

# Echtzeitsysteme

## Ereignisgesteuerte Ablaufplanung periodischer Echtzeitsysteme

**Peter Ulbrich**

Lehrstuhl für Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität Erlangen-Nürnberg

[https://www4.cs.fau.de/Lehre/WS18/V\\_EZS/](https://www4.cs.fau.de/Lehre/WS18/V_EZS/)

22. November 2018



- Was sind die **Prioritäten** der ereignisorientierten Einplanung?
  - Welche Kriterien werden auf Prioritäten abgebildet?
  - **Statische** und **dynamische Verfahren** zur Bestimmung von Prioritäten
  - Wie geht man mit einer knappen Anzahl von Systemprioritäten um?
  
- Optimalität ereignisgesteuerter Ablaufplanung
  - Wie schlagen sich die vorgestellten Verfahren?
  - Wo liegen die Grenzen ereignisgesteuerter Ablaufplanung?
  
- Beurteilung der **Planbarkeit ereignisgesteuerter Systeme**?
  - Mit Hilfe der **maximalen, kumulativen CPU-Auslastung**
  - Fertigstellung von Aufträgen?  $\rightsquigarrow$  **Antwortzeitanalyse**
  - Wie wirken sich zu wenige Systemprioritäten aus?



## 1 Einplanung

- Gebräuchliche Verfahren
- Statische Prioritäten
- Prioritätsabbildung
- Dynamische Prioritäten

## 2 Optimalität

- RM, DM & EDF
- Ereignisgesteuerte Ablaufplanung

## 3 Planbarkeitsanalyse

- CPU-Auslastung
- Antwortzeitanalyse
- Prioritätsabbildung

## 4 Zusammenfassung



## Statische Prioritäten

- RM** Rate Monotonic (dt. *Ratenmonoton*) Folie 5 ff  
→ Je kürzer die **Periode**, desto höher die Priorität
- DM** Deadline Monotonic (dt. *Fristenmonoton*) Folie 7 ff  
→ Je kürzer der **relative Termin**, desto höher die Priorität



## Statische Prioritäten

- RM** Rate Monotonic (dt. *Ratenmonoton*) Folie 5 ff  
→ Je kürzer die **Periode**, desto höher die Priorität
- DM** Deadline Monotonic (dt. *Fristenmonoton*) Folie 7 ff  
→ Je kürzer der **relative Termin**, desto höher die Priorität

## Dynamische Prioritäten

- EDF** Earliest Deadline First (dt. *Frühester Termin zuerst*) Folie 13 ff  
→ Je früher der **Termin**, desto höher die Priorität
- LRT** Latest Release-Time First (dt. *Späteste Auslösezeit zuerst*)  
→ Je später die **Auslösezeit**, desto höher die Priorität
- LST** Least Slack-Time First (dt. *Kleinste Schlupfzeit zuerst*)  
→ Je kürzer die **Schlupfzeit**, desto höher die Priorität





# Kriterien der Prioritätsvergabe

## Statische Prioritäten

- RM** Rate Monotonic (dt. *Ratenmonoton*) Folie 5 ff  
→ Je kürzer die **Periode**, desto höher die Priorität
- DM** Deadline Monotonic (dt. *Fristenmonoton*) Folie 7 ff  
→ Je kürzer der **relative Termin**, desto höher die Priorität

## Dynamische Prioritäten

- EDF** Earliest Deadline First (dt. *Frühester Termin zuerst*) Folie 13 ff  
→ Je früher der **Termin**, desto höher die Priorität
- LRT** Latest Release-Time First (dt. *Späteste Auslösezeit zuerst*)   
→ Je später die Auslösezeit, desto höher die Priorität  
→ **Eigenstudium**
- LST** Least Slack-Time First (dt. *Kleinste Schlupfzeit zuerst*)   
→ Je kürzer die Schlupfzeit, desto höher die Priorität  
→ **Eigenstudium**





## Einplanung gemäß Ausführungsrate

Rate  $1/p_i$  einer Aufgabe  $T_i$  ist die Inverse ihrer Periode  $p_i$

- Bezogen auf die Auslöserate von Arbeitsaufträgen in  $T_i$

→ Je kleiner die Periode, desto höher die **Priorität**  $P_i$  von  $T_i$

- **Beispiel:**  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$ 
  - Perioden  $p = \{4, 5, 20\}$ , Ausführungszeiten  $e = \{1, 2, 5\}$
  - ⚠ Termin und Phase optional bei  $D_i = p_i$  und  $\phi_i = 0$





## Einplanung gemäß Ausführungsrate

Rate  $1/p_i$  einer Aufgabe  $T_i$  ist die Inverse ihrer Periode  $p_i$

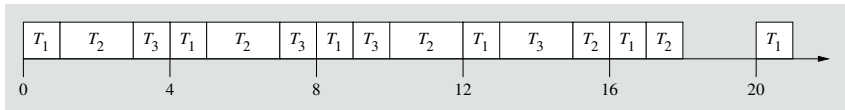
- Bezogen auf die Auslöserate von Arbeitsaufträgen in  $T_i$

→ Je kleiner die Periode, desto höher die **Priorität**  $P_i$  von  $T_i$

- **Beispiel:**  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$ 
  - Perioden  $p = \{4, 5, 20\}$ , Ausführungszeiten  $e = \{1, 2, 5\}$

⚠ Termin und Phase optional bei  $D_i = p_i$  und  $\phi_i = 0$

- **Ablaufplan:**







## Einplanung gemäß Ausführungsrate

Rate  $1/p_i$  einer Aufgabe  $T_i$  ist die Inverse ihrer Periode  $p_i$

- Bezogen auf die Auslöserate von Arbeitsaufträgen in  $T_i$

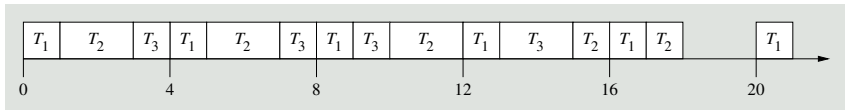
→ Je kleiner die Periode, desto höher die **Priorität**  $P_i$  von  $T_i$

- **Beispiel:**  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$

- Perioden  $p = \{4, 5, 20\}$ , Ausführungszeiten  $e = \{1, 2, 5\}$

⚠ Termin und Phase optional bei  $D_i = p_i$  und  $\phi_i = 0$

- **Ablaufplan:**

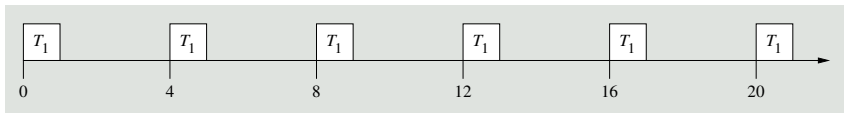


Arbeitsaufträge werden in ihren Aufgabenperioden ausgeführt

→ RM lässt Prozessor bei ausführerbereiten Aufträgen nicht untätig



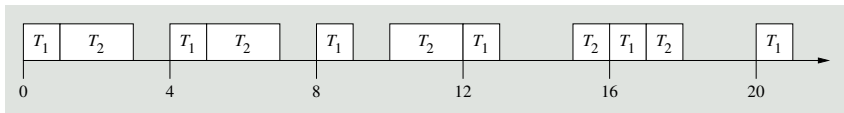
Beispiel:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$



- $T_1$  hat die höchste Rate  $\mapsto$  höchste Priorität  $P_1^{hi}$ 
  - Alle Arbeitsaufträge  $J_{i,j}$  von  $T_1$  werden ausgelöst



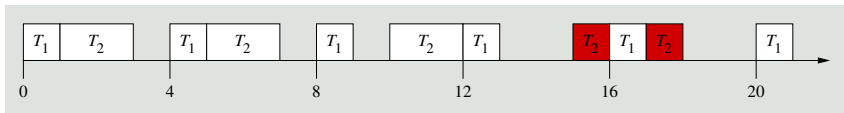
Beispiel:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$



- $T_1$  hat die höchste Rate  $\mapsto$  höchste Priorität  $P_1^{hi}$ 
  - Alle Arbeitsaufträge  $J_{i,j}$  von  $T_1$  werden ausgelöst
- $T_2$  hat die zweithöchste Priorität  $P_2^{med}$  und folgt  $T_1$ 
  - Arbeitsaufträge von  $T_2$  laufen im Hintergrund von  $T_1$
  - $T_2$  startet nach dem ersten Durchlauf von  $T_1$

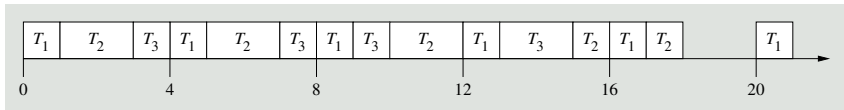


Beispiel:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$



- $T_1$  hat die höchste Rate  $\mapsto$  höchste Priorität  $P_1^{hi}$ 
  - Alle Arbeitsaufträge  $J_{i,j}$  von  $T_1$  werden ausgelöst
- $T_2$  hat die zweithöchste Priorität  $P_2^{med}$  und folgt  $T_1$ 
  - Arbeitsaufträge von  $T_2$  laufen im Hintergrund von  $T_1$
  - $T_2$  startet nach dem ersten Durchlauf von  $T_1$
  - $\rightarrow T_2$  wird zum Zeitpunkt  $t = 16$  von  $T_1$  verdrängt

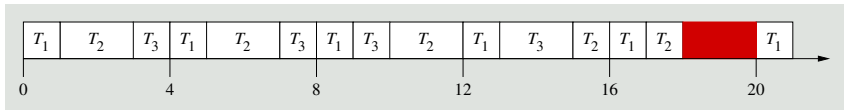
Beispiel:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$



- $T_1$  hat die höchste Rate  $\mapsto$  höchste Priorität  $P_1^{hi}$ 
  - Alle Arbeitsaufträge  $J_{i,j}$  von  $T_1$  werden ausgelöst
- $T_2$  hat die zweithöchste Priorität  $P_2^{med}$  und folgt  $T_1$ 
  - Arbeitsaufträge von  $T_2$  laufen im Hintergrund von  $T_1$
  - $T_2$  startet nach dem ersten Durchlauf von  $T_1$
  - $\rightarrow T_2$  wird zum Zeitpunkt  $t = 16$  von  $T_1$  verdrängt
- $T_3$  hat die dritthöchste Priorität  $P_3^{lo}$  und folgt  $T_2$ 
  - Aufträge von  $T_3$  laufen im Hintergrund von  $T_1$  und  $T_2$
  - $\rightarrow T_3$  läuft nur, wenn kein Auftrag von  $T_1$  und  $T_2$  ausführbereit ist



Beispiel:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$



- 1**  $T_1$  hat die höchste Rate  $\mapsto$  höchste Priorität  $P_1^{hi}$ 
  - Alle Arbeitsaufträge  $J_{i,j}$  von  $T_1$  werden ausgelöst
- 2**  $T_2$  hat die zweithöchste Priorität  $P_2^{med}$  und folgt  $T_1$ 
  - Arbeitsaufträge von  $T_2$  laufen im Hintergrund von  $T_1$
  - $T_2$  startet nach dem ersten Durchlauf von  $T_1$
  - $T_2$  wird zum Zeitpunkt  $t = 16$  von  $T_1$  verdrängt
- 3**  $T_3$  hat die dritthöchste Priorität  $P_3^{lo}$  und folgt  $T_2$ 
  - Aufträge von  $T_3$  laufen im Hintergrund von  $T_1$  und  $T_2$
  - $T_3$  läuft nur, wenn kein Auftrag von  $T_1$  und  $T_2$  ausführbar ist
- 4** **Untätigkeit** im Zeitintervall  $[18, 19]$ 
  - Keine ausführbaren Arbeitsaufträge



## Einplanung gemäß Termin

DM = RM wenn gilt:  $D_i = p_i$

- Beispiel Folie IV-2/5:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$ 
  - Entspricht  $T_1 = (4, 1, 4)$ ,  $T_2 = (5, 2, 5)$ ,  $T_3 = (20, 5, 20)$
  - Relativer Termin und Periode jeder Aufgabe sind identisch



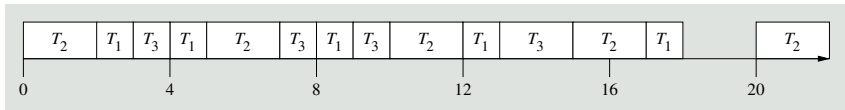
## Einplanung gemäß Termin

DM = RM wenn gilt:  $D_i = p_i$

- Beispiel Folie IV-2/5:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$ 
  - Entspricht  $T_1 = (4, 1, 4)$ ,  $T_2 = (5, 2, 5)$ ,  $T_3 = (20, 5, 20)$
  - Relativer Termin und Periode jeder Aufgabe sind identisch

