ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2015 Prof. Dr. Roger Wattenhofer

Prüfung
Verteilte Systeme
Teil 2

Mittwoch, 27. Januar 2016
15:00 – 18:00

Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Fragen auf Englisch oder auf 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.

Name	Legi-Nr.

Punkte

Frage Nr.	Erreichte Punkte	Maximale Punkte
8		22
9		16
10		24
11		16
12		12
Total		90

8 Multiple Choice (22 Punkte)

Beurteilen Sie, ob die folgenden Aussagen richtig oder falsch sind, und kreuzen Sie die entsprechenden Felder an. Eine richtig beurteilte Aussage gibt 1 Punkt, eine nicht beurteilte Aussage 0 Punkte, eine nicht richtig beurteilte Aussage **-1 Punkt**. Die **gesamte** Aufgabe wird mit minimal 0 Punkten bewertet.

A) Clock Synchronization

Aussage	**Wahr**	**Falsch**
GPS und NTP versuchen den Einfluss der Propagation Delays zu kompensieren um eine genauere Zeitsynchronisation zu erreichen.	☐	☐
GTSP kann einen kleineren maximalen Local Skew als FTSP erreichen falls die Sensorknoten auf einem Grid (20 auf 20 Knoten) angeordnet sind.	☐	☐

B) Consensus & Consistency

Aussage	**Wahr**	**Falsch**
Weil Paxos nur crash failures tolerieren muss, können bis zu $n-1$ viele Server abstürzen und das System kann sich immer noch einigen.	☐	☐
Byzantine Agreement ist für $f = n/3$ nur mit Hilfe eines byzantinischen Quorumsystems lösbar.	☐	☐
Die *Resilience* eines Quorumsystems ist nie grösser als die Grösse des kleinsten Quorums.	☐	☐
Der King-Algorithmus funktioniert nur, weil man immer mindestens einen nicht-byzantinischen König hat.	☐	☐
In Zyzzyva kann kein Kommando complete sein falls der Client selbst es nicht als complete betrachtet.	☐	☐

C) Bitcoin

Aussage	**Wahr**	**Falsch**
Die Summe der Input-Werte einer Transaktion muss kleiner oder gleich der Summe der Output-Werte sein.	☐	☐
Ein Blockchain-Fork wird dadurch aufgelöst dass einer der beiden Äste länger wird und alle Teilnehmer auf diesen Ast wechseln.	☐	☐
Alle Transaktionen im Memory Pool werden früher oder später bestätigt.	☐	☐
Ein Doublespend bedeutet dass zwei wiedersprüchliche Transaktionen in der Blockchain bestätigt werden.	☐	☐
Ein Benutzer kann beliebig viele Adressen erstellen.	☐	☐

D) Locking

Aussage	**Wahr**	**Falsch**
Bei einem Read-Write-Lock darf höchstens ein Leser oder ein Schreiber das Lock halten.	☐	☐
Mutual Exclusion ist ohne atomare RMW-Instruktionen auf Mehrprozessorarchitekturen nicht möglich.	☐	☐
Verwendet man Recursive Split Ordering auf einem Hash Set, so befinden sich die Elemente in der Liste absteigend nach ihrem Schlüssel sortiert.	☐	☐
Hand-über-Hand-Sperren verhindert, dass eine Datenstruktur beim Durchlaufen modifiziert wird.	☐	☐
Backoff-Locks garantieren Fairness nicht.	☐	☐
Queue-Locks spüren den Effekt des "Invalidation Storms", wenn die Flags der einzelnen Prozessoren im Speicher in einem zusammenhängenden Array dicht gepackt vorliegen.	☐	☐

E) Network Updates

Aussage	**Wahr**	**Falsch**
Durch das Einfügen einer geeigneten Forwarding Rule in jedem Switch können jegliche Blackholes vermieden werden.	☐	☐
Es ist NP-vollständig zu überprüfen ob das Einfügen einer neuen Forwarding Rule einen Loop erzeugt.	☐	☐
Mit genügend Speicher in den Switches können Network Updates immer Loop Free durchgeführt werden.	☐	☐
Die Anzahl von Schritten für ein Capacity-Consistent Network Update von Flows is höchstens exponentiell in der Anzahl der Knoten.	☐	☐

