

3. Echtzeit-Scheduling – Grundlagen

3.1. Grundbegriffe, Klassifikation und Bewertung

- **Grundbegriffe**

- **Job**

Planungseinheit für Scheduling

e Ausführungszeit, Bearbeitungszeit (execution time)

$wcet$ maximale Ausführungszeit

r Freigabezeit, Bereitzeit (release time)

d Zeitschranke, Frist (deadline)

- **Task**

Menge „zusammengehörender“ Jobs

speziell: *Jobnetz* oder *periodische Task*

- **Deadline**

hart/weich

„time/utility function“



deterministisch/probabilistisch

- **Ressource (Betriebsmittel BM)**

aktiv (Prozessor) – passiv

exklusiv, i.d.R. entziehbar (Kosten!)

- **Schedule (Ablaufplan)**
zeitliche Zuordnung von Jobs zu Prozessoren
gültig (valid): Zuordnung verletzt keine der gegebenen Bedingungen
ausführbar (feasible): alle Zeitschranken werden eingehalten
- **Scheduling**
 - * *Einplanung*: Vorgehen (Algorithmus), das bei gegebener Taskbeschreibung für jede Taskmenge einen Ablaufplan bestimmt
 - * *Prozessor-Zuordnung*: Auswahl eines Jobs durch Scheduler
- **Einplanbarkeit**
Taskmenge ist *einplanbar (schedulable)* bei einem Scheduling-Algorithmus, wenn der Algorithmus einen ausführbaren Ablaufplan erzeugt
- **Admission (Zulassung)**
Verfahren, das die Einplanbarkeit einer Taskmenge entscheidet
- **Optimalität (bzgl. Einplanbarkeit)**
eines Scheduling-Verfahrens in einer Klasse \mathcal{C} von Verfahren:
erzeugt für jede Taskmenge T einen ausführbaren Ablaufplan, sofern überhaupt T mit irgendeinem Verfahren aus \mathcal{C} einplanbar ist.
- **Klassifikation**
Scheduling für Jobnetze | periodische Tasks
Scheduling für periodische Tasks:
 zeitgesteuert (time driven)
 ereignisgesteuert (event driven)
 statische | dynamische Prioritäten
nichtperiodische Tasks
Nutzung weiterer, nicht entziehbarer Betriebsmittel
Entziehbarkeit
Ein-/Mehrprozessorsysteme

3.2. Schedulingstrategien für Jobnetze

- **Beschreibung**

- **Job**

$J \ e \ (r \ d]$

Jobname – Ausführungszeit – Freigabezeit – Zeitschranke (Deadline)

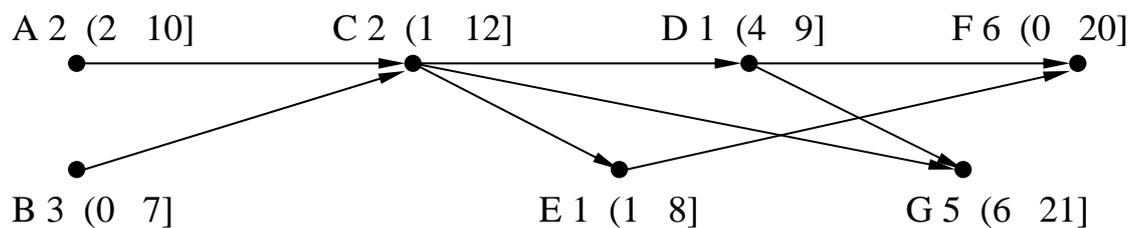
- **Jobnetz**

Jobabhängigkeiten: Präzedenzrelation R

$(A, B) \in R$: B kann erst beginnen, wenn A fertig

Darstellung häufig als Vorgangsknotennetz

Beispiel.