- **Beispiel:**  $T_1 = (4, 1)$ ,  $T_2 = (5, 2, 3)$ ,  $T_3 = (20, 5)$ 
  - Perioden  $p_i = \{4, 5, 20\}$ , Ausführungszeiten  $e_i = \{1, 2, 5\}$
  - Relative Termine  $D_i = \{4, 3, 20\}$

- **Ablaufplan:**



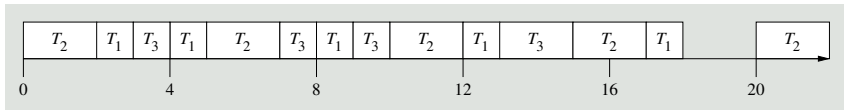


## Einplanung gemäß Termin

DM = RM wenn gilt:  $D_i = p_i$ 

- Beispiel Folie IV-2/5:  $T_1 = (4, 1)$ ,  $T_2 = (5, 2)$ ,  $T_3 = (20, 5)$ 
  - Entspricht  $T_1 = (4, 1, 4)$ ,  $T_2 = (5, 2, 5)$ ,  $T_3 = (20, 5, 20)$
  - Relativer Termin und Periode jeder Aufgabe sind identisch

- **Beispiel:**  $T_1 = (4, 1)$ ,  $T_2 = (5, 2, 3)$ ,  $T_3 = (20, 5)$ 
  - Perioden  $p_i = \{4, 5, 20\}$ , Ausführungszeiten  $e_i = \{1, 2, 5\}$
  - Relative Termine  $D_i = \{4, 3, 20\}$

■ **Ablaufplan:**

Bei beliebigen relativen Terminen arbeitet DM besser als RM

→ DM liefert zulässige Abläufe in Fällen in denen RM scheitert





EZ-Betriebssysteme unterstützen typischerweise nur eine begrenzte Anzahl von **Prioritätsebenen**:

8 im IEEE 802.5 *token ring* [5]

32 im alten QNX, ab Neutrino 256 [4]

140 in Linux 2.5 (mit Ebenen 1–100 reserviert für Echtzeitprozesse)

256 in VxWorks [12] und vielen anderen Echtzeitbetriebssystemen



**Wertebereich ist implementierungsabhängig:** Bitfeld, char





EZ-Betriebssysteme unterstützen typischerweise nur eine begrenzte Anzahl von **Prioritätsebenen**:

8 im IEEE 802.5 *token ring* [5]

32 im alten QNX, ab Neutrino 256 [4]

140 in Linux 2.5 (mit Ebenen 1–100 reserviert für Echtzeitprozesse)

256 in VxWorks [12] und vielen anderen Echtzeitbetriebssystemen



Wertebereich ist **implementierungsabhängig**: Bitfeld, char

## Uneindeutige Prioritäten



Mehr Prioritätsebenen erforderlich, als gegebene Systemplattform unterstützt

- Anzahl unterschiedlicher (eindeutiger) Task-/Jobprioritäten übersteigt die Anzahl unterschiedlicher Prioritäten im System
- Task-/Jobprioritäten lassen sich nicht eindeutig abbilden
- **Uneindeutige Prioritäten** (engl. *nondistinct priorities*)



$\Omega_n$  Anzahl der **logischen Prioritäten** (Aufgaben/Aufträge)

- $1, 2, \dots, \Omega_n$  mit 1 als höchste und  $\Omega_n$  als niedrigste Priorität

$\Omega_s$  Anzahl der **Systemprioritäten**

- $P_1, P_2, \dots, P_{\Omega_s}$  mit  $P_k$  ( $1 \leq k \leq \Omega_s$ ) im Bereich  $[1, \Omega_n]$
- Zusätzlich gilt:  $P_j < P_k$  wenn  $j < k$

- Menge  $\{P_1, P_2, \dots, P_{\Omega_s}\}$  ist **Prioritätsraster**, auf welches die logischen Prioritäten abgebildet werden:
  - Logische Priorität 1 auf  $P_1$
  - Logische Prioritäten im Bereich  $]P_{k-1}, P_k]$  auf  $P_k$  für  $1 < k \leq \Omega_s$



$\Omega_n$  Anzahl der **logischen Prioritäten** (Aufgaben/Aufträge)

- $1, 2, \dots, \Omega_n$  mit 1 als höchste und  $\Omega_n$  als niedrigste Priorität

$\Omega_s$  Anzahl der **Systemprioritäten**

- $P_1, P_2, \dots, P_{\Omega_s}$  mit  $P_k$  ( $1 \leq k \leq \Omega_s$ ) im Bereich  $[1, \Omega_n]$
- Zusätzlich gilt:  $P_j < P_k$  wenn  $j < k$

- Menge  $\{P_1, P_2, \dots, P_{\Omega_s}\}$  ist **Prioritätsraster**, auf welches die logischen Prioritäten abgebildet werden:
  - Logische Priorität 1 auf  $P_1$
  - Logische Prioritäten im Bereich  $]P_{k-1}, P_k]$  auf  $P_k$  für  $1 < k \leq \Omega_s$



Aufträge werden **gemäß ihrer Systempriorität**  $P_k$  abgearbeitet



$\Omega_n$  Anzahl der **logischen Prioritäten** (Aufgaben/Aufträge)

- $1, 2, \dots, \Omega_n$  mit 1 als höchste und  $\Omega_n$  als niedrigste Priorität

$\Omega_s$  Anzahl der **Systemprioritäten**

- $P_1, P_2, \dots, P_{\Omega_s}$  mit  $P_k$  ( $1 \leq k \leq \Omega_s$ ) im Bereich  $[1, \Omega_n]$
- Zusätzlich gilt:  $P_j < P_k$  wenn  $j < k$

- Menge  $\{P_1, P_2, \dots, P_{\Omega_s}\}$  ist **Prioritätsraster**, auf welches die logischen Prioritäten abgebildet werden:
  - Logische Priorität 1 auf  $P_1$
  - Logische Prioritäten im Bereich  $]P_{k-1}, P_k]$  auf  $P_k$  für  $1 < k \leq \Omega_s$



Aufträge werden **gemäß ihrer Systempriorität**  $P_k$  abgearbeitet



Abbildung kann **gleichmäßig** oder **ungleichmäßig** definiert sein





Prioritätsraster **uniform** auf logische Prioritäten legen:

- Sei  $Q$  definiert als Ganzzahl  $\lfloor \Omega_n / \Omega_s \rfloor$ , dann ist die Systempriorität  $P_k = k \cdot Q$  für  $k = 1, 2, \dots, \Omega_s - 1$  und  $P_{\Omega_s} = \Omega_n$





Prioritätsraster **uniform** auf logische Prioritäten legen:

- Sei  $Q$  definiert als Ganzzahl  $\lfloor \Omega_n / \Omega_s \rfloor$ , dann ist die Systempriorität  $P_k = k \cdot Q$  für  $k = 1, 2, \dots, \Omega_s - 1$  und  $P_{\Omega_s} = \Omega_n$
- Für einen Block von max.  $Q$  logischen Prioritäten:
  - Die ersten  $Q$  Tasks/Jobs werden abgebildet auf  $P_1 = Q$
  - Die nächsten  $Q$  Tasks/Jobs werden abgebildet auf  $P_2 = 2Q$
  - Bis alle logischen Prioritäten **gerastert** worden sind





☞ Prioritätsraster **uniform** auf logische Prioritäten legen:

- Sei  $Q$  definiert als Ganzzahl  $\lfloor \Omega_n / \Omega_s \rfloor$ , dann ist die Systempriorität  $P_k = k \cdot Q$  für  $k = 1, 2, \dots, \Omega_s - 1$  und  $P_{\Omega_s} = \Omega_n$

→ Für einen Block von max.  $Q$  logischen Prioritäten:

- Die ersten  $Q$  Tasks/Jobs werden abgebildet auf  $P_1 = Q$
- Die nächsten  $Q$  Tasks/Jobs werden abgebildet auf  $P_2 = 2Q$
- Bis alle logischen Prioritäten **gerastert** worden sind

⚠ Aufgaben verschiedener logischer Prioritäten liegen auf einer Prioritätsebene

- Sie erhalten dieselbe physische Systempriorität
- Aufträge dieser Aufgaben sind einer **linearen Abbildung** unterworfen
- Wichtung erhalten sie durch ihre Position in der linearen Ordnung



# Abbildung durch gleichmäßige Verteilung (Forts.)

Querschneidender Belang von Anwendung und System

- **Beispiel:** Aufgaben mit logischen Prioritäten  $1, 2, \dots, 9$  auf ein System mit Prioritätsebenen  $1, 2, 3$  abbilden ( $Q = 9/3 = 3$ ):

Logische Prioritäten  $\Omega_n$

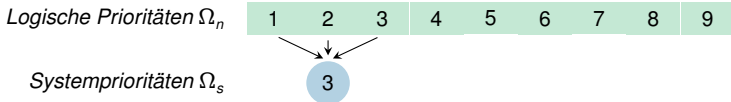
1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



# Abbildung durch gleichmäßige Verteilung (Forts.)

Querschneidender Belang von Anwendung und System

- **Beispiel:** Aufgaben mit logischen Prioritäten  $1, 2, \dots, 9$  auf ein System mit Prioritätsebenen  $1, 2, 3$  abbilden ( $Q = 9/3 = 3$ ):



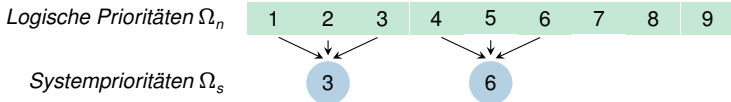
- $[1, 3] \mapsto P_1 = 3$



# Abbildung durch gleichmäßige Verteilung (Forts.)

Querschneidender Belang von Anwendung und System

- **Beispiel:** Aufgaben mit logischen Prioritäten  $1, 2, \dots, 9$  auf ein System mit Prioritätsebenen  $1, 2, 3$  abbilden ( $Q = 9/3 = 3$ ):



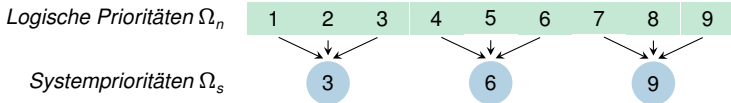
- $[1, 3] \mapsto P_1 = 3$
- $[4, 6] \mapsto P_2 = 6$



# Abbildung durch gleichmäßige Verteilung (Forts.)

Querschneidender Belang von Anwendung und System

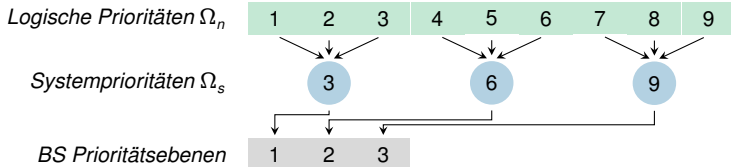
- **Beispiel:** Aufgaben mit logischen Prioritäten  $1, 2, \dots, 9$  auf ein System mit Prioritätsebenen  $1, 2, 3$  abbilden ( $Q = 9/3 = 3$ ):



- $[1, 3] \mapsto P_1 = 3$
- $[4, 6] \mapsto P_2 = 6$
- $[7, 9] \mapsto P_3 = 9$

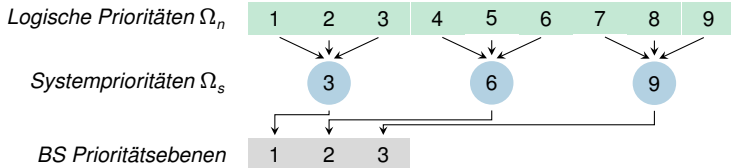


- **Beispiel:** Aufgaben mit logischen Prioritäten 1, 2, ..., 9 auf ein System mit Prioritätsebenen 1, 2, 3 abbilden ( $Q = 9/3 = 3$ ):



- $[1, 3] \mapsto P_1 = 3$
- $[4, 6] \mapsto P_2 = 6$
- $[7, 9] \mapsto P_3 = 9$

- **Beispiel:** Aufgaben mit logischen Prioritäten  $1, 2, \dots, 9$  auf ein System mit Prioritätsebenen  $1, 2, 3$  abbilden ( $Q = 9/3 = 3$ ):



- $[1, 3] \mapsto P_1 = 3$
- $[4, 6] \mapsto P_2 = 6$
- $[7, 9] \mapsto P_3 = 9$

### Problem Fairness:

Aufgaben hoher logischer Priorität werden ggf. gleich behandelt wie solche mit niedrigerer logischer Priorität.

