ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2016 Prof. R. Wattenhofer

Technische Informatik II

Probeklausur

Donnerstag, 02. Juni 2016, 10:15 – 11:45 Uhr

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten und beschriften Sie Skizzen und Zeichnungen verständlich.

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie jedes beschriebene Blatt mit Ihrem Namen und Ihrer Legi-Nummer.

Name	**Legi-Nr.**

Punkte

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1		15
2		15
3		8
4		7
5		13
6		9
7		13
8		10
Summe		**90**

1 Multiple Choice (15 Punkte)

Aussage	**Wahr**	**Falsch**
Man kann die Entschlüsselung von ECB parallelisieren.	☐	☐
Elgamal-public-key-Verschlüsselung schützt gegen replay-Attacken.	☐	☐
`testAndSet()`, `getAndIncrement()`, `compareAndSwap(old, new)` sind Read-Modify-Write-Anweisungen.	☐	☐
Das Anderson-Queue-Lock verwendet linked lists, und jeder Prozess muss warten, bis das Lock seine node in der Liste erreicht.	☐	☐
Locking ist nur auf Mehrkern-Systemen notwendig.	☐	☐
Um einen Knoten in eine Liste hinzuzufügen muss man beim optimistischen Locking nie mehr als zwei Locks beantragen.	☐	☐
UDP baut eine Verbindung zwischen Quelle und Ziel auf, bevor die eigentlichen Daten übertragen werden.	☐	☐
Wenn ein multi-commodity flow network mit den Raten $r_1, \ldots, r_k$ in ein Netwerk passt, dann passt auch ein unsplittable multi-commodity flow mit den selben Raten in das selbe Netzwerk.	☐	☐
Für einen einfachen random walk auf einem Ring mit n Knoten ist die hitting time $T_{i,i} = n$ für jeden Knoten i.	☐	☐
Wenn eine Markovkette eine eindeutige stationäre Verteilung hat, dann ist sie ergodisch.	☐	☐
Damit eine Hashtabelle gute Laufzeiten bietet, muss die verwendete Hashfunktion collision resistant sein.	☐	☐
Ein outer join mit einer leeren Tabelle ändert die Anzahl der Ergebniszeilen nicht.	☐	☐
Wenn man das Internet als Graph modelliert, dann erhält man einen Baum.	☐	☐
Im Gegensatz zu pipes können sockets auch dazu verwendet werden, zwischen Prozessen auf verschiedenen Rechnern im Netzwerk zu kommunizieren.	☐	☐
Ein Betriebssystem ohne preemption kann Prozesse nicht schedulen.	☐	☐

2 Locking (15 Punkte)

Skiplisten (engl. *skip lists*) bauen auf einfach verketteten Listen auf und verbessern die durchschnittliche Laufzeit von Suchoperationen von $O(n)$ auf $O(\log n)$. Dazu befinden sich die Knoten ihrem Schlüssel nach sortiert in der Liste und besitzen zusätzliche Vorwärtsreferenzen.
Eine Skipliste mit k Levels enthält Knoten mit bis zu k Vorwärtsreferenzen. Die Knoten eines Levels $i > 1$ bilden jeweils eine einfach verkettete Liste, die eine Untermenge der Liste von Level $i - 1$ ist. Level 1 enthält schließlich alle Knoten.
Um nun einen bestimmten Schlüssel zu finden, beginnt man die Suche in der Liste des höchsten Levels, und steigt immer ein Level nach unten, wenn man den Schlüssel verpasst hat. Algorithm 1 (nächste Seite) zeigt eine Implementierung der Löschoperation.

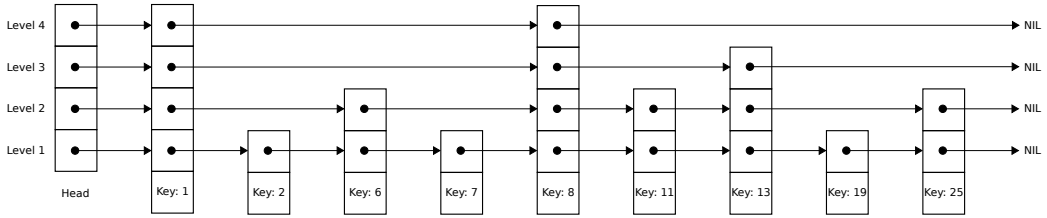

Abbildung 1: Beispiel einer Skipliste mit vier Levels

a) `[3 Punkte]` Ohne Locking können „böse Dinge“ passieren, wenn mehrere Threads gleichzeitig Operationen auf dieser Datenstruktur ausführen. Geben Sie ein Beispiel.

b) `[6 Punkte]` Ergänzen Sie das Grundgerüst von Algorithm 1 um Hand-over-Hand Locking. Verwenden Sie dazu *LOCK(n, i)* und *UNLOCK(n, i)*, die jeweils das Lock für Level i von Knoten n sperren/entsperren. Nicht jede Zeile wird benötigt. Es dürfen mehrere Statements pro Zeile verwendet werden.

c) `[6 Punkte]` Eine stark parallelisierte Anwendung verwendet Skiplisten, um Nachrichten von Produzenten- an Konsumenten-Threads zu übergeben. Viele Konsumenten lesen und entfernen einfach das Element mit dem kleinsten Schlüssel, was zu viel Contention an den Locks des Head-Knotens führt. Wie könnte man die Datenstruktur anpassen, so dass die Löschoperation ohne Locks auskommt? Welche Nachteile handelt man sich damit ein?