- **Effektive Freigabezeit und Deadline**

effektive Freigabezeit eines Jobs:

Maximum der Freigabezeit des Jobs und der effektiven Freigabezeiten seiner Vorgänger

effektive Deadline analog (Minima der Nachfolger)

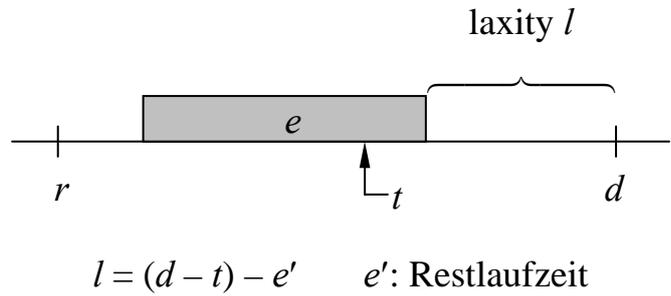
• **Grundlegende Strategien**

EDF Earliest Deadline First

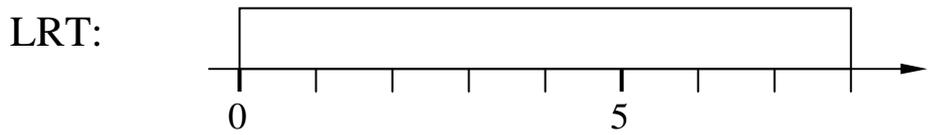
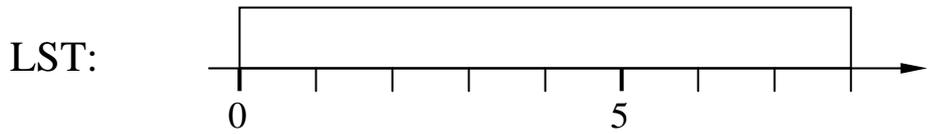
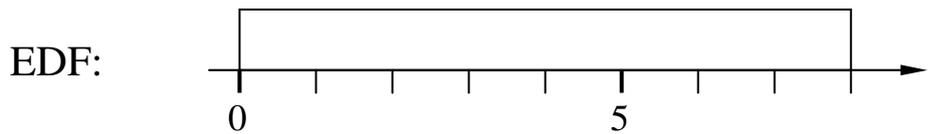
LST Least Slack Time

MLF Minimum Laxity First

LRT Latest Release Time



– **Beispiel.** A 3 (0 5] ● → ● B 2 (1 8]
 C 2 (2 7] ●



– **Bedeutung**

EDF, MLF = LST und LRT sind optimal (bzgl. Einplanbarkeit) in 1-Prozessor-Systemen bei unterbrechbaren Jobs ohne gemeinsame BM.

