

## 4. Ratenmonotones Scheduling – Rate-Monotonic Scheduling (LIU/LAYLAND 1973)

### 4.1. Taskbeschreibung

- **Task**

Planungseinheit. Periodische Folge von Jobs.  $T = \{\tau_1, \dots, \tau_n\}$

- **Taskparameter**

Anforderungszeit, Bereitzeit (release time)  $r_i$

Bearbeitungs-, Ausführungszeit (computation time)  $c_i$

Zeitschranke, Frist (deadline, due date)  $d_i$

Periodendauer  $t_i$

Phasenverschiebung (phase)  $\varphi_i$

Priorität  $p_i$

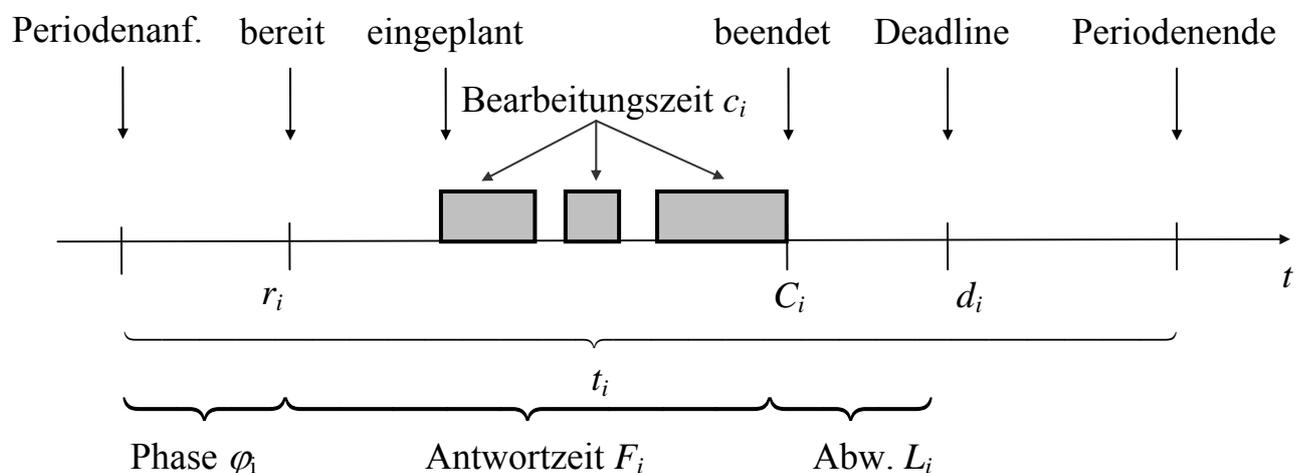
- **Bewertungsgrößen und Bewertungsmaße**

Beendigungszeitpunkt (completion time)  $C_i$   $\max(C_i) \rightarrow \text{Min!}$

Antwortzeit (flow time)  $F_i = C_i - r_i$   $\bar{F}_i \rightarrow \text{Min!}$

Deadline-Abweichung (lateness)  $L_i = C_i - d_i$   $L_i \leq 0 \forall i$

Deadline-Überschreitung (tardiness)  $T_i = \max(L_i, 0)$   $= 0 \forall \tau_i \in T$



## 4.2. Definitionen und Eigenschaften

- **Modellannahmen**

- (1) Anforderungen an Tasks sind periodisch mit konstanter Periodendauer  $t_i$ , d.h. mit konstanter Anforderungsrate  $a_i = t_i^{-1}$ .
- (2)  $\tau_i$  ist bereit zu Periodenbeginn, Ausführungszeit  $c_i$  ist konstant (und höchstens gleich  $t_i$ ).
- (3) Die Deadline einer Task ist gleich deren Periodenende.
- (4) Die Tasks sind voneinander unabhängig.
- (5) Das Scheduling erfolgt auf der Basis fester Prioritäten. Eine Task höherer Priorität verdrängt eine Task niedrigerer Priorität sofort, Overhead wird vernachlässigt.

