Echtzeitbetrieb

Schritthaltende Verarbeitung/Echtzeitbetrieb

- **Fristgerechte Bearbeitung von Anforderungen aus technischem Prozess**
- **Verletzung von Zeitbedingungen ggf. katastrophal**

Echtzeitbetrieb

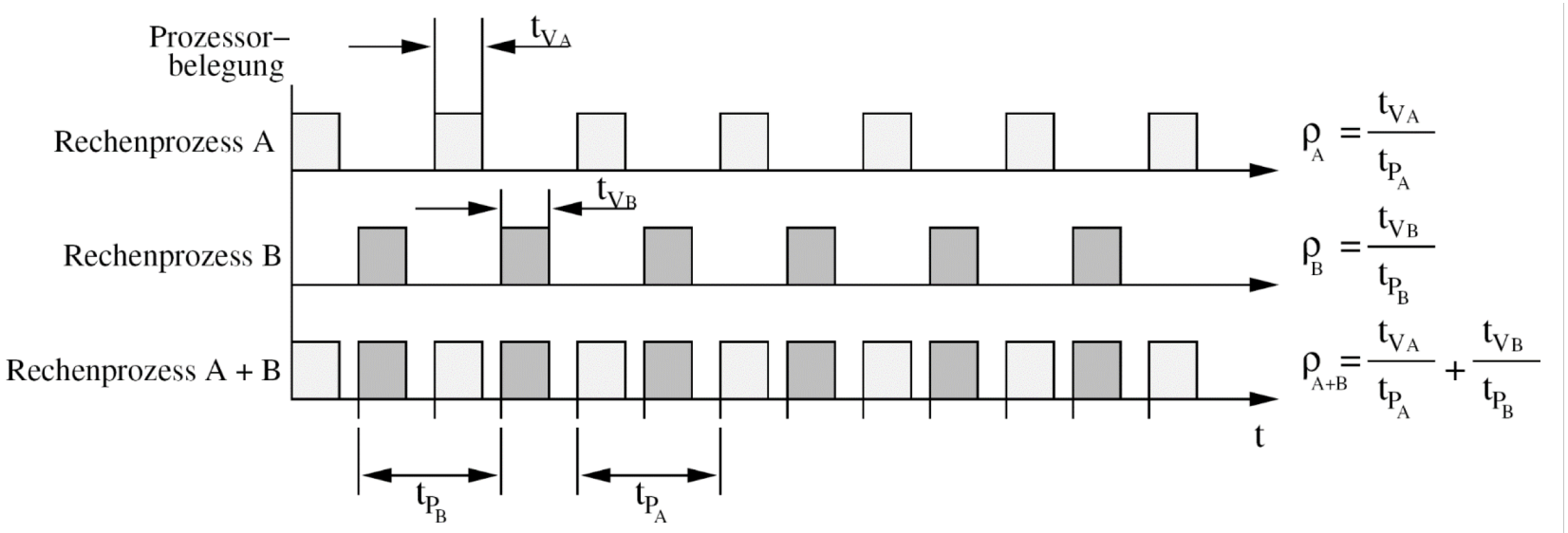
Vorbedingungen für schritthaltende Verarbeitung

- **Bearbeitung von Aufgaben benötigt Zeit (Verarbeitungszeit).**
- **Bei mehreren Aufgaben zugleich: Reihenfolge planen, in der die Arbeiten erledigt werden**
- **Reihenfolge der Aufgaben ist entscheidend, um die fristgerechte Bearbeitung der Aufgaben gewährleisten zu können.**
- **Prioritäten von Aufgaben gemäß ihrer Wichtigkeit als Planungsgrundlage**
- **Die Bearbeitung einer Aufgabe (durch einen Prozess) muss unterbrechbar sein, damit kurzfristige höherpriore Aufgaben erledigt werden können.**

Ziel: Formaler Rahmen, um funktionierende schritthaltende Verarbeitung nachweisen zu können

