

### 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

firm

„time/utility function“



- **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 (zusätzlich)
- **Scheduling**
  - \* *Einplanung*: Vorgehen (Algorithmus), das bei gegebener Taskbeschreibung für jede Taskmenge einen gültigen 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 (Verdrängbarkeit)
  - Ein-/Mehrprozessorsysteme

## 3.2. Schedulingstrategien für Jobnetze

- **Beschreibung**

- **Job**

$J \ e \ (r \ d]$

Jobname – Ausführungszeit – Freigabezeit – Zeitschranke (Deadline)

- **Jobnetz**

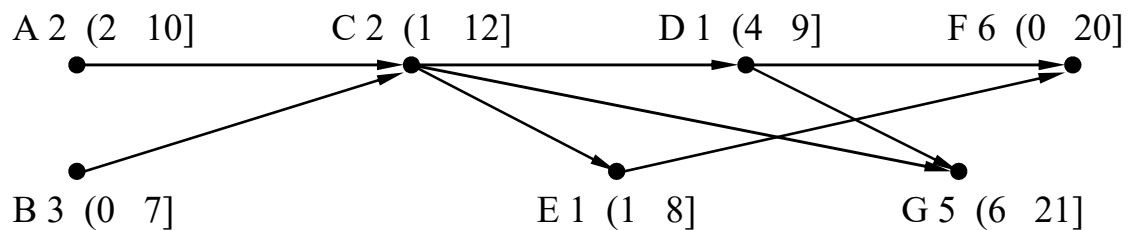
Jobabhängigkeiten:

Präzedenzrelation  $R$ : irreflexive asymmetrische Relation

$(A, B) \in R$ : 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)

→ in 1-Prozessor-Systemen können Abhängigkeiten ignoriert werden!

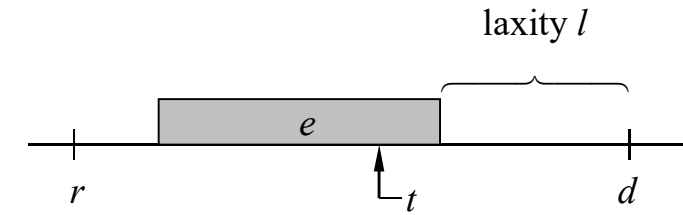
• **Grundlegende Strategien**

EDF Earliest Deadline First

LST Least Slack Time

MLF Minimum Laxity First

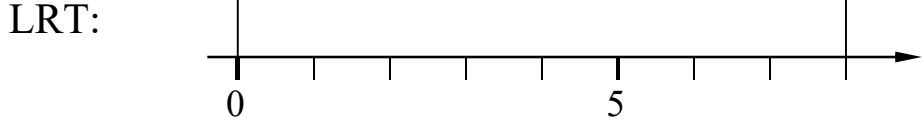
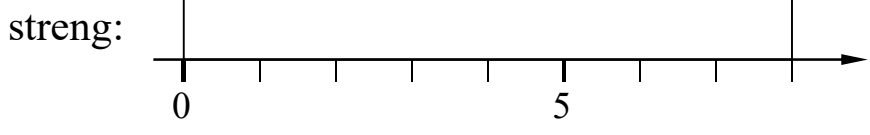
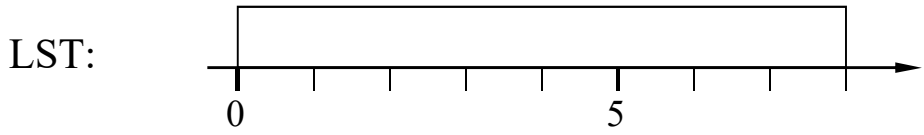
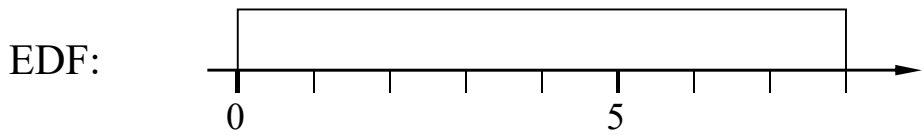
LRT Latest Release Time