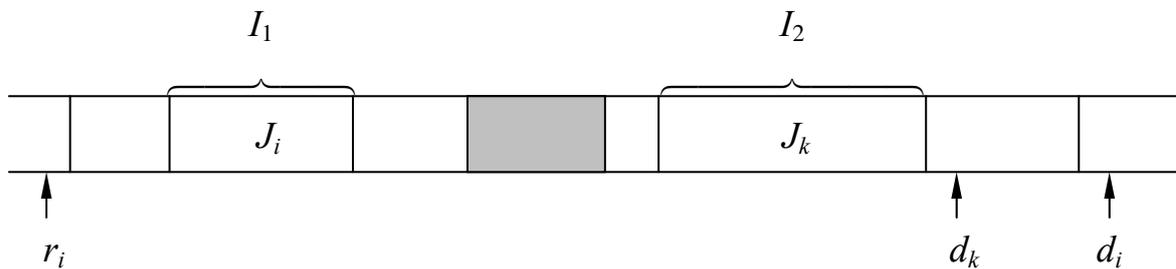
3.3. EDF

- **Optimalität**

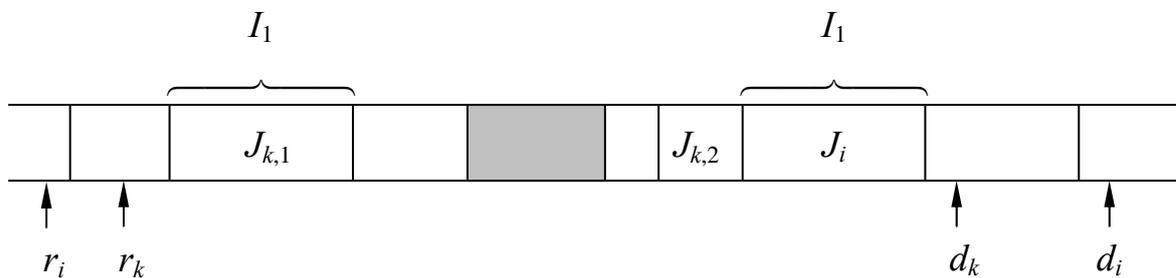
EDF ist optimal in 1-Prozessor-Systemen mit Entzug und ohne gemeinsame BM-Nutzung, d.h.: Wenn es auf irgendeine Weise möglich ist, eine Jobmenge J einzuplanen, so ist dies auch gemäß EDF möglich.

Beweis.

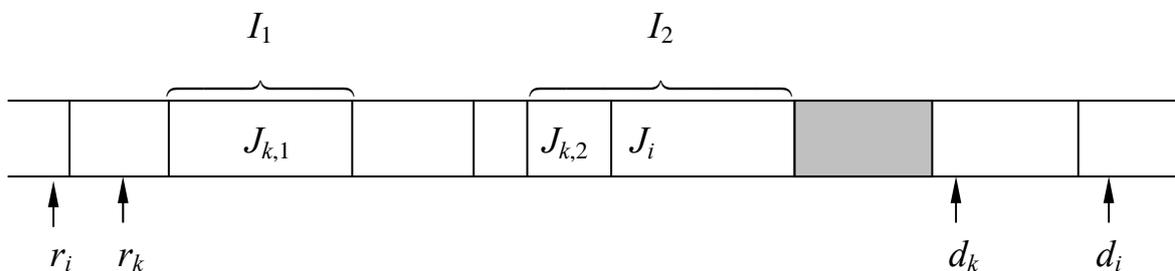
Folgender Ablaufplan sei ausführbar:



Dann ist auch der folgende Ablaufplan ausführbar:



Damit ergibt sich als endgültiger EDF-Plan:



- **Admission-Kriterium für periodische Tasks**

- *Voraussetzungen*

$T = \{ \tau_1, \dots, \tau_n \}$ n unabhängige periodische Tasks, unterbrechbar,
1 Prozessor