9 Mafia Games (16 Punkte)

Sie (Spieler u) sind der Chef des Mafia-Clans "Umberto". Ihr Erzfeind (Spieler v) ist der Chef des Mafia-Clans "Valerio". Sie und Valerio kämpfen um neue 2 Rekruten. Um diese anzulocken, können Sie (aber auch Spieler v) Bierfässer kaufen. Die neuen Rekruten sind sehr naiv, und treten der Mafia bei, die mehr Bier hat. Bei Gleichstand entscheiden sich beide Rekruten gegen eine kriminelle Karriere.

Jeder Rekrut bringt Ihnen einen Nutzen von 1. Jedes Fass Bier, das Sie kaufen, reduziert Ihren Nutzen um 1. Beispiel: Wenn Spieler u 5 Fässer kauft und Spieler v 4 Fässer kauft, hat Spieler u einen Nutzen von -3 und Spieler v einen Nutzen von -4.

A) (8 Punkte) Geben Sie den relevanten Teil des Spiels in Matrixform an. Hat das Spiel ein reines Nash-Gleichgewicht?

Falls ja:

- Geben Sie alle reinen Nash-Gleichgewichte an.
- Bestimmen Sie den (Optimistic) Price of Anarchy.

Falls nein:

- Begründen Sie warum nicht.
- Hat das Spiel ein gemischtes Nash-Gleichgewicht?

B) (8 Punkte) Nun gibt es 3 Rekruten. Ansonsten gibt es keine Änderungen an dem in **A)** beschriebenen Spiel.

Geben Sie den relevanten Teil des Spiels in Matrixform an. Hat das Spiel ein reines Nash-Gleichgewicht?

Falls ja:

- Geben Sie alle reinen Nash-Gleichgewichte an.
- Bestimmen Sie den (Optimistic) Price of Anarchy.

Falls nein:

- Begründen Sie warum nicht.
- Hat das Spiel ein gemischtes Nash-Gleichgewicht?

10 Poor Man's Consensus (24 Punkte)

In dieser Aufgabe betrachten wir das Consensus-Problem auf einem *allgemeinen* Graphen. Das heisst, dass es nicht zwischen jeden zwei Knoten eine Kante geben muss. Um von einem Knoten eine Nachricht an einen nicht direkt verbundenen Knoten zu schicken, muss diese über andere Knoten geschickt werden. Sie können annehmen, dass der Graph zu Beginn der Ausführung verbunden ist.

Annahme: Synchrones Zeitmodell, nur Crash-Failures.

A) (3 Punkte) Nennen Sie die drei Eigenschaften, welche ein Algorithmus erfüllen muss, so dass er Consensus löst, und erklären Sie kurz was die Eigenschaften bedeuten.

Definition (Fehlertoleranz f)**.** *Die Fehlertoleranz f eines Consensus-Systems ist die* ***maximale*** *Anzahl Knoten, so dass Consensus immer möglich ist, unabhängig davon,* ***welche*** *f Knoten abgestürzt sind.*

B) (8 Punkte) Beschreiben Sie einen Algorithmus mit möglichst grosser Fehlertoleranz, welcher Consensus auf *allgemeinen* Graphen (trotz Crash-Failures) löst. Sie können annehmen, dass alle Knoten n kennen.

In den folgenden Aufgaben nehmen wir an, dass wir einen Algorithmus mit optimaler Fehlertoleranz (mit grösstmöglichem f) verwenden.

Wir wollen nun ein konkretes Consensus-System bauen, und dazu Hardware verwenden, welche wir gerade noch rumliegen haben: Das sind 7 Server (Knoten) und 10 Kabel (Kanten). Nehmen Sie an, dass Sie an jeden Server so viele Kabel anschliessen können, wie Sie möchten.

