

Weitere periodische Zusteller

Echtzeitsysteme - Übungen zur Vorlesung

Fabian Scheler, Peter Ulbrich, Niko Böhm

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
`www4.informatik.uni-erlangen.de`

1. Februar 2010

▶ Deferrable Server Algorithmus

- ▶ sehr einfach, relativ einfach zu implementieren
- ▶ **Problem:** Zeitbedarf eines Deferrable Server
 - ▶ kann in System mit **dynamischen und statischen Prioritäten**
 - ▶ den einer entsprechenden **periodischen Aufgabe übersteigen**
- ▶ Szenario: gegeben sein Deferrable Server $D_s = (p_s, e_s)$



t_{-1} ein $P(J_i)$ -Tätigkeitsintervall beginnt

- ▶ Budget = e_s
- ▶ nächste Wiederherstellung: $t_{-1} + e_s$
- ▶ aperiodische Jobs treffen ein

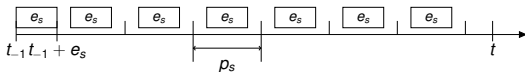
t Termin eines periodischen Jobs J_i

- ▶ bis hier ist D_s immer tätig

$t_{-1} + e_s$ liegt vor den Terminen aller periodischen Jobs

- ▶ die im Intervall $(t_{-1}, t_{-1} + e_s]$ ausführungsbereit sind
- ▶ D_s wird zu jedem Zeitpunkt in $(t_{-1}, t_{-1} + e_s]$ ausgeführt

Zeitbedarf von D_s im Intervall $(t_{-1}, t]$



- ▶ maximaler Zeitbedarf von D_s :

$$e_s + \left\lfloor \frac{t - t_{-1} - e_s}{p_s} \right\rfloor e_s \leq e_s + \frac{e_s}{p_s} (t - t_{-1} - e_s)$$

- ▶ maximaler Zeitbedarf einer periodischen Aufgabe $T_s = (p_s, e_s)$:

$$\frac{e_s}{p_s} (t - t_{-1})$$

- ▶ der Unterschied ist damit:

$$e_s + \frac{e_s}{p_s} (t - t_{-1} - e_s) - \frac{e_s}{p_s} (t - t_{-1}) = \dots = e_s \left(1 - \frac{e_s}{p_s}\right) \geq 0$$

Wie geht man damit um?

- ▶ man behandelt diesen (extra) Zeitbedarf wie **Blockadezeit**
 - ▶ die **jeden anderen Job** beeinträchtigen kann
- ▶ Wie lange wird man maximal *blockiert*?

- ▶ statische Prioritäten: e_s
- ▶ dynamische Prioritäten: $\frac{e_s}{p_s}(p_s - e_s)$

~> **angepasste Tests** für die Überprüfung der Planbarkeit, z.B.

- ▶ Rate Monotonic Approach:

$$U \leq U_{RM/DS}(n) = (n-1) \left[\left(\frac{u_s + 2}{u_s + 1} \right)^{1/(n-1)} - 1 \right]$$

- ▶ $p_s < p_1 < p_2 < \dots < p_n < 2p_s$ und $p_n > p_s + e_s$
- ▶ $u_s = e_s/p_s$
- ▶ Earliest Deadline First:

$$\sum_{k=1}^n \frac{e_k}{\min(D_k, p_k)} + u_s \left(1 + \frac{p_s - e_s}{D_i} \right) \leq 1$$

- ▶ für eine Aufgabe T_i mit relativem Termin D_i

- ▶ benannt nach seinen Erfindern
 - ▶ **S**prunt, **S**ha und **L**ehoczky
- ▶ Idee
 - ▶ Budget bewahren, falls möglich
 - ▶ Budget wird gestückelt
 - ▶ verbrauchte Teile des Budgets so bald wie möglich auffüllen
- ▶ Verhalten
 - ▶ ein SpSL Sporadic Server $D_S = (p_s, e_s)$ verhält sich wie
 - ▶ eine Gruppe periodischer Aufgaben
 - ▶ mit einer gemeinsamen Periode p_s
 - ▶ und einem Gesamtbudget von e_s

Verbrauchs- und Wiederherstellungsregeln

Aufteilung des Budgets

B1 Initial ist Budget = e_s und $t_r = 0$

B2 sobald der SpSL Sporadic Server D_S **suspendiert** wird

- ▶ Budget wird verbraucht (kurz vor der Suspendierung)
- ▶ falls Restbudget vorhanden ist $\leadsto$ Aufteilung des Budget
 1. der **verbrauchte Anteil** $\leadsto$ **nächster** Wiederherstellungszeitpunkt
 2. der **übrige Anteil** $\leadsto$ **letzter** Wiederherstellungszeitpunkt

