Prozessorvergabe - Echtzeitsysteme: Modelle

7 Prozessorvergabe in Echtzeitsystemen

Burns. A.; Wellings, A.: Real-Time Systems and Programming Languages. Addison-Wesley, 2nd Edition, 1990, p. 399 - 440.

7.1 Systemmodelle

❑ Ein einfaches Modell

- Die Anwendung besteht aus einer festen Anzahl von Prozessen.
- Alle Prozesse arbeiten periodisch mit fester, a priori bekannter Periode.
- Die Prozesse sind voneinander vollkommen unabhängig.
- Der Aufwand für Prozeßumschaltungen ist so gering, daß er vernachlässigt werden kann.
- Jeder Prozeß muß innerhalb seiner Periode einen Durchlauf abschließen (d. h. jeder Durchlauf kann eigener Auftrag mit Zielzeit angesehen werden).
- Alle Prozesse besitzen eine a priori bekannte maximale Durchlaufzeit.

◆ Diese Eigenschaften lassen zu, daß alle Prozesse gleichzeitig einen Durchlauf beginnen können. Dieser Fall erzeugt die größte mögliche Last. Ein solcher Zeitpunkt wird als kritischer Moment bezeichnet.

Prozessorvergabe - Echtzeitsysteme: Modelle

❑ Standardnotation für die wichtigsten Prozeßattribute

Notation	Beschreibung
B	Längste auftretende Blockierungszeit
C	Längste auftretende Rechenzeit
D	Zielzeit
I	Maximale Interferenz (durch höherpriore Prozesse)
J	Zulässiger Freigabejitter
N	Zahl der Prozesse im System
P	Priorität (soweit vorhanden)
R	Längste zulässige Verweilzeit
T	Periodenlänge
U	Systembelastung (C / T)

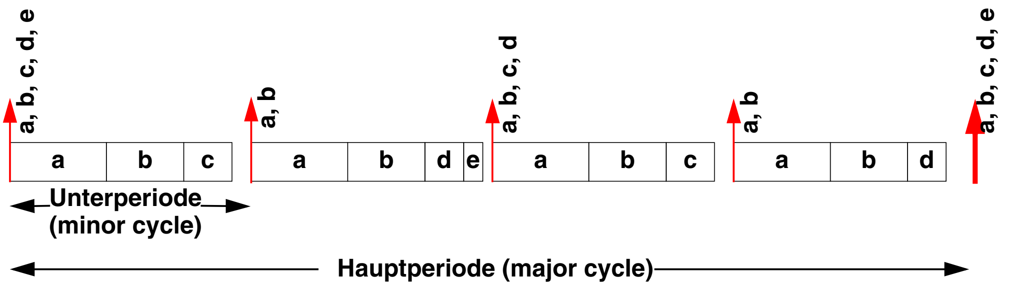
❑ In diesem Kapitel werden Prioritäten durch ganze Zahlen ausgedrückt, wobei größere Werte höhere Priorität bezeichnen!

Vorsicht: Die Literatur ist in dieser Hinsicht uneinheitlich!

7.2 Zyklische Ausführungspläne

- Beispiel zur Vorgehensweise

Prozeß	Periode	Rechenzeit
a	25	10
b	25	8
c	50	5
d	50	4
e	100	2




```
while (true) {
   Procedure_for_a;
   Procedure_for_b;
   Procedure_for_c_1;
   Wait_for_interrupt;

   Procedure_for_a;
   Procedure_for_b;
   Procedure_for_d_1;
   Procedure_for_e;
   Wait_for_interrupt;

   Procedure_for_a;
   Procedure_for_b;
   Procedure_for_c_2;
   Wait_for_interrupt;

   Procedure_for_a;
   Procedure_for_b;
   Procedure_for_d_2;
   Wait_for_interrupt;
}
```

Prozessorvergabe - Echtzeitsysteme: Modelle

❑ Eigenschaften

◆ Vorteilhaft

- Zur Laufzeit nur ein Prozeß, d. h. keine Prozeßumschaltungen
- Die Prozeduren alle in gleichem Adreßraum ausgeführt