$\tau_i = (p_i, e_i)$ p_i Periodenlänge = Deadline, e_i Ausführungszeit

- *Admission-Kriterium für EDF*

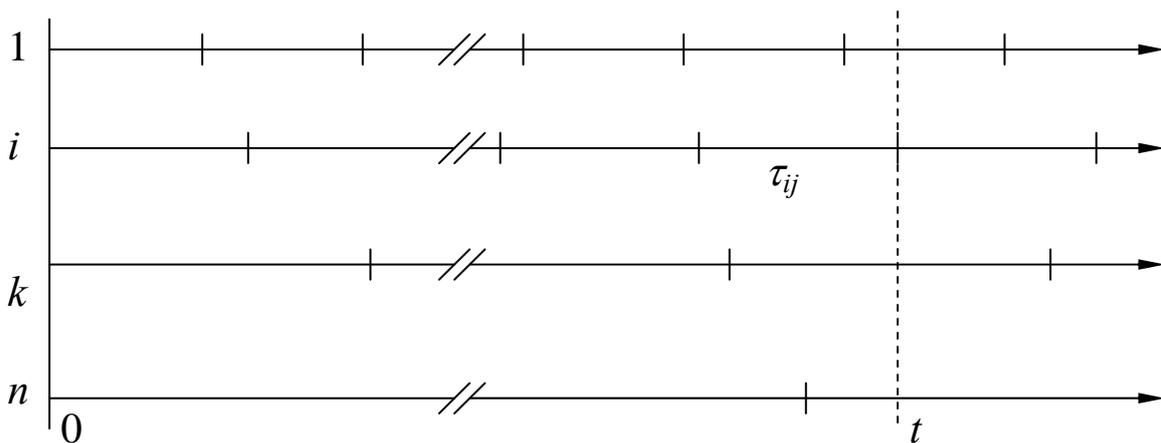
T ist einplanbar genau dann, wenn

$$u = \sum_{i=1}^n \frac{e_i}{p_i} \leq 1$$

- *Beweis-Skizze*

A) $u > 1 \Rightarrow T$ nicht einplanbar (trivial).

B) T lasse sich nicht einplanen $\Rightarrow \exists \tau_{ij}$: τ_{ij} überschreitet seine Deadline.



Zum Zeitpunkt t ist keiner der aktuellen Jobs bisher ausgeführt worden.

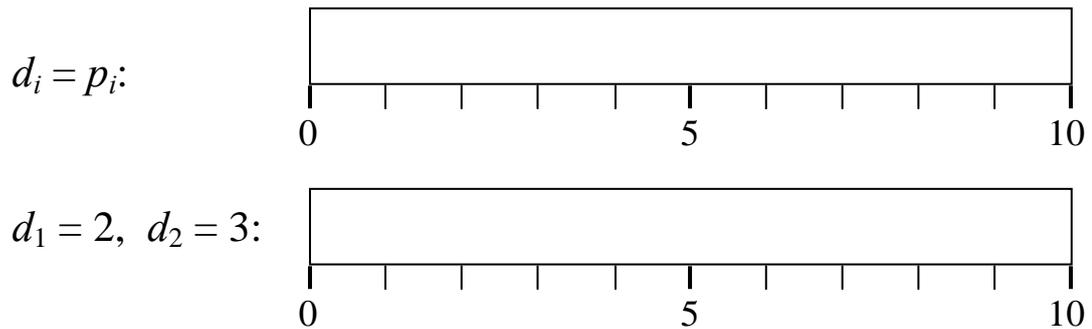
Der Gesamtbedarf an Prozessorzeit bis zu diesem Zeitpunkt übersteigt t :

– **Allgemeiner**

* $\forall i: d_i \geq p_i: T$ einplanbar $\Leftrightarrow u \leq 1$.

* $\exists i: d_i < p_i: T$ muß nicht einplanbar sein auch bei $u \leq 1$.

Beispiel. $\tau_1(2; 1), \tau_2(5; 2,5)$



* **Dichte (density)**

$$\delta_i = \frac{e_i}{\min(d_i, p_i)} \quad \Delta = \sum_{i=1}^n \delta_i$$

Offenbar gilt: $\exists i: d_i < p_i \Rightarrow \Delta > u$.

Es gilt: T ist einplanbar, falls $\Delta \leq 1$.

• Probleme

– Nicht-Optimalität

EDF ist nicht optimal bei

- (1) gemeinsam genutzten Betriebsmitteln
- (2) nicht unterbrechbaren Jobs
- (3) in Mehrprozessorsystemen.

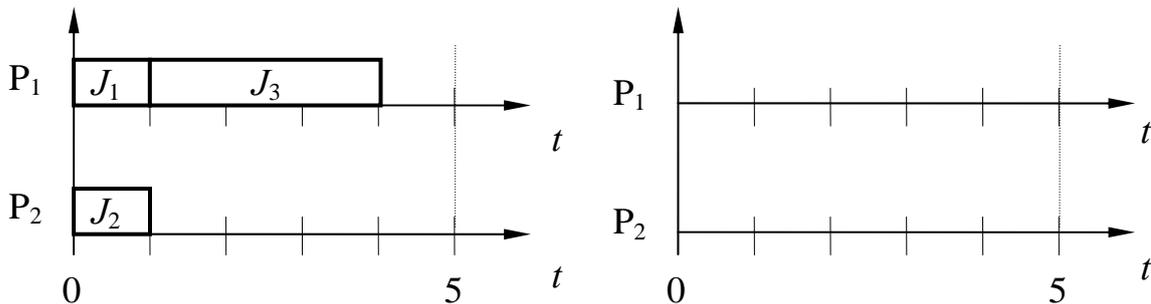
Beispiele.