# Abbildung durch ungleichmäßige Verteilung

(engl. *non-uniform mapping*)



Prioritätsraster **ungleichmäßig** auf logischer Prioritäten legen:

- Methode des *constant ratio mapping* [6]
- Ziel: Verhältnis  $(P_{i-1} + 1)/P_i$  für  $i = 2, 3, \dots, \Omega_s$  bleibt gleich
- Hohen logischen Prioritäten werden mehr Prioritätsebenen reserviert
- Bessere Feinabstufung höher priorisierter Aufgaben

Logische Prioritäten  $\Omega_n$

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

- **Beispiel** (vgl. IV-2/11):  $\Omega_n = 9$ ,  $\Omega_s = 3$ , Verhältnis: 1/2





# Abbildung durch ungleichmäßige Verteilung

(engl. *non-uniform mapping*)



Prioritätsraster **ungleichmäßig** auf logischer Prioritäten legen:

- Methode des *constant ratio mapping* [6]
  - Ziel: Verhältnis  $(P_{i-1} + 1)/P_i$  für  $i = 2, 3, \dots, \Omega_s$  bleibt gleich
  - Hohen logischen Prioritäten werden mehr Prioritätsebenen reserviert
- Bessere Feinabstufung höher priorisierter Aufgaben

Logische Prioritäten  $\Omega_n$



Systemprioritäten  $\Omega_s$



- **Beispiel** (vgl. IV-2/11):  $\Omega_n = 9$ ,  $\Omega_s = 3$ , Verhältnis: 1/2
- $P_1$  wird direkt abgebildet



# Abbildung durch ungleichmäßige Verteilung

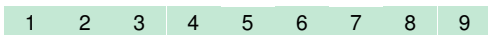
(engl. *non-uniform mapping*)



Prioritätsraster **ungleichmäßig** auf logischer Prioritäten legen:

- Methode des *constant ratio mapping* [6]
- Ziel: Verhältnis  $(P_{i-1} + 1)/P_i$  für  $i = 2, 3, \dots, \Omega_s$  bleibt gleich
- Hohen logischen Prioritäten werden mehr Prioritätsebenen reserviert
- Bessere Feinabstufung höher priorisierter Aufgaben

Logische Prioritäten  $\Omega_n$



Systemprioritäten  $\Omega_s$



- **Beispiel** (vgl. IV-2/11):  $\Omega_n = 9$ ,  $\Omega_s = 3$ , Verhältnis:  $1/2$

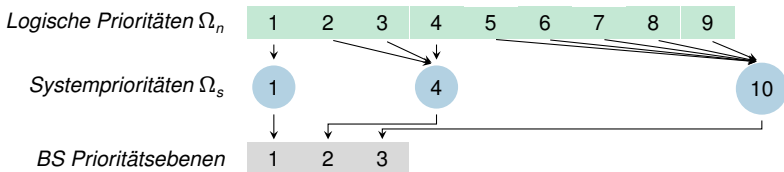
- $P_1$  wird direkt abgebildet
- $(P_1 + 1)/P_2 = 2/4 = 1/2$





Prioritätsraster **ungleichmäßig** auf logischer Prioritäten legen:

- Methode des *constant ratio mapping* [6]
- Ziel: Verhältnis  $(P_{i-1} + 1)/P_i$  für  $i = 2, 3, \dots, \Omega_s$  bleibt gleich
- Hohen logischen Prioritäten werden mehr Prioritätsebenen reserviert
- Bessere Feinabstufung höher priorisierter Aufgaben



■ **Beispiel** (vgl. IV-2/11):  $\Omega_n = 9$ ,  $\Omega_s = 3$ , Verhältnis:  $1/2$

- $P_1$  wird direkt abgebildet
- $(P_1 + 1)/P_2 = 2/4 = 1/2$
- $(P_2 + 1)/P_3 = 5/10 = 1/2$



## Einplanung gemäß absolutem Termin

Ordnet Arbeitsaufträge nach ihrem **absoluten Termin  $d$**

- Je näher der absolute Termin, umso höher die Priorität
- Verschiedene Aufträge derselben Aufgabe mit unterschiedlicher Priorität



## Einplanung gemäß absolutem Termin

Ordnet Arbeitsaufträge nach ihrem **absoluten Termin  $d$**

- Je näher der absolute Termin, umso höher die Priorität
- Verschiedene Aufträge derselben Aufgabe mit unterschiedlicher Priorität

- **Beispiel:**  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$ 
  - Perioden  $p_i = \{4, 5, 20\}$ , Ausführungszeiten  $e_i = \{2, 1, 5\}$
  - Relative Termine  $D_i = \{4, 3, 20\}$



## Einplanung gemäß absolutem Termin

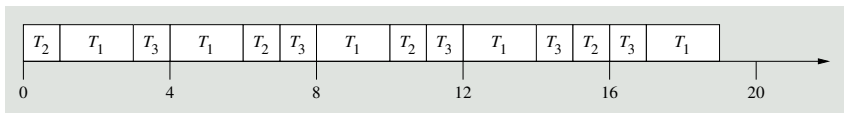
Ordnet Arbeitsaufträge nach ihrem **absoluten Termin  $d$**

- Je näher der absolute Termin, umso höher die Priorität
- Verschiedene Aufträge derselben Aufgabe mit unterschiedlicher Priorität

■ **Beispiel:**  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$

- Perioden  $p_i = \{4, 5, 20\}$ , Ausführungszeiten  $e_i = \{2, 1, 5\}$
- Relative Termine  $D_i = \{4, 3, 20\}$

■ **Ablaufplan:**



## Einplanung gemäß absolutem Termin

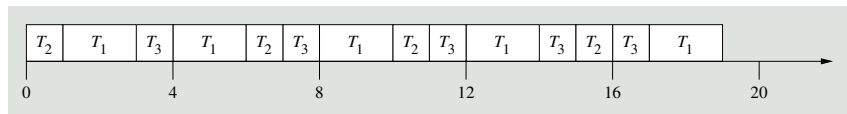
Ordnet Arbeitsaufträge nach ihrem **absoluten Termin  $d$**

- Je näher der absolute Termin, umso höher die Priorität
- Verschiedene Aufträge derselben Aufgabe mit unterschiedlicher Priorität

■ **Beispiel:**  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$

- Perioden  $p_i = \{4, 5, 20\}$ , Ausführungszeiten  $e_i = \{2, 1, 5\}$
- Relative Termine  $D_i = \{4, 3, 20\}$

■ **Ablaufplan:**



Arbeitsaufträge werden möglichst auslösezeitnah gestartet

- Lässt den Prozessor bei ausführerbereiten Aufträgen nicht untätig





## EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$$T_1 = (4, 2)$$

$t_0$  Auslösung,  $d_1 = 4$

$$T_2 = (5, 1, 3)$$

$t_0$  Auslösung,  $d_2 = 3$

$$T_3 = (20, 5)$$

$t_0$  Auslösung,  $d_3 = 20$







## EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$$T_1 = (4, 2)$$

$t_0$  Auslösung,  $d_1 = 4$

$t_1$  Start –  $t_3$  Ende

$$T_2 = (5, 1, 3)$$

$t_0$  Auslösung,  $d_2 = 3$ , Start

$t_1$  Ende

$$T_3 = (20, 5)$$

$t_0$  Auslösung,  $d_3 = 20$

$t_3$  Start





## EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$$T_1 = (4, 2)$$

$t_0$  Auslösung,  $d_1 = 4$

$t_1$  Start –  $t_3$  Ende

$t_4$  Auslösung,  $d_1 = 8$

$$T_2 = (5, 1, 3)$$

$t_0$  Auslösung,  $d_2 = 3$ , Start

$t_1$  Ende

$t_5$  Auslösung,  $d_2 = 8$

$$T_3 = (20, 5)$$

$t_0$  Auslösung,  $d_3 = 20$

$t_3$  Start





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

$t_0$  Auslösung,  $d_1 = 4$

$t_1$  Start –  $t_3$  Ende

$t_4$  Auslösung,  $d_1 = 8$ , Start

$t_6$  Ende

$T_2 = (5, 1, 3)$

$t_0$  Auslösung,  $d_2 = 3$ , Start

$t_1$  Ende

$t_5$  Auslösung,  $d_2 = 8$

$t_6$  Start –  $t_7$  Ende

$T_3 = (20, 5)$

$t_0$  Auslösung,  $d_3 = 20$

$t_3$  Start –  $t_4$  Verdrängung

$t_7$  Fortsetzung





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

$t_0$  Auslösung,  $d_1 = 4$

$t_1$  Start –  $t_3$  Ende

$t_4$  Auslösung,  $d_1 = 8$ , Start

$t_6$  Ende

$t_8$  Auslösung,  $d_1 = 12$

$T_2 = (5, 1, 3)$

$t_0$  Auslösung,  $d_2 = 3$ , Start

$t_1$  Ende

$t_5$  Auslösung,  $d_2 = 8$

$t_6$  Start –  $t_7$  Ende

$t_{10}$  Auslösung,  $d_2 = 13$

$T_3 = (20, 5)$

$t_0$  Auslösung,  $d_3 = 20$

$t_3$  Start –  $t_4$  Verdrängung

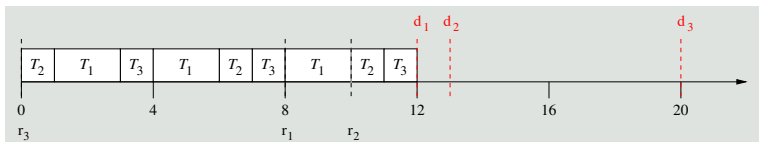
$t_7$  Fortsetzung





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

$t_0$  Auslösung,  $d_1 = 4$

$t_1$  Start –  $t_3$  Ende

$t_4$  Auslösung,  $d_1 = 8$ , Start

$t_6$  Ende

$t_8$  Auslösung,  $d_1 = 12$ , Start

$t_{10}$  Ende

$T_2 = (5, 1, 3)$

$t_0$  Auslösung,  $d_2 = 3$ , Start

$t_1$  Ende

$t_5$  Auslösung,  $d_2 = 8$

$t_6$  Start –  $t_7$  Ende

$t_{10}$  Auslösung,  $d_2 = 13$ , Start

$t_{11}$  Ende

$T_3 = (20, 5)$

$t_0$  Auslösung,  $d_3 = 20$

$t_3$  Start –  $t_4$  Verdrängung

$t_7$  Fortsetzung

$t_8$  Verdrängung

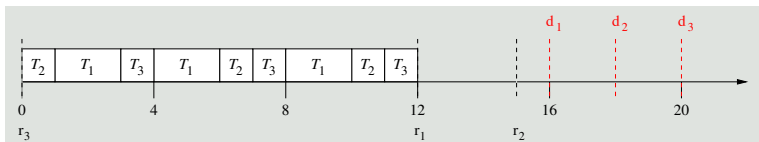
$t_{11}$  Fortsetzung





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

- $t_0$  Auslösung,  $d_1 = 4$
- $t_1$  Start –  $t_3$  Ende
- $t_4$  Auslösung,  $d_1 = 8$ , Start
- $t_6$  Ende
- $t_8$  Auslösung,  $d_1 = 12$ , Start
- $t_{10}$  Ende
- $t_{12}$  Auslösung,  $d_1 = 16$

$T_2 = (5, 1, 3)$

- $t_0$  Auslösung,  $d_2 = 3$ , Start
- $t_1$  Ende
- $t_5$  Auslösung,  $d_2 = 8$
- $t_6$  Start –  $t_7$  Ende
- $t_{10}$  Auslösung,  $d_2 = 13$ , Start
- $t_{11}$  Ende
- $t_{15}$  Auslösung,  $d_2 = 18$

$T_3 = (20, 5)$

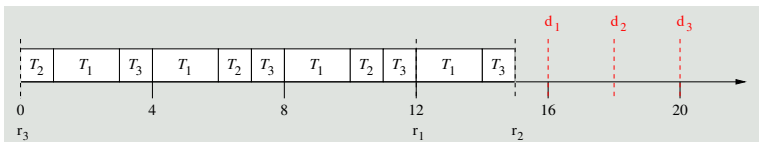
- $t_0$  Auslösung,  $d_3 = 20$
- $t_3$  Start –  $t_4$  Verdrängung
- $t_7$  Fortsetzung
- $t_8$  Verdrängung
- $t_{11}$  Fortsetzung





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

- $t_0$  Auslösung,  $d_1 = 4$
- $t_1$  Start –  $t_3$  Ende
- $t_4$  Auslösung,  $d_1 = 8$ , Start
- $t_6$  Ende
- $t_8$  Auslösung,  $d_1 = 12$ , Start
- $t_{10}$  Ende
- $t_{12}$  Auslösung,  $d_1 = 16$ , Start
- $t_{14}$  Ende