Verbrauch

C1 D_S verbraucht die Budgetteile in der Reihenfolge ihrer Auffüllung

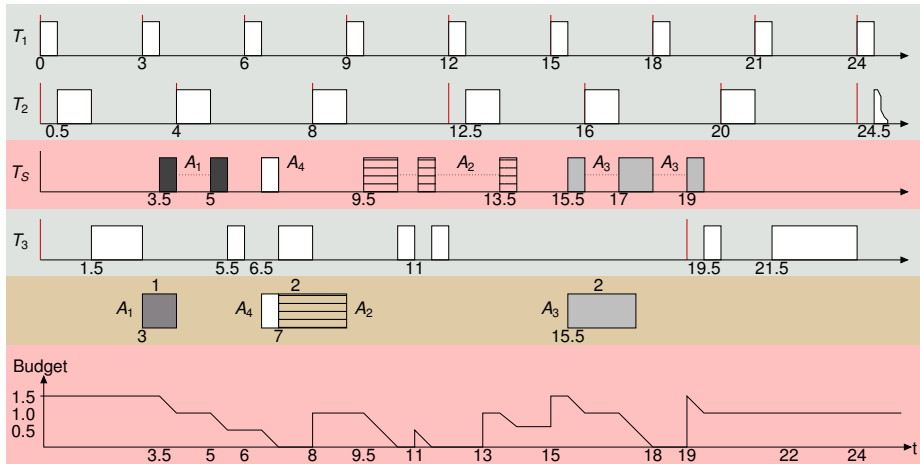
C2 D_S verbraucht sein Budget nur, wenn D_S ausgeführt wird

Wiederherstellung

- ▶ Zeitpunkte: **R2** und **R3** Simple Sporadic Server
 - ▶ nur der jeweilige, verbrauchte Anteil wird aufgefüllt
- ▶ Vereinigung verschiedener Budgetteile
 - ▶ wenn sie zum selben Zeitpunkt aufgefüllt werden

Beispiel: SpSL-Sporadic Server

$T_1 = (3, 0.5)$, $T_2 = (4, 1)$, $T_3 = (19, 4.5)$ und $T_S = (5, 1.5)$; RM



Beispiel: SpSL-Sporadic Server (Forts.)

Budgetverbrauch und -auffüllung

$t_{5.5}$ T_H wird untätig, aber D_S behält sein Budget

$t_{6.5}$ A_4 trifft ein, D_S kann diesen Job behandeln

- ▶ weil noch 0.5 Zeiteinheiten des Budgets übrig sind

t_8 1.0 Zeiteinheiten von e_s werden aufgefüllt

- ▶ nicht die vollen 1.5 Zeiteinheiten

t_{11} 0.5 Zeiteinheiten von e_s werden aufgefüllt

- ▶ D_S beginnt Ausführung zum Zeitpunkt $t_{6.5} = t_f$

- ▶ $t_{begin} = 6.0$ (T_1 startet, T_H wird tätig) $\leadsto$ Auffüllung bei t_{11}

t_{13} analog zu t_8

$t_{13.5}$ A_2 wird fertig behandelt

- ▶ von e_s sind noch 0.5 Zeiteinheiten übrig

$\leadsto$ das Budget von D_S kann in 3 Teile geteilt werden:

- ▶ ein Teil wurde an t_{11} aufgefüllt und an $t_{11.5}$ verbraucht
- ▶ ein Teil wurde an t_{13} aufgefüllt und bis t_{14} verbraucht
- ▶ ein Teil ist an t_{14} noch übrig

t_{14} T wird untätig $\leadsto e_s$ wird an t_{15} aufgefüllt

Generalized Processor Sharing Servers

Idee idealisiertes Round-Robin-Verfahren

- ▶ rundenbasiert
- ▶ Zuteilung einer beliebigen Menge Laufzeit je Runde
- ▶ Laufzeit direkt proportional zum Budget

Effekt zeitliche Isolation mehrerer Server

- ▶ maximale Antwortzeiten unabhängig

Einsatzgebiet in Kombination mit EDF-Scheduling

- ▶ weiche Echtzeitsysteme
- ▶ Quality of Service (QoS)
 - ▶ Multi-Media Systeme
 - ▶ Streaming

Constant Utilization Server

Verhalten

- ▶ sporadische Aufgabe mit konstanter **momentanen Auslastung**
- ▶ Größe: maximale momentane Auslastung $\tilde{u}_s = \max_j(e_{s,j}/p_{s,j})$