C) (6 Punkte) Da Sie gelernt haben, dass die Fehlertoleranz von der Anzahl Server abhängt, entwerfen Sie ein Netzwerk welches alle 7 Server verwendet. Wie müssen die Kabel zwischen den Servern verbunden werden, so dass die Fehlertoleranz f des Systems möglichst gross wird? Wie gross ist f in diesem Fall?

Sie sind vom f, welches Ihr System in **C)** erreicht hat, enttäuscht. Mit der vorhandenen Hardware müsste doch eigentlich mehr möglich sein...

D) (4 Punkte) Können Sie ein grösseres f erreichen, indem Sie weniger als 7 Server verwenden?

Falls ja: Geben Sie ein Netzwerk an (Zeichnung) und bestimmen Sie das dazugehörige f.

Falls nein: Begründen Sie warum f mit weniger Servern nicht besser sein kann.

E) (3 Punkte) Wie gross kann f im Allgemeinen überhaupt sein? Sie können Kennzahlen des Graphs als gegeben annehmen, wie z.B. Anzahl Knoten, Anzahl Kanten, grösster Knotengrad, etc.

11 Quorumsysteme (16 Punkte)

Falls benötigt können Sie die Vorlagen auf den nächsten Seiten verwenden.

Die Firma Hexa & Göhne hat sich auf die Entwicklung von Quorumsystemen spezialisiert. Ihr aktuellstes Produkt ist `Hex 4`, welches aus 37 Hexagonen (Servern) besteht, welche in einem grossen Hexagon mit Seitenlänge 4 angeordnet sind (siehe Abbildung 1).

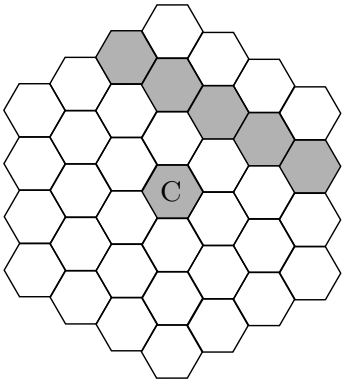


Figure 1: Quorumsystem `Hex 4`.

In `Hex 4` beinhaltet jedes Quorum alle Server einer Reihe des grossen Hexagons, und zusätzlich den zentralen Server C. Ein solches Quorum ist in Abb. 1 grau eingezeichnet.

A) (5 Punkte) Ist `Hex 4` ein gültiges Quorumsystem?

Falls ja: Begründen Sie kurz warum es ein gültiges Quorumsystem ist, und argumentieren Sie ob es ein gutes oder schlechtes Quorumsystem ist.

Falls nein: Begründen Sie kurz warum es kein gültiges Quorumsystem ist.

Hexa & Göhne bleibt immer am Ball, und so haben sie ein neues System entwickelt. Im Quorumsystem `Hex 12` besteht jedes Quorum aus 12 Hexagonen welche als Ring in Hexagonform mit Seitenlänge 3 angeordnet sind. Ein solches Quorum ist in Abb. 2.

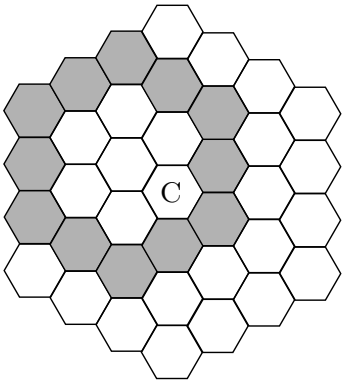
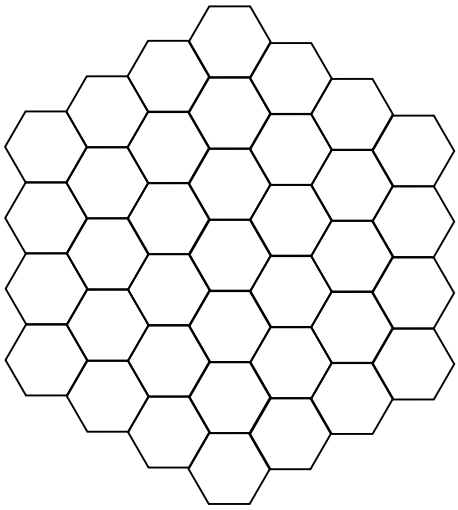


Figure 2: Quorumsystem `Hex 12`.