- **Taskbeschreibung**

$$\tau_i: (t_i, c_i, p_i) \quad i = 1, \dots, n$$

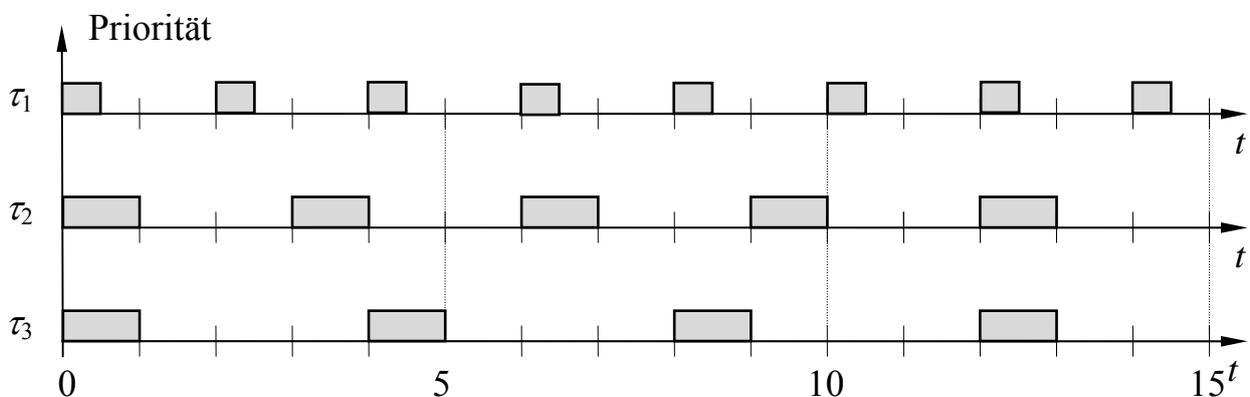
- **Kritischer Moment einer Task**

Zeitpunkt, zu dem die Anforderung einer Task zu deren größtmöglicher Antwortzeit führt (falls Ausführung stets in derselben Periode beendet).

– **Kritische Zeit** einer Task (kritisches Intervall):

Zeit zwischen kritischem Moment und Beendigung der Task.

– **Beispiel 4.1.**  $t_1 = 2, t_2 = 3, t_3 = 4, c_1 = \frac{1}{2}, c_2 = c_3 = 1; p_1 \succ p_2 \succ p_3$ .



• **Satz 4.1.**

Für eine Task tritt dann ein kritisches Moment ein, wenn die Task gleichzeitig mit allen Tasks höherer Priorität angefordert wird.

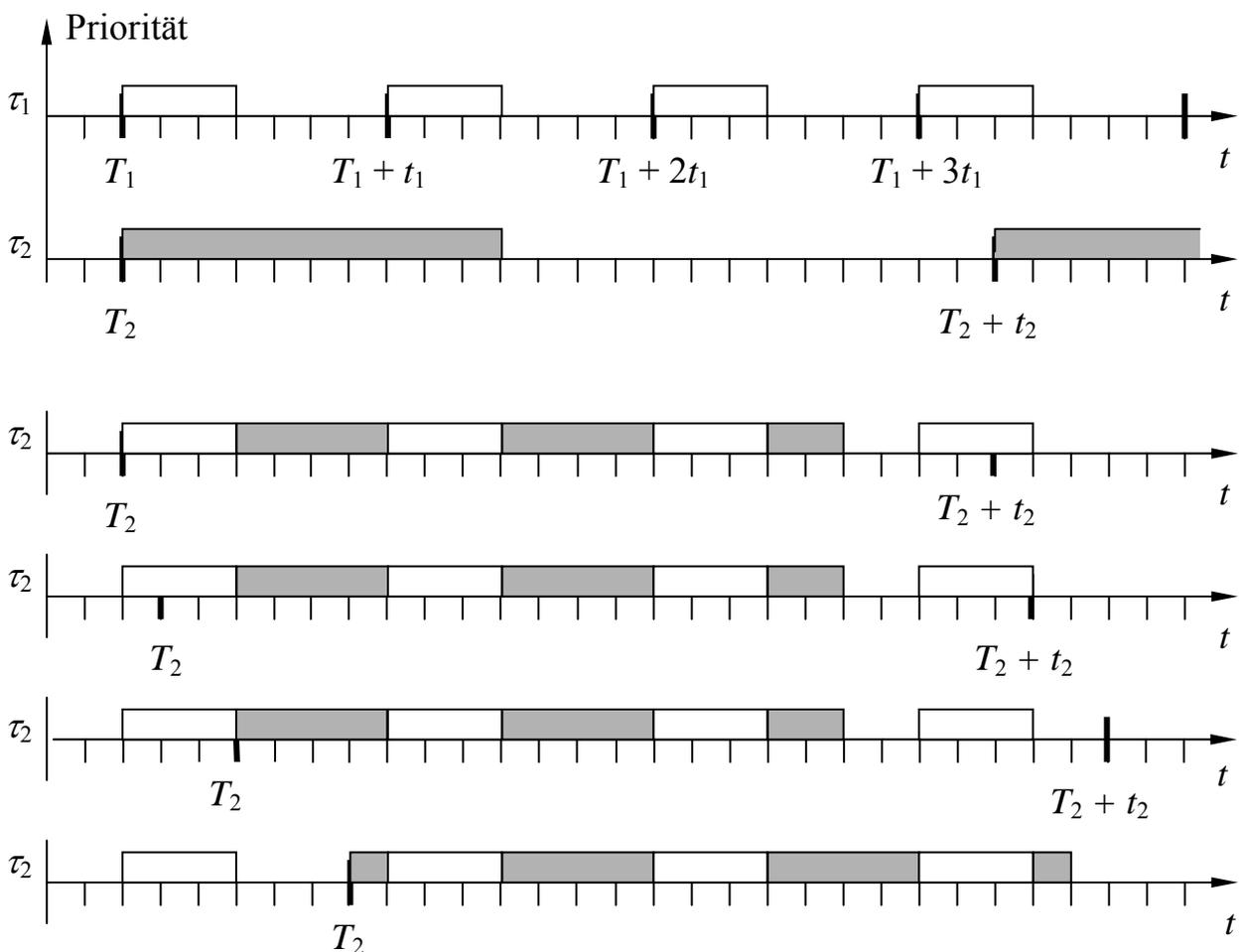
**Beweis ( $n = 2$ ).**

Sei  $p_1 \succ p_2$ . Zur Zeit  $T_2$  erfolge eine Anforderung an  $\tau_2$ , und für  $\tau_1$  erfolge eine Anforderung zu den Zeiten