Echtzeitbetrieb

Echtzeitbedingungen

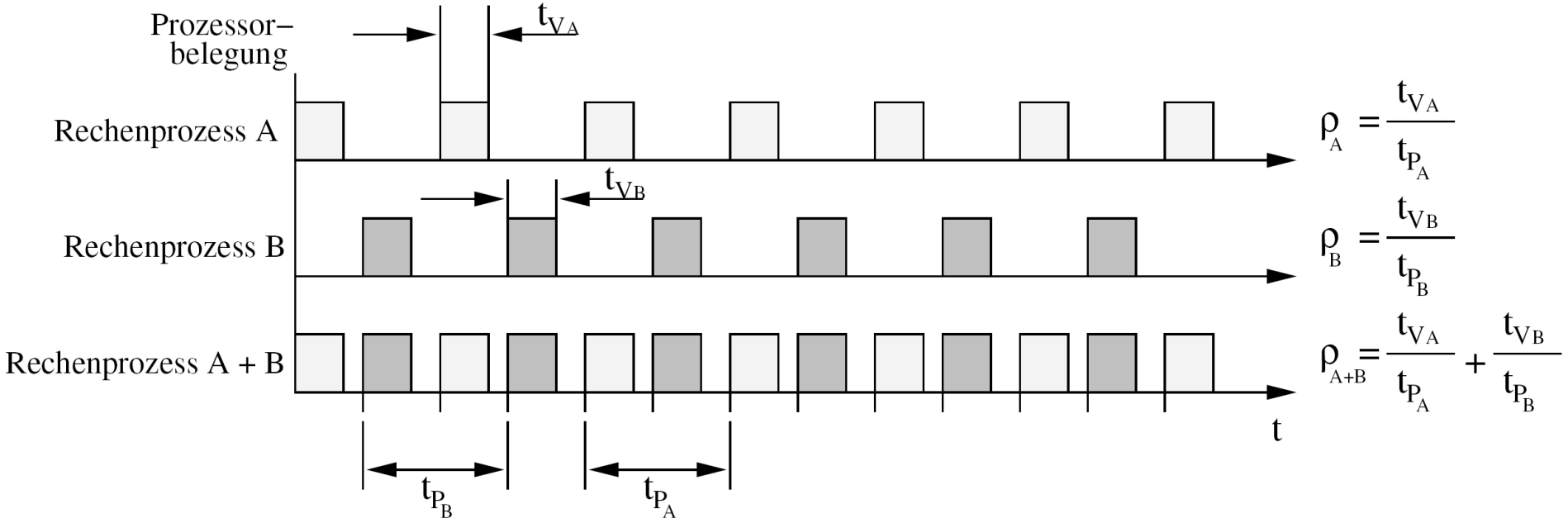


- **Prozesszeit t_P : Abstand zwischen zwei Anforderungen (*jobs*) desselben Typs**
  - **konstant: zyklischer/periodischer Prozess**
- **Verarbeitungszeit t_V**
- **Auslastung $\rho = t_V / t_P$**
- **Gesamtauslastung bei *n* Prozessen:**

$$\rho = \sum_{i=0}^{n} \frac{t_{V_i}}{t_{P_i}}$$

Echtzeitbetrieb

Echtzeitbedingungen

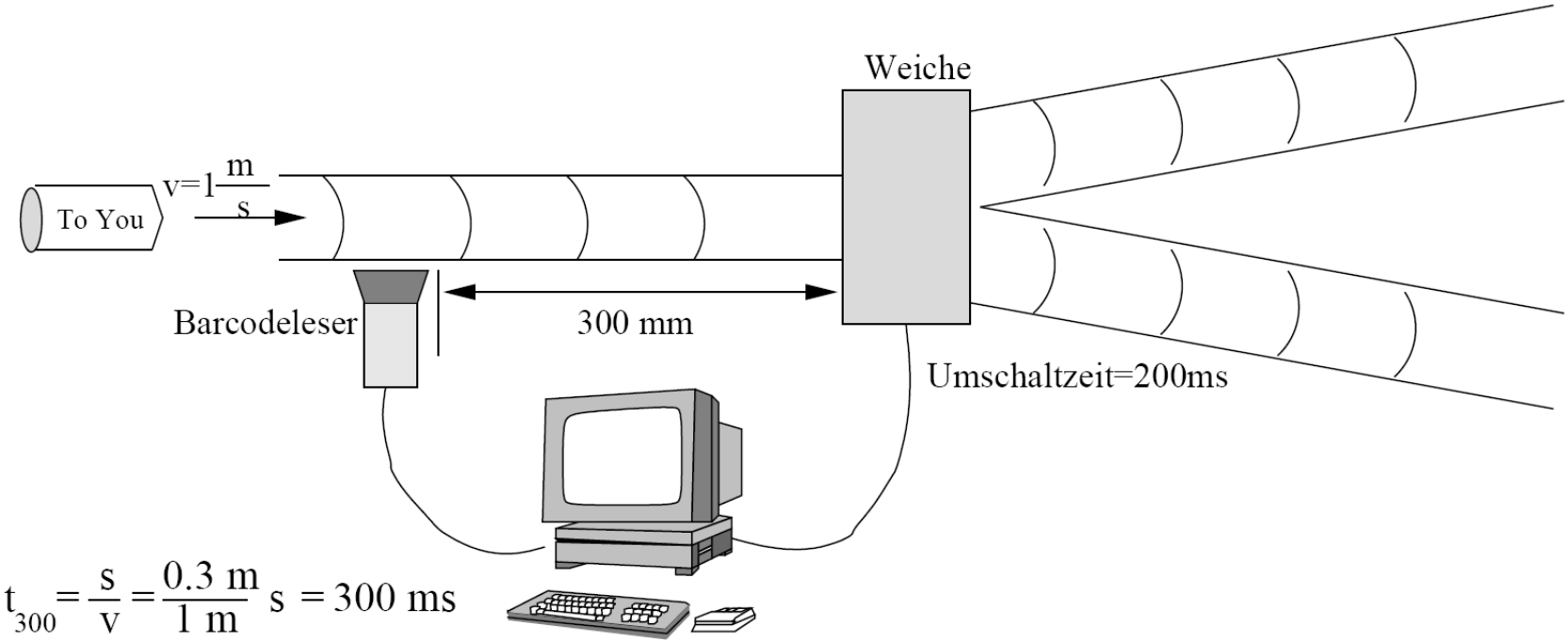


- **1. Echtzeitbedingung: Auslastung eines Rechensystems muss kleiner oder gleich 100% sein**

