Scheinprüfung Dependable Distributed Systems SS04

18. April 2005

Prüfer: Prof. Gärtner

Beisitzer: Martin Mink

Dauer: angesetzt 25 min, letztendlich waren es etwa 20 min

Verlauf: Prof. Gärtner fragte mich zu Beginn, ob ich in Verlässliche Verteilte Systeme vertiefen würde und auf meine positive Antwort, ob er die Fragen wie in der Diplom-Prüfung stellen soll. Ich bejate, somit sollte dieses Protokoll auch für die Diplom-Prüfungsvorbereitung nützlich sein.

G.: Gibt es ein Thema, das sie besonders interessant fanden, mit dem sie gerne anfangen würden?

I.: Interessant fand ich den CHT-Beweis, aber ob ich damit anfangen möchte..., vielleicht können wir langsam in die Richtung gehen?

G.: Consensus ist ja sehr wichtig, wir hatten das ja oft auch für andere Abstraktionen verwendendet. Wie ist den Consensus definiert?

I.: Validity: Jeder Wert, der entschieden wird...., Agreement: ...

G.: Wie kann man den NBAC mit Consensus implementieren?

I.: Jeder Prozess sendet seinen Wert an alle anderen, wartet auf die Werte oder die Crashmeldungen aller anderen, gibt 1 in Consensus ein, falls er von allen 1 erhalten hat, sonst 0, entscheidet das Consensus-Ergebnis.

G.: Welcher Fehlerdetektor wird dafür verwendet?

I.: Ein perfekter Fehlerdetektor.

G.: Kann man auch einen eventually perfect failure detector verwenden?

I.: Nein, korrekte Prozesse können eine Prozess der 1 vorschlägt und falsch detektiert wird nicht unbedingt von einem Prozess unterscheiden, der 0 vorschlägt, aber abstürzt. (Dazu Bilder gezeichnet)

G.: Gibt es dann einen schwächeren Fehlerdetektor als den perfekten, um NBAC zu implementieren?

I.: Ja, den Fehlerdetektor $?P + \chi$. $?P$ ist der anonyme perfekte Fehlerdetektor.

G.: Wie ist der denn definiert?

I.: Wenn ein Fehler auftritt, dann wird der irgendwann gemeldet, allerdings nicht welcher Prozess. Solange kein Fehler auftritt, meldet er auch nichts.

G.: Wenn wir Consensus haben, reicht es dann zusätzlich $?P$ zu haben um NBAC zu implementieren?

I.: (nach einigen Überlegen) Ja, in unserem Algorithmus reicht es zu wissen, ob ein Prozess fehlerhaft ist, damit er terminieren kann.

G.: Was müsste man denn zeigen, um zu zeigen dass ein gegebener Fehlerdetektor der schwächste Fehlerdetektor für NBAC ist?

I.: Einerseits, dass es einen Algorithmus gibt, der unter Verwendung dieses Fehlerdetektors D (und keines anderen) NBAC implementiert und andererseits, dass bei für jeden Fehlerdetektor D' der mittels Algorithmus A NBAC implementiert, D sich mittels D' und A implementieren lässt.

G.: Was ist denn der schwächste Fehlerdetektor für Consensus.

I.: Ω, $\Diamond S$, $\Diamond W$

G.: Wie lässt sich denn $\Diamond S$ mittels Ω implementieren?

I.: Ω liefert einen Prozess als Ausgabe und man weiß, dass irgendwann alle Prozesse die gleiche Ausgabe haben und dieser korrekt ist. Um $\Diamond S$ zu implementiert werden nun alle anderen Prozesse, bis auf die Ausgabe von Ω verdächtigt. (noch kurze Erläuterung, warum das $\Diamond S$ implementiert)

G.: Und wie lässt sich Ω mit $\Diamond S$ implementieren? Das ist schwieriger.

I.: Jeder Prozess hat für jeden Prozess einen Zähler. Regelmäßig wird der Fehlerdetektor abgefragt und für verdächtige Prozesse der Zähler erhöht. Die Zähler werden ausgetauscht und jeweils das Maximum genommen. Der Prozess mit dem kleinsten Zähler wird als Ausgabe für Ω gewählt.

G.: Wenn wir CHT-Beweis mit dem FLP-Beweis vergleichen, dann haben diese eine Gemeinsamkeit. Ist ihnen da etwas aufgefallen?

I.: Beide benutzen die Konfigurationen $c0, c1, \ldots, cn$, dass heißt die Konfigurationen ci, bei denen Prozess 1 bis i 1 vorschlagen, der Rest 0. Bei FLP wird damit gezeigt, dass es eine non-uniforme Initialkonfiguration gibt, bei CHT, dass wird damit der kritische Index definiert.

G.: Wie ist den Consensus mit byzantischen Prozessen definiert.

I.: Definition aufgezählt, wobei ich als Persistancy folgendes gesagt hatte: Wenn alle korrekten Prozesse den gleichen Wert vorschlagen, muss dieser Wert gewählt werden.

G.: Nur die korrekten?

I.: Man kann es auch für alle Prozesse fordern das ist äquivalent.

G.: Aber die Bedingungen sind doch nicht äquivalent?

I.: Nein, aber ein Algorithmus der das eine löst, muss auch das andere lösen.

G.: OK

Fazit: Die Prüfungsatmosphere war ziemlich locker und ich denke in einer Diplom-Prüfung wäre das genauso.