Praktikum

# Rechnernetze und Verteilte Systeme

Block 4

# DHT & P2P

25.11. — 8.12.2019

## 1 Präsenzaufgaben

Die folgenden Aufgaben werden im Termin besprochen. Sie dienen auch der Vorbereitung der ISIS-Tests.

**Aufgabe 1:**

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

(a) Welches minimale Interface bieten DHTs mindestens an (3 Funktionen) und was tun diese Funktionen?

(b) Wieso ist es einfach, verschiedene Anwendungen mit der selben DHT Software zu betreiben? Funktioniert das auch gleichzeitig?

(c) Welche Probleme gibt es mit der Dynamizität und der Größe solcher DHTs? Wie sind jeweils die Lösungen, die in der Vorlesung vorgestellt werden?

(d) Erinnern Sie sich an die Struktur von DNS. Welche Strukturen haben DHTs im Vergleich zu DNS?

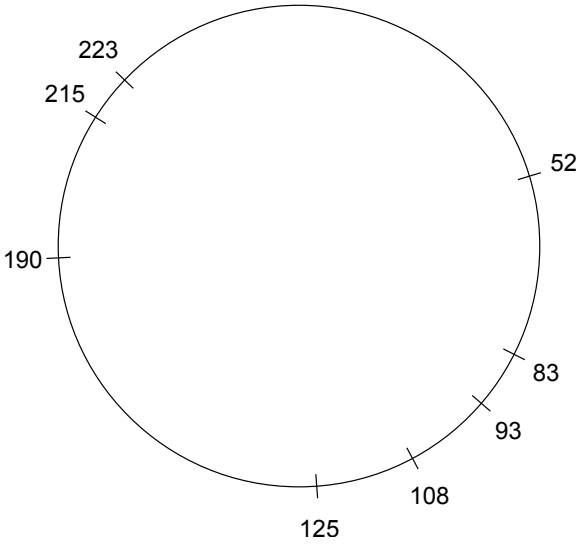
(e) Wie funktioniert der “Chord Lookup”?

(f) Wie funktioniert die “Chord Joining Operation”?

(g) Was versteht man unter “latency stretch” (Formel und Erklärung)?

**Aufgabe 2:**

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




(a) Was ist die allgemeine Formel zum Ausrechnen des $i$-ten Wertes in der Finger Table des Knotens mit der ID $n$ bei Chord?

(b) Stellen Sie die Finger Table des Knoten mit der ID $n = 52$ auf.

(c) Wie vereinfacht eine Finger Table den Chord Lookup?

**Aufgabe 3:**

Beantworten Sie die folgenden Fragen rund um P2P:

(a) Nennen sie zwei wesentliche Herausforderungen, die ein Peer-to-Peer-System zu lösen hat.

(b) Vergleichen Sie die beiden Technologien Client/Server und P2P und stellen Sie gegenüber, welche Vor- bzw. Nachteil diese jeweils haben.

(c) Wie findet ein Knoten im Gnutella-Netzwerk eine Datei? Was ist dabei das Problem?

(d) Was ist im Kontext von BitTorrent ein Seeder, Leecher und ein Tracker?

**Aufgabe 4:**

Betrachten Sie eine verteilte Hashtabelle (engl. distributed hash table, 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)?

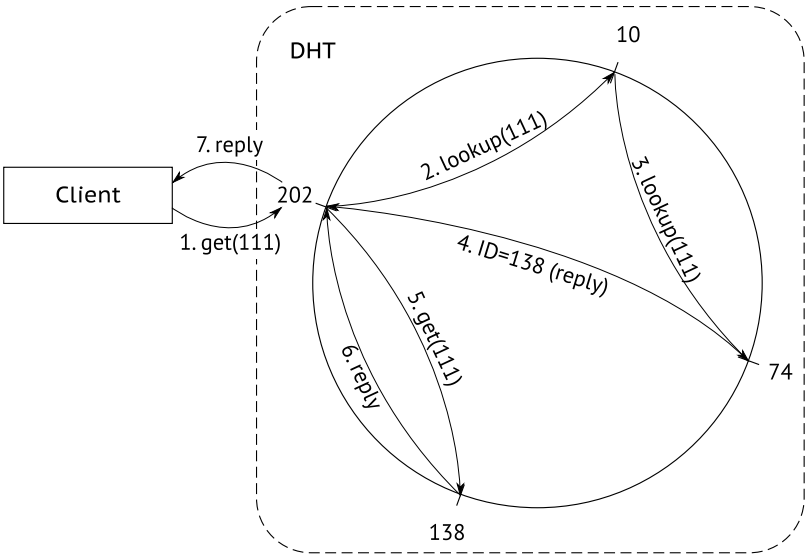
# 2 Praktische Aufgaben

Die praktischen Aufgaben sind in Kleingruppen von i. d. R. 3 Personen zu lösen. Die Ergebnisse besprechen Sie im zweiten Termin mit dem Tutor. Reichen Sie deshalb bitte den Quelltext bzw. Lösungen dieser Aufgaben bis Sonntag vor dem zweiten Termin (30..11.) 18:00 Uhr per ISIS ein. Aufgaben werden sowohl auf Plagiate als auch auf Richtigkeit hin automatisch getestet, halten Sie sich deshalb genau an das vorgegebene Abgabeformat.

