständig ist. Zeichnen sie die notwendigen Nachrichten für die Abfrage in die Grafik von Aufgabe 3 ein, wenn *keine* Finger Tables benutzt werden. Welche Knoten-ID wird dem anfragenden Knoten gemeldet?
3. Wie viele Nachrichten werden **ohne** Benutzung von Finger Tables maximal benötigt, um im dargestellten System von einem beliebigen Knoten einen beliebigen Key abzufragen?

Lösung

1.
   - `lookup(99)` → `108`
   - `lookup(240)` → `52`



2.

3. Maximal 7 Nachrichten → 6 für den Lookup (Vorgänger kann antworten), 1 für die Antwort

Übung 4 *Mesh-Overlay und DHT*

Betrachten Sie eine verteilte Hashtabelle (DHT) mit einem Mesh-Overlay-Netzwerk (das heißt, alle Peers kennen alle anderen Peers im System). Was sind die Vor- und Nachteile eines solchen Systems? Was sind die Vor- und Nachteile einer DHT mit Ringstruktur (ohne Finger Table)?

Lösung

Mesh-Overlay-Netzwerk: Ein vollvermaschtes Netzwerk könnte Anfragen jeweils direkt an den zuständigen Knoten weiterleiten und hätte daher geringe Latenzzeiten. Der Anzahl der verschickten Nachrichten pro Anfrage wäre konstant, also $\mathcal{O}(1)$. Jedoch müsste jeder Knoten eine Liste mit potentiell tausenden Knoten speichern und vor allem aktuell halten. Das Beitreten und Verlassen des Netzwerks von Knoten erzeugt somit großen Aufwand und skaliert schlecht mit der Größe des Netzwerks.

DHT mit Ringstruktur: In einer DHT mit Ringstruktur wie z.B. Chord, dauert es ggf. länger eine Anfrage an den richtigen Knoten zuzustellen, als in einem vollvermaschten Netzwerk. Der Anzahl der verschickten Nachrichten pro Anfrage stiege linear mit der Anzahl der Knoten, also $\mathcal{O}(n)$. Jedoch muss jeder Knoten jeweils nur seinen Vorgänger bzw. Nachfolger kennen, was den Prozess des Beitretens und Verlassens erheblich vereinfacht, da der Zustand von nur zwei Knoten aktualisiert werden muss.