$T_2 = (5, 1, 3)$

- $t_0$  Auslösung,  $d_2 = 3$ , Start
- $t_1$  Ende
- $t_5$  Auslösung,  $d_2 = 8$
- $t_6$  Start –  $t_7$  Ende
- $t_{10}$  Auslösung,  $d_2 = 13$ , Start
- $t_{11}$  Ende
- $t_{15}$  Auslösung,  $d_2 = 18$

$T_3 = (20, 5)$

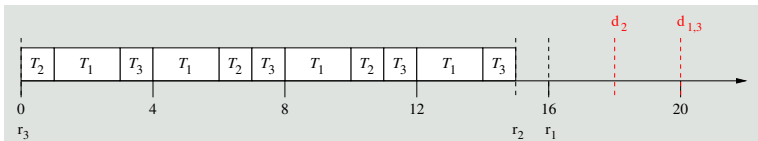
- $t_0$  Auslösung,  $d_3 = 20$
- $t_3$  Start –  $t_4$  Verdrängung
- $t_7$  Fortsetzung
- $t_8$  Verdrängung
- $t_{11}$  Fortsetzung
- $t_{12}$  Verdrängung
- $t_{14}$  Fortsetzung





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

- $t_0$  Auslösung,  $d_1 = 4$
- $t_1$  Start –  $t_3$  Ende
- $t_4$  Auslösung,  $d_1 = 8$ , Start
- $t_6$  Ende
- $t_8$  Auslösung,  $d_1 = 12$ , Start
- $t_{10}$  Ende
- $t_{12}$  Auslösung,  $d_1 = 16$ , Start
- $t_{14}$  Ende
- $t_{16}$  Auslösung,  $d_1 = 20$

$T_2 = (5, 1, 3)$

- $t_0$  Auslösung,  $d_2 = 3$ , Start
- $t_1$  Ende
- $t_5$  Auslösung,  $d_2 = 8$
- $t_6$  Start –  $t_7$  Ende
- $t_{10}$  Auslösung,  $d_2 = 13$ , Start
- $t_{11}$  Ende
- $t_{15}$  Auslösung,  $d_2 = 18$ , Start

$T_3 = (20, 5)$

- $t_0$  Auslösung,  $d_3 = 20$
- $t_3$  Start –  $t_4$  Verdrängung
- $t_7$  Fortsetzung
- $t_8$  Verdrängung
- $t_{11}$  Fortsetzung
- $t_{12}$  Verdrängung
- $t_{14}$  Fortsetzung

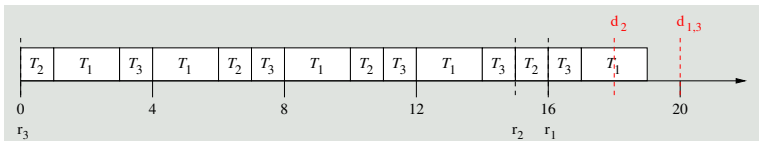






# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

- $t_0$  Auslösung,  $d_1 = 4$
- $t_1$  Start –  $t_3$  Ende
- $t_4$  Auslösung,  $d_1 = 8$ , Start
- $t_6$  Ende
- $t_8$  Auslösung,  $d_1 = 12$ , Start
- $t_{10}$  Ende
- $t_{12}$  Auslösung,  $d_1 = 16$ , Start
- $t_{14}$  Ende
- $t_{16}$  Auslösung,  $d_1 = 20$
- $t_{17}$  Start –  $t_{19}$  Ende

$T_2 = (5, 1, 3)$

- $t_0$  Auslösung,  $d_2 = 3$ , Start
- $t_1$  Ende
- $t_5$  Auslösung,  $d_2 = 8$
- $t_6$  Start –  $t_7$  Ende
- $t_{10}$  Auslösung,  $d_2 = 13$ , Start
- $t_{11}$  Ende
- $t_{15}$  Auslösung,  $d_2 = 18$ , Start
- $t_{16}$  Ende

$T_3 = (20, 5)$

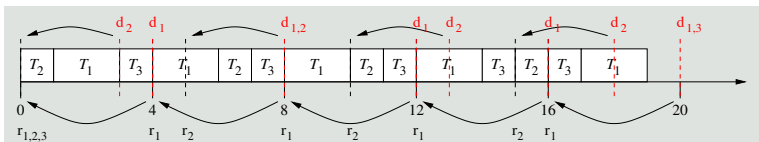
- $t_0$  Auslösung,  $d_3 = 20$
- $t_3$  Start –  $t_4$  Verdrängung
- $t_7$  Fortsetzung
- $t_8$  Verdrängung
- $t_{11}$  Fortsetzung
- $t_{12}$  Verdrängung
- $t_{14}$  Fortsetzung
- $t_{15}$  Verdrängung
- $t_{16}$  Fortsetzung
- $t_{17}$  Ende





# EDF – Frühester Termin zuerst (Forts.)

Beispiel:  $T_1 = (4, 2)$ ,  $T_2 = (5, 1, 3)$ ,  $T_3 = (20, 5)$



$T_1 = (4, 2)$

- $t_0$  Auslösung,  $d_1 = 4$
- $t_1$  Start –  $t_3$  Ende
- $t_4$  Auslösung,  $d_1 = 8$ , Start
- $t_6$  Ende
- $t_8$  Auslösung,  $d_1 = 12$ , Start
- $t_{10}$  Ende
- $t_{12}$  Auslösung,  $d_1 = 16$ , Start
- $t_{14}$  Ende
- $t_{16}$  Auslösung,  $d_1 = 20$
- $t_{17}$  Start –  $t_{19}$  Ende

$T_2 = (5, 1, 3)$

- $t_0$  Auslösung,  $d_2 = 3$ , Start
- $t_1$  Ende
- $t_5$  Auslösung,  $d_2 = 8$
- $t_6$  Start –  $t_7$  Ende
- $t_{10}$  Auslösung,  $d_2 = 13$ , Start
- $t_{11}$  Ende
- $t_{15}$  Auslösung,  $d_2 = 18$ , Start
- $t_{16}$  Ende

$T_3 = (20, 5)$

- $t_0$  Auslösung,  $d_3 = 20$
- $t_3$  Start –  $t_4$  Verdrängung
- $t_7$  Fortsetzung
- $t_8$  Verdrängung
- $t_{11}$  Fortsetzung
- $t_{12}$  Verdrängung
- $t_{14}$  Fortsetzung
- $t_{15}$  Verdrängung
- $t_{16}$  Fortsetzung
- $t_{17}$  Ende



- 1 Einplanung
  - Gebräuchliche Verfahren
  - Statische Prioritäten
  - Prioritätsabbildung
  - Dynamische Prioritäten
- 2 Optimalität
  - RM, DM & EDF
  - Ereignisgesteuerte Ablaufplanung
- 3 Planbarkeitsanalyse
  - CPU-Auslastung
  - Antwortzeitanalyse
  - Prioritätsabbildung
- 4 Zusammenfassung





## Rahmenbedingungen

Der RM-Algorithmus ist **optimal** für Systeme unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind in **Phase** (engl. *synchron*) (d.h.  $\phi_i = 0$ )
- Alle Perioden sind harmonisch  $\mapsto$  **Einfach periodisches** (engl. *simple periodic*) Aufgabensystem (d.h.  $p_i = k \cdot p_1$ )





## Rahmenbedingungen

Der RM-Algorithmus ist **optimal** für Systeme unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind in **Phase** (engl. *synchron*) (d.h.  $\phi_i = 0$ )
- Alle Perioden sind harmonisch  $\mapsto$  **Einfach periodisches** (engl. *simple periodic*) Aufgabensystem (d.h.  $p_i = k \cdot p_1$ )



Die Planbarkeit ist unabhängig von der Zahl der Aufgaben bis zu einer CPU-Auslastung (s. Folie 26) von 100 % garantiert

---

<sup>1</sup>Die Prioritäten zweier Aufgaben  $T_1$  und  $T_2$ , die das RM-Schema verletzen (für die also  $T_1 > T_2$  gilt, obwohl  $p_1 > p_2$ ), lassen sich tauschen ohne dabei die Zulässigkeit des Systems zu zerstören.





## Rahmenbedingungen

Der RM-Algorithmus ist **optimal** für Systeme unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind in **Phase** (engl. *synchron*) (d.h.  $\phi_i = 0$ )
- Alle Perioden sind harmonisch  $\mapsto$  **Einfach periodisches** (engl. *simple periodic*) Aufgabensystem (d.h.  $p_i = k \cdot p_1$ )



Die Planbarkeit ist unabhängig von der Zahl der Aufgaben bis zu einer CPU-Auslastung (s. Folie 26) von 100 % garantiert

- **Beweisidee** (Baruah [1])
  - Gegeben sei ein System mit den Aufgaben  $\{T_1, T_2, T_3, \dots, T_n\}$
  - Prioritäten  $T_1 \succ T_2 \succ \dots \succ T_n$  (nicht RM-konform)
  - Erzeuge einen zulässigen Ablaufplan
  - Prioritäten können hinsichtlich RM umgeformt werden<sup>1</sup> ohne die Zulässigkeit des Ablaufplans zu zerstören

<sup>1</sup>Die Prioritäten zweier Aufgaben  $T_1$  und  $T_2$ , die das RM-Schema verletzen (für die also  $T_1 \succ T_2$  gilt, obwohl  $p_1 > p_2$ ), lassen sich tauschen ohne dabei die Zulässigkeit des Systems zu zerstören.





# Nichtoptimalität des RM-Algorithmus

## Rahmenbedingungen

Der RM-Algorithmus ist nicht optimal unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind **phasenversetzt** (engl. *asynchron*) (d.h.  $\exists \phi_i > 0$ )
- Perioden **nicht harmonisch**  $\mapsto$  **Komplex periodisch** (engl. *complex periodic*)





# Nichtoptimalität des RM-Algorithmus

## Rahmenbedingungen

Der RM-Algorithmus ist **nicht optimal** unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind **phasenversetzt** (engl. *asynchron*) (d.h.  $\exists \phi_i > 0$ )
- Perioden **nicht harmonisch**  $\mapsto$  **Komplex periodisch** (engl. *complex periodic*)

## ■ Beweis (Baruah [1])

- Betrachte  $T_1 = (10, 7, 10, 0)$ ,  $T_2 = (15, 3, 15, 4)$ ,  $T_3 = (16, 1, 16, 0)$
- RM:  $T_1 \succ T_2 \succ T_3$







# Nichtoptimalität des RM-Algorithmus

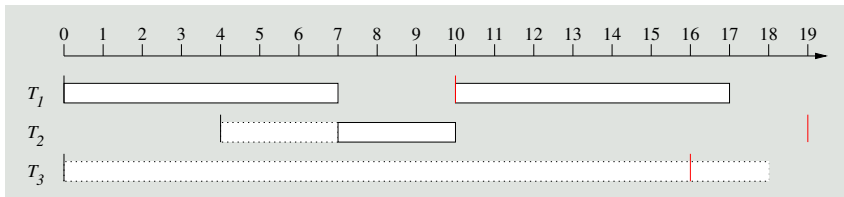
## Rahmenbedingungen

Der RM-Algorithmus ist **nicht optimal** unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind **phasenversetzt** (engl. *asynchron*) (d.h.  $\exists \phi_i > 0$ )
- Perioden **nicht harmonisch**  $\mapsto$  **Komplex periodisch** (engl. *complex periodic*)

## ■ Beweis (Baruah [1])

- Betrachte  $T_1 = (10, 7, 10, 0)$ ,  $T_2 = (15, 3, 15, 4)$ ,  $T_3 = (16, 1, 16, 0)$
- RM:  $T_1 \succ T_2 \succ T_3$



- $T_3$  verpasst bei  $t_{16}$  seinen Termin
- $T_1 \succ T_3 \succ T_2$  würde funktionieren





# Nichtoptimalität des RM-Algorithmus

## Rahmenbedingungen

Der RM-Algorithmus ist **nicht optimal** unter den Bedingungen:

- Voraussetzungen **A1 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)
- Aufgaben sind **phasenversetzt** (engl. *asynchron*) (d.h.  $\exists \phi_i > 0$ )
- Perioden **nicht harmonisch**  $\mapsto$  **Komplex periodisch** (engl. *complex periodic*)