◆ Nachteilig

- Alle Perioden müssen Vielfache der Unterperiode sein
- Einordnung sporadischer Prozesse sehr schwierig
- Vorgänge mit sehr langen Perioden müssen evtl. in sehr viele kurzlaufende Teilstücke zerlegt werden
- Konstruktion des Ablaufplans schwierig, NP-vollständig

Prozessorvergabe - Echtzeitsysteme: Modelle

7.3 Prozeßbasierte Zuordnung

❑ Jeder periodische Vorgang als eigener Prozeß

◆ Zuordnung:

- nicht-verdrängend
- verdrängend
- verzögert verdrängend (kooperativ)

❑ Abarbeitung nach abnehmenden Raten (rate monotonic priority)

- $T_i < T_j \Rightarrow P_i > P_j$

S7.1 Wenn das Zuordnungsproblem durch statische Prioritäten gelöst werden kann, dann wird es auch durch Abarbeitung nach abnehmenden Raten gelöst.

❑ Wann führt diese Vorgehensweise zum Ziel?

7.4 **Nutzungsbasierte Tests**

S7.2 **(Liu and Layland, 1973): Wenn die Bedingung**

$$\sum_{i=1}^{N} \frac{C_i}{T_i} < N(2^{1/N} - 1)$$

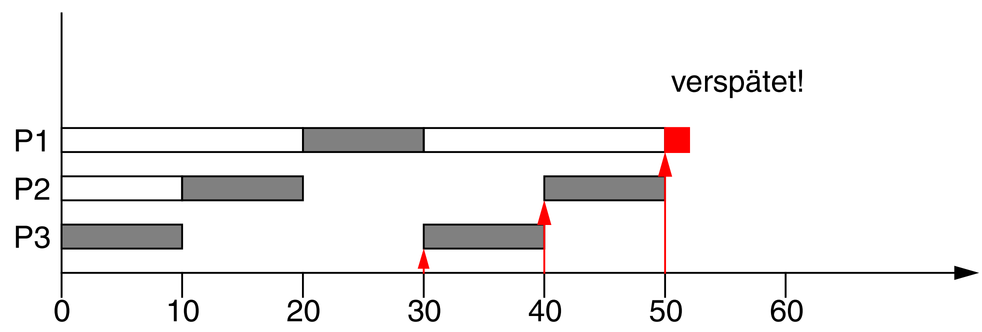
erfüllt ist, dann werden bei Zuordnung nach abnehmenden Raten die Zielzeiten eingehalten.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

❑ **Beispiele**

◆ **Beispiel A: Bedingung nicht erfüllt; Perioden nicht einhaltbar**

Prozeß	T	C	P	U
P1	50	12	1	0.24
P2	40	10	2	0.25
P3	30	10	3	0.33

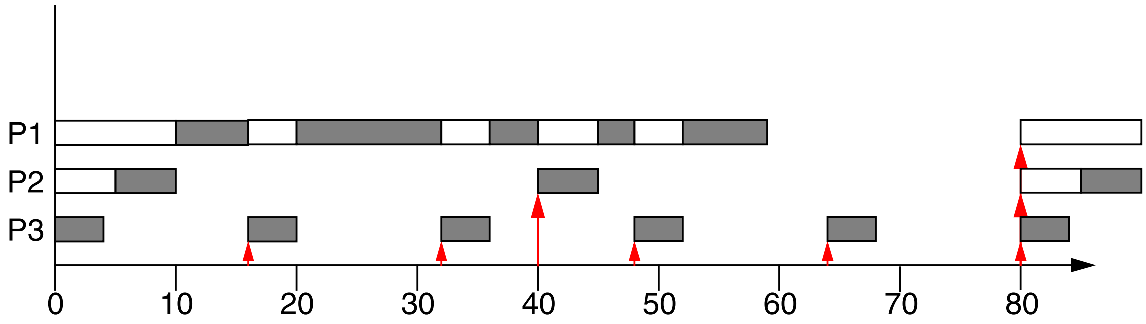


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Prozessorvergabe - Echtzeitsysteme: Modelle

