Vertiefungsprüfung verlässliche Verteilte Systeme

Verlässliche Verteilte Systeme I, V4
Verlässliche Verteilte Systeme II, V3
Datenkommunikation, V3
Verteilte Systeme, V2

Prüfer: Prof. F. Freiling
Beisitzer: Thorsten Holz
Datum: 13.12.2006

Die Prüfungsathmospäre war sehr angenehm. Die ruhige Art von Prof. Freiling nimmt einem recht schnell die Nervosität und die Prüfung selbst wirkt eher wie ein Gespräch anstatt einer typischen Frage-Antwort-Prüfung. Meine Prüfung wurde mit 1.0 bewertet.

Verlässliche Verteilte Systeme II, ca. 20min

FF: Was versteht man unter Consensus und einem Fehlerdetektor?

Ich: *Consensus zuerst umgangssprachlich erklärt, danach die vier formalen Eigenschaften (Validity, Integrity, Termination, Agreement) erläutert. Fehlerdetektoren als verteilte Orakel beschrieben, die Aussagen über abgestürzte Prozesse machen.*

FF: Wie findet man denn den schwächsten Fehlerdetektor D für ein Problem P?

Ich: *Man muss zwei Richtungen zeigen: 1. Man kann mit D tatsächlich P lösen (hinreichender Teil) und 2. jeder andere Fehlerdetektor, der P löst, kann D emulieren (notwendiger Teil).*

FF: Erklären Sie mal CHT.

Ich: *Komplett erklärt, inklusive des Beweises aus dem Anhang des CHT-Plays.*

FF: Nach der Vorlesung kommen immer mal wieder Studenten zu mir, die einen angeblich einfacheren Weg gefunden haben Ω zu simulieren: Jeder Prozess bringt seine PID in den Consensusalgorithmus A ein. Consensus wird dann dafür sorgen, dass sich alle Prozesse auf den gleichen Prozess einigen. Ist das tatsächlich Ω?

Ich: *Nein, denn Consensus gibt keine Garantien darüber, dass der ausgegebene Prozess auch korrekt ist.*

FF: Das sage ich den Studenten dann auch. Aber ein paar Tage später kommen sie mit einer erweiterten Version wieder zu mir: Die Prozesse führen periodisch Consensus aus, wobei sie auch wieder ihre PID voschlagen. Die gecrashten Prozesse werden irgendwann nicht mehr an den Consensusläufen teilnehmen, d.h. es werden nur noch PIDs von korrekten Prozessen vorgeschlagen. Ist denn das nun Ω?

Ich: *Nein, das ist immer noch nicht* Ω*, denn Consensus könnte in jedem Durchlauf einen anderen korrekten Prozess entscheiden.* Ω *verlangt aber, dass schließlich immer der gleiche Prozess ausgegeben wird.*

FF: Wir hatten in der Vorlesung ja einen Pseudo-Consensus-Algorithmus. Erklären Sie den doch mal.

Ich: *Den Algorithmus mit den drei Prozessen erklärt.*

FF: Könnte man denn diesen Algorithmus auf n Prozesse erweitern? Z.B. auf 4 Prozesse *(malt eine Anordung von 4 Prozessen)*?

Ich: *(nach kurzem Überlegen) Nein, denn wenn zwei benachbarte Prozesse 0 vorschlagen und die anderen beiden 1, dann kann in diesem Lauf sowohl 0 als auch 1 entschieden werden.*

Verteilte Systeme, ca. 15min

FF: In der Vorlesung wurden Vector-Timestamps vorgestellt. Was hat es denn damit auf sich?

Ich: *Mit Vector-Timstamps kann man eine kausale Ordnungsrelation auf einer Menge von Ereignissen erzeugen. Dann die Implementierung anhand von ein paar Skizzen erklärt.*

FF: Man kann Lamport-Timestamps als eine vereinfachte Variante der Vector-Timestamps betrachten. Wo ist denn der Unterschied?

Ich: *Bei den Lamport-Timestamps wird jede Nachricht mit einem Timestamp verschickt. Wenn dann ein Prozess eine Nachricht aus der Zukunft erhält, wird die lokale Uhr einfach auf den Timestamp der Nachricht+1 gesetzt.*

FF: Was sagen sie aus? Können Sie ein Anwendungsbeispiel geben?

Ich: *Ein wenig schwammig rumgeredet. Schließlich wollte er darauf hinaus, dass ein Lamport-Timestamp den längsten Pfad von Ereignissen in der Vergangenheit des aktuellen Ereignisses bezeichnet. Als Anwendungsbeispiel die Koordination von Datenbankzugriffen genannt.*

FF: Nun haben wir ja viel über logische Zeitsynchronisation gesprochen. Manchmal muss man aber auch die reale Zeit sychronisieren. Was gibt es denn da für Möglichkeiten?

Ich: *Es gibt verteilte Algorithmen wie z.B. den von Berkeley (kurz erklärt) und zentral gesteuerte, wie z.B. Cristians Algorithmus (auch erklärt). Außerdem gibt es natürlich NTP.*

FF: Bei NTP gibt es ja verschiedene Layer?

Ich: *Ja, die Timeserver sind hierarchisch angeordnet. Ausserdem berechnen die NTP-Server das Delay und den Offset zu anderen Servern. Beides grob erläutert.*

Datenkommunikation, ca. 5min

FF: Welches Protokoll benutzt NTP?

Ich: *Da man nur einzelne Pakete so schnell wie möglich verschicken möchte, benutzt NTP UDP.*

FF: Was macht UTP beim Verbindungsaufbau?

Ich: *Nichts, es gibt keinen.*

FF: Ok, aber TCP stellt ja eine Verbindung her. Was passiert denn da?

Ich: *Den 3-way-handshake erklärt.*

FF: Sie haben das SYN und das ACK flag genannt. Es werden aber auch Pakete mit einem gesetzten RST flag geschickt. Wann?

Ich: *Wenn man zu einem Port verbinden möchte, der geschlossen ist.*

Verlässliche Verteilte Systeme I, ca. 5min

FF: Gut, manchmal kommt aber gar kein Paket zurück. Woran kann denn das liegen?

Ich: *Einerseits können die Pakete von einer Firewall geschluckt werden und andererseits sieht die TCP Spezifikation vor, dass Pakete, die ungültige Flags gesetzt haben, von einem offenen Port ignoriert werden. Microsofts TCP-Stack ignoriert aber immer solche Pakete, egal ob der Port offen oder geschlossen ist.*

FF: Richtig, dann kann man mit Tools wie nmap OS-Fingerprinting machen. Sagen sie abschließend doch noch was zu Firewalls

Ich: *Iptables erläutert.*

FF: Gut, dann gehen Sie mal bitte kurz vor die Tür