**Aufgabe 5:**

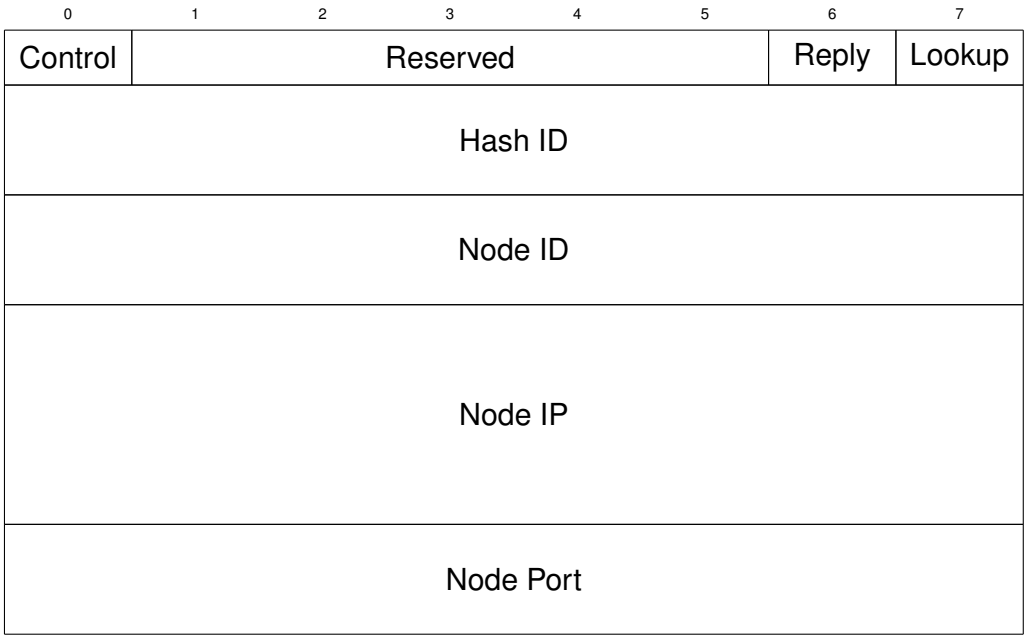
In der Vorlesung wurde Ihnen das Chord-Protokoll vorgestellt. Damit lässt sich eine Hashtabelle - bspw. die, die sie auf dem letzten Aufgabenblatt implementiert haben - dezentralisiert, d.h. über mehrere Server (sog. Knoten oder Peers) verteilt, speichern und als Distributed Hash Table (kurz: DHT) betreiben. Ziel dieser Aufgabe ist es, einen Server zu entwickeln, der als solch ein Knoten in einem Chord-Ring fungieren kann.

Nehmen Sie als Ausgangspunkt Ihre Implementierung aus der letzten Aufgabe. Die Funktionalität soll nun im wesentlichen gleich bleiben, jedoch die Speicherung der Werte über mehrere Prozesse bzw. Rechner verteilt werden. Ihre (jetzt verteilte) Hashtabelle sollte nach wie vor das gleiche Interface nach außen besitzen, also immer noch die Befehle `set()` um einen Wert zu speichern oder zu aktualisieren, `get()` um einen Wert auszulesen und `delete()` um einen Wert zu löschen anbieten. Die Speicherung der Werte soll allerdings eine DHT auf Basis von einem vereinfachten Chord-Ring übernehmen. Eine DHT ist dabei ein Peer-to-Peer (P2P) Netz, also alle Knoten der DHT fungieren als gleichwertige Peers, die sowohl Anfragen entgegennehmen als auch weiterleiten können. Da sich das Interface nicht geändert hat, sollen Sie ihren Client aus der letzten Aufgabe unverändert verwenden. Die Interaktion zwischen Client und einem der Peers der DHT soll beispielhaft für eine gleichmäßige Topologie aus 4 Peers wie folgt aussehen:




Das Paketformat zwischen Client und Peers bleibt dabei erhalten. Der Befehl wird von einem beliebigen Peer entgegengenommen und von diesem Knoten stellvertretend bearbeitet. Auch innerhalb des Rings soll das Nachrichtenformat für die bekannten `get()`, `set()` und `delete()` Befehle erhalten bleiben.