- **Beispiel B: Bedingung erfüllt; Perioden eingehalten**

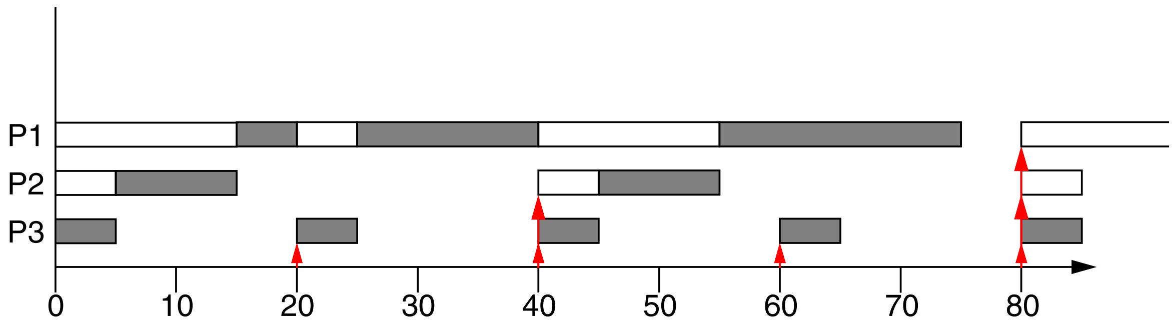
Prozeß	T	C	P	U
P1	80	32	1	0.400
P2	40	5	2	0.125
P3	16	4	3	0.250



Prozessorvergabe - Echtzeitsysteme: Modelle

- **Beispiel C: Bedingung nicht erfüllt; Perioden einhaltbar**

Prozeß	T	C	P	U
P1	80	40	1	0.50
P2	40	10	2	0.25
P3	20	5	3	0.25



Prozessorvergabe - Echtzeitsysteme: Modelle

7.5 **Antwortzeitanalyse**

- **Prozeß höchster Priorität:** $R = C$
- **Für andere:** $R_i = C_i + I_i$
- **Maximale Verdrängungszeit tritt ein, wenn alle höherprioren Prozesse gleichzeitig mit dem betrachteten eine Periode beginnen; o. B. d. A. Beginn zum Zeitpunkt 0.**

◆ **Betrachtet wird das Intervall** $[0, R_i)$

- **Zahl der Deblockierungen von Prozeß j:** $\left\lceil \frac{R_i}{T_j} \right\rceil$
- **Maximale Interferenz:** $\left\lceil \frac{R_i}{T_j} \right\rceil C_j$

also $$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j,$$

dabei bezeichnet hp(i) die Menge der Prozesse, deren Priorität größer i ist.

- **Durch Substitution:** $$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Prozessorvergabe - Echtzeitsysteme: Modelle

◆ **Lösung obiger impliziter Gleichung für** R_i

- **Kleinster Fixpunkt liefert schlechteste Antwortzeit**
- **Iteratives Verfahren**

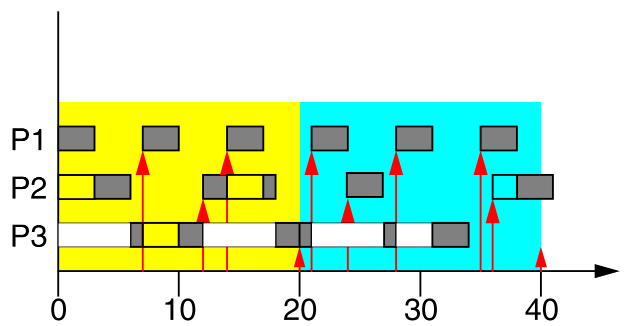
$$w_i^{(n+1)} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^{(n)}}{T_j} \right\rceil C_j$$

- **Algorithmus:**

```
for (int i = 0; i < N; i++) {
    int n =  0;
    w[n][i] = C[i];
    while (true) {
      w[n + 1][i] = C[i];
      for (int j = 0; j < N; j++) {
        if (P[j] > P[i]) {
          w[n + 1][i] += Math.ceil((float)w[n][i] / T[j]) * C[j];
        }
      }
      if (w[n + 1][i] == w[n][i]) {
        R[i] = w[n][i];
        break;
      }
      if (w[n + 1][i] > T[i]) {
        // Not schedulable
        break;
      }
      n++;
    }
}
```


❑ **Beispiel D:**

Prozeß	T	C	P	$w^{(0)}$	$w^{(1)}$	$w^{(2)}$	$w^{(3)}$	$w^{(4)}$	$w^{(5)}$
P1	7	3	3	3	3	3	3	3	3
P2	12	3	2	3	6	6	6	6	6
P3	20	5	1	5	11	14	17	20	20