## ■ Beweis (Baruah [1])


- Betrachte  $T_1 = (10, 7, 10, 0)$ ,  $T_2 = (15, 3, 15, 4)$ ,  $T_3 = (16, 1, 16, 0)$
- RM:  $T_1 \succ T_2 \succ T_3$
- $T_3$  verpasst bei  $t_{16}$  seinen Termin
- $T_1 \succ T_3 \succ T_2$  würde funktionieren



Hinreichende Planbarkeitsbedingung ist nunmehr nur bis zu einer Prozessorauslastung von  $\ln(2) \approx 69,3\%$  gegeben

- CPU-Auslastung für n Aufgaben:  $u = \sum_{i=1}^n \frac{e_i}{p_i} \leq n \cdot (\sqrt{2} - 1)$

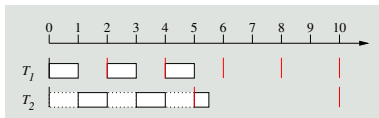


-  **Beispiel:** Betrachte  $T_1 = (2, 1)$  und  $T_2 = (5, 2.5)$   
→ Sei  $T_1 \succ T_2$  (RM-konform)

# Nichtoptimalität statischer Prioritäten

☞ **Beispiel:** Betrachte  $T_1 = (2, 1)$  und  $T_2 = (5, 2.5)$

→ Sei  $T_1 \succ T_2$  (RM-konform)



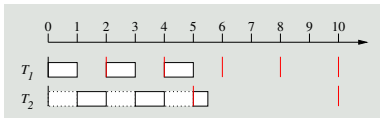
$t_5$   $T_2$  verpasst Termin



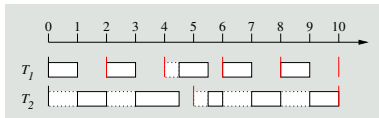
# Nichtoptimalität statischer Prioritäten

☞ **Beispiel:** Betrachte  $T_1 = (2, 1)$  und  $T_2 = (5, 2.5)$

→ Sei  $T_1 \succ T_2$  (RM-konform)



$t_5$   $T_2$  verpasst Termin



$t_4$   $T_2 \succ T_1$

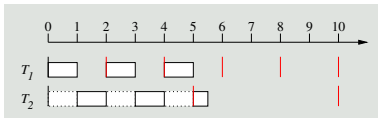
$t_{10}$  Hyperperiode



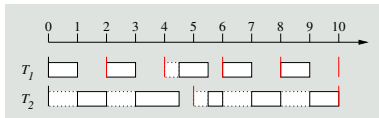
# Nichtoptimalität statischer Prioritäten

☞ **Beispiel:** Betrachte  $T_1 = (2, 1)$  und  $T_2 = (5, 2.5)$

→ Sei  $T_1 \succ T_2$  (RM-konform)



$t_5$   $T_2$  verpasst Termin



$t_4$   $T_2 \succ T_1$

$t_{10}$  Hyperperiode

- Vor dem Zeitpunkt  $t_4$  müsste gelten  $T_1 \succ T_2$
- Zum Zeitpunkt  $t_4$  müsste gelten  $T_2 \succ T_1$

⚠ **Widerspruch zur statischen Vergabe von Prioritäten**



## Rahmenbedingungen

Der EDF-Algorithmus ist **optimal** unter den Bedingungen:

- Aufgaben haben beliebige Auslösezeiten
  - Sporadisch/periodisch
  - Synchron/asynchron
- Aufgaben besitzen beliebige Termine
  - Länger oder kürzer als die entsprechende Periode
- Voraussetzungen **A2** und **A4 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)



## Rahmenbedingungen

Der EDF-Algorithmus ist **optimal** unter den Bedingungen:

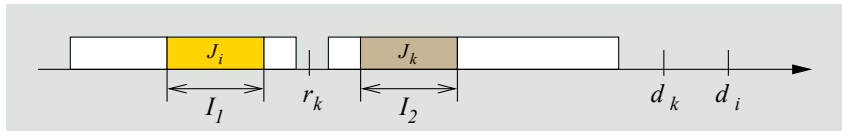
- Aufgaben haben beliebige Auslösezeiten
  - Sporadisch/periodisch
  - Synchron/asynchron
- Aufgaben besitzen beliebige Termine
  - Länger oder kürzer als die entsprechende Periode
- Voraussetzungen **A2** und **A4 - A7** sind erfüllt (siehe Folie IV-1/9 bzw. 43)

■ **Beweis** siehe Liu [8, S.67 f]:

→ Jeder **zulässige** Ablaufplan für solche Systeme lässt sich in einen EDF-Ablaufplan umformen



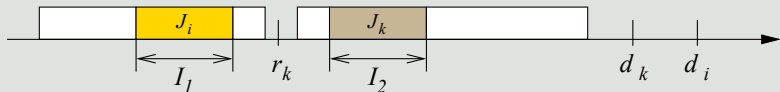




## Beispiel einer Umformung eines Ablaufplans

- Betrachte alle Paare von Arbeitsaufträgen  $J_i$  und  $J_k$
- Arbeitsauftrag  $J_i$  wird im Intervall  $I_1$ ,  $J_k$  im Intervall  $I_2$  eingeplant
- Der Termin von  $J_k$  sei vor dem Termin von  $J_i$ :  $d_k < d_i$
- Das Intervall  $I_1$  liegt komplett vor  $I_2$ :  $I_1 < I_2$





## Beispiel einer Umformung eines Ablaufplans

- Betrachte alle Paare von Arbeitsaufträgen  $J_i$  und  $J_k$
- Arbeitsauftrag  $J_i$  wird im Intervall  $I_1$ ,  $J_k$  im Intervall  $I_2$  eingeplant
- Der Termin von  $J_k$  sei vor dem Termin von  $J_i$ :  $d_k < d_i$
- Das Intervall  $I_1$  liegt komplett vor  $I_2$ :  $I_1 < I_2$

### Fall 1: $r_k \geq \text{Ende}(I_1)$

- $J_k$  kann nicht in  $I_1$  eingeplant werden
- Der Ablaufplan hat bereits EDF-Form

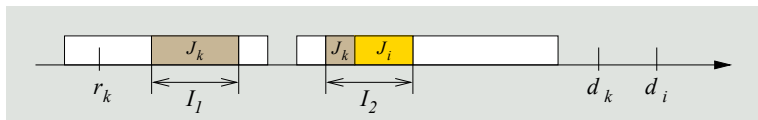


Fall 2:  $r_k < \text{Ende}(I_1)$

(ohne Beschränkung der Allgemeinheit:  $r_k \leq \text{Anfang}(I_1)$ )

1 Tausche  $J_i$  und  $J_k$

Fall 2a:  $l_1 < l_2$   $J_k$  passend stückeln (Verdrängung!),  $l_1 > l_2$  analog

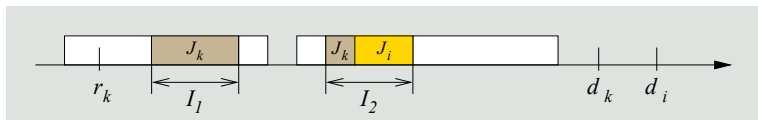


Fall 2:  $r_k < \text{Ende}(I_1)$

(ohne Beschränkung der Allgemeinheit:  $r_k \leq \text{Anfang}(I_1)$ )

1 Tausche  $J_i$  und  $J_k$

Fall 2a:  $l_1 < l_2$   $J_k$  passend stückeln (Verdrängung!),  $l_1 > l_2$  analog



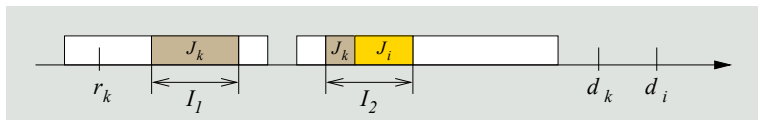
Fall 2b:  $l_1 = l_2$  trivial

Fall 2:  $r_k < \text{Ende}(I_1)$

(ohne Beschränkung der Allgemeinheit:  $r_k \leq \text{Anfang}(I_1)$ )

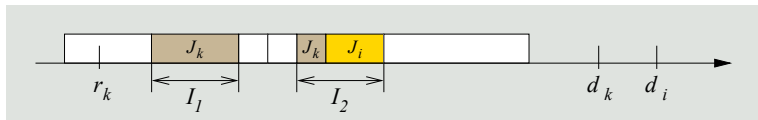
1 Tausche  $J_i$  und  $J_k$

Fall 2a:  $l_1 < l_2$   $J_k$  passend stückeln (Verdrängung!),  $l_1 > l_2$  analog



Fall 2b:  $l_1 = l_2$  trivial

2 Verbliebene Ruheintervalle durch Verschiebung auffüllen



Beliebige (in diesem Fall nicht-verdrängbare) Aufgaben

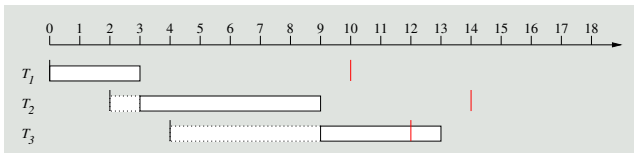
- Beispiel:  $T_1 = (10, 3, 10, 0)$ ,  $T_2 = (14, 6, 12, 2)$ ,  $T_3 = (12, 4, 8, 4)$

---

<sup>2</sup>Dies ist nicht abwegig, da beispielsweise in Mehrkernsystemen die Verdrängbarkeit von Tasks auf unterschiedlichen Kernen i.A. nicht gegeben ist!



- Beispiel:  $T_1 = (10, 3, 10, 0)$ ,  $T_2 = (14, 6, 12, 2)$ ,  $T_3 = (12, 4, 8, 4)$

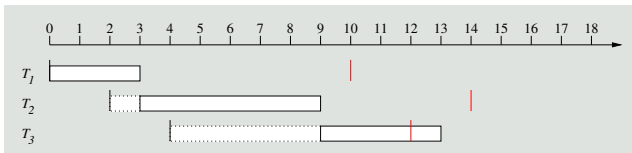


⚠ EDF versagt bei diesem System (Achtung: A7 außer Kraft<sup>2</sup>)

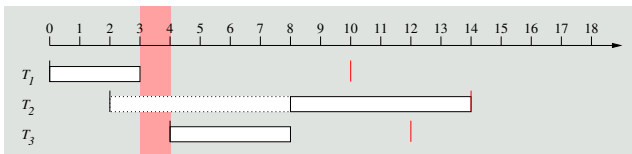
<sup>2</sup>Dies ist nicht abwegig, da beispielsweise in Mehrkernsystemen die Verdrängbarkeit von Tasks auf unterschiedlichen Kernen i.A. nicht gegeben ist!

Beliebige (in diesem Fall nicht-verdrängbare) Aufgaben

- Beispiel:  $T_1 = (10, 3, 10, 0)$ ,  $T_2 = (14, 6, 12, 2)$ ,  $T_3 = (12, 4, 8, 4)$



⚠ EDF versagt bei diesem System (**Achtung: A7 außer Kraft<sup>2</sup>**)



👉 Obwohl ein zulässiger Ablaufplan existiert

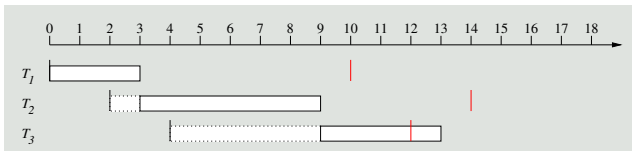
<sup>2</sup>Dies ist nicht abwegig, da beispielsweise in Mehrkernsystemen die Verdrängbarkeit von Tasks auf unterschiedlichen Kernen i.A. nicht gegeben ist!



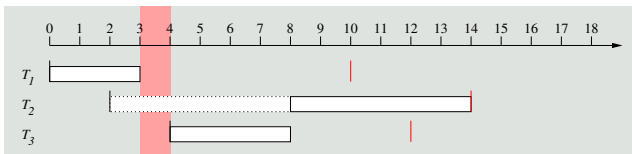


Beliebige (in diesem Fall nicht-verdrängbare) Aufgaben

- Beispiel:  $T_1 = (10, 3, 10, 0)$ ,  $T_2 = (14, 6, 12, 2)$ ,  $T_3 = (12, 4, 8, 4)$



⚠ EDF versagt bei diesem System (**Achtung: A7 außer Kraft<sup>2</sup>**)



Obwohl ein zulässiger Ablaufplan existiert

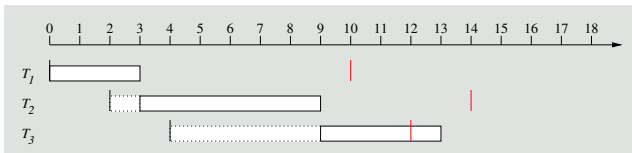
→ Dieser lässt allerdings den Prozessor kurz untätig

<sup>2</sup>Dies ist nicht abwegig, da beispielsweise in Mehrkernsystemen die Verdrängbarkeit von Tasks auf unterschiedlichen Kernen i.A. nicht gegeben ist!

