TKN Telecommunication Networks Group

Praktikum
Rechnernetze und Verteilte Systeme

Block 4

— DHT & P2P —

Termin: 27.11.-1.12.2017 & 4.-8.12.2017

1 Theoretische Vorbereitungsaufgaben

Die folgenden Aufgaben sollen Ihnen helfen, sich auf den Vorbereitungstest vorzubereiten. Klären Sie bitte mögliche Fragen oder Unklarheiten unbedingt vor den ISIS-Testaten!

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?

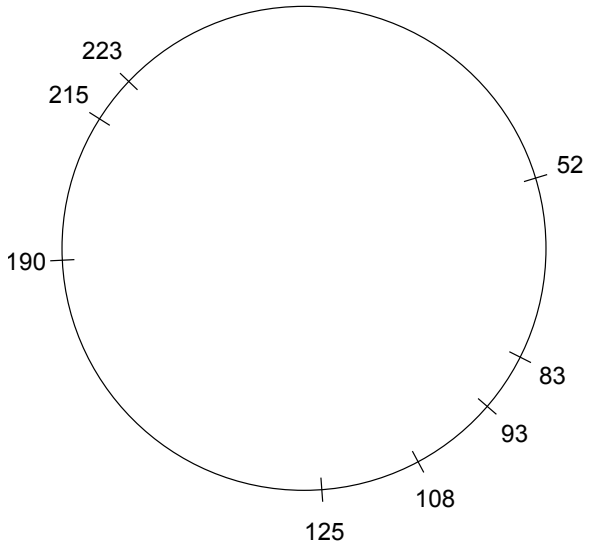
d) Erinnern Sie sich an die Struktur von DNS. Welche Strukturen haben DHTs im Vergleich zu DNS?

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) Welches ist der Eintrag mit $i = 3$ in der Finger Table des Knoten mit der ID $n = 52$?

c) Wie vereinfacht eine Finger Table den Chord Lookup?

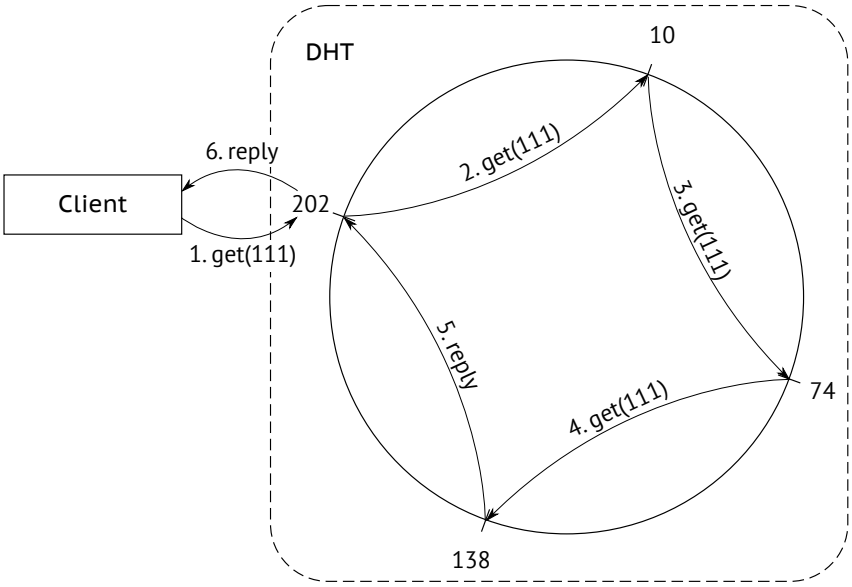
2 Praktische Aufgaben

Die praktischen Aufgaben sind in Kleingruppen von i. d. R. 4 Personen zu lösen. Sie haben dieses Mal 2 Wochen zur Bearbeitung der Aufgaben Zeit. **Fangen Sie dennoch rechtzeitig an!** Im zweiten Termin haben Sie die Möglichkeit, ihrem Tutor wertvolle Fragen zu stellen. Reichen Sie bitte den Quelltext bzw. Lösungen dieser Aufgaben bis Sonntag nach dem zweiten Termin 23:55 Uhr per ISIS ein.

Aufgabe 3:
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.

