8. Einplanung einzelner Jobs durch nichtperiodische Tasks in prioritätsbasierten Systemen

8.1. Modellannahmen und Vorgehen

- **Voraussetzungen**

– ***Jobs / nichtperiodische Tasks***

unterbrechbar, voneinander unabhängig

sporadisch: harte Deadline

aperiodisch: keine (oder weiche) Deadline

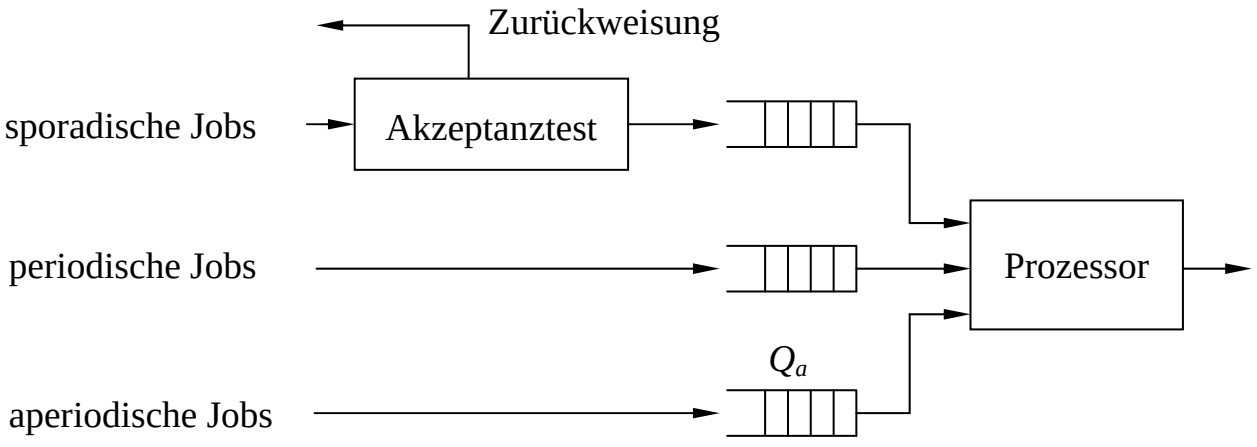
Abstand sowie Ausführungszeit und Deadline beliebig
(Werte zur Ankunftszeit bekannt)

- **Vorgehen**

periodisch: Zulassung $\rightarrow$ garantierte Deadline-Einhaltung

sporadisch: Akzeptanztest

aperiodisch: Beendigung „baldmöglichst“



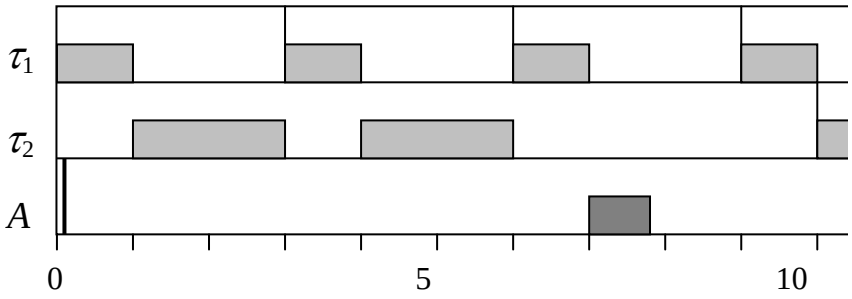
- **Einfache Ansätze für aperiodische Jobs**

  Annahme: keine sporadischen Jobs im System

– ***Ausführung im Hintergrund***

  dann, wenn keine periodischen Tasks bearbeitet werden

  Beispiel 8.1. a) $\tau_1 = (3;\ 1)$, $\tau_2 = (10;\ 4)$ $A = (0{,}1;\ 0{,}8)$.