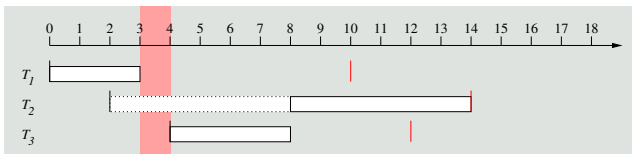


Beliebige (in diesem Fall nicht-verdrängbare) Aufgaben

- Beispiel:  $T_1 = (10, 3, 10, 0)$ ,  $T_2 = (14, 6, 12, 2)$ ,  $T_3 = (12, 4, 8, 4)$



- ⚠ EDF versagt bei diesem System (**Achtung: A7 außer Kraft<sup>2</sup>**)



- 👉 Obwohl ein zulässiger Ablaufplan existiert

→ Dieser lässt allerdings den Prozessor kurz untätig

- ⚠ **Vorranggesteuerte Algorithmen sind stets gefräßig (engl. greedy)!**

<sup>2</sup>Dies ist nicht abwegig, da beispielsweise in Mehrkernsystemen die Verdrängbarkeit von Tasks auf unterschiedlichen Kernen i.A. nicht gegeben ist!



- 1 Einplanung
  - Gebräuchliche Verfahren
  - Statische Prioritäten
  - Prioritätsabbildung
  - Dynamische Prioritäten
- 2 Optimalität
  - RM, DM & EDF
  - Ereignisgesteuerte Ablaufplanung
- 3 Planbarkeitsanalyse
  - CPU-Auslastung
  - Antwortzeitanalyse
  - Prioritätsabbildung
- 4 Zusammenfassung



- Gegeben sei eine Menge periodischer Aufgaben  $T_i = (p_i, e_i, D_i, \phi_i)$  mit

$p_i$  der Periode

$e_i$  der maximalen Ausführungszeit

$D_i$  dem relativen Termin

$\phi_i$  der Phase

der jeweiligen Aufgabe.



- Gegeben sei eine Menge periodischer Aufgaben  $T_i = (p_i, e_i, D_i, \phi_i)$  mit

$p_i$  der Periode

$e_i$  der maximalen Ausführungszeit

$D_i$  dem relativen Termin

$\phi_i$  der Phase

der jeweiligen Aufgabe.

## Fragestellung:

Ist diese Menge von Aufgaben **zulässig**?



- CPU-Auslastung (engl. *loading factor*)
  - Zu welchem Prozentsatz wird der Prozessor **maximal** beansprucht?
  - Bevorzugte Methode für **dynamische Prioritäten**



- CPU-Auslastung (engl. *loading factor*)
  - Zu welchem Prozentsatz wird der Prozessor **maximal** beansprucht?
  - Bevorzugte Methode für **dynamische Prioritäten**
  
- Antwortzeitanalyse (engl. *response time analysis*)
  - Wie lange benötigt eine Aufgabe **maximal** bis zur Fertigstellung?
  - Präzise Methode für **statische Prioritäten**



- CPU-Auslastung (engl. *loading factor*)
  - Zu welchem Prozentsatz wird der Prozessor **maximal** beansprucht?
  - Bevorzugte Methode für **dynamische Prioritäten**
- Zeitbedarfsanalyse (engl. *processor demand*)  $\leadsto$  **Eigenstudium** 🏠
  - Wieviel Rechenzeit wird innerhalb eines Zeitintervalls benötigt?
  - Neuere Methode für **dynamische Prioritäten**
- Antwortzeitanalyse (engl. *response time analysis*)
  - Wie lange benötigt eine Aufgabe **maximal** bis zur Fertigstellung?
  - Präzise Methode für **statische Prioritäten**





- CPU-Auslastung (engl. *loading factor*)
  - Zu welchem Prozentsatz wird der Prozessor **maximal** beansprucht?
  - Bevorzugte Methode für **dynamische Prioritäten**
- Zeitbedarfsanalyse (engl. *processor demand*)  $\leadsto$  **Eigenstudium** 🏠
  - Wieviel Rechenzeit wird innerhalb eines Zeitintervalls benötigt?
  - Neuere Methode für **dynamische Prioritäten**
- Antwortzeitanalyse (engl. *response time analysis*)
  - Wie lange benötigt eine Aufgabe **maximal** bis zur Fertigstellung?
  - Präzise Methode für **statische Prioritäten**
- Simulation (engl. *simulation*)  $\leadsto$  **Eigenstudium** 🏠
  - Wird in einem bestimmten Intervall eine Deadline verfehlt?
  - Bevorzugte Methode für **statische Prioritäten**



## Bestimmung der CPU-Auslastung

Die CPU-Auslastung  $u_{[t_1, t_2[}$  einer Menge von Arbeitsaufträgen während eines Intervalls  $[t_1, t_2[$ , ist der Anteil des Rechenzeitbedarfs  $h_{[t_1, t_2[}$ , der nötig ist, um diese Arbeitsaufträge auszuführen:

$$u_{[t_1, t_2[} = \frac{h_{[t_1, t_2[}}{t_2 - t_1}$$



# CPU-Auslastung (engl. utilisation)

## Bestimmung der CPU-Auslastung

Die CPU-Auslastung  $u_{[t_1, t_2[}$  einer Menge von Arbeitsaufträgen während eines Intervalls  $[t_1, t_2[$ , ist der Anteil des Rechenzeitbedarfs  $h_{[t_1, t_2[}$ , der nötig ist, um diese Arbeitsaufträge auszuführen:

$$u_{[t_1, t_2[} = \frac{h_{[t_1, t_2[}}{t_2 - t_1}$$



Für die Zulässigkeit einer Menge von Aufgaben  $T$  ist die maximale CPU-Auslastung (engl. *maximum utilisation*) entscheidend

→ Maximale CPU-Auslastung über alle Intervalle  $[t_1, t_2[$

## Maximale CPU-Auslastung

$$u = \max_{0 \leq t_1 < t_2} u_{[t_1, t_2[}$$



➔ Rechenzeitbedarf einer Aufgabenmenge  $T$  im Zeitintervall  $[t_1, t_2[$ :

Bestimmung des Rechenzeitbedarfs

$$h_{[t_1, t_2[} = \sum_{t_1 \leq r_k, D_k \leq t_2} e_k$$

- Maximale Ausführungszeit aller Arbeitsaufträge, deren
  - Auslösezeitpunkt und
  - absoluter Termininnerhalb dieses Intervalls liegt.



## Zulässigkeitstest von Liu und Layland [7]

Für jede Menge von  $n$  synchronen, periodischen Aufgaben, die den Kriterien A1 - A7 (siehe IV-1/9 bzw. 43) entsprechen, findet der EDF Algorithmus einen zulässigen Ablaufplan, **gdw** für die CPU-Auslastung gilt:

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

- Dies gilt auch für asynchrone Aufgaben (entkräftet A2) [3]
- EDF-Algorithmus ist **optimal** [11] unter folgenden Bedingungen:
  - Aufgaben wie oben
  - Synchron oder asynchron
  - Kriterium:  $U \leq 1$



Analoge Tests existieren auch für RMA und DMA [8, S. 146]





Systeme, die den Bedingungen **A1 - A7** genügen, sind mit **polynomiell**em Aufwand analysierbar.





Systeme, die den Bedingungen **A1 - A7** genügen, sind mit **polynomiell**em Aufwand analysierbar.



**Starke Konsequenzen** bei Lockerung dieser Einschränkungen:

- Verzicht auf **A3**  $\leadsto$  **stark NP-hart** (Baruah [?])
  - Termine sind **kürzer** als die Perioden der Aufgaben.
- Verzicht auf **A4**  $\leadsto$  **stark NP-hart** (Richard [10])
  - Aufgaben **legen sich schlafen** (engl. *self-suspension*).
- Verzicht auf **A5**  $\leadsto$  **stark NP-hart** (Mok [9])
  - Der **gegenseitige Ausschluss** wird durch Semaphore gesichert.
- Verzicht auf **A7**  $\leadsto$  **stark NP-hart** (Cai [?])
  - Harmonische, periodische Aufgaben sind **nicht verdrängbar**.





Systeme, die den Bedingungen **A1 - A7** genügen, sind mit **polynomiell**em Aufwand analysierbar.



**Starke Konsequenzen** bei Lockerung dieser Einschränkungen:

- Verzicht auf **A3**  $\leadsto$  **stark NP-hart** (Baruah [?])
  - Termine sind **kürzer** als die Perioden der Aufgaben.
- Verzicht auf **A4**  $\leadsto$  **stark NP-hart** (Richard [10])
  - Aufgaben **legen sich schlafen** (engl. *self-suspension*).
- Verzicht auf **A5**  $\leadsto$  **stark NP-hart** (Mok [9])
  - Der **gegenseitige Ausschluss** wird durch Semaphore gesichert.
- Verzicht auf **A7**  $\leadsto$  **stark NP-hart** (Cai [?])
  - Harmonische, periodische Aufgaben sind **nicht verdrängbar**.



**Dies hat auch Auswirkungen auf die Zulässigkeitstests!**





- $D_i \geq p_i$ : Periodische Aufgaben mit großen Terminen
  - Kriterien von Layland/Liu und Coffman gelten nach wie vor [2]
  - Diese Kriterien sind **notwendig** und **hinreichend**



- $D_i \geq p_i$ : Periodische Aufgaben mit großen Terminen
  - Kriterien von Layland/Liu und Coffman gelten nach wie vor [2]
  - Diese Kriterien sind **notwendig** und **hinreichend**
- $D_i < p_i$ : Aperiodische Aufgaben sind möglich
  - **Hybride** Menge: periodische und aperiodische Aufgaben
  - Diese Kriterium ist **nur hinreichend!**

## Rahmenbedingungen nach Baruah [2]

Für eine hybride Menge von  $n$  Aufgaben  $T$ , findet der EDF-Algorithmus einen zulässigen Ablaufplan, wenn gilt:

$$U = \sum_{i=1}^n \frac{e_i}{\min\{D_i, p_i\}} \leq 1$$

- **Beispiel:**  $T_1 = (4, 3, 4, 0)$ ,  $T_2 = (20, 2, 18, 0)$ ,  $T_3 = (10, 1, 3, 0)$ 
  - $\sum_i \frac{e_i}{\min\{D_i, p_i\}} = \frac{3}{4} + \frac{2}{18} + \frac{1}{3} = \frac{43}{36} > 1$

## Dieser Test ist pessimistisch ...

■ **Beispiel:**  $T_1 = (4, 3, 4, 0)$ ,  $T_2 = (20, 2, 18, 0)$ ,  $T_3 = (10, 1, 3, 0)$

■  $\sum_i \frac{e_i}{\min\{D_i, p_i\}} = \frac{3}{4} + \frac{2}{18} + \frac{1}{3} = \frac{43}{36} > 1$

⚠ System ist laut des Tests (siehe Folie IV-2/30) **nicht zulässig!**



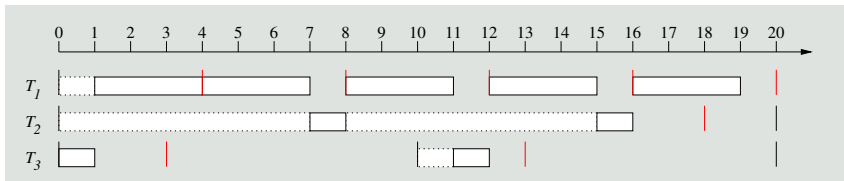
## Dieser Test ist pessimistisch ...

