TKN Telecommunication Networks Group

Praktikum
Rechnernetze und Verteilte Systeme

Block 5

— Distributed Hash Tables II & HTTP/REST —

Termin: 10.–14.12.2018 & 17.12.2018–21.12.2018

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:
Beantworten Sie im Kontext vom Hypertext Transfer Protocol (HTTP) folgende Fragen:

a) Welche HTTP Methoden gibt es und was machen diese?

b) Welche HTTP Methoden sind idempotent?

c) Was sind persistente und nicht-persistente Verbindungen?

d) HTTP ist ein zustandsloses Protokoll. Was bedeutet das? Mit welcher Technik können Server z.B. trotzdem den Warenkorb einer Nutzers speichern?

Aufgabe 2:
Beantworten Sie im Kontext vom Caching im WWW folgende Fragen:

a) Warum wird Caching genutzt?

b) Welche Nachteile hat Caching?

c) Welche verschiedenen Arten von Caching werden im WWW eingesetzt?

d) Was bedeutet es für das Caching, wenn Methoden idempotent sind?

e) Mit welchen HTTP-Headern kann das Caching gesteuert werden? Was sind dabei die Unterschiede?

Aufgabe 3:
Beantworten Sie im Kontext vom RESTful Webservices folgende Fragen:

a) Was ist REST? Was genau bedeutet “Representational State Transfer”?

b) Was sind die Haupt-Prinzipien von REST, die in der Vorlesung angesprochen werden?

c) Welche Eigenschaften verspricht man sich von der Nutzung von REST?

d) Was kann bei REST (k)eine Ressource sein? Was ist dabei der Unterschied zu klassischen Webservices aufbauend auf WSDL/SOAP?

e) Welche Methoden können auf Ressourcen ausgeführt werden?

f) Wann ist eine Ressource cachable?

2 Praktische Aufgaben

Die praktischen Aufgaben sind in Kleingruppen von i. d. R. 3 Personen zu lösen. Die Ergebnisse des ersten Termins führen Sie im zweiten Termin dem Tutor vor. Reichen Sie bitte des weiteren den Quelltext bzw. Lösungen bis Sonntag vor dem dem zweiten Termin 23:55 Uhr per ISIS ein.

2.1 In der ersten Woche zu lösen

Aufgabe 4:
Im letzten Block haben Sie bereits eine statische DHT implementiert. In dieser Aufgabe soll nun der Chord-Join implementiert werden. Dabei können Sie vereinfachend davon ausgehen, dass nach dem Eingang des ersten `set`-Befehls dem Chord-Ring keine weiteren Knoten hinzugefügt werden.

a) Schreiben Sie hierzu zunächst Ihren Server wie folgt um:

- Werden nur IP-Adresse und Port übergeben, soll der Knoten später als Basis für einen neuen Ring fungieren. Optional soll eine beliebige (16-Bit) ID übergeben werden können, unter der der Knoten später arbeitet; als Standard-ID ist andernfalls stets 0 zu verwenden.
- Soll sich der Knoten in einen bestehenden Ring einklinken, so benötigt er hierfür zusätzlich IP-Adresse und Port eines beliebigen anderen, bereits im Ring aktiven Knotens, sowie eine freie (16-Bit) ID. Sie können davon ausgehen, dass die IDs eindeutig vergeben werden und müssen daher nicht gesondert prüfen ob die gewählte ID noch frei ist. Der Aufruf soll dann wie folgt aussehen:

```
./peer IP Port [ID] [Peer-IP Peer-Port]
```

b) Implementieren Sie anschließend das Chord-Joining-Verfahren, wie in der Vorlesung vorgestellt. Hierzu müssen `join`-, `notify`- und `stabilize`-Nachrichten ausgetauscht werden. Verwenden Sie für alle Nachrichten bitte ein modifiziertes erstes Byte im Header mit zusätzlichen Flags für die neuen Nachrichtentypen:

0	1	2	3	4	5	6	7
Internal	Join	Notify	Stabilize	Ack	Get	Set	Delete

Überlegen Sie sich hier auch, welche IP-Adresse, Port und ID jeweils bei den neuen Befehlen geschickt werden müssen.