$$\rho = \sum_{i=0}^{n} \frac{t_{V_i}}{t_{P_i}} \leq 1$$

Echtzeitbedingungen

Pünktlichkeit



$$t_{300} = \frac{s}{v} = \frac{0.3\ m}{1\ m}\ s = 300\ ms$$

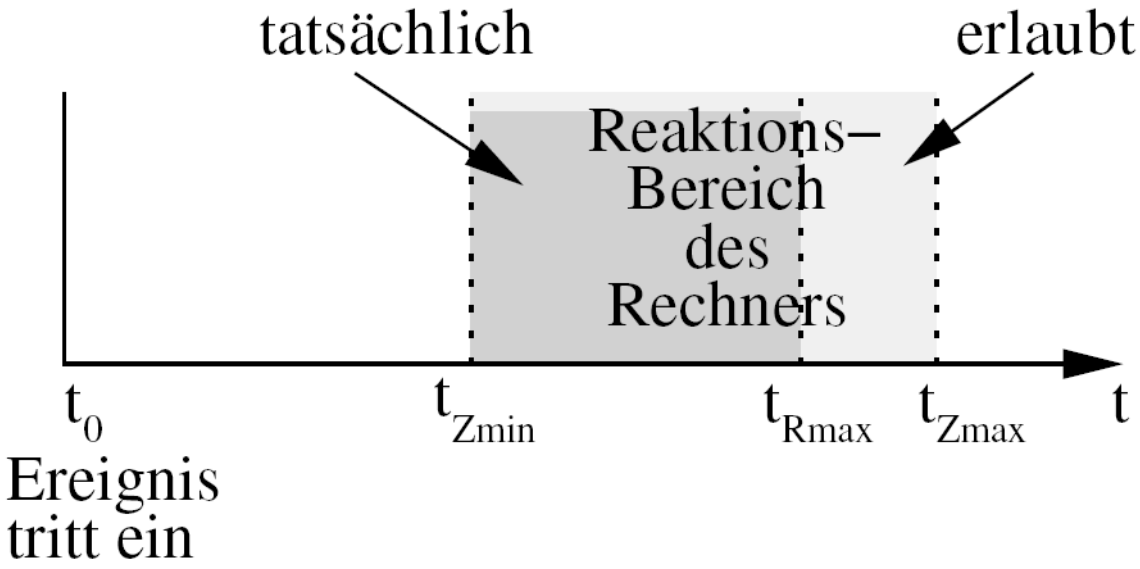
$$t_{Zmax} = t_{300} - t_U = 100\ ms$$

Echtzeitbetrieb

Echtzeitbedingungen

Pünktlichkeit

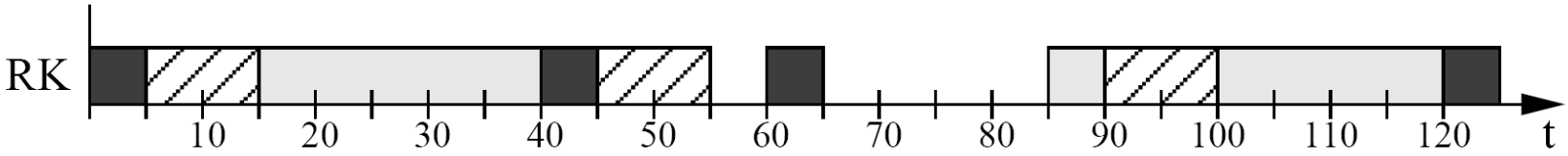
- **Aufgabe nicht vor spezifiziertem Zeitpunkt t_{Zmin} erledigt (meist unwichtig oder trivial)**
- **spätestens bis Zeitpunkt t_{Zmax} erledigt (Rechtzeitigkeit)**



Echtzeitbedingungen

Pünktlichkeit

- Reaktionszeit $t_R = t_V + t_W$ (Verarbeitungszeit t_V + Wartezeit t_W)
- Wartezeit : Zeit bis Rechnerkern frei



- 2. Echtzeitbedingung: Um Aufgaben rechtzeitig zu erledigen, muss die Reaktionszeit t_R zwischen der minimal und maximal zulässigen Reaktionszeit liegen

$$t_{Zmin} \leq t_{Rmin} \leq t_R \leq t_{Rmax} \leq t_{Zmax} \text{ (relative Deadline)}$$

Echtzeitbetrieb

Echtzeitbedingungen

Harte und weiche Echtzeit

- **Harte Echtzeit: Verletzung der Rechtzeitigkeit hat katastrophale Folgen (z.B. Airbag, Rohrpost,...)**
- **Weiche Echtzeit: Schlechteres Ergebnis (z.B. ruckelnde Videowiedergabe,...)**