(1) A [5 R(2 3)] (0 9) B [3 R(0 1)] (3 7)

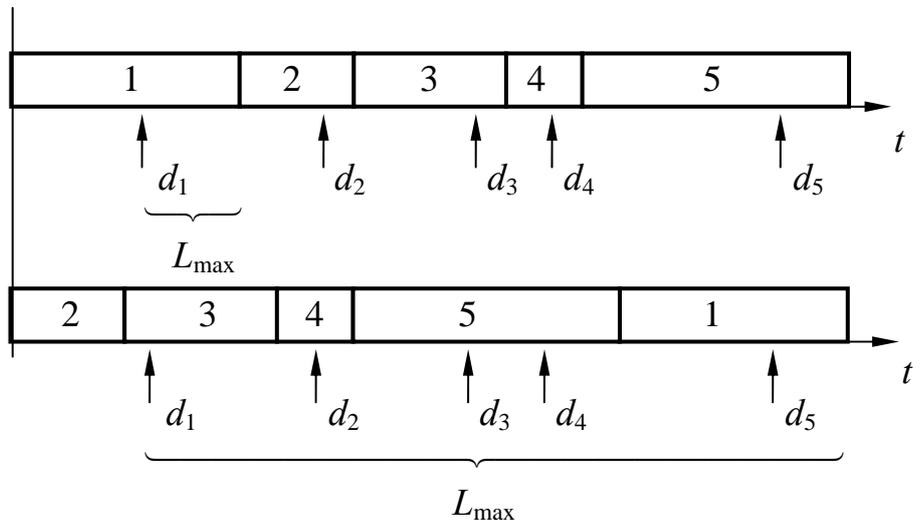


(2) A 3 (0 4] B 1 (1 3]

(3) Prozessoren P₁, P₂; 3 Jobs (e_i, d_i): (1; 1) (1; 2) (3; 3,5); r_i = 0

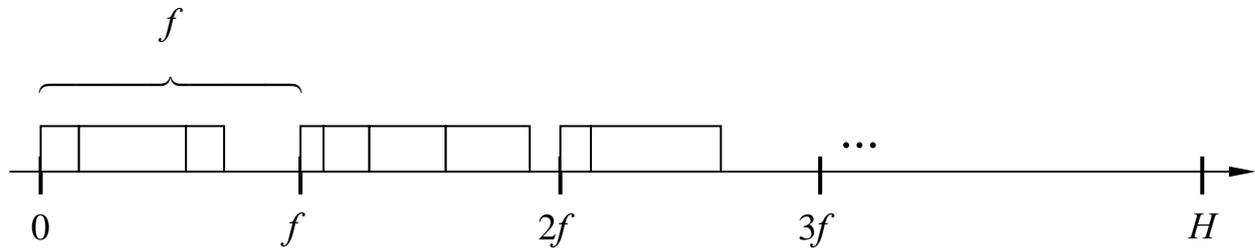


– „Domino-Effekt“:



- **Rahmenbasiertes zyklisches Scheduling**

Scheduling-Entscheidungen sollen periodisch erfolgen auf der Basis von Rahmen (frames) der Größe f .