$$T_1 = T_2, \quad T_1 + t_1, \quad T_1 + 2t_1, \dots$$

Dann erhöht sich die Antwortzeit von  $\tau_2$  um  $\left\lceil \frac{c_2}{t_1 - c_1} \right\rceil \cdot c_1$ .

Weiter: Ein Vergrößern (oder Verkleinern) von  $T_2$  läßt die Antwortzeit von  $\tau_2$  konstant oder verringert sie. □



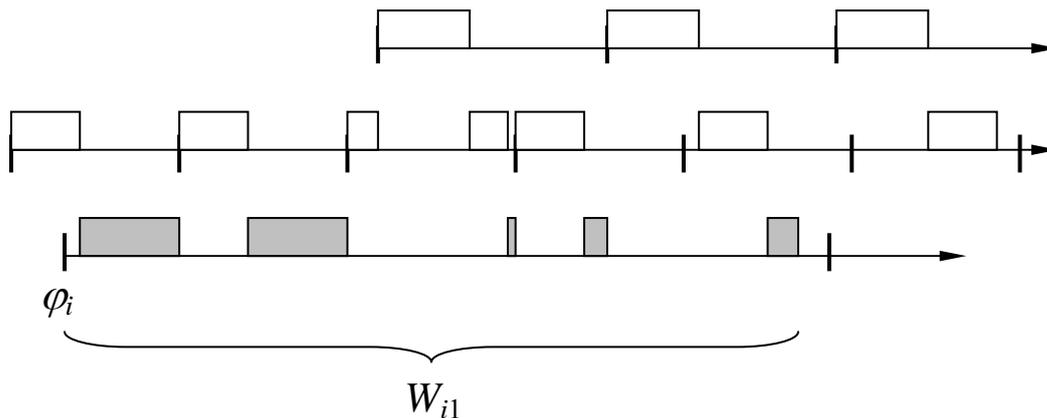
– Formal: Analyse des „Bedarfs an Prozessorzeit“

$$c_i + \sum_{k=1}^{i-1} \left\lceil \frac{W_{i1} + \varphi_i - \varphi_k}{t_k} \right\rceil \cdot c_k, \quad W_{i1}: \text{Antwortzeit von } \tau_{i1}$$

Annahmen:

- \*  $t = 0$  bei  $\min_{k \leq i}(\varphi_k)$
- \* bis  $\varphi_i$  kein Prozessor-Leerlauf
- \* Betrachtung der Antwortzeit  $W_{i1}$  des 1. Jobs  $\tau_{i1}$  von Task  $\tau_i$ .

Dann ist  $\left\lceil \frac{W_{i1} + \varphi_i - \varphi_k}{t_k} \right\rceil$  die Anzahl der Aktivierungen von  $\tau_k$  ( $k < i$ ) bis zur Beendigung von  $\tau_{i1}$ .



Damit ist  $W_{i1}$  die kleinste Lösung der Gleichung

$$W_{i1} = c_i + \sum_{k=1}^{i-1} \left\lceil \frac{W_{i1} + \varphi_i - \varphi_k}{t_k} \right\rceil \cdot c_k - \varphi_i.$$

$W_{i1}$  wird maximal für  $\varphi_k = 0 \quad \forall k < i$  und somit auch für  $\varphi_i = 0$ .

– Weiter folgt: Die maximale Antwortzeit von  $\tau_i$  ist die kleinste Lösung  $t \in [0, t_i]$  der Gleichung

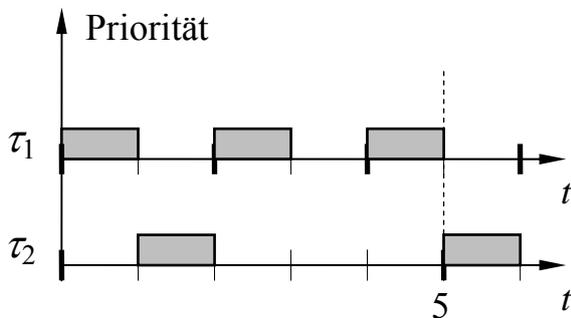
$$t = c_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{t_k} \right\rceil \cdot c_k.$$