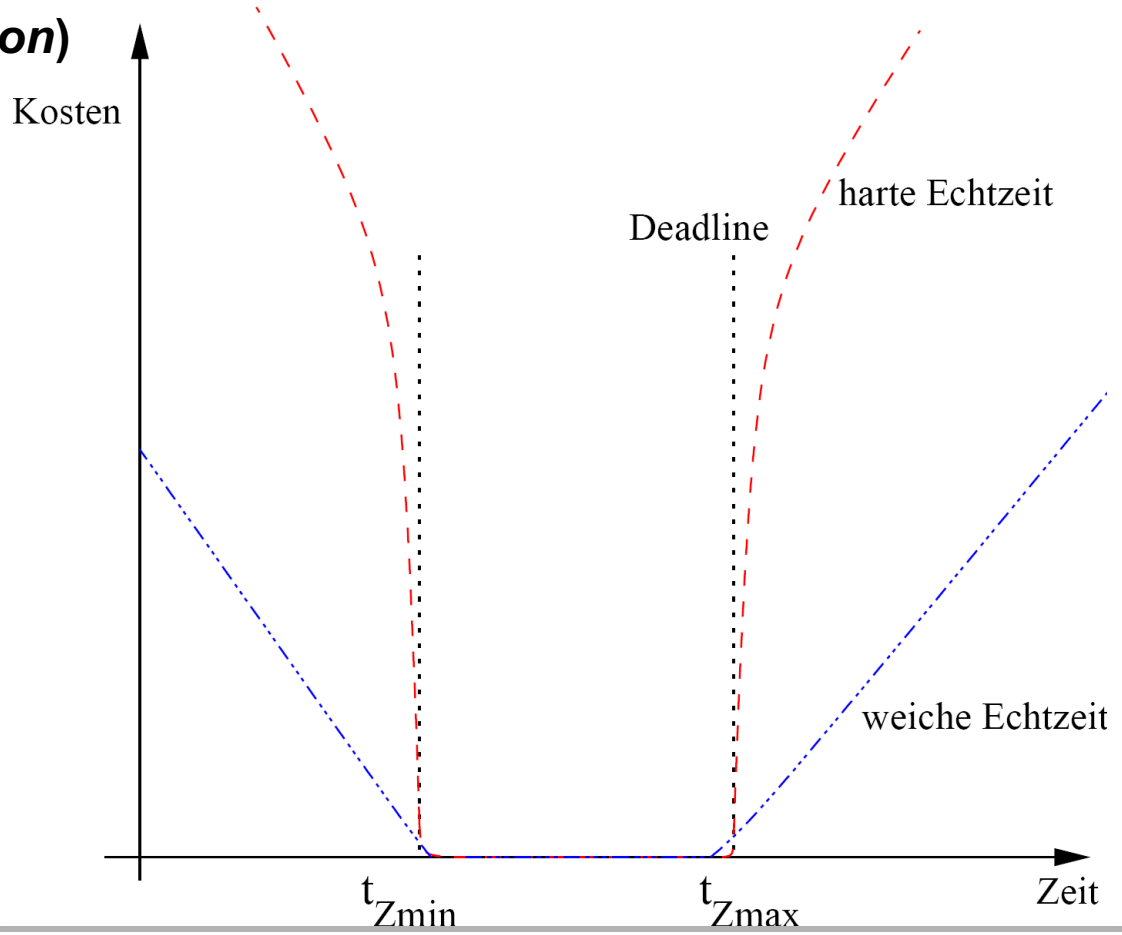
- **Häufig Graubereich (Buffer Underflow beim DVD brennen???)**