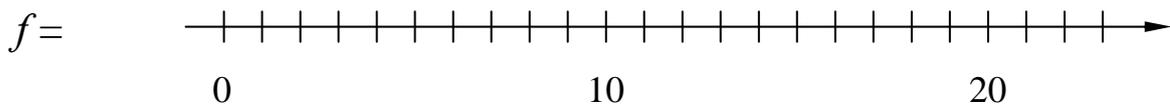
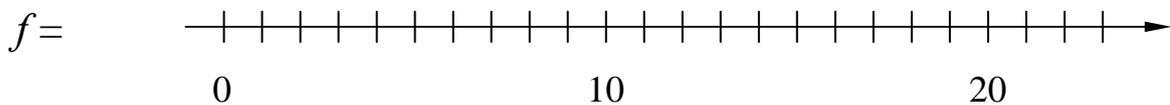
- **Bedingungen für f**

(1) $f \geq e_i \quad \forall i = 1, \dots, n$

(2) $\exists i \in \{1, \dots, n\}: f \mid p_i$

(3) $2f - \text{ggT}(p_i, f) \leq d_i \quad \forall i = 1, \dots, n.$

$T_1(6, 3), \quad T_2(7, 3) \rightarrow f =$



- **Jobscheiben (job slices)**

$T_1(8, 3), \quad T_2(5, 2) \rightarrow$ kleinster Rahmen gemäß (1)/(2): $f = 4.$

Aber: $2 \cdot 4 - \text{ggT}(5, 4) = 7 \leq 5 ?$ ●*

Jobscheiben für T_1 : $(8, 2), \quad (8, 1).$

- **Sporadische Tasks:** Einplanung mittels „slack stealing“

3.5. Prioritätsbasiertes Scheduling für periodische Tasks

- **Voraussetzungen**

wie in 3.4.

- **Wichtige Strategien und Ergebnisse**

- *Statische Prioritäten*

RMS ist optimal bei $d_i = p_i$ „rate monotonic“

DMS ist optimal bei $d_i < p_i$ „deadline monotonic“

Admission-Kriterium für RMS: $u = \sum_{i=1}^n \frac{e_i}{p_i}$;

$u \leq n \cdot (\sqrt[n]{2} - 1)$ hinreichend, $u \leq 1$ bei harmonischen Tasks.

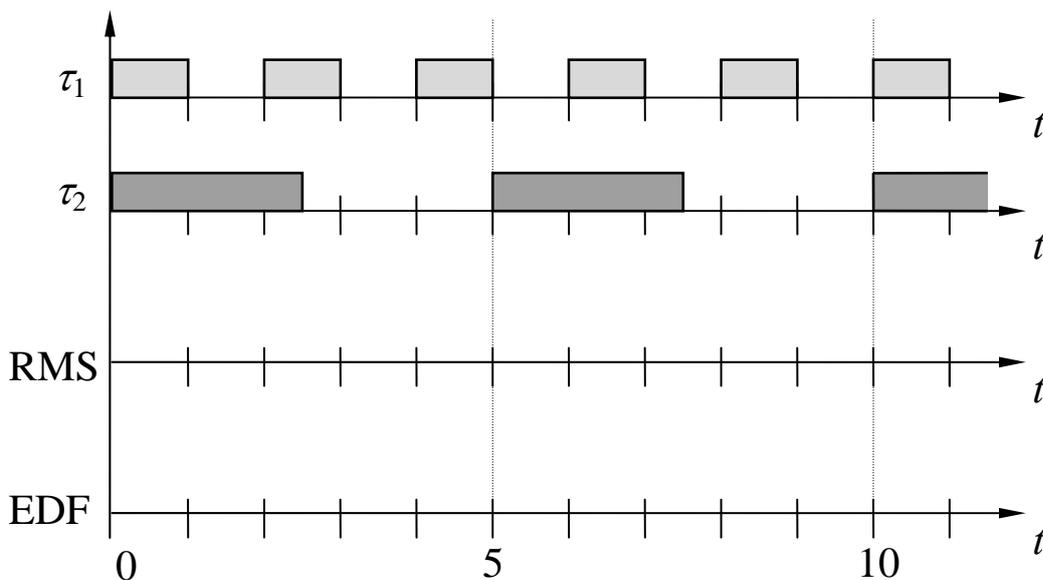
Exakt: LEHOCZKY-Kriterium bzw. Analyse des Zeitbedarfs.