Am Ende dieser Aufgabe sollte es möglich sein, eine beliebige Anzahl an Knoten zu starten, die sich immer wieder zu einem gültigen Chord-Ring zusammensetzen. Dabei sollte jeder Knoten seinen korrekten Nachfolger und Vorgänger im Ring kennen und kontaktieren können. Sie können vereinfachend davon ausgehen, dass kein Knoten je den Ring verlässt. *Achtung: Welchen bereits aktiven Knoten ein neu hinzukommender Knoten beim Chord-Joining kontaktiert, darf nicht durch Sie vorausgesetzt werden und sollte das Ergebnis niemals beeinflussen!*

Reichen Sie bitte ihre Lösungen bis Sonntag vor dem zweiten Termin 23:55 Uhr per ISIS ein. Die Lösungen werden **automatisch überprüft** und müssen ohne Ausnahme in einer Datei mit dem Namen **Block5a.TXXGYY.tar.gz** (XX = Praktikumstermin, YY = Gruppennummer) eingereicht werden. Darin enthalten erwarten wir in einem Ordner mit dem Namen Block5a.TXXGYY ein Makefile, welches die Binärdatei `peer` entsprechend kompiliert.

2.2 In der zweiten Woche zu lösen

Aufgabe 5:
Ihre DHT benutzt noch keine Finger-Tables, um schneller entfernte Knoten auf dem Ring zu kontaktieren. In dieser Aufgabe sollen diese implementiert und evaluiert werden.

a) Auf ISIS wird Ihnen ein Skript bereitgestellt, welches eine Vielzahl von Client-Aufrufen startet mit zufälligen `SET` Anfragen an ihre Peers. Anschließend werden die selben Anfragen noch einmal getätigt. Dabei wird die Zeit gemessen die für den gesamten Vorgang benötigt wird und anschließend ausgegeben. Da der Effekt einer Finger-Table erst mit steigender Ringgröße sichtbar wird, starten Sie zunächst lokal einen Ring mit 10 Knoten mit jeweils aufsteigendem Port (bspw. 8000–8009). Starten Sie anschließend das Skript mit dem ersten Port als Parameter:
`./measure 8000`
Notieren Sie sich die ausgegebene Zeit.

b) Passen Sie Ihre Peers so an, dass diese kontinuierlich ihre Finger-Table aufbauen. Sie können dafür ausnutzen, dass die zuständigen Peers für einen Wert auf eine Anfrage mit Ihrer eigenen IP-Adresse bzw. ihrem eigenen Port antworten werden.

c) Benutzen Sie Ihre Finger Table, um Anfragen effizienter an Ihr Ziel zu bringen.

d) Wiederholen Sie Ihren Test aus dem ersten Aufgabenteil, und notieren Sie sich wieder die Zeit.

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 **Block5b.TXXGYY.tar.gz** (XX = Praktikumstermin, YY = Gruppennummer) eingereicht werden. Darin enthalten erwarten wir in einem Ordner mit dem Namen Block5b.TXXGYY ein Makefile, welches die Binärdatei `peer` entsprechend kompiliert. Fügen Sie diesmal bitte auch eine Textdatei mit den Messergebnissen aus beiden Versuchen hinzu.[1]

[1] Da diese nur auf Plausibilität überprüft wird ist kein vordefiniertes Format notwendig.

3 Vertiefungsaufgaben

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

Aufgabe 6:
Nennen Sie die wesentlichen Unterschiede zwischen HTML, XML und XHTML!

Aufgabe 7:
Laden Sie die WSDL-Datei des TKN-Sensornetzwerk-Gateways von der ISIS-Seite herunter (inzwischen wird der Service zu Gunsten von REST nicht mehr angeboten).

a) Interpretieren Sie knapp die allgemeine Bedeutung der folgenden Elemente[2]:
   - `<types>`
   - `<message>`
   - `<interface>/<portType>`[3]
   - `<binding>`
   - `<service>`

b) Welche Adresse hat der in der WSDL-Datei beschriebene Web-Service?

c) In welcher Programmiersprache ist der Web-Service implementiert? In welcher Programmiersprache ist der Web-Service-Client zu implementieren?

Aufgabe 8:
UDDI war nie so weit verbreitet, wie die Erfinder gehofft hatten. IBM, Microsoft und SAP kündigten bereits im Januar 2006 an, ihre öffentlichen UDDI-Knoten zu schließen. Kann ein Web-Service auch ohne "Universal Description, Discovery and Integration" (UDDI)-Eintrag angeboten werden? Ergibt das ggf. Sinn?

Frohe Weihnachten und ein frohes neues Jahr! ☺

[2] Hilfreich ist dabei die z.B. die WSDL-Spezifikation: `http://www.w3.org/TR/wsdl`
[3] Beide Bezeichner werden genutzt; im konkreten Beispiel `<portType>`