$l = (d - t) - e'$   $e'$ : Restlaufzeit

„EDF rückwärts“

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



– **Bedeutung**

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

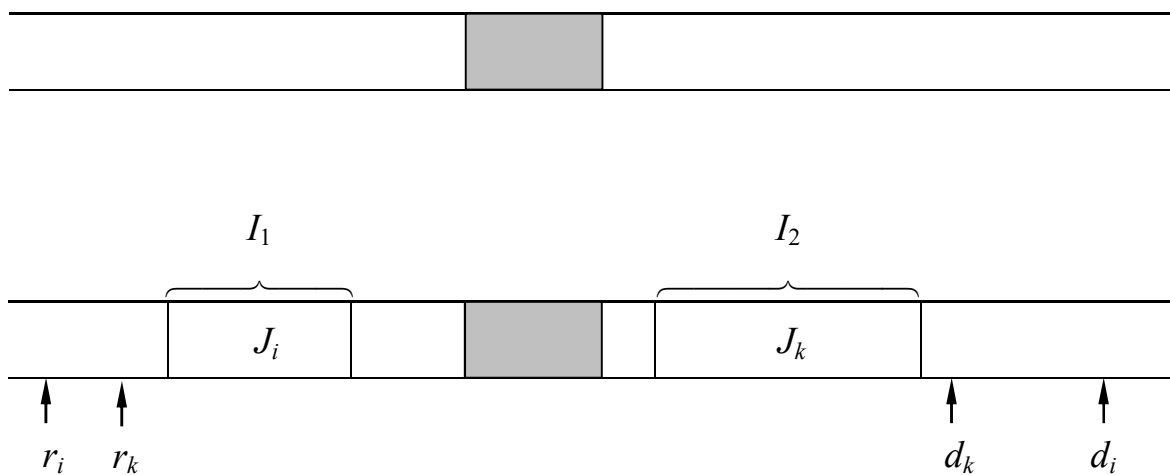
### 3.3. EDF

- **Optimalität**

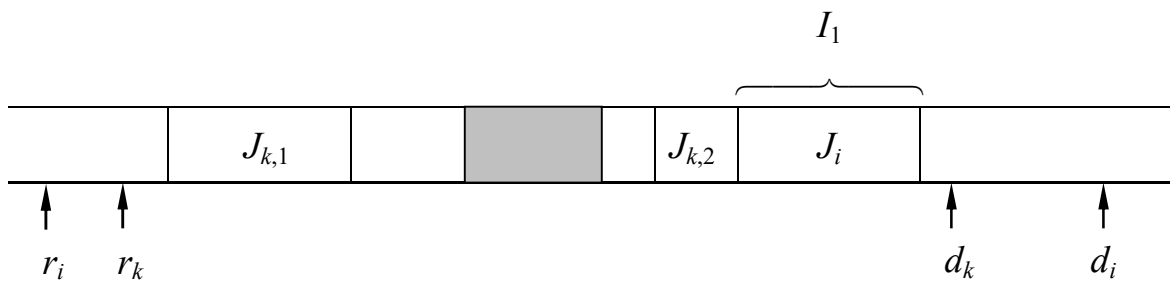
EDF ist optimal in 1-Prozessor-Systemen mit Entzug und ohne gemeinsame BM-Nutzung, d.h.: Wenn es unter diesen Bedingungen auf irgendeine Weise möglich ist, eine Jobmenge  $J$  einzuplanen, so ist dies auch gemäß EDF möglich.