Ausgangspunkt: Kritischer Zeitpunkt einer Task.

- *Dynamische Prioritäten*

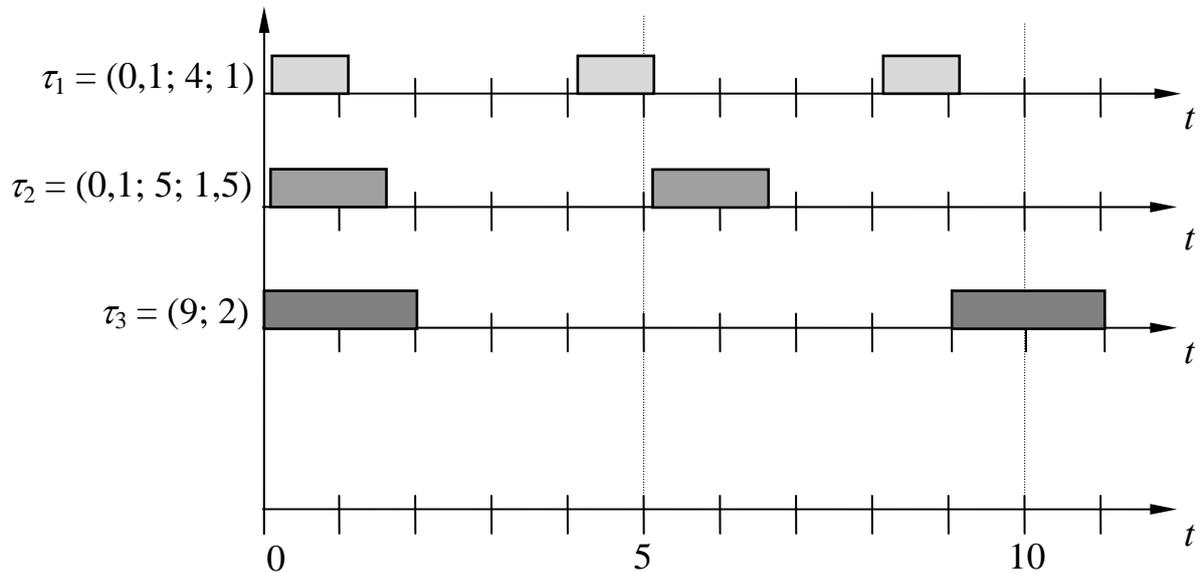
EDF ist optimal. Kriterium: $u \leq 1$.

– *Beispiel.* $\tau_1: p_1 = 2 \quad e_1 = 1 \quad \tau_2: p_2 = 5 \quad e_2 = 2,5. \quad u =$



3.6. Blockierzeiten

- Nicht-Unterbrechbarkeit (Nonpreemptibility)