Hinzu kommen noch die Kontrollnachrichten für den Lookup bzw. die entsprechende Antwort. Im Vergleich zur letzten Aufgabe enthält das Nachrichtenformat für diese beiden Nachrichten zusätzlich Felder für ID, IP und Port, aber enthalten keinen Value mehr. Der Key wurde durch den jeweiligen Hash-Wert ersetzt:




Dabei ist “Control” der Indikator für eine Kontrollachricht innerhalb des Rings, also

einen Lookup oder dessen Antwort. Die IP-, Port-, und ID-Felder sind beim Lookup mit den Daten des ersten Knotens zu füllen, der den Befehl von außen erhalten hat und jetzt stellvertretend nach dem zuständigen Knoten sucht. Wir beschränken uns dabei auf IPv4. Bei dieser Version sind die Adressen 32bit lang. Der dazugehörige Port und die ID sind jeweils 16bit lang. Achten sie wie immer darauf, dass Zahlen in Network-Byte-Order versendet werden. Wenn ein Lookup beantwortet wird, schreibt der antwortende Peer jeweils die ID, die Adresse und den Port des zuständigen Knotens in die Antwort.

Schreiben Sie Ihren Hash Table Server so um, dass er als Peer in einem DHT P2P Netz teilnimmt. Dabei liegt der Fokus auf der Lookup-Operation. Chord Joining muss zunächst nicht implementiert werden. Sie sollten sich im Vorfeld eine Ringtopologie aus 4 Prozessen/Knoten mit jeweiligen IDs ausdenken, dabei darf allerdings keine der IDs 0 sein! Der Knoten sollte beim Aufruf seine eigene ID, IP und Port, die ID, IP und Port des Vorgängers, sowie die ID, IP und Port des Nachfolgers übergeben bekommen. Ein Beispielaufruf für einen der Peers wäre dabei:

```
./peer 15 127.0.0.1 4711 245 127.0.0.1 4710 112 127.0.0.1 4712
```

So weiß jeder Peer, für welchen Bereich er zuständig ist und wer seine Vor- und Nachfolger sind.

Achten sie bei der Implementierung besonders auf die folgenden Punkte:

(a) Als Hashing-Funktion im Ringe sollen der Einfachheit halber die zwei ersten Bytes des Keys genutzt werden. Sollte der Key weniger als 2 Bytes lang sein, werden die fehlenden Bytes mit Nullbytes aufgefüllt. Überlegen Sie sich, wie groß dabei die Variable $m$ aus der Vorlesung ist, was sie bedeutet und wie viele IDs es auf dem Ring höchstens gibt.

(b) Der Peer muss nun mehrere Verbindungen gleichzeitig halten können. Bei der Anfrage eines Clients bleibt dessen Verbindung zum DHT-Peer offen, während dieser ggf. die Anfrage zu anderen Peers weiterleitet und auf eine Verbindungsanfrage für die Antwort wartet. Sie sollten sich deshalb mit der Funktion `select` vertraut machen, um mit mehreren gleichzeitigen Verbindungen umzugehen[1].

[1] `https://beej.us/guide/bgnet/examples/selectserver.c`

(c) Um bei Antworten den richtigen Client zuzuordnen, muss ihr Programm sich mindestens die Zuordnung des Keys zum Socket-Deskriptor speichern. Für eine Zuordnung von solchen `<key, value>` Paaren bietet sich natürlich eine interne Hashtable an.

(d) Sie sollen in Ihrem Projekt CMake benutzen. Der folgende Aufruf soll im Unterverzeichnis `build` eine ausführbare Datei mit dem Namen `peer` erstellen:

```
$ mkdir build; cd build
$ cmake .. && make
```

Erstellen Sie aus Ihrer Lösung ein tar.gz Archiv, z.B. so:

```
tar -czvf Block4.TXXGYY.tar.gz Block4
```

Laden Sie dieses Archiv bei ISIS bei der entsprechenden Abgabe hoch.

# 3 Vertiefungsaufgaben

Diese Aufgaben sind zu Ihrer eigenen Vertiefung in Hinblick auf die Klausurvorbereitung gedacht:

**Aufgabe 6:**

Eine Distributed Hash Table benutzt Chord als Implementierung identisch mit Aufgabe 3. Die Keys haben eine Länge von 8 Bits. Es sind 8 Knoten vorhanden.

(a) In welchen Knoten sind die Werte mit den Keys 99 bzw. 240 jeweils gespeichert?

(b) 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?

(c) Wie viele Nachrichten werden **ohne** Benutzung von Finger Tables maximal benötigt, um im dargestellten System von einem beliebigen Knoten einen beliebigen Key abzufragen?