S7.3 **Notwendig und hinreichend für die Einhaltbarkeit der Zielzeiten ist, daß für alle Prozesse i die Beziehung $R_i <= T_i$ gilt.**

Prozessorvergabe - Echtzeitsysteme: Modelle

7.6 Worst-case- Ausführungszeit

- ❑ **Bislang angenommen, daß für jeden Prozeß i die maximal benötigte Rechenzeit C_i bekannt ist.**
- ❑ **Wie wird sie bestimmt?**
  - **Messung; Einfluß von Pufferspeichern!**
  - **Analyse (abzählen der benötigten Takte); setzt genaues Prozessormodell voraus!**
  - **Behandlung von Schleifen? Evtl. semantisches Wissen einbeziehen**

```
for (int i = 1; i <= 10; i++) {
   if (condition)
      ... ; // Block, der 100 Zeiteinheiten benötigt
   else
      ... ; // Block, der 10 Zeiteinheiten  benötigt
}
```

  - Vielleicht bekannt, daß erster Block höchstens dreimal durchlaufen wird.
  - Alle Iterationen und Rekursionen müssen a priori bekannte Schranken besitzen.

Prozessorvergabe - Echtzeitsysteme: Modelle

7.7 Sporadische und aperiodische Prozesse

- ❑ **Erweiterung des Systemmodells**
  - **Zusätzlich sporadische Aufträge**
    - T interpretiert als kleinstes Zeitintervall zwischen zwei Auslösungen
  - **Zielzeit kann vor Periodenende liegen: D < T**

- ❑ **Vorangehender Algorithmus auch hier anwendbar, wenn die Bedingung `w[n + 1][i] > T[i]` durch `w[n + 1][i] > D[i]` ersetzt wird.**

  Funktioniert für jede Prioritätenzuordnung!

- ❑ **Harte und weiche Echtzeitbedingungen (hard and soft real-time)**

  In manchen Fällen ist Einhalten von Zielzeiten erwünscht, aber nicht unbedingt erforderlich.

  Dann sind Strategien möglich, die nicht immer mit der schlechtesten Situation arbeiten, z. B.
  - **Prozesse sollen mit mittleren Ausführungszeiten und Startraten einplanbar sein**
  - **Harte Echtzeitprozesse sollen einplanbar sein**

7.8 **Zuordnung nach *deadline monotonic priority***

Für D < T: In vorangehenden Überlegungen T durch D ersetzen

S7.4 **Zuordnung nach *deadline monotonic priority* ist für jede Menge Q von Prozessen, die nach einer prioritätsgeteuerten Strategie einplanbar ist, eine optimale prioritätsgeteuerte Strategie.**

7.9 **Interaktion und Blockierung**

Zusätzlich zu berücksichtigen: Prozesse können sich gegenseitig koordinieren

❑ **Es kann zu Prioritätsinversionen (*priority inversion*) kommen, d. h. ein hochpriorer Prozeß muß auf Freigabe durch einen niederprioren warten.**

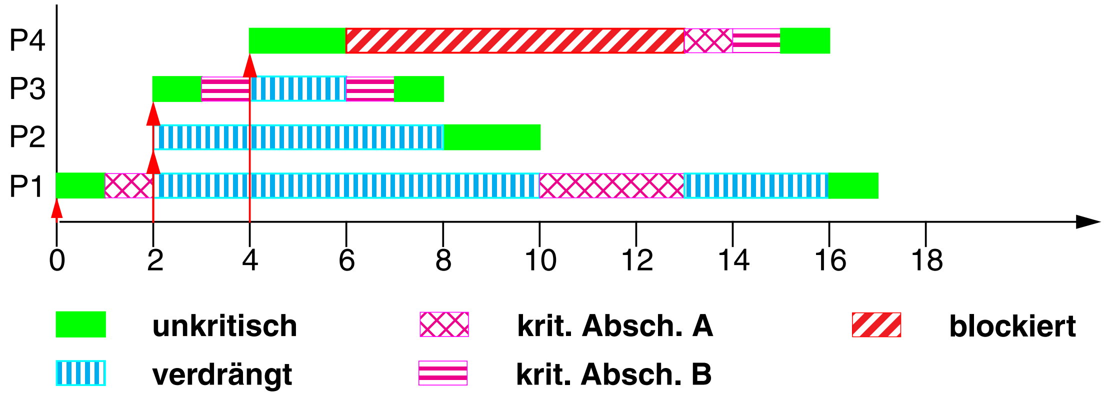
- **Meist nicht vollkommen vermeidbar, aber nachteilige Effekte können minimiert werden.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Prozessorvergabe - Echtzeitsysteme: Modelle

