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 (zusätzlich)

– ***Scheduling***
  * *Einplanung:* Vorgehen (Algorithmus), das bei gegebener Task-beschreibung 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 $\boldsymbol{T}$ einen ausführbaren Ablaufplan, sofern überhaupt $\boldsymbol{T}$ 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***

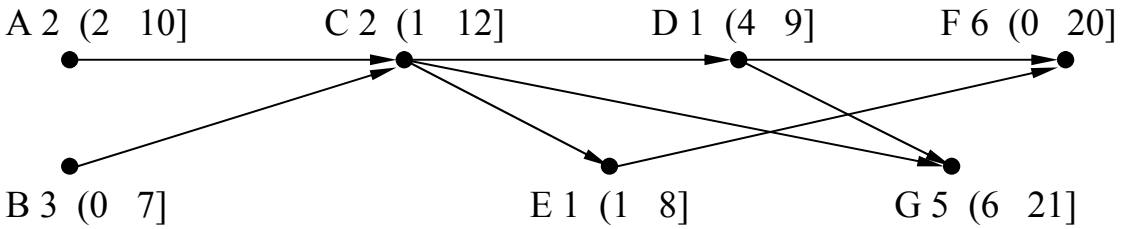
Jobabhängigkeiten:

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)

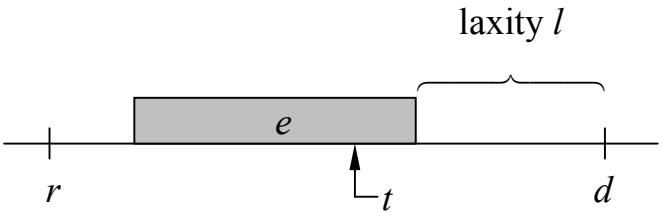
→ in 1-Prozessor-Systemen können Abhängigkeiten ignoriert werden!

- **Grundlegende Strategien**

EDF Earliest Deadline First

LST Least Slack Time

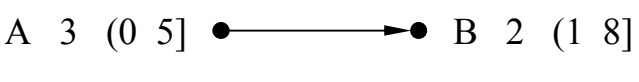
MLF Minimum Laxity First



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

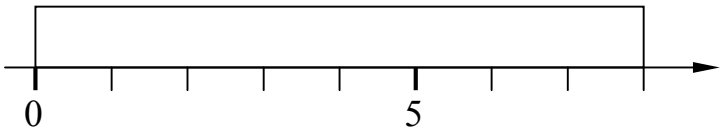
LRT Latest Release Time „EDF rückwärts"

– ***Beispiel.***



EDF:

LST:



streng:

LRT:

– ***Bedeutung***

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

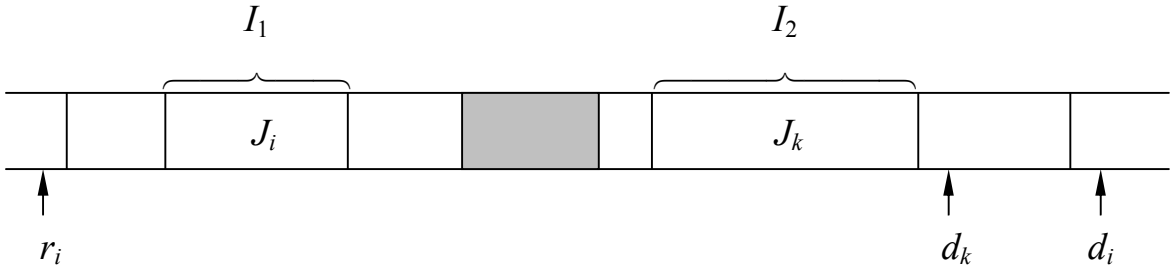
3.3. EDF

- **Optimalität**

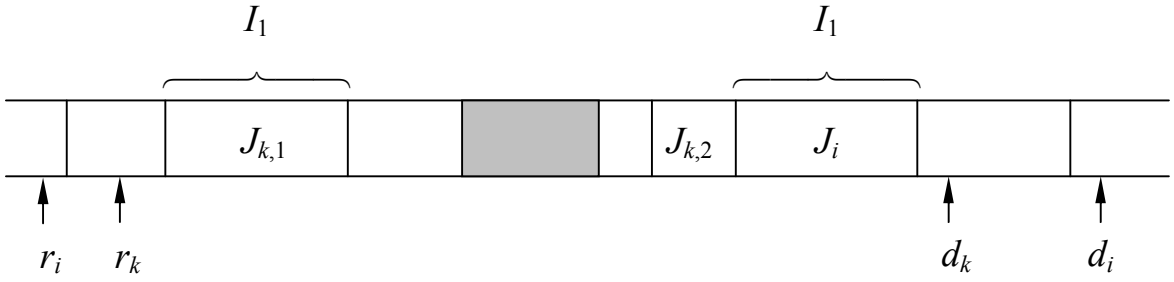
  EDF ist optimal in 1-Prozessor-Systemen mit Entzug und ohne gemeinsame BM-Nutzung, d.h.: Wenn es auf irgendeine Weise möglich ist, eine Jobmenge ***J*** einzuplanen, so ist dies auch gemäß EDF möglich.

Beweis.

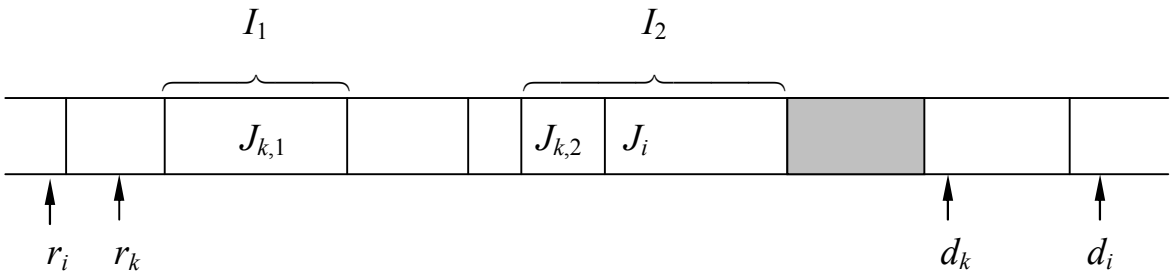
Folgender Ablaufplan sei ausführbar:



Dann ist auch der folgende Ablaufplan ausführbar:



Damit ergibt sich als endgültiger EDF-Plan:



- **Admission-Kriterium für periodische Tasks**

– ***Voraussetzungen***

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

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

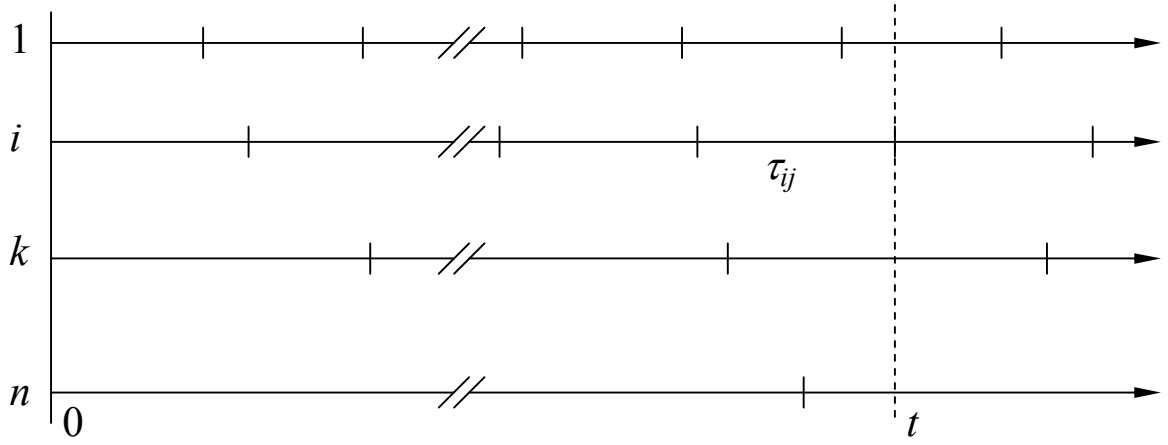
– ***Admission-Kriterium für EDF***

T ist einplanbar genau dann, wenn $\boxed{u = \sum_{i=1}^{n} \frac{e_i}{p_i} \leq 1}$

– ***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:

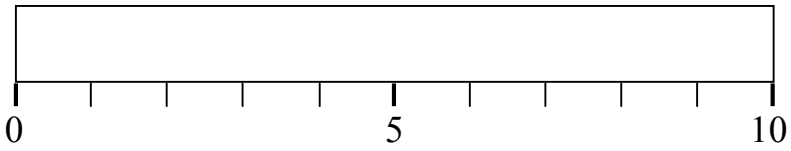
– ***Allgemeiner***

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

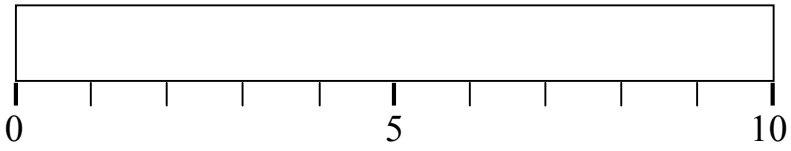
* $\exists i$: $d_i < p_i$: T muß nicht einplanbar sein auch bei $u \leq 1$.

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

  $d_i = p_i$:



  $d_1 = 2,\ d_2 = 3$:

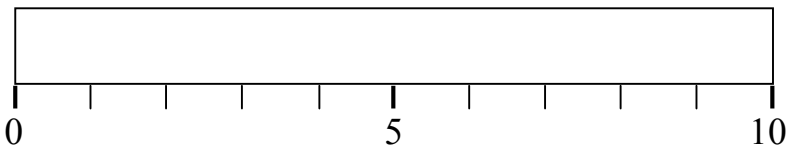


* ***Dichte (density)***

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

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

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



* ***In summa***

- **Probleme**

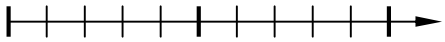
– ***Nicht-Optimalität***

EDF ist nicht optimal bei
(1) gemeinsam genutzten Betriebsmitteln
(2) nicht verdrängbaren Jobs
(3) in Mehrprozessorsystemen.

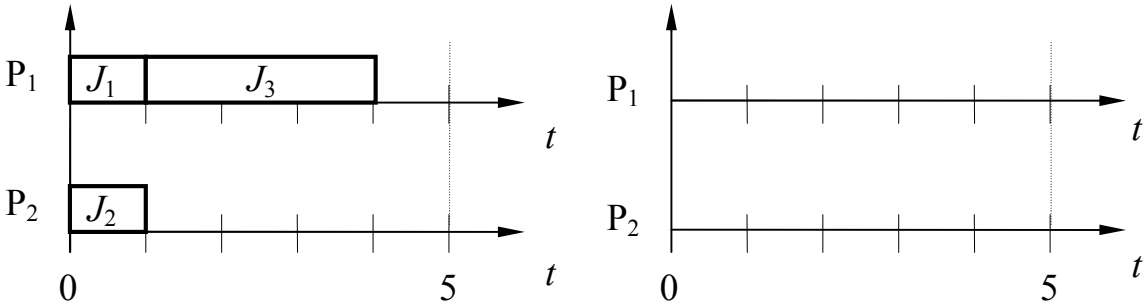
Beispiele.

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

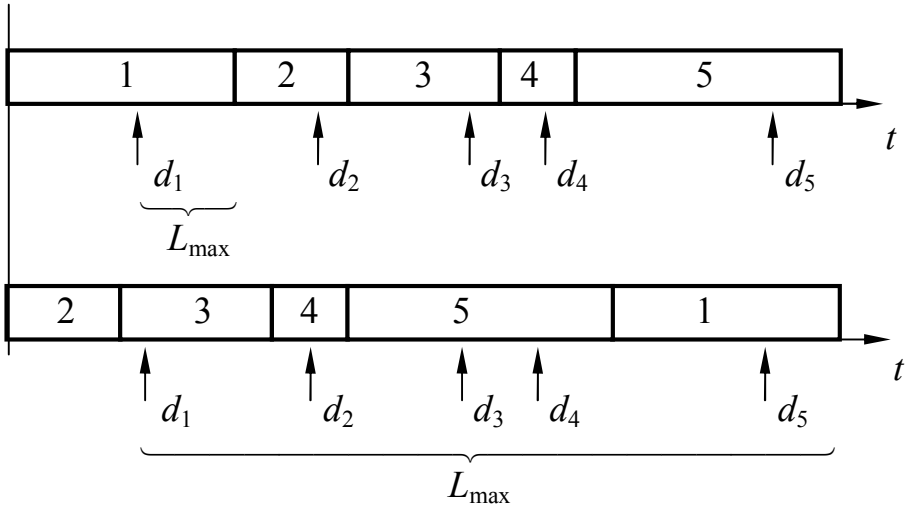
(2) A 3 (0 4] B 1 (1 3]