– **Folgerung 4.2.**

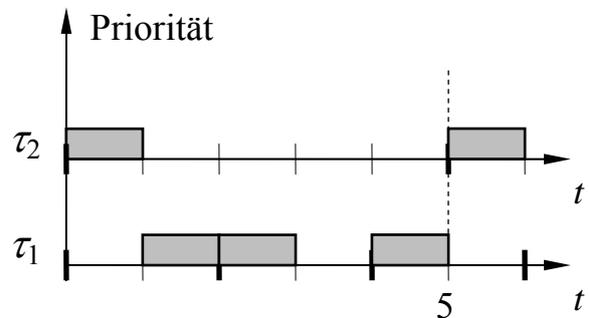
Für die Existenz eines Ablaufplans ist es hinreichend (und notwendig), daß alle Tasks, die in kritischen Momenten angefordert werden, ihre Deadline einhalten. Damit genügt es bei der Untersuchung der Ausführbarkeit eines Ablaufplans, die gleichzeitige Anforderung von Tasks zu betrachten.

– **Beispiel 4.2.**  $\tau_1: (2, 1, p_1), \quad \tau_2: (5, 1, p_2).$

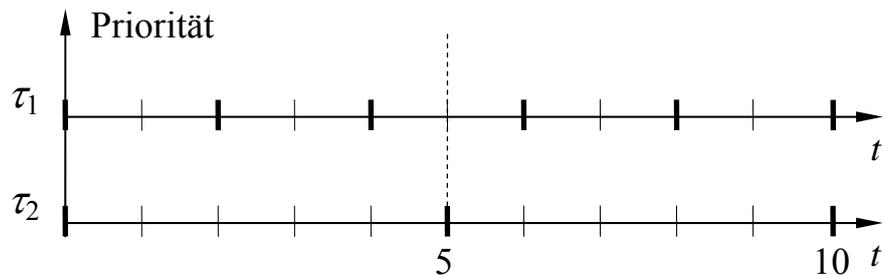
a)  $p_1 \succ p_2$ :



b)  $p_2 \succ p_1$ :



c) Modifikation:  
 $c_1 =$   
 (max.)



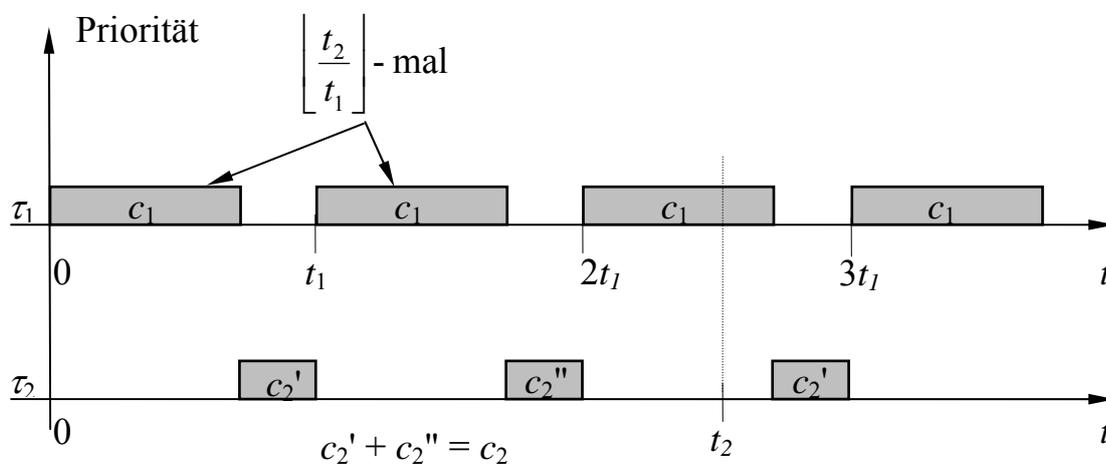
• **Lemma 4.3 (LIU/LAYLAND).**

Sei  $T = \{\tau_1, \tau_2\}$  mit  $t_1 < t_2$ . Gibt es einen ausführbaren Ablaufplan bei  $p_2 \succ p_1$ , so gibt es einen solchen Plan auch bei  $p_1 \succ p_2$ .

**Beweis.**

Notwendig (aber nicht hinreichend) für die Existenz eines Ablaufplans bei  $p_1 \succ p_2$  ist mit  $m = \left\lfloor \frac{t_2}{t_1} \right\rfloor$  die Bedingung (man beachte Folg. 4.2)

$$mc_1 + c_2 \leq t_2. \quad (*)$$



Sei nun  $p_2 \succ p_1$ . Dann muß offenbar gelten:

$$c_1 + c_2 \leq t_1.$$

Diese Bedingung impliziert wegen  $m \geq 1$  Bedingung (\*):

$$mc_1 + c_2 \leq mc_1 + mc_2 \leq mt_1 \leq t_2.$$

• **Lemma 4.3.**

Für  $T = \{\tau_1, \dots, \tau_n\}$  sei eine statische Prioritätszuordnung  $p$  gegeben; o.B.d.A. sei  $p(\tau_i) = i$ ,  $i = 1, \dots, n$  (1: höchste Priorität). Weiter gebe es ein  $k \in \{1, \dots, n-1\}$  mit  $t_k > t_{k+1}$ , und  $T$  sei unter diesen Bedingungen einplanbar. Dann ist  $T$  auch einplanbar, wenn die Prioritäten von  $\tau_k$  und  $\tau_{k+1}$  vertauscht werden.