Algorithm 1: Element aus Skipliste mit k Levels entfernen

```
1  n ← Head // Zuerst suchen wir nach dem Schlüssel
2  .....
3  for i ← k to 1 do
4      .....
5      while n.next[i] ≠ NIL and n.next[i].key < key do
6          .....
7          temp ← n.next[i]
8          .....
9          n ← temp
10         .....
11     previous[i] ← n // Vorgänger für jedes Level merken
12     .....
13 .....
14 n ← n.next[i]
15 if n ≠ NIL and n.key = key then
       // Haben wir den Schlüssel gefunden?
16     .....
17     for i ← 1 to n.height do
18         .....
19         previous[i].next[i] ← n.next[i] // Umsetzen der Zeiger
20         .....
21     .....
22 .....                    // Locks wieder freigeben (Aufgabenteil B)
23 .....
24 .....
25 .....
26 .....
```

3 Disks (8 Punkte)

You are given two HDDs with the following specifications: Du willst für deinen Rechner zuhause schnellere Schreibzeiten und entscheidest dich, zwei HDDs mit den folgenden Eigenschaften zu verbauen:

- HDD A: 5ms average seek, 9000 rounds per minute, transfer rate 100MB/s
- HDD B: 2ms average seek, 6000 rounds per minute, transfer rate 150MB/s

Nimm an, dass beide HDDs eine Sektorgrösse (sector size) von 4KB haben, und dass 1000KB = 1 MB. Dein Betriebssystem ist dazu in der Lage, Dateien beim Schreiben in jedem beliebigem Verhältnis (beispielsweise zu 40% auf A und zu 60% auf B) auf die beiden HDDs aufzuteilen.

a) `[5 Punkte]` Wie lange dauert es, 100MB uniformverteilte Schreibzugriffe auf die beiden HDDs umzusetzen, wenn jede HDD genau 50% der Befehle verarbeiten muss?

b) `[3 Punkte]` In welchem Verhältnis müssen die Schreibbefehle auf die beiden HDDs aufgeteilt werden, um die geringstmögliche Zeit zu benötigen?

4 Dateisysteme (7 Punkte)

Wir haben ein inode-basiertes Dateisystem, in dem jeder Block $2048 = 2^{11}$ Bytes gross ist und Blocks mittels 32-bit Adressen referenziert werden. Jede inode verwaltet bis zu 12 direct pointers, und je einen singly-indirect, doubly-indirect, und triply-indirect pointer.

Hinweis: Sie müssen Exponenten nicht auswerten; ein vereinfachter Term genügt für jede Teilaufgabe als Lösung.

a) `[2 Punkte]` Wie viel Speicherplatz kann dieses Dateisystem maximal verwalten?

b) `[2 Punkte]` Wie gross kann eine Datei maximal sein?

c) `[3 Punkte]` Wieviele Datenblocks werden insgesamt benötigt, um eine Datei von 2^{30} Bytes zu speichern?

5 Medium Access Control (13 Punkte)

In einem Sensornetzwerk sollen kabellos gelegentlich Daten von allen Sensorknoten zu einer Basisstation gesandt werden. Dazu steht nur ein einziges Frequenzband zur Verfügung. Um Strom zu sparen, soll die Datenübermittlung so effizient wie möglich ablaufen; insbesondere soll es so wenig Kollisionen wie möglich geben, und es soll kein Coding Division verwendet werden. Da die Sensorknoten sich zwischen Messungen oder Übermittlungen selbstständig in einen Schlafmodus schalten, besteht keine Uhrensynchronisierung zwischen den Knoten.

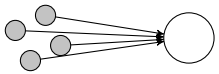

Abbildung 2:

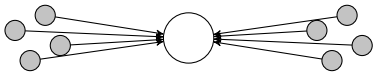

Abbildung 3:

a) `[3 Punkte]` Das Sensornetzwerk ist räumlich wie in Abbildung 2 ausgelegt. Beschreibe ein MAC-Schema, das Kollisionen in diesem Szenario (fast) immer vermeiden kann.

b) `[4 Punkte]` Nun gibt es zwei Gruppen von Sensorknoten, die so weit von einander weg sind, dass sie sich gegenseitig nicht hören können, siehe Abbildung 3. Zu welchem Problem führt diese Konstellation? Funktioniert deine Lösung aus **a)** noch? Wenn nein, beschreibe ein MAC-Schema, das mit dieser neuen Situation umgehen kann.