**Beweis.**

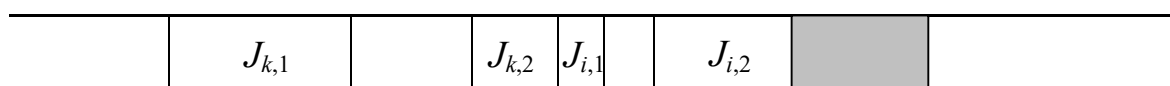
Folgender Ablaufplan sei ausführbar:



Dann ist auch der folgende Ablaufplan ausführbar:



Damit ergibt sich als gültiger EDF-Plan bzgl.  $J_i$  und  $J_k$ :



Gilt offenbar auch bei  $I_1 > I_2$  oder wenn  $r_k$  innerhalb von  $I_1$  liegt.

• **Admission-Kriterium für periodische Tasks**

– *Voraussetzungen*

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

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

– *Admission-Kriterium für EDF*

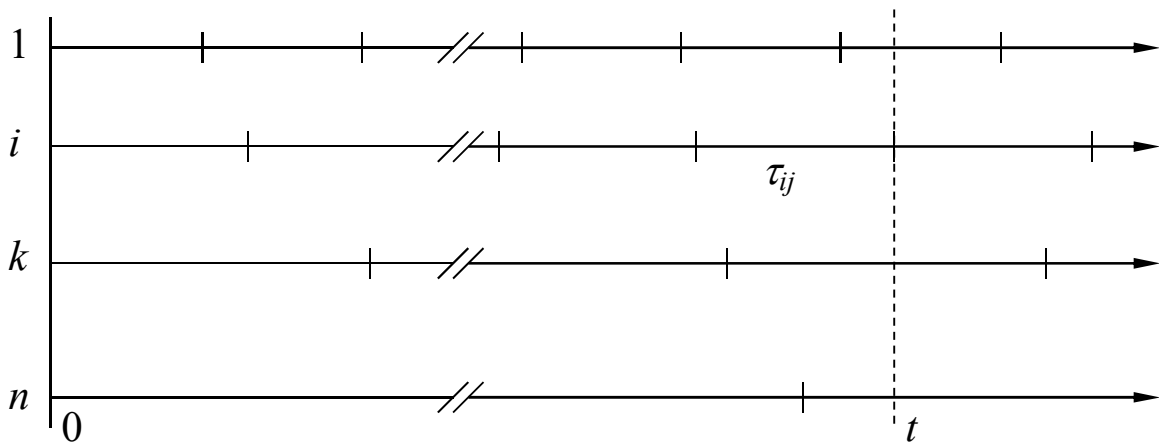
$T$  ist einplanbar genau dann, wenn

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

– *Beweis-Skizze*

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

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



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

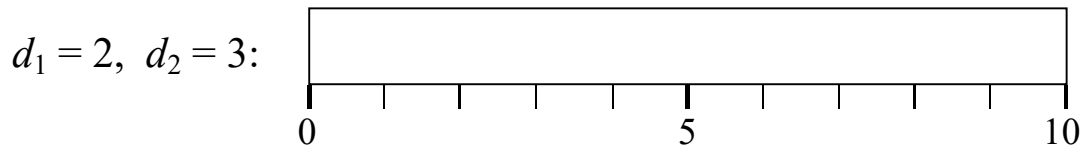
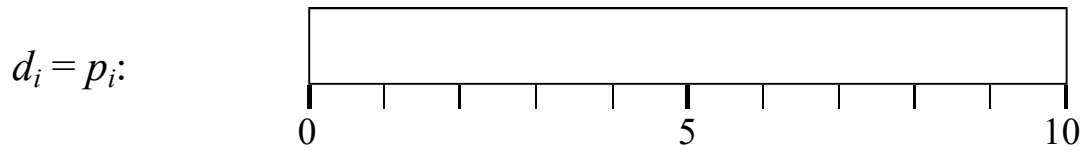
Der Gesamtbedarf an Prozessorzeit bis zu diesem Zeitpunkt übersteigt  $t$ :