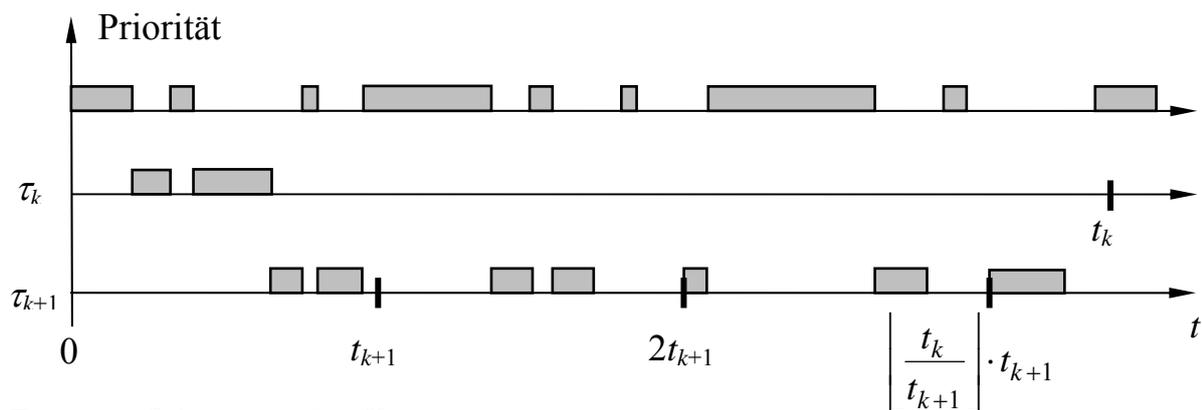
**Beweis.**

Sei 
$$m = \left\lfloor \frac{t_k}{t_{k+1}} \right\rfloor$$

und 
$$c_0 = \sum_{i=1}^{k-1} c_i^*, \quad c_i^*: \text{Gesamtausführungszeit aller höher priorisierten } \tau_i \text{ in } [0, m \cdot t_{k+1}].$$

Notwendig für die Einplanbarkeit von  $T$  bei  $p$  ist

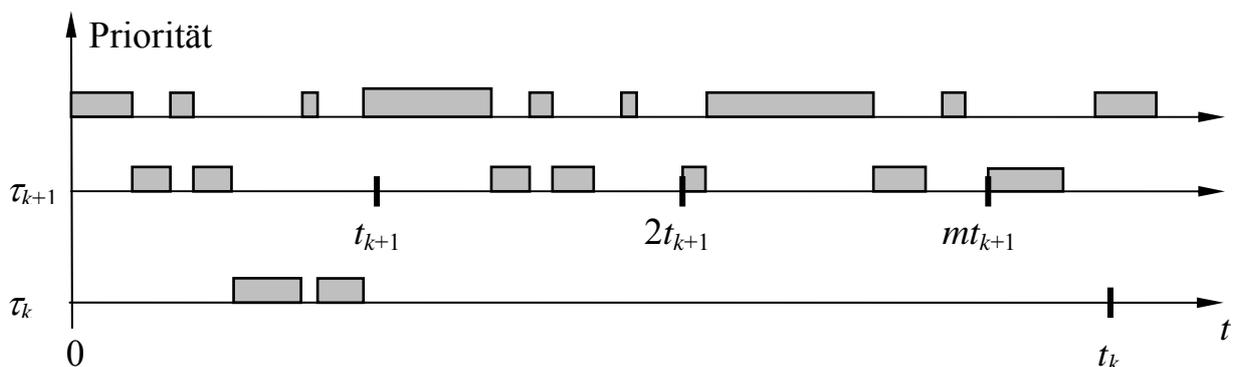
$$c_0 + c_k + m c_{k+1} \leq m t_{k+1}.$$



Daraus folgt unmittelbar

$$c_k \leq m(t_{k+1} - c_{k+1}) - c_0,$$

was (unter Beachtung von Folg. 4.2) hinreichend für die Einplanbarkeit von  $T$  nach Vertauschen der Prioritäten von  $\tau_k$  und  $\tau_{k+1}$  ist. ■



- **Ratenmonotone Prioritätszuordnung RMS**

Den Tasks einer Taskmenge  $T$  wird eineindeutig eine Priorität in der Reihenfolge ihrer Anforderungsraten zugeordnet (höchste Rate entspricht höchster Priorität; bei gleicher Rate werden unterschiedliche, aber aufeinanderfolgende Prioritäten festgelegt).

- **Satz 4.4. (Optimalitätseigenschaft von RMS)**

Sei  $T$  eine Taskmenge, und sei  $\Sigma(T)$  die Gesamtheit aller statischen Prioritätszuordnungen für  $T$ . Dann gilt: Gibt es irgendeine Zuordnung aus  $\Sigma(T)$ , die zu einem ausführbaren Ablaufplan führt, so erzeugt auch RMS einen solchen Plan.