Idee

- ▶ Server bekommt genügend Budget
 - ▶ für die Ausführung des ersten Jobs in der Warteschlange
- ▶ Wiederherstellungszeitpunkt ist der Termin d ($\leadsto$ EDF)
- $\leadsto$ Anpassung von d
 - ▶ verhindert eine Überschreitung von $\tilde{u}_s$

Konsum- und Wiederherstellungsregeln

Konsumregeln

- ▶ der Server verbraucht sein Budget, wenn er ausgeführt wird

Wiederherstellungsregeln

R1 Initial: $e_s = 0$ und $d_s = 0$

R2 Zeitpunkt t : Server ist untätig, aperiodisches Ereignis trifft ein

a) $t < d$: nichts

b) $t \geq d$: $d = t + e/\tilde{u}_s$; $e_s = e = \text{WCET des Jobs}$

R3 Zeitpunkt d :

a) Server ist zurück gestellt: $d = t + e/\tilde{u}_s$; $e_s = e$

b) Server untätig ist: nichts

Total Bandwidth Server

- ▶ Variante des Constant Utilization Server
- ▶ Problem:
 - ▶ das System ist untätig
 - ▶ es stehen aperiodische Jobs zur Behandlung an
 - ▶ Server hat kein Budget
- ▶ Lösung:
 - ▶ Kombination mit einem Background Server
 - ▶ Modifikation der Wiederherstellungsregeln
 - ▶ Termine werden wie bisher gesetzt
 - ▶ Server bekommt in R2 immer Budget

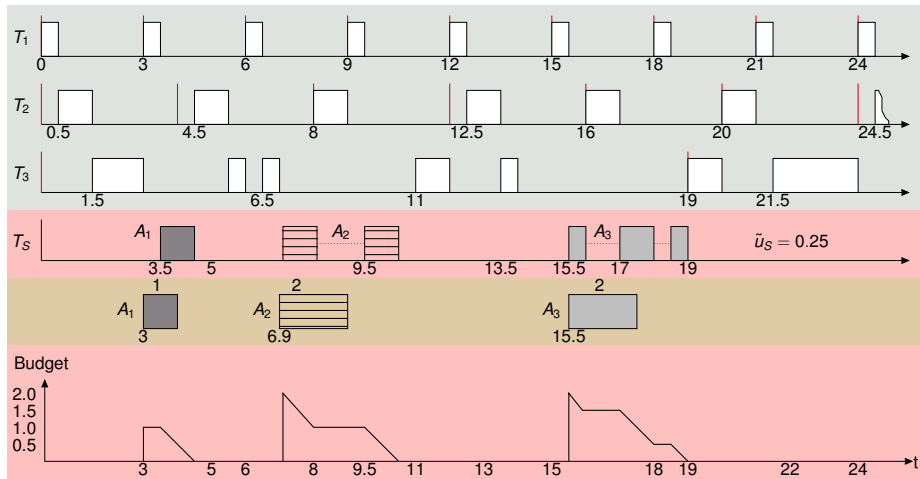
Wiederherstellungsregeln

R2 Zeitpunkt t : aperiodisches Ereignis mit WCET e trifft ein

- ▶ Server ist untätig
- ▶ $d = \max(d, t) + e/\tilde{u}_s$; $e_s = e$

Beispiel: Constant Utilization Server

$T_1 = (3, 0.5)$, $T_2 = (4, 1)$, $T_3 = (19, 4.5)$, EDF



Beispiel: Constant Utilization Server (Forts.)

Budgetverbrauch und -auffüllung

t_0 initial: $e_s = 0, d_s = 0$

t_3 A_1 trifft ein, der Server ist untätig $\leadsto$ R2 b)

- ▶ Budget: $e_s = e_1 = 1$

- ▶ Wiederherstellung: $d_s = t + e_1/\tilde{u}_s = 3 + 1/0.25 = 7$

$t_{6.9}$ A_2 trifft ein, der Server ist untätig

- ▶ $6.9 = t < d = 7 \leadsto$ R2 a) nichts

t_7 Wiederherstellungszeitpunkt, Server ist zurückgestellt $\leadsto$ R3 a)

- ▶ Budget: $e_s = e_2 = 2.0$

- ▶ Wiederherstellung: $d_s = t + e_2/\tilde{u}_s = 7 + 2/0.25 = 15$

t_{15} Wiederherstellungszeitpunkt, Server ist untätig $\leadsto$ R3 b)

$T_{15.5}$ A_3 trifft ein, Server ist untätig $\leadsto$ R2 b)

- ▶ Budget: $e_s = e_3 = 2$

- ▶ Wiederherstellung: $d_s = t + e_3/\tilde{u}_s = 15.5 + 2/0.25 = 23.5$