  einfache Implementation, aber schlechtes Antwortzeitverhalten

– ***Ausführung durch Unterbrechung***

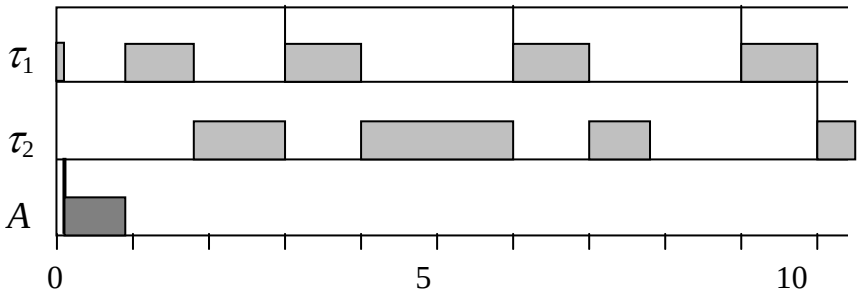
  sofortige Ausführung bei Ankunft

  minimale Antwortzeit, aber Gefährdung der Jobs mit harter Deadline

– ***Ausführung durch Aufschieben periodischer Jobs – „slack stealing“***

  durch „slack stealer“ mit höchster Priorität, sofern Zeit frei

  Beispiel 8.1. b)



  effizient, aber aufwendige Analyse der verfügbaren Zeit (slack)

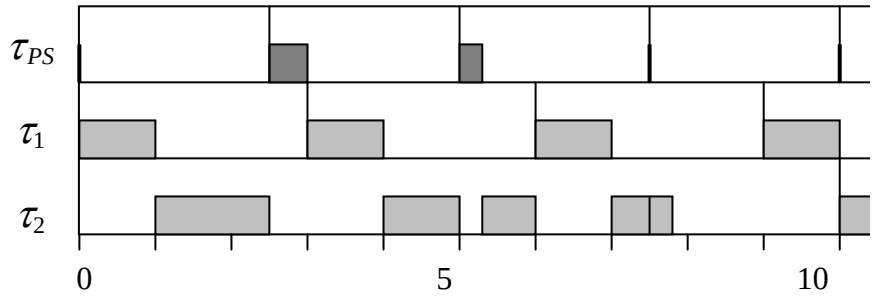
– ***Periodische Ausführung mittels „Polling Server“***

PS polling server: periodische Task (t_s; e_s).

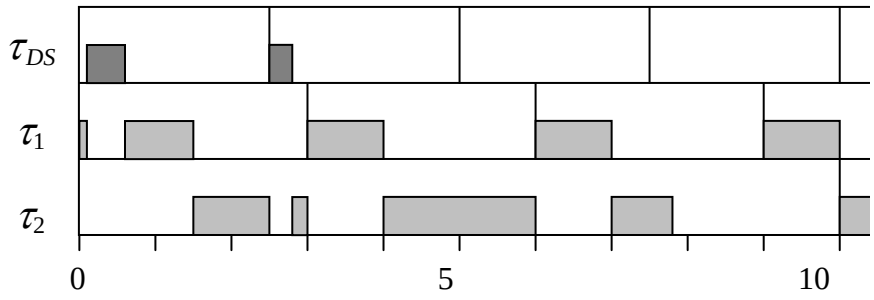
Wenn in neuer Periode Q_a bei Ausführungsbeginn leer ist, suspendiert sich PS für restliche Periode selbst. Andernfalls Abarbeitung der Jobs aus Q_a längstens für die Dauer e_s.

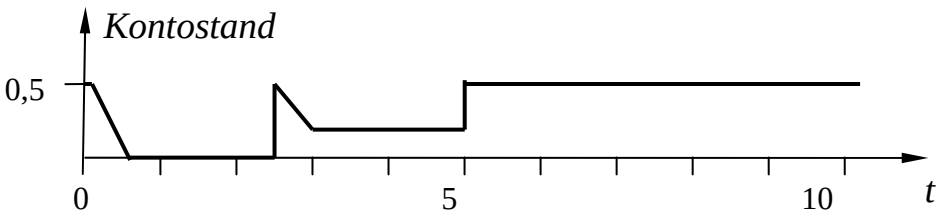
Beispiel 8.1. c) $\tau_1 = (3;\ 1)$, $\tau_2 = (10;\ 4)$, $A = (0{,}1;\ 0{,}8)$.