***Beweis.***

Sei

$p: T \rightarrow \{1, \dots, n\}$  eineindeutig (1: höchste Priorität),  
so daß es einen Ablaufplan für  $T$  gebe. Weiter existiere ein  
 $k \in \{1, \dots, n-1\}$  mit

$$t_k > t_{k+1}, \quad p(\tau_k) \succ p(\tau_{k+1}),$$

und  $T$  sei unter diesen Bedingungen einplanbar. Dann ist nach Lemma 4.3  $T$  auch einplanbar, wenn die Prioritäten von  $\tau_k$  und  $\tau_{k+1}$  vertauscht werden.

RMS ist eine Permutation der ursprünglichen Prioritätszuordnung, jede Permutation läßt sich als Produkt von Transpositionen darstellen. ■

### 4.3. Prozessorauslastung und Existenz von Ablaufplänen – Admission-Kriterien für RMS

Gegeben sei eine Taskmenge  $T = \{\tau_1, \dots, \tau_n\}$ ,  $\tau_i = (t_i, c_i, p_i)$ ,  $i = 1, \dots, n$ .

- **Begriffe**

- *Prozessorauslastung von T*

$$U = 1 - p_0 = \sum_{i=1}^n \frac{c_i}{t_i} \quad p_0: \text{Stillstandswahrscheinlichkeit}$$

- Eine Taskmenge  $T$  heißt **gerade noch einplanbar** (*fully utilizing the processor, difficult-to-schedule*), wenn es für  $T$  bei der gegebenen Prioritätszuordnung einen ausführbaren Ablaufplan gibt, der jedoch bei Vergrößerung der Bearbeitungszeit einer Task nicht mehr ausführbar ist.

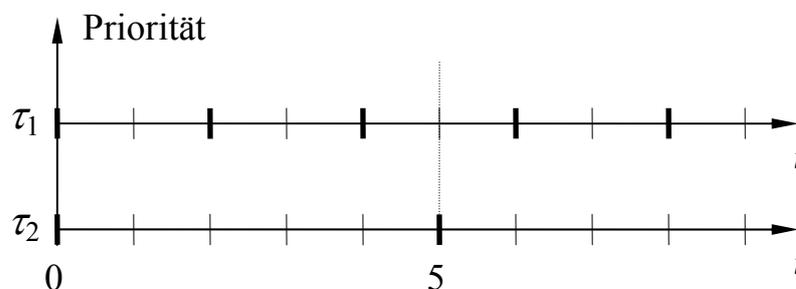
Aufgabe: Bestimmung eines Wertes  $U_g$ , so daß es für alle Taskmengen mit  $U \leq U_g$  stets einen Ablaufplan gibt. Da RMS optimal, genügt es,  $U_g$  bzgl. RMS zu bestimmen.

- **Grenzauslastung, obere Grenze  $U_g$  der Prozessorauslastung** (*maximum schedulable utilization, utilization bound*)

Minimum von  $U$  über alle Taskmengen (über alle möglichen Werte von  $t_i$  und  $c_i$ ), die bei RMS den Prozessor voll auslasten.

- **Beispiel. 4.2. c)**  $U =$

**4.2. d)**  $t_1 = 2 \quad c_1 = 0,9 \quad t_2 = 5 \quad c_2 = 2,5 \quad U =$



- **Lemma 4.5.** Für zwei Tasks ist die obere Grenze  $U_g$  der Prozessorauslastung unter RMS

$$U_g = 2(\sqrt{2} - 1).$$

**Beweis.**

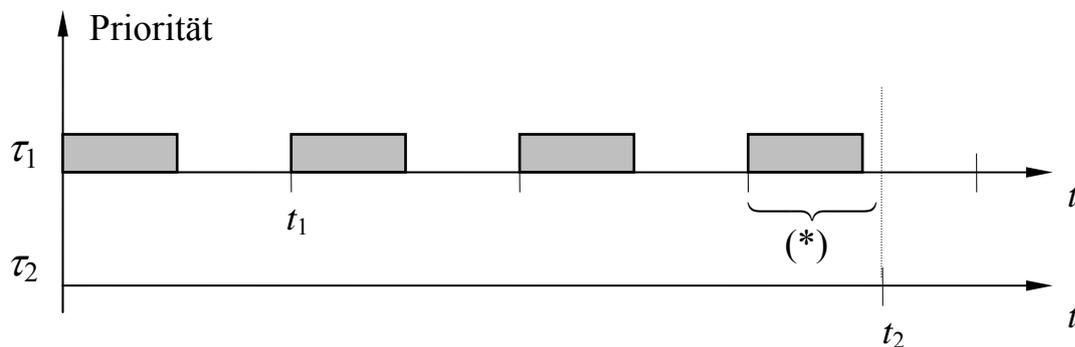
Sei  $T = \{\tau_1, \tau_2\}$  eine Menge von zwei Tasks mit  $t_1 < t_2$ , mithin  $p_1 \succ p_2$ . Dann ist  $I := [0, t_2)$  das kritische Intervall für  $\tau_2$ .