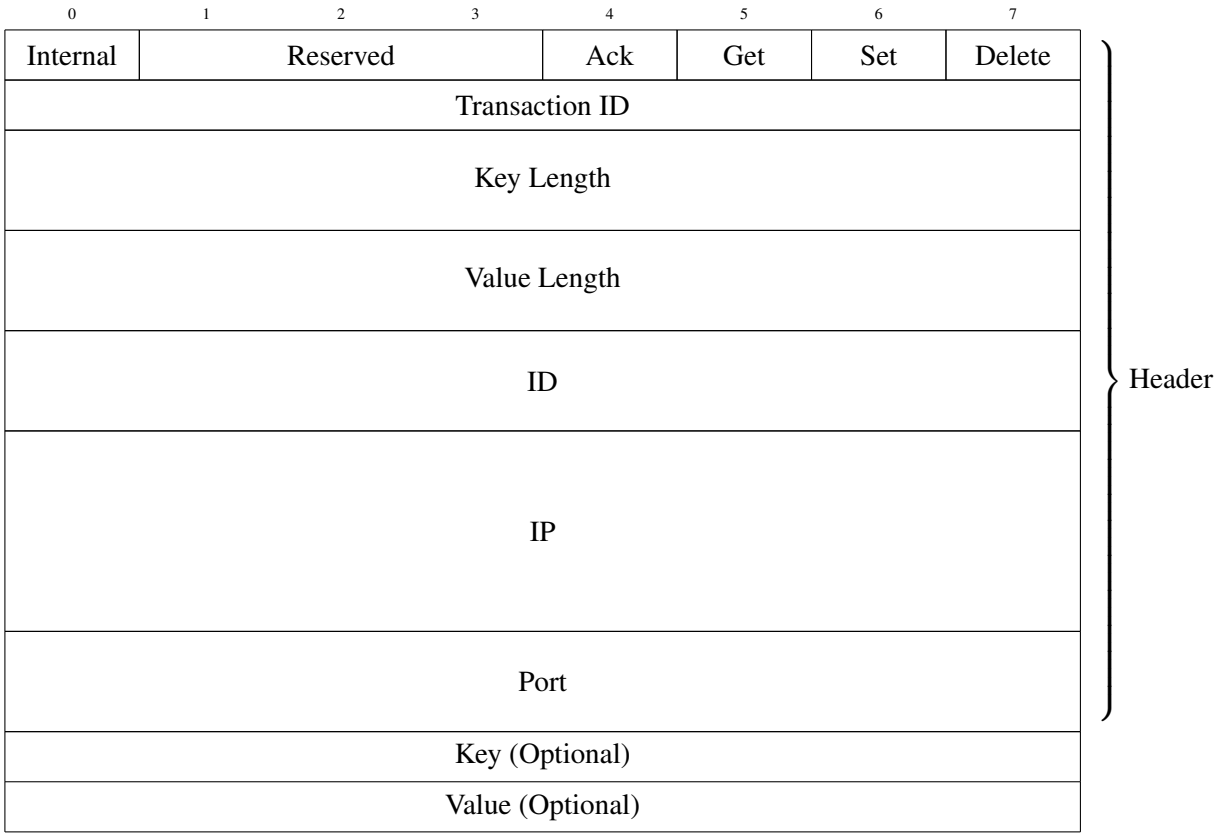
Nehmen Sie als Ausgangspunkt Ihre Implementierung aus der letzten Aufgabe. Die Funktionalität soll nun im wesentlichen gleich bleiben, jedoch die Speicherung der Werte über mehrere Prozesse bzw. Rechner verteilt werden. Ihre (jetzt verteilte) Hashtabelle sollte nach wie vor das gleiche Interface nach 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 dort ggf. an den Knoten weitergeleitet, der dafür zuständig ist, die jeweiligen Daten zu speichern bzw. auszulesen. Bitte beachten: Dies ist etwas anders als in der Vorlesung! In der Vorlesung ist der Ergebnis des Chord-Lookups lediglich, welcher Peer für welchen key zuständig ist. Eine Abfrage müsste gesondert stattfinden. Zur Vereinfachung passieren diese Schritte hier zusammen. Die theoretischen Aufgaben bzw. die Klausur nehmen jeweils bezug auf den Chord-Lookup im Sinne der Vorlesungsunterlagen.

Im Vergleich zur letzten Aufgabe enthält das Nachrichtenformat jetzt zusätzlich Felder für ID, IP und Port:



Dabei ist “internal” der Indikator für eine interne Nachricht innerhalb des Rings. Die IP ist zunächst die IP des ersten Knotens, der den Befehl von außen erhalten hat. Wir beschränken uns hier auf IPv4. Bei dieser Version sind die Adressen 32bit lang. Der dazugehörige Port ist 16bit lang. Achten sie wie immer darauf, dass Zahlen in Network-Byte-Order versendet werden. Wenn eine Anfrage beantwortet wird, schreibt der antwortende Peer jeweils seine eigene Adresse und seinen eigenen Port in die Antwort.

Fur die Rücksprache sollte ein Knoten mit jeder erhaltenen Nachricht folgendes ausgeben:

- Art der Nachricht
- ID, IP-Adresse und Port des Absenders
- ID, IP-Adresse und Port des aktuellen Vorgänger-Knotens
- ID, IP-Adresse und Port des aktuellen Nachfolge-Knotens

Schreiben Sie Ihren Hash Table Server so um, dass er als Peer in einem DHT P2P Netz teilnimmt. Dabei liegt der Fokus auf dem Auslesen der Werte. 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.

Zu beachten ist bei der Aufgabe, dass der Peer nun mehrere Verbindungen gleichzeitig bearbeiten können muss: 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]. Um anhand der Transaction-ID Antworten den richtigen Clients

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

zuzuordnen, müssen die Peers mindestens die Zuordnung der Transaction-ID zum Socket-Deskriptor speichern. Ggf. macht es Sinn, auch weitere Zustandsdaten zu speichern. Für solch eine Zuordnung von solchen `<key, value>` Paaren bietet sich natürlich eine interne Hashtable an.

Reichen Sie bitte ihre Lösungen bis Sonntag nach dem zweiten Termin 23:55 Uhr per ISIS ein. Die Lösungen werden **automatisch überprüft** und müssen ohne Ausnahme in einer Datei mit dem Namen **Block4.TXXGYY.tar.gz** (XX = Praktikumstermin, YY = Gruppennummer) eingereicht werden. Darin enthalten erwarten wir in einem Ordner mit dem Namen Block4.TXXGYY ein Makefile, welches die Binärdatei `peer` entsprechend kompiliert.

Hinweis: Da Sie zum Testen mehrere Server auf unterschiedlichen Ports betreiben und regelmäßig neu starten müssen, empfiehlt es sich, den folgenden Code-Ausschnitt im Server zu verwenden:

```
1 int yes = 1;
2 setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes));
```

3 Vertiefungsaufgaben

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

Aufgabe 4:
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 5:
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)?

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?