$\tau_{PS} = (2{,}5;\ 0{,}5)$



Beispiel 8.1. d) Verzögerte Ausführung





- **Begriffe**

– ***Periodischer Server*** (t_s; e_s):
Task mit einem *Konto K* zur periodischen Ausführung aperiodischer Jobs; definiert durch Regeln für ***Verbrauch*** und ***Auffüllung*** des Kontos.

Scheduling als periodische Task; dabei verbraucht Server seinen Kredit, bis Konto ***ausgeschöpft*** ist.

e_s: ***Kredit*** (execution budget) gespeichert auf einem *Konto*

t_s: Abstand der Konto-Auffüllung (replenishment)

Konto wird zum Auffüllungszeitpunkt **auf** e_s aufgefüllt.

$u_s = \frac{e_s}{t_s}$: ***Größe*** (Umfang, size) des Servers

Server ist ***im Rückstand, beschäftigt*** (backlogged), wenn $Q_a \neq \varnothing$

im Leerlauf (idle), wenn $Q_a = \varnothing$

(ausführungs-)bereit (eligible for execution)
bei positivem Kontostand und wenn er im Rückstand ist.

– ***Bandbreiten-erhaltender Server*** (bandwidth-preserving server):
modifizierter Polling-Server mit besserer Leistung.

Formen:

* deferrable servers
* sporadic servers
* constant utilization / total bandwidth servers
* weighted fair-queueing servers.

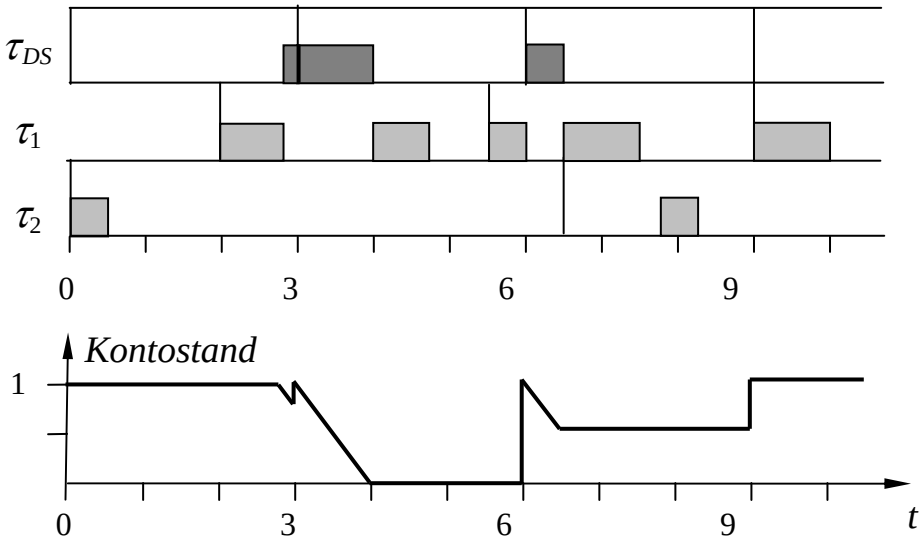
8.2. Verzögerte Ausführung – „Deferrable Server“

- **Regeln**

  Konto wird zu jedem Periodenbeginn $k \cdot t_s$ auf e_s aufgefüllt;
  es wird verbraucht, wenn Server arbeitet.

- **Beispiel 8.2.**

  $\tau_1 = (2;\ 3{,}5;\ 1{,}5)$ $\tau_2 = (6{,}5;\ 0{,}5)$ $\tau_{DS} = (3;\ 1)$ $A = (2{,}8;\ 1{,}7)$



- **Eigenschaften**

– Hinreichende Bedingung für kritischen Zeitpunkt komplizierter.

– Grenzauslastung für Einplanbarkeit mit statischen Prioritäten (schedulable utilization) nur bekannt für RMS und dann nur, falls Deadlines und Periodenenden übereinstimmen und wenn gilt:

$$t_s < t_1 < \ldots < t_n < 2t_s \quad \text{und} \quad t_n > t_s + e_s.$$

– Analyse des Zeitbedarfs: $w(t)$ ergänzen um $\left(1 + \left\lceil \frac{t - e_s}{t_s} \right\rceil\right) \cdot e_s$

- **Bewertung**

  einfach, aber längere Antwortzeit für niedriger priorisierte Jobs möglich.

