ulm university universität
uulm

Entwurfsmethodik Eingebetter Systeme — **Übungsblatt**
Institut für Eingebettete Systeme/Echtzeitsysteme — F. Slomka, V. Pollex

Nummer	**Ausgabe**	**Abgabe**	**Besprechung**
5	28. November 2014	05. Dezember 2014	05. Dezember 2014

Aufgabe 5-1: Auslastungsanalyse und Testgrenzen

Tabelle 1: Taskmenge Γ_1

Task	p	c^+	d
τ_1	10	5	10
τ_2	55	10	55
τ_3	100	15	100
τ_4	200	20	200

Tabelle 2: Taskmenge Γ_2

Task	p	c^+	d
τ_1	10	5	5
τ_2	55	10	40
τ_3	100	15	75
τ_4	200	20	150

Gegeben seien die Taskmengen Γ_1 und Γ_2 aus Tabelle 1 und 2 mit der jeweiligen Periode p, maximale Nettoausführungszeit c^+ und Frist d. Die Tasks werden nach ihren Fristen geplant.

a) Für jede Task τ aus der Taskmenge Γ_1 gilt, dass die Frist gleich der Periode ist ($d_\tau = p_\tau$). Daher kann die Auslastungsanalyse angewendet werden.

   Beurteilen Sie unter Anwendung der Auslastungsanalyse ob die Taskmenge Γ_1 alle gesetzten Fristen einhält.

b) Sind Tasks vorhanden bei welchen die Frist d ungleich der Periode p sind, wie in der Taskmenge Γ_2, so kann die Auslatungsanalyse nicht länger verwendet werden. Stattdessen muss ein anderer Test wie der Rechenzeitanforderungstest durchgeführt werden. Dazu wird eine Testgrenze benötigt.

   Berechnen Sie für die Taskmenge Γ_2 die Hyperperiode, die Testgrenze nach Baruah und nach Ripoll. Berechnen Sie außerdem für alle 3 Testschranken die Anzahl der Punkte welche getestet werden müssten. Wählen Sie eine der 3 berechneten Schranken aus und wenden Sie den Rechenzeitanforderungstest auf die Taskmenge Γ_2 an. Beurteilen Sie ob alle Fristen eingehalten werden.

Aufgabe 5-2: Approximation durch straffen der Periode

Neben der Verkürzung der Testgrenze kann die Rechenzeitdichte approximiert werden um die Analyse effizienter zu gestalten. Gegeben Sei die Taskmenge Γ_3 aus Tabelle 3. Die Tasks werden gemäß Ihrer Fristen geplant.

Tabelle 3: Taskmenge Γ_3

Task	p	c^+	d
τ_1	22	4	15
τ_2	35	10	30
τ_3	57	18	45

a) Berechnen Sie die Hyperperiode der Taskmenge Γ_3 und die Anzahl der Punkte welche bei der Anwendung des Rechenzeitanforderungstests mit der Hyperperiode getestet werden müssten.

b) Wenden Sie die Approximation durch straffen der Periode an. Beurteilen sie dann mit Hilfe des Rechenzeitanforderungstests unter Verwendung der Hyperperiode (der Taskmenge mit den gestrafften Perioden) als Testgrenze ob alle Fristen eingehalten werden. Dazu ist es ratsam die gestrafften Perioden so zu wählen, dass die Anzahl der Punkte die getestet werden gering ist.

c) Das Finden von geeigneten gestrafften Perioden ist ein algorithmisch hartes Problem. Für Taskmengen für die gilt, dass die Frist und die Periode einer Task gleich sind ($d = p$) existiert die Auslastungsanalyse. Es ist wünschenswert einen ähnlich einfachen Test zu besitzen welcher auch für Taskmengen geeignet ist, bei welchen die Fristen der Tasks kleiner als deren Perioden sein können ($d \leq p$).

   Versuchen Sie solch einen Test zu entwickeln. Gehen Sie dabei von der Approximation durch straffen der Periode aus. Zeigen Sie anschließend die Korrektheit Ihres entwickelten Tests und wenden Sie den Test auf die Taskmenge Γ_3 an. Korrektheit bedeutet hier, gibt Ihr Test aus: “Alle Fristen werden eingehalten”, dann muss dies auch tasächlich der Fall sein.

Aufgabe 5-3: Approximation durch Superposition

Eine weitere Möglichkeit die Analyse effizienter zu gestalten ist die Approximation durch Superposition.

a) Analysieren Sie die Taskmenge Γ_2 mit der Approximation durch Superposition. Verwenden Sie einmal $k = 1$ Testpunkte pro Task und ein zweites mal $k = 2$ Testpunkte pro Task. Beurteilen Sie jedes mal ob die Taskmenge Γ_2 alle Fristen einhält.

b) Beim Rechenzeitanforderungstest werden unter Umständen sehr viele Punkte getestet, welches sehr lange dauern kann. Bei der Approximation durch Superposition können bereits einige wenige Testpunkte pro Task ausreichen um festzustellen, dass eine Taskmenge alle Fristen einhält. Je mehr Testpunkte pro Task verwendet werden, desto kleiner wird der Fehler durch die Approximation. Da bei der Approximation immer ein Fehler vorhanden sein wird kann es immer vorkommen, dass eine Taskmenge ihre Fristen eigentlich einhält aber dies durch die Approximation nicht erkannt wird.

   Versuchen Sie einen Algorithmus zu entwickeln welcher den Rechenzeitanforderungstest mit der Approximation durch Superposition kombiniert. Dieser Algorithmus soll exakt sein. Dies bedeutet, falls eine Taskmenge ihre Fristen tatsächlich einhält bzw. nicht einhält so soll dies der Algorithmus zuverlässig erkennen. Zusätzlich soll der Algorithmus möglichst wenig Testpunkte für diese Entscheidung benötigen und ihn dadurch potentiell schneller zu machen als der Rechenzeitanforderungstest.