- **Beispiel E**

Prozeß	Priorität	Ausführungsfolge	Startzeit
P4	4	RRABR	4
P3	3	RBBR	2
P2	2	RR	2
P1	1	RAAAAR	0



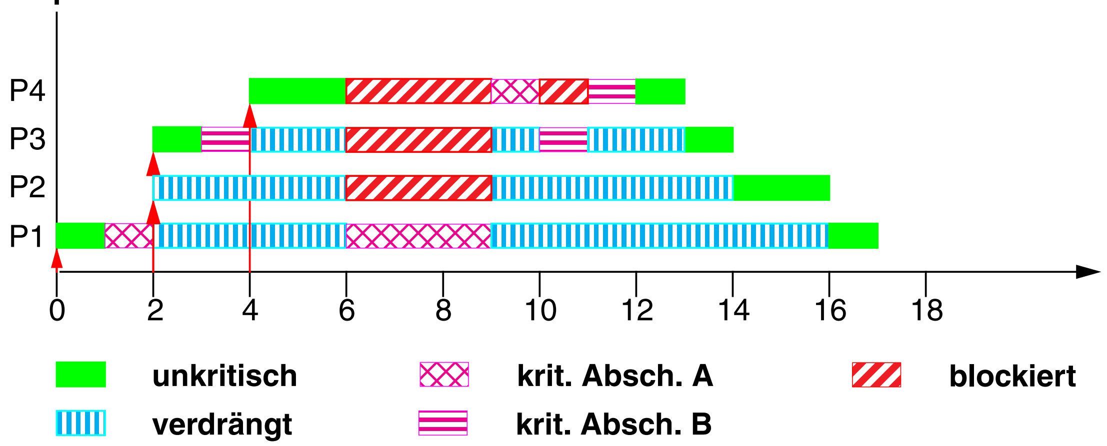
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

Prozessorvergabe - Echtzeitsysteme: Modelle

❑ **Vermeidung dieses Effektes durch Prioritätsvererbung**

Wenn Prozeß P blockiert wird, weil ein Prozeß Q mit niedriger Priorität im kritischen Abschnitt ist, dann bekommt Q bis zum Verlassen des kritischen Abschnitts die gleiche Priorität wie P

Beispiel E



Diese Vererbung kann sich über mehrere Prozesse erstrecken!

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann

BP 2 ## Prozessorvergabe - Echtzeitsysteme: Modelle

S7.5 **Sha e. a., 1990:**

Wenn ein Prozeß m kritische Abschnitte besitzt und n Prozesse niedrigerer Priorität existieren, dann wird er bei Prioritätsvererbung höchstens min(m, n)-mal blockiert.

❑ **Bestimmung der maximalen Blockierungszeit**

Es sei $\text{nutzung}(k, i) = 1$**, wenn Betriebsmittel k von wenigstens einem Prozeß geringerer Priorität und wenigstens einem Prozeß mit gleicher oder höherer Priorität benutzt wird.**

Ansonsten gelte $\text{nutzung}(k, i) = 0$**.**

K bezeichne die Zahl verschiedener Betriebsmittel.

Dann gilt für die maximale Blockierungszeit B_i

$$B_i \le \sum_{k=1}^{K} \text{nutzung}(k, i)\text{Zahl\_k\_A}(k)$$

BP 2 ## Prozessorvergabe - Echtzeitsysteme: Modelle

❑ **Antwortzeiten bei Berücksichtigung von Blockierungen**

- **Frühere Berechnung ergänzt zu**

$$R = C + B + I$$

- **also**

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- **Daraus ergibt sich die Rekursion**

$$w_i^{(n+1)} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^{(n)}}{T_j} \right\rceil C_j$$

Diese Formel ist wegen der Art, in der die Blockierungen abgeschätzt wurden, pessimistisch.