In  $I$  treten  $\left\lceil \frac{t_2}{t_1} \right\rceil =: l$  Anforderungen an  $\tau_1$  auf. Weiter sei  $k := \left\lfloor \frac{t_2}{t_1} \right\rfloor$ .

– **Fall a):**

Alle in  $I$  auftretenden Anforderungen an  $\tau_1$  werden vollständig in  $I$  erfüllt. Das bedeutet (s. Abb.)

$$c_1 \leq t_2 - kt_1. \tag{*}$$



Dann lastet  $T$  den Prozessor genau dann voll aus, wenn

$$c_2 = t_2 - lc_1,$$

und damit ist

$$\begin{aligned} U &= \frac{c_1}{t_1} + \left( \frac{t_2}{t_2} - l \cdot \frac{c_1}{t_2} \right) = \\ &= 1 + c_1 \left( \frac{1}{t_1} - \frac{l}{t_2} \right), \end{aligned} \tag{a}$$

und wegen

$$l = \left\lceil \frac{t_2}{t_1} \right\rceil \geq \frac{t_2}{t_1}$$

folgt

$$\frac{l}{t_2} \geq \frac{1}{t_1}$$

und damit ist (a) monoton fallend in  $c_1$ .

– **Fall b):**

Bei

$$c_1 \geq t_2 - kt_1$$

lastet  $T$  den Prozessor genau dann voll aus, wenn

$$c_2 = k(t_1 - c_1),$$

so daß

$$\begin{aligned} U &= \frac{c_1}{t_1} + k \cdot \frac{t_1 - c_1}{t_2} = \\ &= k \cdot \frac{t_1}{t_2} + c_1 \left( \frac{1}{t_1} - \frac{k}{t_2} \right). \end{aligned} \tag{b}$$

Dabei ist wegen  $\lfloor x \rfloor \leq x$  der Faktor von  $c_1$  nichtnegativ und somit  $U$  monoton steigend in  $c_1$ .

– **Also:**

$U_g$  tritt bei  $c_1 = t_2 - kt_1$  auf, und dann gilt

$$U_g = 1 - \frac{t_1}{t_2} \left( l - \frac{t_2}{t_1} \right) \left( \frac{t_2}{t_1} - k \right).$$

Sei

$$r = \frac{t_2}{t_1} - k \neq 0.$$

Dann ist

$$\frac{t_2}{t_1} = k + r, \quad l = k + 1$$

und mithin

$$\begin{aligned} U_g &= 1 - \frac{1}{k+r} \cdot (k+1 - k - r) \cdot r = \\ &= 1 - \frac{r(1-r)}{k+r} = f(k, r) \rightarrow \text{Min.}! \end{aligned}$$

Da  $U_g$  monoton in  $k$  steigt und  $k \in \mathbb{N}^+$  gilt, bedeutet dies

$$U_g = 1 - \frac{r(1-r)}{r+1} \rightarrow \text{Min.}!$$

Daraus resultiert

$$r = \sqrt{2} - 1$$

und schließlich

$$U_g = 2(\sqrt{2} - 1).$$

q.e.d.

- **Satz 4.6 (LIU/LAYLAND, 1973).**

Für eine Menge von  $n$  Tasks mit der Auslastung  $U$  gibt es einen Ablaufplan gemäß RMS, wenn  $U \leq U_g$  gilt mit

$$U_g = U_g(n) = n(\sqrt[n]{2} - 1)$$

(„obere Grenze der Prozessorauslastung“).

$$n = 2: U_g \approx 0,828$$

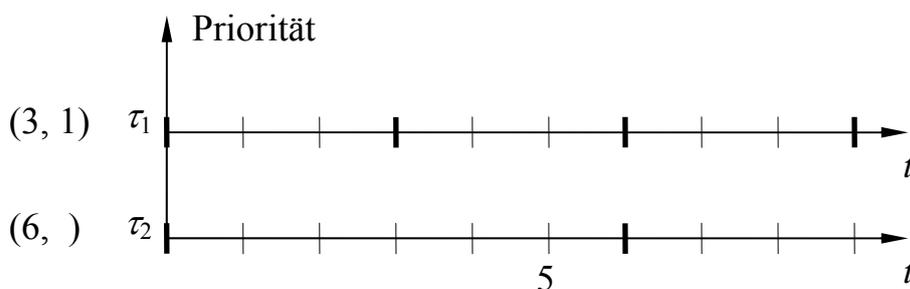
$$n = 3: U_g \approx 0,780$$

$$n \rightarrow \infty: U_g \rightarrow \approx 0,693.$$

- **Folgerung 4.7.**

Ist  $\frac{t_j}{t_i} \in \mathbb{N}$  für alle  $i, j$  mit  $1 \leq i \leq j, i, j = 1, \dots, n$ , so gibt es genau

dann einen Ablaufplan, wenn  $\sum_{i=1}^n \frac{c_i}{t_i} \leq 1$ .