- **Kosten- und Nutzenfunktion**

Echtzeitbedingungen

Harte und weiche Echtzeit

- **Kostenfunktion (*cost function*)**

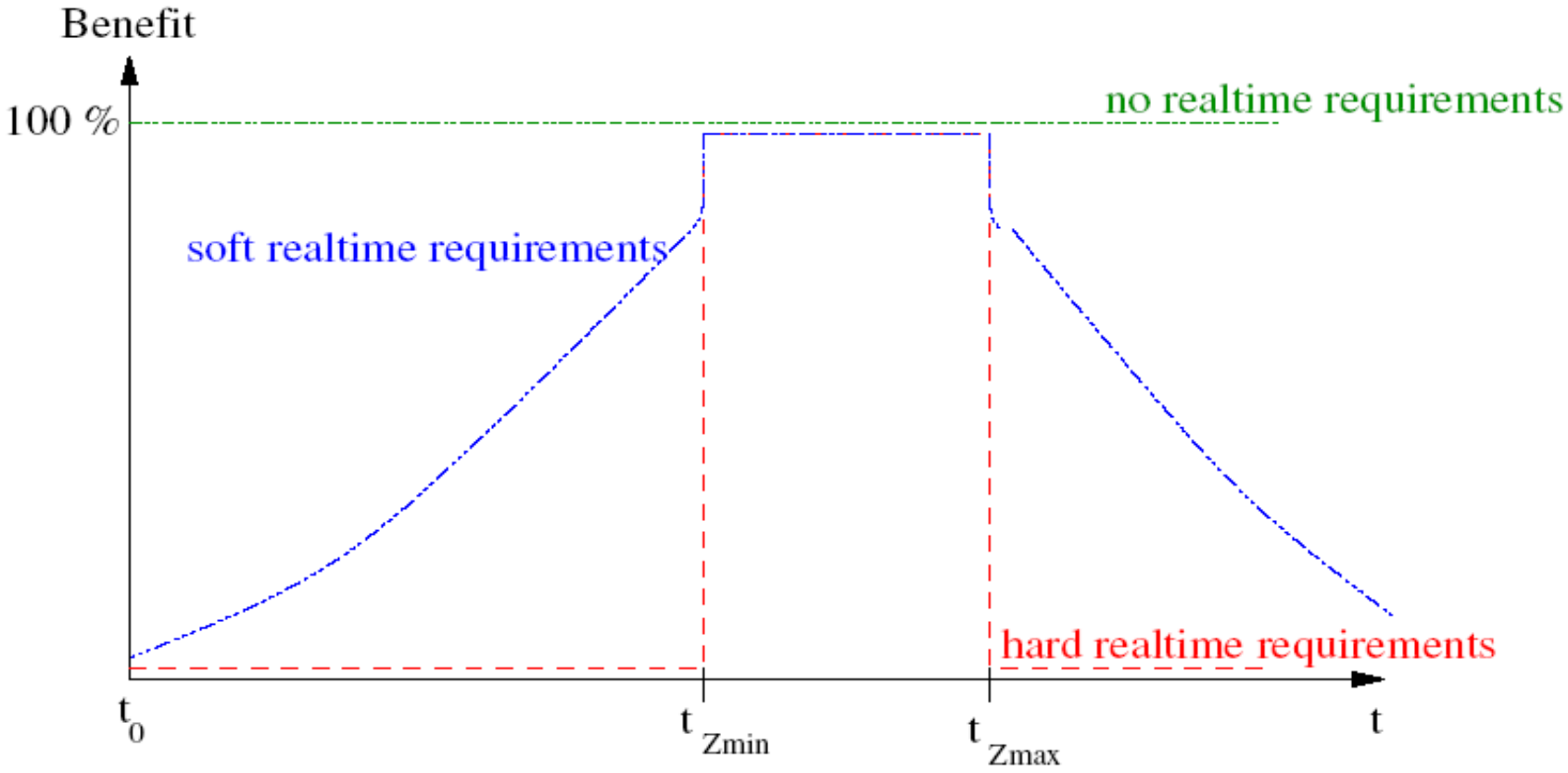


Echtzeitbetrieb

Echtzeitbedingungen

Harte und weiche Echtzeit

- **Nutzenfunktion (*benefit function*)**



Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

Ein Meßwerterfassungssystem soll im Abstand von 1 ms (t_{P1}=1ms) kontinuierlich Meßwerte aufnehmen. Dazu benötigt der zugehörige Rechenprozess eine Rechenzeit von t_{V1}=500 μs. Jeweils 100 Meßwerte (t_{P2}=100t_{P1}=100ms) ergeben einen Datensatz, der vorverarbeitet und zur Archivierung weitergeleitet wird. Dazu ist eine Rechenzeit von t_{V2}=40ms notwendig.

Unterbrechbarkeit und Prioritäten
Beispiel Messwertaufnahme

Ein Meßwerterfassungssystem soll im Abstand von 1 ms (t_{P1}=1ms) kontinuierlich Meßwerte aufnehmen. Dazu benötigt der zugehörige Rechenprozess eine Rechenzeit von t_{V1}=500 μs. Jeweils 100 Meßwerte (t_{P2}=100t_{P1}=100ms) ergeben einen Datensatz, der vorverarbeitet und zur Archivierung weitergeleitet wird. Dazu ist eine Rechenzeit von t_{V2}=40ms notwendig.