  Verbesserung durch „Sporadic Server“ (verbraucht nie mehr Zeit als eine entsprechende periodische Task).

8.3. Sporadische Server – „Sporadic Servers“

- **Voraussetzungen, Begriffe und Bezeichnungen**

  Scheduling mittels fester Prioritäten

  1 Server S mit t_s, e_s und Priorität pr_s

  $T_H = \{\tau \mid pr_\tau \succ pr_s\}$

– ***Arbeitsintervall (busy interval)***

  von S: jedes maximale Intervall mit $Q_a \neq \varnothing$

  bzgl. T_H: jedes maximale Intervall, in dem Tasks aus T_H aktiv sind

– ***Zeiten***

t_r	letzter Zeitpunkt einer Auffüllung des Kontos
$t_f > t_r$	Aktivierungszeitpunkt von S
t_e	letzter *effektiver* Zeitpunkt einer Kontoauffüllung
t	aktuelle Zeit
BEGIN	Beginn eines Arbeitsintervalls bzgl. T_H
END	Ende eines Arbeitsintervalls bzgl. T_H, falls vor t (sonst ∞)

- **Varianten**

  Hintergrund-Arbeit (Sporadic/Background Server)

  kumulative Auffüllung

  „originale“ sporadische Server (SPRUNT / SHA / LEHOCZKY, 1989)

  Prioritätstausch

  Systeme mit dynamischen Prioritäten

- **Einfacher periodischer Server (Simple periodic server)**

– ***Verbrauch***

Konto wird verbraucht in den Fällen:

(C1) S ist aktiv.

(C2) S hat seit t_r (irgendwann) gearbeitet und END < t.

– ***Auffüllung***

(R1) Anfangs und bei jeder Auffüllung: $K := e_s$, $t_r := t$.

(R2) S wird aktiv ($t = t_f$):

$$t_e := \begin{cases} t_f & \text{falls END} < t_f \\ \max(t_r, \text{BEGIN}) & \text{falls END} = t_f \end{cases} \qquad t_r := t_e + t_s$$

(R3) Nächste Auffüllung erfolgt zum nächsten Auffüllungszeitpunkt (aktueller Wert von t_r) außer

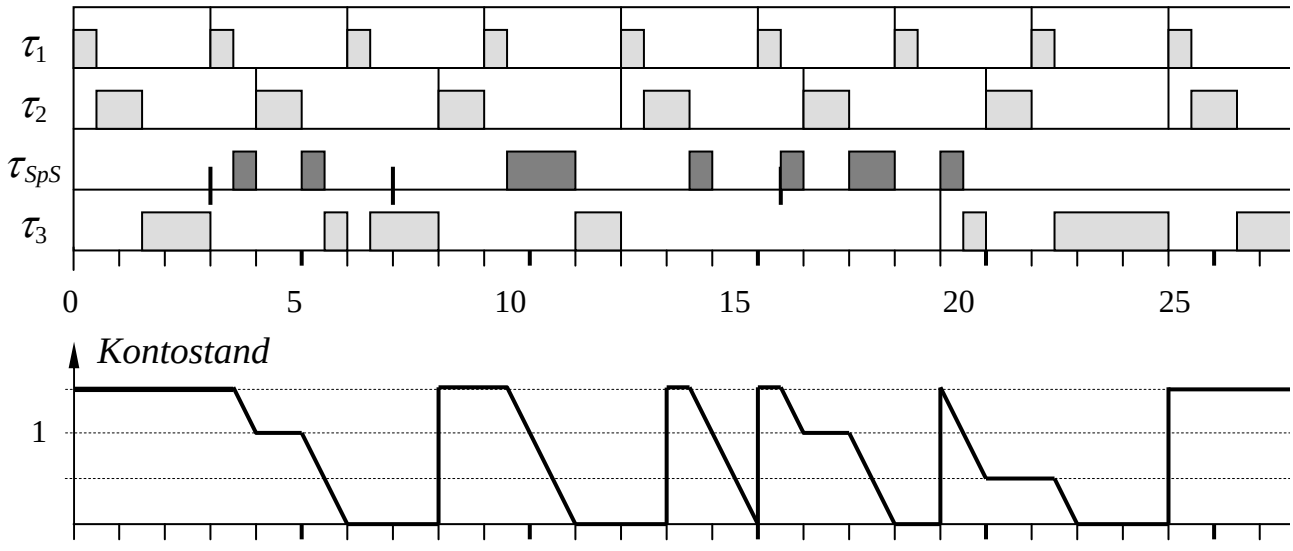
(a) $t_e + t_s < t_f$: Auffüllung, wenn Konto erschöpft