- **Beispiel:**  $T_1 = (4, 3, 4, 0)$ ,  $T_2 = (20, 2, 18, 0)$ ,  $T_3 = (10, 1, 3, 0)$

- $\sum_i \frac{e_i}{\min\{D_i, p_i\}} = \frac{3}{4} + \frac{2}{18} + \frac{1}{3} = \frac{43}{36} > 1$

⚠ System ist laut des Tests (siehe Folie IV-2/30) **nicht zulässig!**

☞ EDF findet jedoch einen zulässigen Ablaufplan:



- Antwortzeit  $\omega_i$ 
  - Zeitdauer zwischen Auslösezeit und Terminationszeitpunkt (siehe III-2/26)
- Idee: Antwortzeitanalyse
  - Terminationszeitpunkt vor dem absoluten Termin  $d_i$
  - Antwortzeit  $\omega_i$  kürzer als der relative Termin  $D_i$
  - Für jeden Auftrag  $J_{i,j}$  in der Aufgabe:  $T_i : \omega_{i,j} \leq D_{i,j}$
- Voraussetzungen
  - Bedingungen A1 - A7 müssen eingehalten werden
  - Konzept ist jedoch erweiterbar



- Antwortzeit  $\omega_i$ 
  - Zeitdauer zwischen Auslösezeit und Terminationszeitpunkt (siehe III-2/26)
- Idee: Antwortzeitanalyse
  - Terminationszeitpunkt vor dem absoluten Termin  $d_i$
  - Antwortzeit  $\omega_i$  kürzer als der relative Termin  $D_i$
  - Für jeden Auftrag  $J_{i,j}$  in der Aufgabe:  $T_i : \omega_{i,j} \leq D_{i,j}$
- Voraussetzungen
  - Bedingungen A1 - A7 müssen eingehalten werden
  - Konzept ist jedoch erweiterbar

## Probleme

- Wie berechnet man die Antwortzeit?
- Wann wird die maximale Antwortzeit erreicht?



- Antwortzeit  $\omega_i$  der Aufgabe  $T_i$  berechnet sich zu:

$$\omega_i(t) = e_i$$

- Aufgabe terminiert bevor das Ereignis (Periode) erneut eintritt
- Setzt sich zusammen aus:
  - WCET  $e_i$  von  $T_i$



- Antwortzeit  $\omega_i$  der Aufgabe  $T_i$  berechnet sich zu:

$$\omega_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k; 0 < t \leq p_i$$

- Aufgabe terminiert bevor das Ereignis (Periode) erneut eintritt
- Setzt sich zusammen aus:
  - WCET  $e_i$  von  $T_i$
  - WCETs  $e_1, \dots, e_{i-1}$  der Aufgaben  $T_1, \dots, T_{i-1}$  höherer Priorität



- Antwortzeit  $\omega_i$  der Aufgabe  $T_i$  berechnet sich zu:

$$\omega_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k; 0 < t \leq p_i$$

- Aufgabe terminiert bevor das Ereignis (Periode) erneut eintritt
- Setzt sich zusammen aus:
  - WCET  $e_i$  von  $T_i$
  - WCETs  $e_1, \dots, e_{i-1}$  der Aufgaben  $T_1, \dots, T_{i-1}$  höherer Priorität
- Prüfung:  $\omega_i(t) \leq t$ 
  - $t = jp_k; \quad k = 1, 2, \dots, i; \quad j = 1, 2, \dots, \lfloor \min(p_i, D_i) / p_k \rfloor$
  - Zeitbedarf erhöht sich nur bei Auslösung dringlicherer Aufgaben
  - Bis das Ereignis erneut eintritt/der Termin der Aufgabe erreicht ist



- Antwortzeit  $\omega_i$  der Aufgabe  $T_i$  berechnet sich zu:

$$\omega_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k; 0 < t \leq p_i$$

- Aufgabe terminiert bevor das Ereignis (Periode) erneut eintritt
- Setzt sich zusammen aus:
  - WCET  $e_i$  von  $T_i$
  - WCETs  $e_1, \dots, e_{i-1}$  der Aufgaben  $T_1, \dots, T_{i-1}$  höherer Priorität
- Prüfung:  $\omega_i(t) \leq t$ 
  - $t = jp_k; k = 1, 2, \dots, i; j = 1, 2, \dots, \lfloor \min(p_i, D_i) / p_k \rfloor$
  - Zeitbedarf erhöht sich nur bei Auslösung dringlicherer Aufgaben
  - Bis das Ereignis erneut eintritt/der Termin der Aufgabe erreicht ist



Ist die Ungleichung für **einen** Zeitpunkt  $t$  erfüllt, ist  $T_i$  **zulässig**



## Beispiel: Berechnung der maximalen Antwortzeit

Aufgaben:  $T_1 = (3, 1, 3, \phi_1)$ ,  $T_2 = (5, 1.5, 5, \phi_2)$ ,  $T_3 = (7, 1.25, 7, \phi_3)$ ,  $T_4 = (9, 0.5, 9, \phi_4)$

- Antwortzeit  $\omega_1$  von  $T_1$ 
  - $\omega_1(3) = 1 \leq 3 \leadsto$  zulässig



## Beispiel: Berechnung der maximalen Antwortzeit

Aufgaben:  $T_1 = (3, 1, 3, \phi_1)$ ,  $T_2 = (5, 1.5, 5, \phi_2)$ ,  $T_3 = (7, 1.25, 7, \phi_3)$ ,  $T_4 = (9, 0.5, 9, \phi_4)$

- Antwortzeit  $\omega_1$  von  $T_1$ 
  - $\omega_1(3) = 1 \leq 3 \leadsto$  zulässig
- Antwortzeit  $\omega_2$  von  $T_2$ 
  - $\omega_2(3) = 1.5 + \lceil \frac{3}{3} \rceil 1 = 2.5 \leq 3 \leadsto$  zulässig



## Beispiel: Berechnung der maximalen Antwortzeit

Aufgaben:  $T_1 = (3, 1, 3, \phi_1)$ ,  $T_2 = (5, 1.5, 5, \phi_2)$ ,  $T_3 = (7, 1.25, 7, \phi_3)$ ,  $T_4 = (9, 0.5, 9, \phi_4)$

- Antwortzeit  $\omega_1$  von  $T_1$ 
  - $\omega_1(3) = 1 \leq 3 \rightsquigarrow$  zulässig
- Antwortzeit  $\omega_2$  von  $T_2$ 
  - $\omega_2(3) = 1.5 + \left\lceil \frac{3}{3} \right\rceil 1 = 2.5 \leq 3 \rightsquigarrow$  zulässig
- Antwortzeit  $\omega_3$  von  $T_3$ 
  - $\omega_3(3) = 1.25 + \left\lceil \frac{3}{3} \right\rceil 1 + \left\lceil \frac{3}{5} \right\rceil 1.5 = 3.75 > 3$
  - $\omega_3(5) = 1.25 + \left\lceil \frac{5}{3} \right\rceil 1 + \left\lceil \frac{5}{5} \right\rceil 1.5 = 4.75 \leq 5 \rightsquigarrow$  zulässig



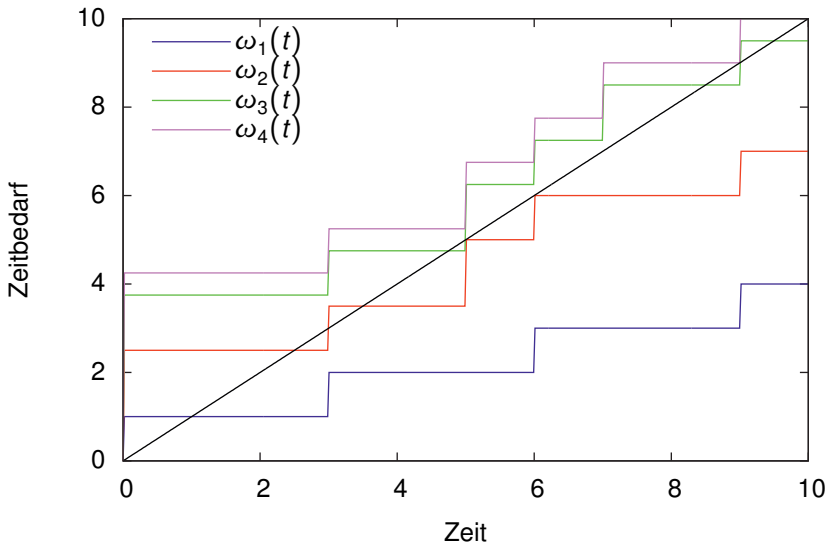
# Beispiel: Berechnung der maximalen Antwortzeit

Aufgaben:  $T_1 = (3, 1, 3, \phi_1)$ ,  $T_2 = (5, 1.5, 5, \phi_2)$ ,  $T_3 = (7, 1.25, 7, \phi_3)$ ,  $T_4 = (9, 0.5, 9, \phi_4)$

- Antwortzeit  $\omega_1$  von  $T_1$ 
  - $\omega_1(3) = 1 \leq 3 \rightsquigarrow$  zulässig
- Antwortzeit  $\omega_2$  von  $T_2$ 
  - $\omega_2(3) = 1.5 + \lceil \frac{3}{3} \rceil 1 = 2.5 \leq 3 \rightsquigarrow$  zulässig
- Antwortzeit  $\omega_3$  von  $T_3$ 
  - $\omega_3(3) = 1.25 + \lceil \frac{3}{3} \rceil 1 + \lceil \frac{3}{5} \rceil 1.5 = 3.75 > 3$
  - $\omega_3(5) = 1.25 + \lceil \frac{5}{3} \rceil 1 + \lceil \frac{5}{5} \rceil 1.5 = 4.75 \leq 5 \rightsquigarrow$  zulässig
- Antwortzeit  $\omega_4$  von  $T_4$ 
  - $\omega_4(3) = 0.5 + \lceil \frac{3}{3} \rceil 1 + \lceil \frac{3}{5} \rceil 1.5 + \lceil \frac{3}{7} \rceil 1.25 = 4.25 > 3$
  - $\omega_4(5) = 0.5 + \lceil \frac{5}{3} \rceil 1 + \lceil \frac{5}{5} \rceil 1.5 + \lceil \frac{5}{7} \rceil 1.25 = 5.25 > 5$
  - $\omega_4(6) = 0.5 + \lceil \frac{6}{3} \rceil 1 + \lceil \frac{6}{5} \rceil 1.5 + \lceil \frac{6}{7} \rceil 1.25 = 6.75 > 6$
  - $\omega_4(7) = 0.5 + \lceil \frac{7}{3} \rceil 1 + \lceil \frac{7}{5} \rceil 1.5 + \lceil \frac{7}{7} \rceil 1.25 = 7.75 > 7$
  - $\omega_4(9) = 0.5 + \lceil \frac{9}{3} \rceil 1 + \lceil \frac{9}{5} \rceil 1.5 + \lceil \frac{9}{7} \rceil 1.25 = 9.00 \leq 9 \rightsquigarrow$  zulässig



# Zeitbedarfsfunktionen der Aufgaben $T_1$ , $T_2$ , $T_3$ und $T_4$







**Kritischer Zeitpunkt** (engl. *critical instant*)  $\mapsto$  **maximale Antwortzeit**

$\rightarrow$  Auslösung eines Arbeitsauftrags an seinem kritischen Zeitpunkt





**Kritischer Zeitpunkt** (engl. *critical instant*)  $\mapsto$  **maximale Antwortzeit**

$\rightarrow$  Auslösung eines Arbeitsauftrags an seinem kritischen Zeitpunkt

■ An seinem kritischen Zeitpunkt ausgelöster Auftrag  $J_{i,j}$  einer Aufgabe  $T_i$ :

$\rightarrow$  Erfährt die **maximale Antwortzeit** aller Aufträge in  $T_i$

– Falls alle Arbeitsaufträge ihre Termine einhalten

$\rightarrow$  **Verpasst seinen Termin**

– Falls irgendein Arbeitsauftrag in  $T_i$  seinen Termin verpasst





**Kritischer Zeitpunkt** (engl. *critical instant*)  $\mapsto$  **maximale Antwortzeit**

$\rightarrow$  Auslösung eines Arbeitsauftrags an seinem kritischen Zeitpunkt

- An seinem kritischen Zeitpunkt ausgelöster Auftrag  $J_{i,j}$  einer Aufgabe  $T_i$ :

$\rightarrow$  Erfährt die **maximale Antwortzeit** aller Aufträge in  $T_i$

- Falls alle Arbeitsaufträge ihre Termine einhalten

$\rightarrow$  **Verpasst seinen Termin**

- Falls irgendein Arbeitsauftrag in  $T_i$  seinen Termin verpasst



**Kritischer Zeitpunkt in Systemen mit statischen Prioritäten** [7]

$\rightarrow$  Falls **zusammen** mit einem Arbeitsauftrag der Aufgabe  $T_i$  Aufträge aller Aufgaben **höherer Priorität**  $T_1, \dots, T_{i-1}$  ausgelöst werden





**Kritischer Zeitpunkt** (engl. *critical instant*)  $\mapsto$  **maximale Antwortzeit**

$\rightarrow$  Auslösung eines Arbeitsauftrags an seinem kritischen Zeitpunkt

- An seinem kritischen Zeitpunkt ausgelöster Auftrag  $J_{i,j}$  einer Aufgabe  $T_i$ :