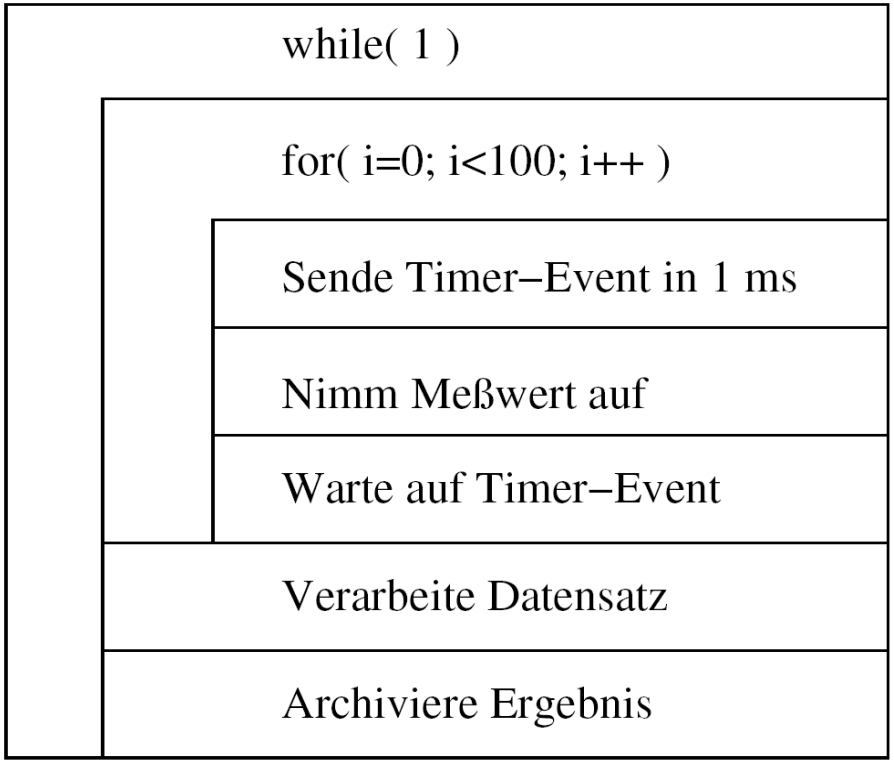
- **Analyse der Auslastung**

$$\rho = t_{V1}/t_{P1} + t_{V2}/t_{P2} = 0.5 + 0.4 = 0.9 = 90\%$$

Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

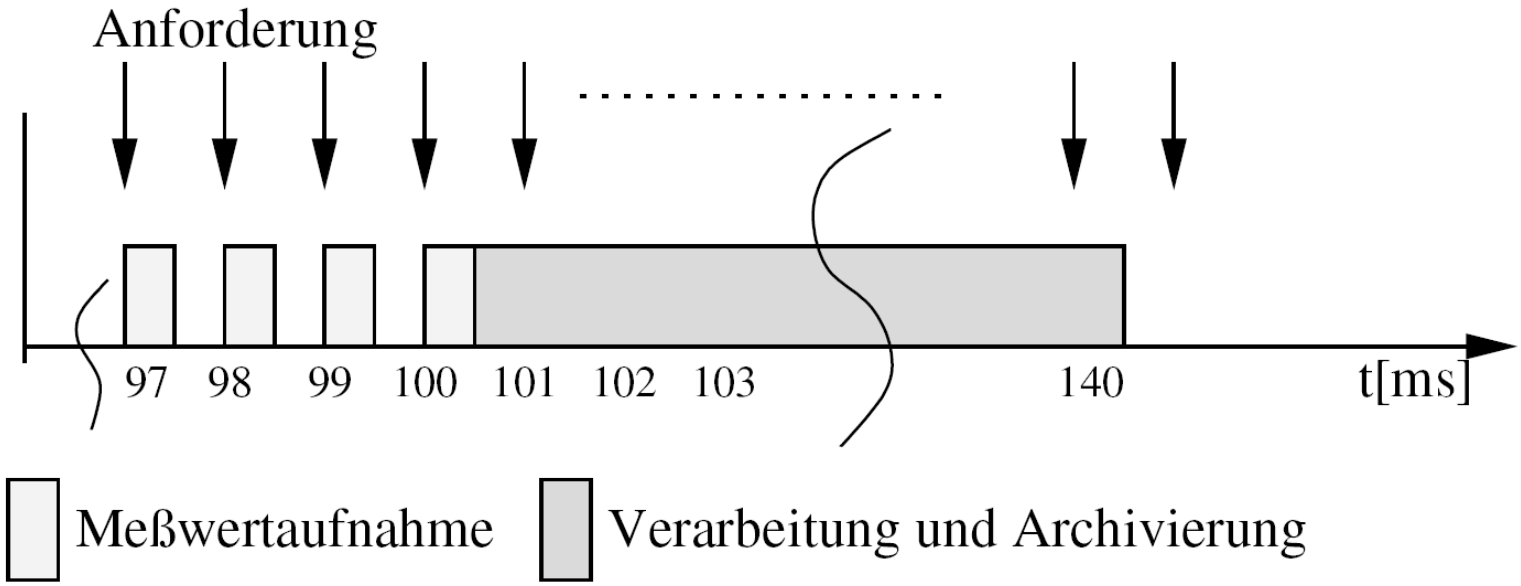
- **Sequentiell, Jobs nicht unterbrechbar**



Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

- **Sequentiell, Jobs nicht unterbrechbar**



- **Rechtzeitigkeit nicht erfüllt !!!!**

Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

- Lösung: zwei Rechenprozesse
- Interprozesskommunikation (IPC)

Rechenprozess "Erfassung"

```
while( 1 )
    for( i=0; i<100; i++ )
        Sende Timer–Event in 1 ms
        Nimm Meßwert auf
        Warte auf Timer–Event
    SendMsg an "Verarbeitung"
```

Rechenprozess "Verarbeitung"

```
while( 1 )
    RcvMsg von "Erfassung"
    Verarbeitung der Daten
    Archiviere Ergebnis
```

Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

- **Lösung: zwei Rechenprozesse**
- **Interprozesskommunikation (IPC)**
- **Rechenprozess „Erfassung“ *wichtiger* (höher prior, hat höhere Priorität)**
- **Rechenprozess „Verarbeitung“ muss unterbrechbar sein**

Rechenprozess "Erfassung"

```
while( 1 )
    for( i=0; i<100; i++ )
        Sende Timer–Event in 1 ms
        Nimm Meßwert auf
        Warte auf Timer–Event
    SendMsg an "Verarbeitung"
```

Rechenprozess "Verarbeitung"

```
while( 1 )
    RcvMsg von "Erfassung"
    Verarbeitung der Daten
    Archiviere Ergebnis
```

Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

- **Lösung: zwei Rechenprozesse**
- **Interprozesskommunikation (IPC)**
- **Rechenprozess „Erfassung“ *wichtiger* (höhere Priorität)**
- **Rechenprozess „Verarbeitung“ muss unterbrechbar sein**

Echtzeitbetriebssystem unterstützt diese Forderungen

Rechenprozess "Erfassung"

```
while( 1 )
    for( i=0; i<100; i++ )
        Sende Timer–Event in 1 ms
        Nimm Meßwert auf
        Warte auf Timer–Event
    SendMsg an "Verarbeitung"
```