(b) System wird leer vor $t_e + t_s$, wieder beschäftigt bei t_b:
Auffüllung bei $\min(t_e + t_s, t_b)$.

- **Beispiel 8.3.**

$\tau_1 = (3;\ 0{,}5)$ $\tau_2 = (4;\ 1)$ $\tau_3 = (19;\ 4{,}5)$ $\tau_{SpS} = (5;\ 1{,}5)$

$A = (3;\ 1)$ $B = (7;\ 2)$ C = (15,5; 2)



8.4. Einplanung mittels dynamischer Prioritäten

System-Scheduling nach EDF.

- **„Constant Utilization Server“ (CU-Server)**

– ***Bezeichnungen und Voraussetzungen***

u_s Größe (definiert den Server)

e_s Konto / Kredit des Servers;
Server ist ausführungsbereit bei positivem Kontostand

d Deadline

e Ausführungszeit des „1.“ Jobs von Q_a;
ein Job wird erst bei seiner Beendigung aus Q_a gelöscht.

– ***Regeln***

Verbrauch nur während der Ausführung.

Auffüllung:

(1) Initialisierung: $e_s = 0$, $d = 0$.

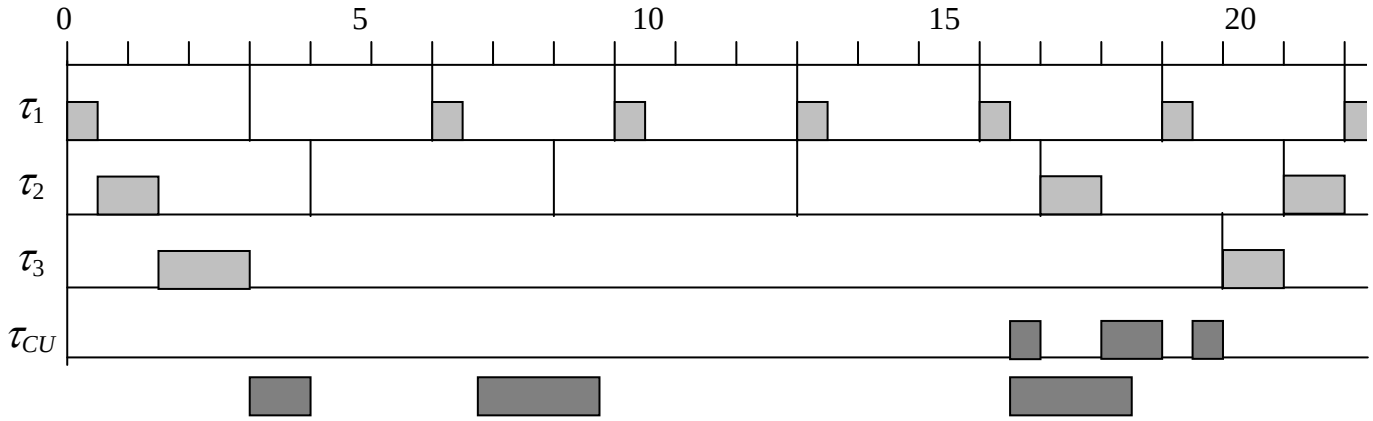
(2) Bei Ankunft eines Jobs mit Ausführungszeit e zur Zeit t,
falls $Q_a = \varnothing$ und $t \geq d$:
$d := t + e/u_s$, $e_s := e$ (andernfalls nichts).