– **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.  $\tau_1(2; 1), \tau_2(5; 2,5)$

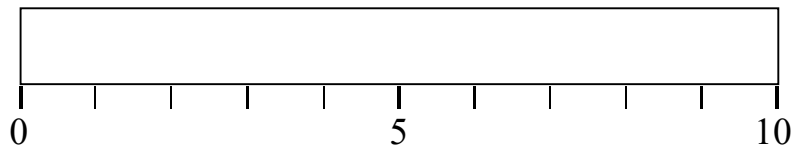


\* **Dichte (density)**

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

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

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



\* **In summa**

• Probleme

– Nicht-Optimalität

EDF ist nicht optimal bei

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

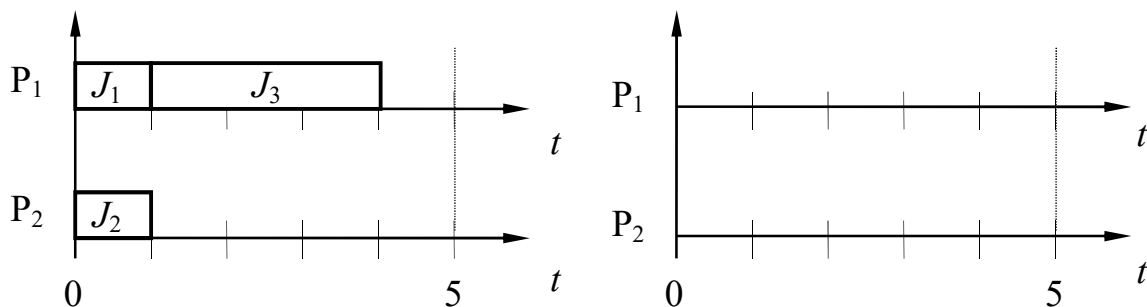
**Beispiele.**

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

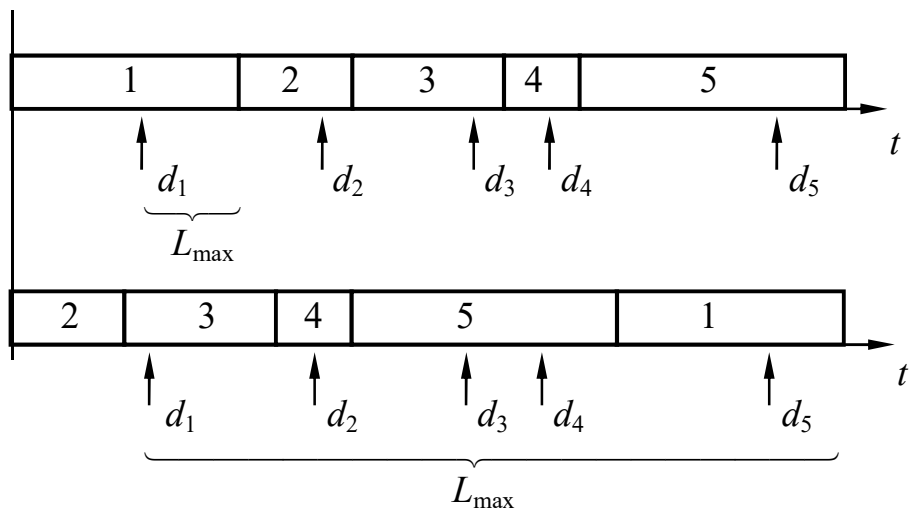


(2) A 3 (0 4]    B 1 (1 3]     

(3) Prozessoren P<sub>1</sub>, P<sub>2</sub>; 3 Jobs (e<sub>i</sub>, d<sub>i</sub>): (1; 1) (1; 2) (3; 3,5); r<sub>i</sub> = 0



– „Domino-Effekt“:



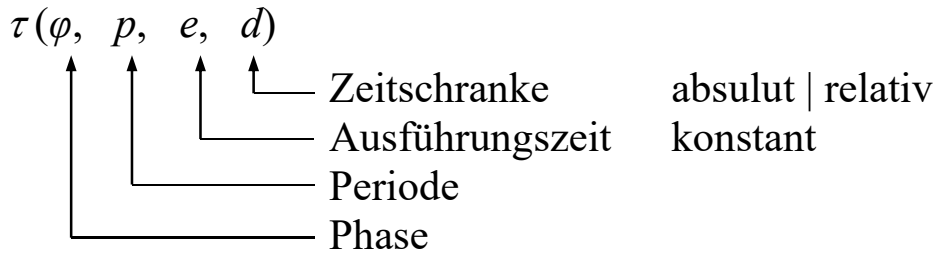


### 3.4. Zeitgesteuertes Scheduling für periodische Tasks

- **Periodische Tasks – Begriffe**

- *Beschreibung*

$$T = \{\tau_1, \dots, \tau_n\} \quad n \in \mathbb{N} \text{ fest}; \quad \tau_i = (\tau_{ij})_{j=1,2,\dots}$$



unabhängig, d.h.  $R = \emptyset$  und keine gemeinsamen BM!

- *Hyperperiode*

$$H = \text{kgV}(p_1, \dots, p_n)$$

- *Harmonische Perioden*

$$p_i \leq p_j \Rightarrow p_i | p_j \quad \forall i, j = 1, \dots, n$$

- *Klassifikation (nach J. LIU)*

periodisch: konstant | minimaler Abstand

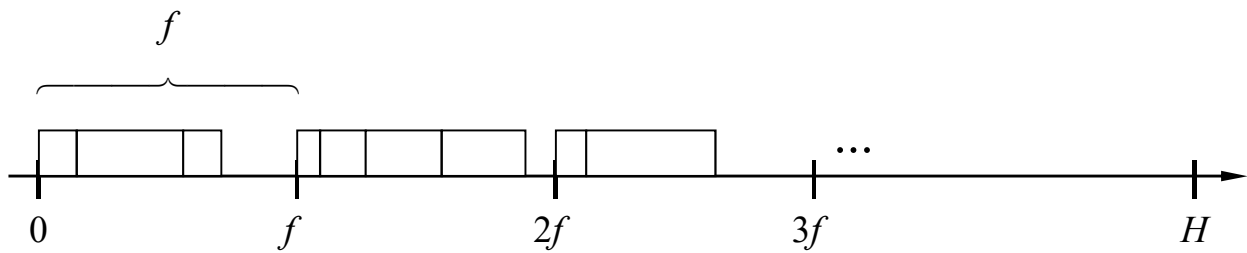
sporadisch: beliebig dicht, harte Deadline

aperiodisch: keine (oder weiche) Deadline

- *Off-line Scheduling*

- **Rahmenbasiertes zyklisches Scheduling**

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



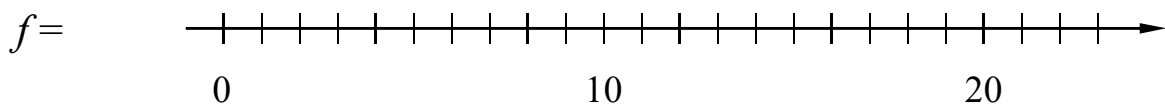
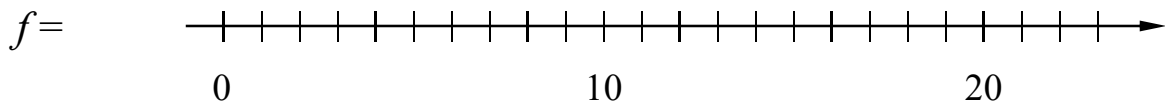
- **Bedingungen für  $f$**

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

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

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

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



- **Jobscheiben (job slices)**

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

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

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

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