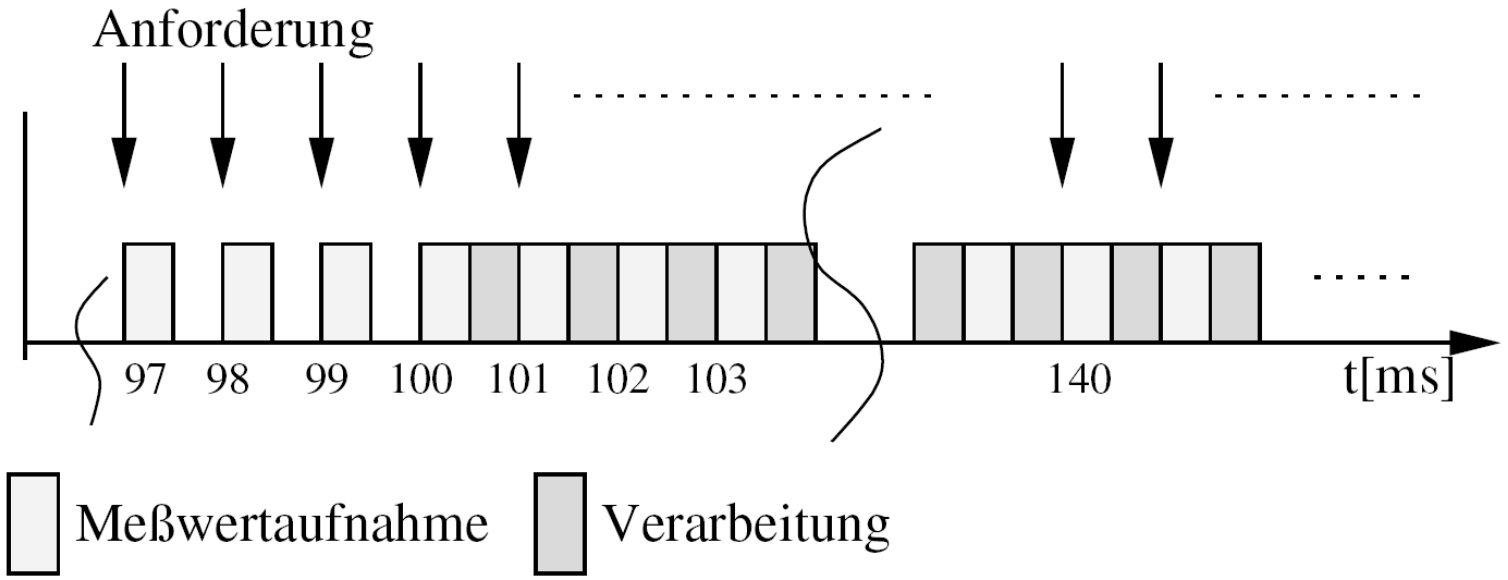
Rechenprozess "Verarbeitung"

```
while( 1 )
    RcvMsg von "Erfassung"
    Verarbeitung der Daten
    Archiviere Ergebnis
```

Unterbrechbarkeit und Prioritäten

Beispiel Messwertaufnahme

- **Lösung: zwei Rechenprozesse**
- **Interprozesskommunikation (IPC)**
- **Rechenprozess „Erfassung“ *wichtiger* (höhere Priorität)**
- **Rechenprozess „Verarbeitung“ muss unterbrechbar sein**



Echtzeitnachweis

???

Echtzeitnachweis

Formaler Nachweis, dass

- **1. Echtzeitbedingung in jeder Situation erfüllt ist**
- **2. Echtzeitbedingung in jeder Situation erfüllt ist**

Analyse der Echtzeit-Bedingungen für den Fall der höchsten Last

- **Prozesszeiten möglichst kurz**
- **Verarbeitungszeiten möglichst lang**
- **alle sonstigen Ereignisse im System, die im aktuellen Prozesszustand möglich sind, treten gleichzeitig (t = 0) auf**

Echtzeitnachweis

Sehr schwieriges Problem, Methoden hier nur skizziert

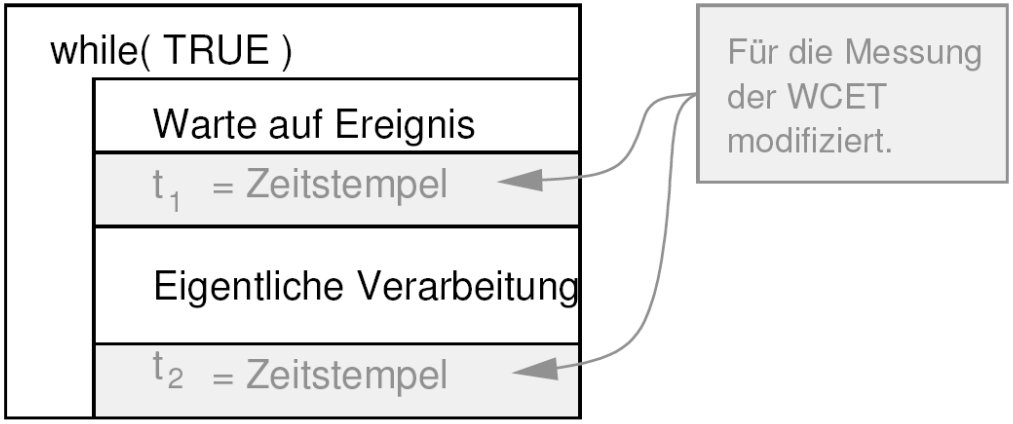
Prinzipiell:

- **relevante Kenndaten des technischen Prozesses ermitteln,**
  - **Anzahl der unterschiedlichen Anforderungen**
  - **Minimale Prozesszeit für jede Anforderung**
  - **Minimal zulässige Reaktionszeit t_{Zmin} für jede Anforderung**
  - **Maximal zulässige Reaktionszeit t_{Zmax} für jede Anforderung**
  - **Abhängigkeiten zwischen den Ereignissen**
- **maximale Verarbeitungszeit t_{Vmax} (WCET) für jede Anforderung identifizieren**
- **Auslastungsbedingung überprüfen**
- **Rechtzeitigkeitsbedingung verifizieren (Bestimmung von t_{Rmin} und t_{Rmax}).**

Echtzeitnachweis

Abschätzung der Worst Case Execution Time, Messen

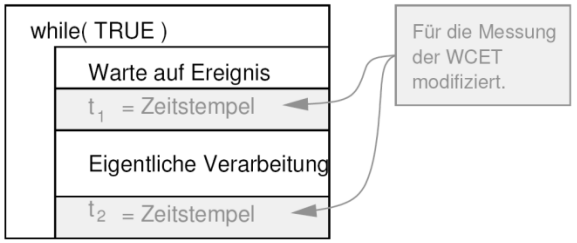
- **Messen durch Code-Instrumentierung**