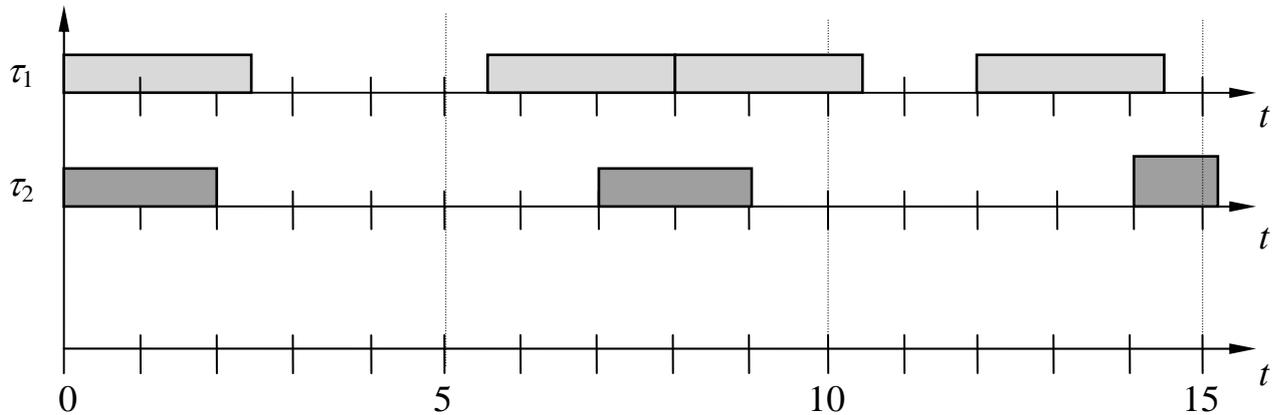
$b_i(nu)$: Blockierzeit von Task τ_i aufgrund von Nicht-Unterbrechbarkeit

$\theta_i(nu)$: längste Dauer der nichtunterbrechbaren Abschnitte der Jobs von τ_i

$$b_i(nu) \leq \max_{i+1 \leq k \leq n} \theta_k(nu)$$

- **Selbstunterbrechung (Self-Suspension)**

$$\tau_1 = (4; 2,5 [1,5]) \quad \tau_2 = (7; 2)$$



$b_i(su)$: Blockierzeit von Task τ_i aufgrund von Selbstunterbrechbarkeit

$\theta_i(su)$: längste Selbstunterbrechungszeit eines Jobs von τ_i

$$b_i(su) \leq \theta_i(su) + \sum_{k=1}^{i-1} \min(e_k, \theta_k(su))$$

k_i : maximale Anzahl von Selbstunterbrechungen eines Jobs von τ_i

$$b_i = b_i(su) + (k_i + 1) \cdot b_i(nu)$$

- **Kontextwechsel (Context Switches)**

$\theta(kw)$: Dauer eines Kontextwechsels

e_i ersetzen durch $e_i + 2 \cdot \theta(kw)$ ohne Selbstunterbrechung

e_i ersetzen durch $e_i + 2(k_i + 1) \cdot \theta(kw)$ mit Selbstunterbrechung

3.7. Prioritätszuordnung

- **Problemstellung**

Taskprioritäten $1, \dots, n$ sind auf Systemprioritäten $1, \dots, m$ aufzuteilen mit $m < n$.

- **Vorgehen**

a_1, \dots, a_m sei eine Aufteilung (d.h. $1 \leq a_1 < \dots < a_m = n$). Möglichkeiten:

- gleichmäßig (linear)

- in gleichmäßigem Verhältnis $\frac{a_{i-1} + 1}{a_i}$, $i = 2, \dots, m$.

- **Hintergrund**

- **Grenzauslastung für gesicherte Einplanbarkeit (schedulable utilization)**

Kleinste Auslastung \tilde{u} , so daß jede Taskmenge mit $u \leq \tilde{u}$ einplanbar ist.

RMS: $\tilde{u} = n(\sqrt[n]{2} - 1)$, EDF: $\tilde{u} = 1$.

Wird bei unzureichenden Systemprioritäten eingeschränkt.

- **RMS**

$$g = \min_{i=2, \dots, m} \frac{a_{i-1} + 1}{a_i}, \quad n \text{ groß:}$$

$$\tilde{u} = \begin{cases} \ln(2g) + 1 - g & g > 1/2 \\ g & g \leq 1/2 \end{cases} \quad \text{LEHOCZKY/SHA, 1986}$$

Es gilt stets $\tilde{u} \leq \ln 2$.

Beispiel. $n = 100\,000$, $m = 256$: $\tilde{u} = 0,9986 \cdot \ln 2$;

Aufteilung: $1 - 2_{16} - 3_{25} - \dots - 91.398 - 95.602 - 100.000$.