Jetzt wurden die Sensorknoten mit GPS-Empfängern ausgestattet, die ihnen jederzeit Zugang zu einer akkuraten „globalen Uhr“ geben. Messungen sollen nun immer gleichzeitig genommen werden.

c) `[3 Punkte]` Wie erlaubt dies, Kollisionen weiter zu verringern oder gar vollständig zu eliminieren?

d) `[3 Punkte]` Betrachte nun den Fall mit 4 Gruppen (siehe Abbildung 4). Wie kann man den Stromverbrauch sowie die Dauer bis zum Abschluss aller Übertragungen einer Messungsrunde reduzieren, wenn man die Sendestärke der Knoten variieren kann?

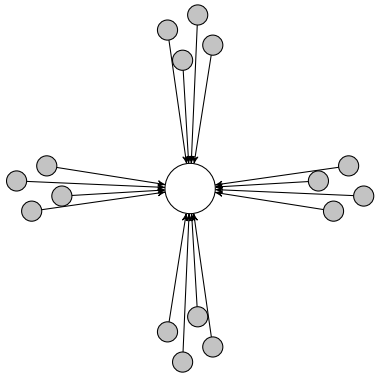

Abbildung 4:

6 Hashing (9 Punkte)

a) `[6 Punkte]` Wir betrachten die parametrisierte Hashfuntion $h_i(k) = h(k)+i+i^2 \mod m$ für eine beliebige Hashfuntion $h : U \to [m]$ und eine Primzahl $m > 2$. Zeigen Sie, dass die probing sequence für jedes $k \in [m]$ höchstens $\lceil \frac{m}{2} \rceil$ Elemente von $[m]$ abdeckt.

b) `[3 Punkte]` Sei unser Universum $U = [2^{2w}]$ die Menge aller Bistrings der Länge $2w > 0$, sei unsere Hashtabelle das Array $[2^w]$, und sei $h : U \to [2^w]$ eine Hashfunktion. Zeigen Sie, dass es eine Menge von mindestens 2^w keys gibt, die alle denselben Hash haben.

7 Nachtwache (13 Punkte)

Um ihre schlechte finanzielle Lage aufzubessern arbeiten Darya und Sebastian auch nachts. Ihre Aufgabe besteht darin, eine berühmte Serverfarm zu bewachen, deren Architektur folgendermassen aussieht:

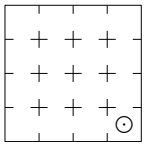

Abbildung 5: Räume der Serverfarm. Es gibt 4x4 Räume, die alle wie eingezeichnet mit Türen verbunden sind.

In einem ersten Szenario patrouillieren Darya und Sebastian immer zusammen. Sie beginnen im Raum oben links. Jede Minute gehen sie in einen anderen Raum, den sie uniformverteilt aus den benachbarten Räumen auswählen.

a) `[8 Punkte]` Berechnen Sie die Wahrscheinlichkeit in der stationären Verteilung, dass Darya und Sebastian in dem Raum sind, durch den der Offline-Angreifer die Serverfarm betritt (markiert durch $\odot$)!

b) `[5 Punkte]` Da Darya und Sebastian sehr stark sind, kann jeder von ihnen den Angreifer auch alleine stellen. Sie patrouillieren darum in einem zweiten Szenario unabhängig voneinander. Sie sind sich bewusst, dass ihre Wahrscheinlichkeitsverteilung nicht notwendigerweise gegen die stationäre Verteilung konvergiert, und überlegen, in welchen Räumen sie starten sollten. Sei X der Raum, in dem der Angreifer die Farm betritt, und sei $t \geq 60$ eine ganze Zahl. Wenn Darya im Raum oben links startet, wo kann Sebastian dann starten, sodass die Wahrscheinlichkeit, dass sich nach genau t Minuten mindestens einer der beiden Wächter in X befindet, strikt grösser als 0 ist?

8 Key Exchange Variante (10 Punkte)

Betrachten Sie die folgende Variante eines Key-Exchange-Algorithmus:

Algorithm 2: Neues Schlüsselaustauschverfahren mit Forward Secrecy

Input : Alices und Bobs gemeinsamer geheimer Schlüssel k_{shared}.
Ergebnis: Ein Diffie-Hellman Schlüsselaustauschverfahren (key exchange), das hoffentlich nicht gegen eine man-in-the-middle-Attacke anfällig ist und zusätzlich forward secrecy bietet.

1 Alice und Bob führen den Algorithmus Diffie-Hellman Key Exchange aus, um Rundenschlüssel (round key) $g^{k_A k_B}$ zu erhalten; k_A und k_B sind die privaten Exponenten von Alice und Bob
2 Alice und Bob verschlüsseln $g^{k_A k_B}$ mit k_{shared} und senden sich das Ergebnis gegenseitig zu
3 Falls die Entschlüsselung $g^{k_A k_B}$ ergibt, so akzeptieren beide $g^{k_A k_B}$ als den Rundenschlüssel

a) `[2 Punkte]` Inwiefern ist Diffie-Hellman gegen eine man-in-the-middle-Attacke anfällig?

b) `[3 Punkte]` Hat Alrogithmus 2 die Eigenschaft forward secrecy?

c) `[5 Punkte]` Ist Algorithmus 2 anfällig gegen eine man-in-the-middle-Attacke?