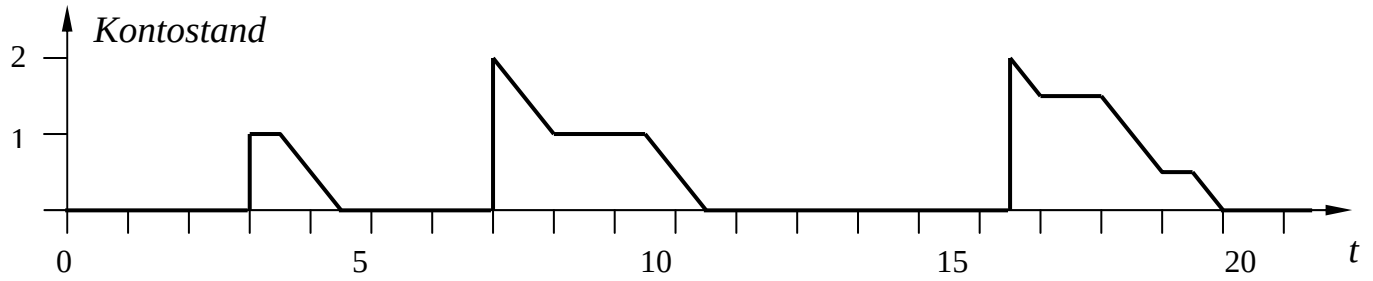
(3) Ist der Server zu seinem Deadline-Zeitpunkt d beschäftigt:
$d := t + e/u_s$, $e_s := e$ (andernfalls nichts).

– ***Beispiel 8.4. a)***

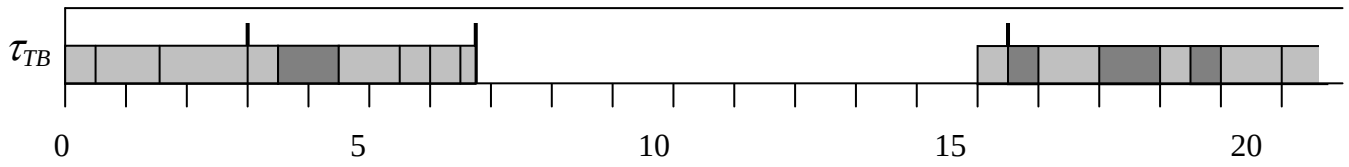
$\tau_1 = (3;\ 0{,}5)$ $\tau_2 = (4;\ 1)$ $\tau_3 = (19;\ 4{,}5)$

$A = (3;\ 1)$ $B = (6{,}75;\ 2)$ $C = (15{,}5;\ 2)$ $u_s = 0{,}25$





– ***Beispiel 8.4. b)***



– ***Bewertung***

besseres Antwortzeitverhalten als sporadischer Server

günstigere Einplanbarkeit

Problem: Verschwendung von Leerlaufzeiten des Prozessors

- **„Total Bandwidth Server“ (TB-Server)**

  Voraussetzungen und Verbrauchs-Regel wie bisher.

– ***Auffüllung:***

(1) Initialisierung: $e_s = 0$, $d = 0$.

(2) Bei Ankunft eines Jobs mit Ausführungszeit e zur Zeit t:

$d := \max(d, t) + e/u_s$, $e_s := e$ im Fall $Q_a = \varnothing$ (sonst nichts).

(3) Bleibt der Server bei Beendigung eines aperiodischen Jobs beschäftigt:

$d := d + e/u_s$, $e_s := e$ (sonst nichts).

– ***Beispiel 8.4. b)***

– ***Bewertung***

Gleiche Deadlines wie CU-Server, aber frühere Auffüllung und damit geringere Antwortzeiten.

Einplanbarkeit: Gegeben sei ein System mit

* unabhängigen, periodischen Tasks, Gesamt-Dichte D
* (einem oder mehreren) CU- oder TB-Servern, Gesamt-Größe U
* unterbrechbaren Tasks bzw. Jobs
* Scheduling nach EDF.

In einem solchen System halten alle Tasks und alle Server stets ihre Deadlines ein, wenn

$$D + U \leq 1.$$

Probleme: Fairneß und Aushungern.