(3) Prozessoren P_1, P_2; 3 Jobs (e_i, d_i): (1; 1) (1; 2) (3; 3,5); $r_i = 0$



– ***„Domino-Effekt“:***



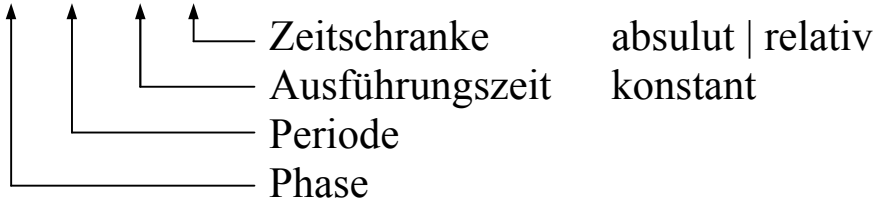
3.4. Zeitgesteuertes Scheduling für periodische Tasks

- **Periodische Tasks – Begriffe**

– ***Beschreibung***

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

$\tau(\varphi,\ p,\ e,\ d)$



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

– ***Hyperperiode***

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

– ***Harmonische Perioden***

$p_i \le p_j \ \Rightarrow\ p_i \mid p_j \quad \forall i,j = 1,\ldots,n$

– ***Klassifikation (nach J. LIU)***

periodisch: konstant | minimaler Abstand

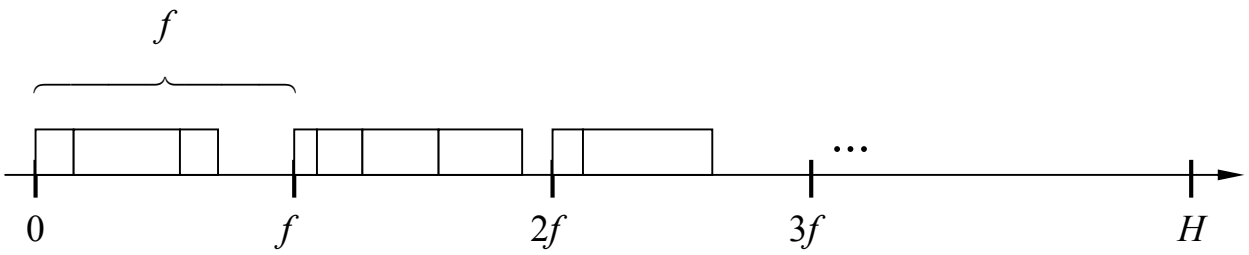
sporadisch: beliebig dicht, harte Deadline

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.



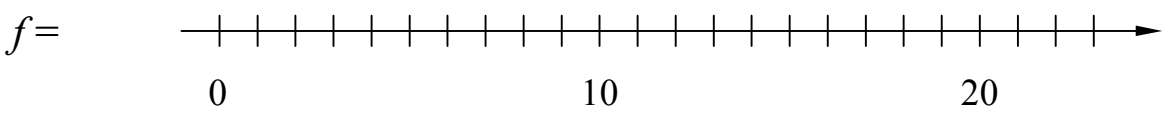
– ***Bedingungen für f***

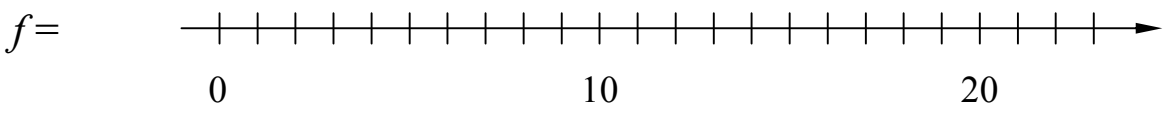
(1) $f \geq e_i \quad \forall i = 1,...,n$

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

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

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

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

Exakt: LEHOCZKY-Kriterium bzw. Analyse des Zeitbedarfs.

Ausgangspunkt: Kritischer Zeitpunkt einer Task.

– ***Dynamische Prioritäten***

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

– ***Beispiel.*** τ_1: $p_1 = 2$ $e_1 = 1$ τ_2: $p_2 = 5$ $e_2 = 2{,}5$. $u =$