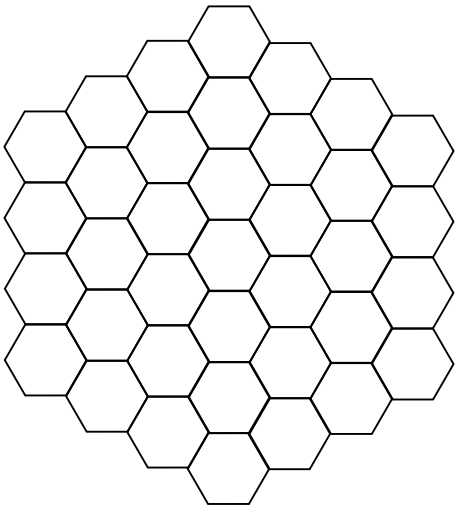
B) (3 Punkte) Das Quorumsystem besteht aus allen möglichen Quoren mit der genannten Form. Wieviele verschiedene Quoren sind in `Hex 12`?

C) (6 Punkte) Nehmen Sie an, dass alle Quoren mit der gleichen Wahrscheinlichkeit angefragt werden. Was sind *load, work* und *resilience* von `Hex 12`?

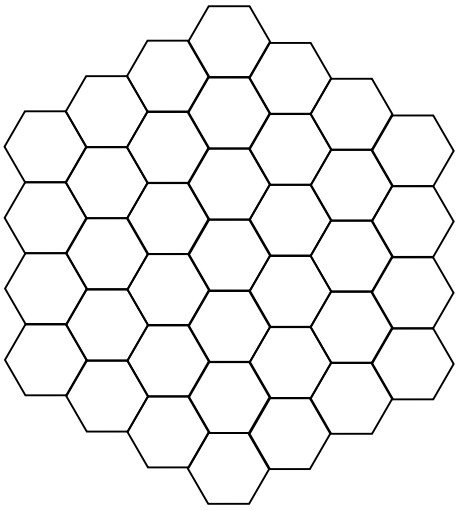
D) (2 Punkte) Alle Quoren in `Hex 12` werden gleich oft angefragt. Sind dann alle Server im Erwartungswert gleich viel beansprucht, d.h., in gleich vielen Anfragen involviert?



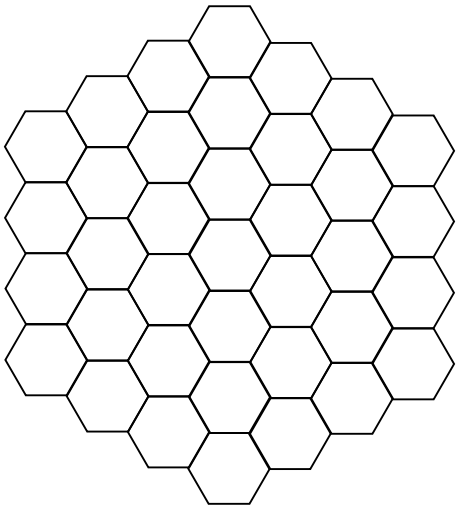
(a)



(b)



(c)



(d)

Figure 3: Kopien des Hexagons

12 Bitcoin vs. Strong Consistency (12 Punkte)

A) (4 Punkte) Vergleichen Sie die Konsistenz-Garantien von Zyzzyva mit denen von Bitcoin. Nennen Sie zwei Unterschiede.

B) (3 Punkte) Weshalb kann Bitcoin nicht Zyzzyva verwenden um Transaktionen zu bestätigen?

C) (5 Punkte) In einem Bitcoin-Netzwerk mit n Knoten hat jeder Knoten maximal 8 Nachbarn. Transaktionen und Blöcke werden nur an direkte Nachbarn weitergeleitet. Vergleichen Sie die Worst-Case Nachrichten-Komplexität eines solchen Bitcoin-Netzwerks mit der eines Zyzzyva-Systems, ebenfalls mit n Knoten.