$\rightarrow$  Erfährt die **maximale Antwortzeit** aller Aufträge in  $T_i$

- Falls alle Arbeitsaufträge ihre Termine einhalten

$\rightarrow$  **Verpasst seinen Termin**

- Falls irgendein Arbeitsauftrag in  $T_i$  seinen Termin verpasst



**Kritischer Zeitpunkt** in Systemen mit **statischen Prioritäten** [7]

$\rightarrow$  Falls **zusammen** mit einem Arbeitsauftrag der Aufgabe  $T_i$  Aufträge aller Aufgaben **höherer Priorität**  $T_1, \dots, T_{i-1}$  ausgelöst werden



**Kritischer Zeitpunkt** in Systemen mit **dynamischen Prioritäten**

- Lässt sich ein solcher kritischer Zeitpunkt **nicht** identifizieren

$\rightarrow$  Antwortzeitanalyse ist hier **ungeeignet**



# Relative Planbarkeit – Prioritätsabbildung

Einfluss der Anzahl von Systemprioritäten auf die Planbarkeit eines Systems



Verschlechterung der Planbarkeit bei wenigen verfügbaren Systemprioritäten  
( $\Omega_n \gg \Omega_s$ ) zu erwarten



# Relative Planbarkeit – Prioritätsabbildung

Einfluss der Anzahl von Systemprioritäten auf die Planbarkeit eines Systems



Verschlechterung der Planbarkeit bei wenigen verfügbaren Systemprioritäten ( $\Omega_n \gg \Omega_s$ ) zu erwarten

- Sei  $g$  das Minimum von Verhältniswerten des Prioritätsrasters (s. IV-2/12)





Verschlechterung der Planbarkeit bei wenigen verfügbaren Systemprioritäten ( $\Omega_n \gg \Omega_s$ ) zu erwarten

- Sei  $g$  das Minimum von Verhältniswerten des Prioritätsrasters (s. IV-2/12)
- Für RM für große  $n$  und  $D_i = p_i$  für alle  $i$  wurde gezeigt [6], dass für die **planbare Auslastung** (engl. *schedulable utilization*) gilt:

$$\begin{aligned} \ln(2g) + 1 - g & \text{ falls } g > 1/2 \\ g & \text{ falls } g \leq 1/2 \end{aligned}$$





Verschlechterung der Planbarkeit bei wenigen verfügbaren Systemprioritäten ( $\Omega_n \gg \Omega_s$ ) zu erwarten

- Sei  $g$  das Minimum von Verhältniswerten des Prioritätsrasters (s. IV-2/12)
- Für RM für große  $n$  und  $D_i = p_i$  für alle  $i$  wurde gezeigt [6], dass für die **planbare Auslastung** (engl. *schedulable utilization*) gilt:  
$$\ln(2g) + 1 - g \quad \text{falls } g > 1/2$$
$$g \quad \text{falls } g \leq 1/2$$
- Das Verhältnis dieser Auslastung zu  $\ln(2)$  ist ein Maß für die **relative Planbarkeit** des gegebenen Systems





# Relative Planbarkeit – Prioritätsabbildung

Einfluss der Anzahl von Systemprioritäten auf die Planbarkeit eines Systems



Verschlechterung der Planbarkeit bei wenigen verfügbaren Systemprioritäten ( $\Omega_n \gg \Omega_s$ ) zu erwarten

- Sei  $g$  das Minimum von Verhältniswerten des Prioritätsrasters (s. IV-2/12)
- Für RM für große  $n$  und  $D_i = p_i$  für alle  $i$  wurde gezeigt [6], dass für die **planbare Auslastung** (engl. *schedulable utilization*) gilt:  
$$\ln(2g) + 1 - g \quad \text{falls } g > 1/2$$
$$g \quad \text{falls } g \leq 1/2$$
- Das Verhältnis dieser Auslastung zu  $\ln(2)$  ist ein Maß für die **relative Planbarkeit** des gegebenen Systems

- **Beispiel:** 100 000 Aufgaben (evtl. noch vielmehr Aufträge), d.h.,  $\Omega_n = 100\,000$ 
  - Die relative Planbarkeit bei  $\Omega_s = 256$  ist gleich 0.9986



# Relative Planbarkeit – Prioritätsabbildung

Einfluss der Anzahl von Systemprioritäten auf die Planbarkeit eines Systems



Verschlechterung der Planbarkeit bei wenigen verfügbaren Systemprioritäten ( $\Omega_n \gg \Omega_s$ ) zu erwarten

- Sei  $g$  das Minimum von Verhältniswerten des Prioritätsrasters (s. IV-2/12)
- Für RM für große  $n$  und  $D_i = p_i$  für alle  $i$  wurde gezeigt [6], dass für die **planbare Auslastung** (engl. *schedulable utilization*) gilt:  
$$\ln(2g) + 1 - g \quad \text{falls } g > 1/2$$
$$g \quad \text{falls } g \leq 1/2$$
- Das Verhältnis dieser Auslastung zu  $\ln(2)$  ist ein Maß für die **relative Planbarkeit** des gegebenen Systems

- **Beispiel:** 100 000 Aufgaben (evtl. noch vielmehr Aufträge), d.h.,  $\Omega_n = 100\,000$ 
  - Die relative Planbarkeit bei  $\Omega_s = 256$  ist gleich 0.9986



**RM:** Für komplexeste Taskssysteme reichen schon **256 Prioritätsebenen**



- 1 Einplanung
  - Gebräuchliche Verfahren
  - Statische Prioritäten
  - Prioritätsabbildung
  - Dynamische Prioritäten
- 2 Optimalität
  - RM, DM & EDF
  - Ereignisgesteuerte Ablaufplanung
- 3 Planbarkeitsanalyse
  - CPU-Auslastung
  - Antwortzeitanalyse
  - Prioritätsabbildung
- 4 Zusammenfassung





**Ablaufplanung** gebräuchliche, ereignisgesteuerte Verfahren

- **statische Prioritäten**  $\rightsquigarrow$  RM, DM
  - **Prioritätsabbildung** im Falle nicht ausreichender Systemprioritäten
- **dynamische Prioritäten**  $\rightsquigarrow$  EDF



**Ablaufplanung** gebräuchliche, ereignisgesteuerte Verfahren

- **statische Prioritäten**  $\rightsquigarrow$  RM, DM
  - **Prioritätsabbildung** im Falle nicht ausreichender Systemprioritäten
- **dynamische Prioritäten**  $\rightsquigarrow$  EDF

**Optimalität und Nichtoptimalität** von RM, DM und EDF

- **Hängt** von den Eigenschaften der betrachteten Aufgaben ab
- **Nichtoptimalität** von statischen Prioritäten und Ereignissteuerung

**Ablaufplanung** gebräuchliche, ereignisgesteuerte Verfahren

- **statische Prioritäten**  $\leadsto$  RM, DM
  - Prioritätsabbildung im Falle nicht ausreichender Systemprioritäten
- **dynamische Prioritäten**  $\leadsto$  EDF

**Optimalität und Nichtoptimalität** von RM, DM und EDF

- Hängt von den Eigenschaften der betrachteten Aufgaben ab
- Nichtoptimalität von statischen Prioritäten und Ereignissteuerung

**Planbarkeitsanalyse** ereignisgesteuerter Ablaufplanungsverfahren

- maximalen, kumulativen CPU-Auslastung und Antwortzeitanalyse
- relative Planbarkeit im Falle nicht ausreichender Systemprioritäten



- [1] *Kapitel 28.*  
In: Baruah, S. ; Goossens, J. :  
*Scheduling Real-time Tasks: Algorithms and Complexity.*  
Chapman & Hall/CRC, 2004 (Computer and Information Science series)
  
- [2] Baruah, S. ; Mok, A. ; Rosier, L. :  
Preemptively scheduling hard-real-time sporadic tasks on one processor.  
(1990), Dez., S. 182–190.  
<http://dx.doi.org/10.1109/REAL.1990.128746>. –  
DOI 10.1109/REAL.1990.128746
  
- [3] Coffman, E. G.:  
*Computer and Job-shop Scheduling Theory.*  
John Wiley & Sons Inc, 1976. –  
ISBN 978–0471163190
  
- [4] Hildebrand, D. :  
An Architectural Overview of QNX.  
In: *Proceedings of the USENIX Workshop on Microkernels and Other Kernel Architectures.*  
Seattle, WA, USA, Apr. 27–28, 1992, S. 113–126
  
- [5] IEEE Standard 802.5:  
*Token Ring Access Method and Physical Layer Specification.*  
IEEE, New York, 1989






- [6] Lehoczky, J. P. ; Sha, L. :  
Performance of Real-Time Bus Scheduling Algorithms.  
In: *ACM Performance Evaluation Review* 14 (1986), Mai, Nr. 1, S. 44–55
- [7] Liu, C. L. ; Layland, J. W. :  
Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment.  
In: *Journal of the ACM* 20 (1973), Nr. 1, S. 46–61.  
<http://dx.doi.org/http://doi.acm.org/10.1145/321738.321743>. –  
DOI <http://doi.acm.org/10.1145/321738.321743>. –  
ISSN 0004–5411
- [8] Liu, J. W. S. :  
*Real-Time Systems*.  
Englewood Cliffs, NJ, USA : Prentice Hall PTR, 2000. –  
ISBN 0–13–099651–3
- [9] Mok, A. K. :  
*Fundamental design problems of distributed systems for the hard real-time environment*, MIT,  
Diss., 1983
- [10] Richard, P. :  
On the complexity of scheduling real-time tasks with self-suspensions on one processor.  
In: *Proceedings. 15th Euromicro Conference on Real-Time Systems (ECRTS 2003)* (2003), Jul.,  
S. 187–194



- [11] Spuri, M. :  
*Earliest Deadline Scheduling in Real-Time Systems*, Scuola Superiore S. Anna, Pisa,  
Dissertation, 1996
- [12] Wind River Systems, Inc.:  
*Wind River Homepage*.  
<http://www.windriver.com>,



 Mathematische Ansätze zur zeitlichen Analyse periodischer Echtzeitsysteme bedingen häufig starke Einschränkungen:

- A1** Alle Aufgaben sind periodisch
- A2** Alle Arbeitsaufträge können an ihren Auslösezeitpunkten eingeplant und ausgeführt werden
- A3** Termine und Perioden sind identisch
- A4** Kein Arbeitsauftrag gibt die Kontrolle über den Prozessor ab
- A5** Alle Aufgaben sind unabhängig<sup>3</sup>
- A6** Die Kosten durch Unterbrechungen, Ablaufplanung und Verdrängung sind vernachlässigbar
- A7** Alle Aufgaben verhalten sich voll-präemptiv

---

<sup>3</sup>D.h. die einzige gemeinsame Ressource ist die CPU und es existieren keine Einschränkungen hinsichtlich der Auslösezeiten der Arbeitsaufträge voneinander.



## Typographische Konvention

Der erste Index gibt die Aufgabe an (z.B.  $D_i$ ), der Zweite (optional) bezieht sich auf den Arbeitsauftrag (z.B.  $d_{i,j}$ ). Exponenten zeigen verschiedene Varianten einer Eigenschaft an (z.B.  $T^{HI}, T^{MED}, T^{LO}$ ). Funktionen beschreiben zeitlich variierende Eigenschaften (z.B.  $P(t)$ ).

## Eigenschaften

$t$  (Real-)Zeit  
 $d$  Zeitverzögerung (engl. delay)

## Strukturelemente

$E_i$  Ereignis (engl. event)  
 $R_i$  Ergebnis (engl. result)  
 $T_i$  Aufgabe (engl. task)  
 $J_{i,j}$  Arbeitsauftrag (engl. job) der Aufgabe  $T_i$

## Temporale Eigenschaften

Allgemein

$r_i$  Auslösezeitpunkt  
(engl. release time)  
 $e_i$  Maximale Ausführungszeit (WCET)  
 $D_i$  Relativer Termin (engl. deadline)  
 $d_i$  Absoluter Termin  
 $\omega_i$  Antwortzeit (engl. response time)  
 $\sigma_i$  Schlupf (engl. slack)

Periodische Aufgaben

$p_i$  Periode (engl. period)  
 $\phi_i$  Phase (engl. phase)

## Aufgaben – Tupel

$T_p = (\rho, e, D, \phi)$  Periodische Aufgabe ohne Priorität (zeitgesteuert oder dynamische Taskpriorität),  $D = p$  und  $\phi = 0$  sind der Reihe nach optional

## Ablaufplanung

$P_i$  Priorität (engl. priority) der Aufgabe  
 $T_i$   
 $\Omega_i$  Prioritätsebenen (engl. number of priorities)  
 $h_{\Delta_i}$  Rechenzeitbedarf (engl. demand)  
 $u_{\Delta_i}$  CPU-Auslastung (engl. utilisation)  
 $U$  Absolute CPU-Auslastung