- **Struktur des Beweises für  $n \geq 2$**

0. Es werden ausschließlich Taskmengen  $T$  betrachtet, die „gerade noch einplanbar“ sind (die den Prozessor maximal auslasten, schwer einplanbar sind). Für diese Taskmengen wird die kleinstmögliche Auslastung bestimmt.

o.B.d.A. sei  $T$  nach Periodenlängen geordnet, d.h.  $t_1 < t_2 < \dots < t_n$ .

1. Sei  $t_n \leq 2t_1$ .

– Bei gegebenen  $t_i$  wird eine Taskmenge konstruiert, die gerade noch einplanbar ist.

– Bei gleichen  $t_i$  hat jede andere Taskmenge eine höhere Auslastung.

2. Die Auslastung der in 1. konstruierten Taskmenge ist abhängig von den Periodenlängen, genauer von den Periodenverhältnissen  $\frac{t_{i+1}}{t_i}$ . Für diese

Taskmengen ist der kleinstmögliche Auslastungswert  $U_n = n(\sqrt[n]{2} - 1)$ .

3.  $T$  sei eine Taskmenge mit  $t_n > 2t_1$ .

– Konstruktion einer Taskmenge  $T'$ :  $(t'_i, c_i)$  für  $i < n$ ,  $(t_n, c'_n)$  mit  $t_n \leq 2t'_i \quad \forall i \neq n$ :

$$t'_i = (\lceil t_n / t_i \rceil - 1) \cdot t_i,$$

$c'_n$ : Summe aller „restlichen“ Bearbeitungszeiten.

Die Auslastung von  $T'$  ist nach 2. mindestens  $U_n$ .

– Die Auslastung der ursprünglichen Taskmenge  $T$  ist (echt) größer als die Auslastung von  $T'$ .

- **Beispiel 4.3.**

$$n = 2$$

- **Verallgemeinerungen und Ergänzungen**

- $d_i < t_i$ :

RMS ist nicht mehr optimal. Aber: Zuordnung

$$\text{Priorität} \sim d_i^{-1}$$

ist optimal: **Deadline-monotones Scheduling DMS.**

Einplanbarkeit: Satz 4.8,  $t \in (0, d_i)$ .

- $d_i > t_i$ :

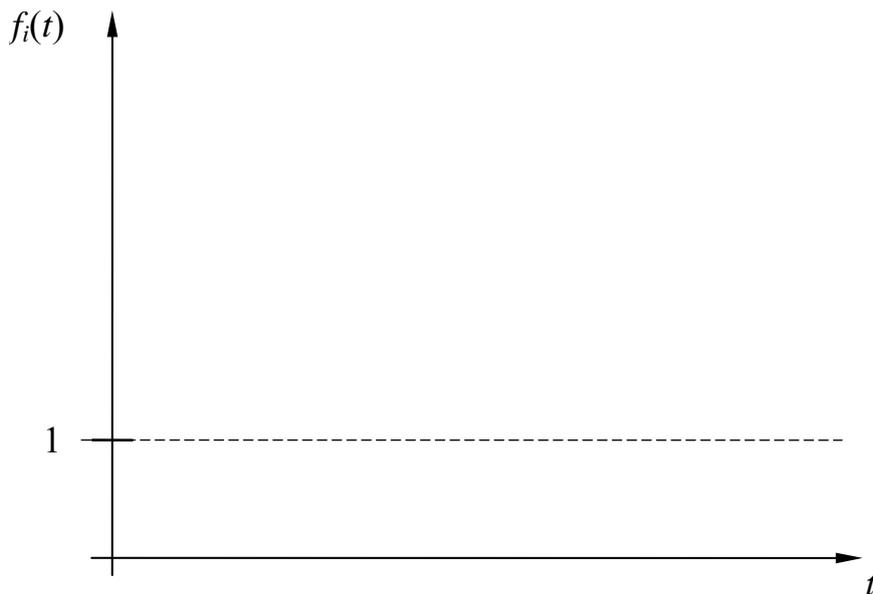
weder raten- noch deadline-monotones Scheduling sind optimal.

- **Satz 4.8 (LEHOCZKY, 1989).**

Sei  $T = \{\tau_1, \dots, \tau_n\}$  geordnet nach aufsteigender Periode. Dann gilt:

Für  $T$  gibt es genau dann einen Ablaufplan gemäß RMS, wenn

$$\max_{1 \leq i \leq n} \min_{t \in (0, t_i]} f_i(t) \leq 1 \quad \text{mit} \quad f_i(t) = \frac{1}{t} \cdot \sum_{j=1}^i \left\lceil \frac{t}{t_j} \right\rceil \cdot c_j.$$



• **Analyse des Zeitbedarfs (time-demand analysis)**

$$? \quad \forall i \quad \exists t : \quad w_i(t) = c_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{t_j} \right\rceil c_j \leq t$$

**Beispiel 4.4.**  $T = \{(2; 1), (4; 0,5), (5; 0,5), (6; 1,5)\}$