- **Externe Messung von Ereignis und Reaktion (z.B. mit Oszilloskop)**
- **Vorsicht!!! Messung kann Zeitverhalten verändern (Profiler, die Prolog und Epilog austauschen)**

Echtzeitnachweis
Abschätzung der Worst Case Execution Time, Messen

+ Sprachunabhängig
+ meist einfach realisierbar



- Aussagekraft der Messungen abhängig von vielen Randbedingungen (Prozesszustand, Cache, Schleifen, Verzweigungen)
- Theoretisch sämtliche Kombinationen aus Inputdaten erforderlich (Test-Überdeckung)
- Produktiver Code auf Zielplattform oder Simulator erforderlich
- Messumgebung/Testrahmen erforderlich
- Instrumentierung modifiziert den Code

Echtzeitnachweis

Abschätzung der Worst Case Execution Time, Analyse

Aus Quell- oder Zielcode wird ein Strukturgraph des Codestücks erstellt
Mit Beschreibung der Zielhardware wird der längste Pfad durch den Graphen gesucht

- **+ Analyse (auch nach Codemodifikationen) schnell durchführbar (Anpassung der Analysetools an Zielhardware und –Sprache ist aufwändig)**
- **+ Analyse ist frühzeitig ohne vorhandene Zielhardware möglich**
- **+ Analyse unter größtmöglicher Abdeckung der Inputs**
- **- Komplexe Analysewerkzeuge erforderlich**
- **- Verifikation/Aussagekraft der Analyseergebnisse („stimmt die Hardwarebeschreibung?“,...)**

Echtzeitbetrieb

Abschätzung der Best Case Execution Time (BCET)

- Bestimmung der BCET erforderlich, wenn Codesequenz nicht vor einer minimal zulässigen Reaktionszeit t_{Zmin} abgearbeitet worden sein darf
- Bestimmung der BCET einfacher als Bestimmung der WCET (kürzester Pfad)
- Einfachere Codeanalyse
- Messung wie bei WCET, mit möglichst geringer Systemlast

Echtzeitnachweis

Unterscheidung nach Scheduling-Arten

- prioritätengesteuertes ratenmonotones Scheduling
- Deadline-Scheduling → „zuerst die Task, deren Deadline am nächsten liegt“

Verteilung der Taskprioritäten so, daß entweder:

a) Kürzere Prozesszeit t_p entspricht höherer Priorität → Bezeichnung: Rate Monotonic Fixed Priority Scheduling (RMS) oder

b) Kürzere maximal zulässige Reaktionszeit t_{Zmax} entspricht höherer Priorität
Bezeichnung: Deadline Monotonic Fixed Priority Scheduling (DMS)

Prinzipiell: Verifikation, dass Rechtzeitigkeitsbedingung für alle Ereignisse eingehalten

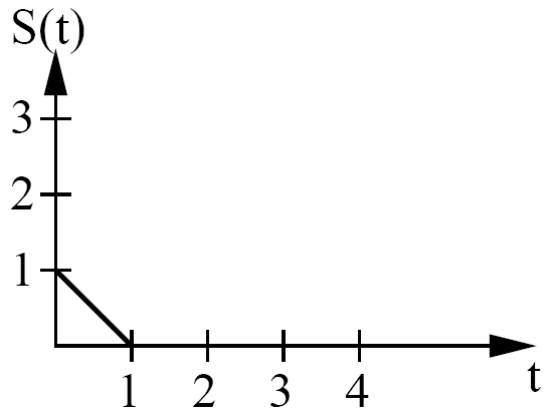
Erforderliche Kenndaten

- Minimale und maximale Verarbeitungszeiten (t_V, BCET und WCET) der Jobs
- Minimal und maximal zulässige Reaktionszeiten (t_{Zmin} und t_{Zmax}) für jeden Job
- Minimale und maximale Reaktionszeiten (t_{Rmin} und t_{Rmax}) (???)

BCET und WCET für die Berechnung der minimalen/maximalen Reaktionszeiten
Mit Hilfe der Reaktionszeiten wird der eigentliche Nachweis durchgeführt
Hier: grafisches Verfahren

Echtzeitnachweis

Grafische Bestimmung der maximalen Reaktionszeit t_{Rmax}



Voraussetzungen

- prioritätengesteuertes (preemptives) Scheduling
- alle Ereignisse sind unabhängig voneinander

Koordinatensystem

- X-Achse: Zeit
- Y-Achse: *noch zu erbringende* Rechenzeit `S(t)`
  (S=Supplied Computation Time, Systemlast)

Dimension von `S(t): 1S kann` in einer Zeiteinheit abgearbeitet werden

- Die Rechenzeitanforderungen sind die Anforderungen des Worst Case
- Bei ungestörter Abarbeitung mit der Steigung `-1` fallende Gerade
- Bei jeder Anforderung: Gerade um t_V nach oben verschieben
- Schnittpunkt Gerade/X-Achse entspricht der maximalen Reaktionszeit des niedrigpriorsten Prozesses, wenn t_{Zmax} nicht verletzt wurde.
- Für den nächstniedrigprioren Prozess: X-Achse um t_V des vorhergehenden Prozesses nach oben verschieben
- Zu jedem Zeitpunkt lässt sich ablesen, wie viel Rechenarbeit noch zu leisten ist

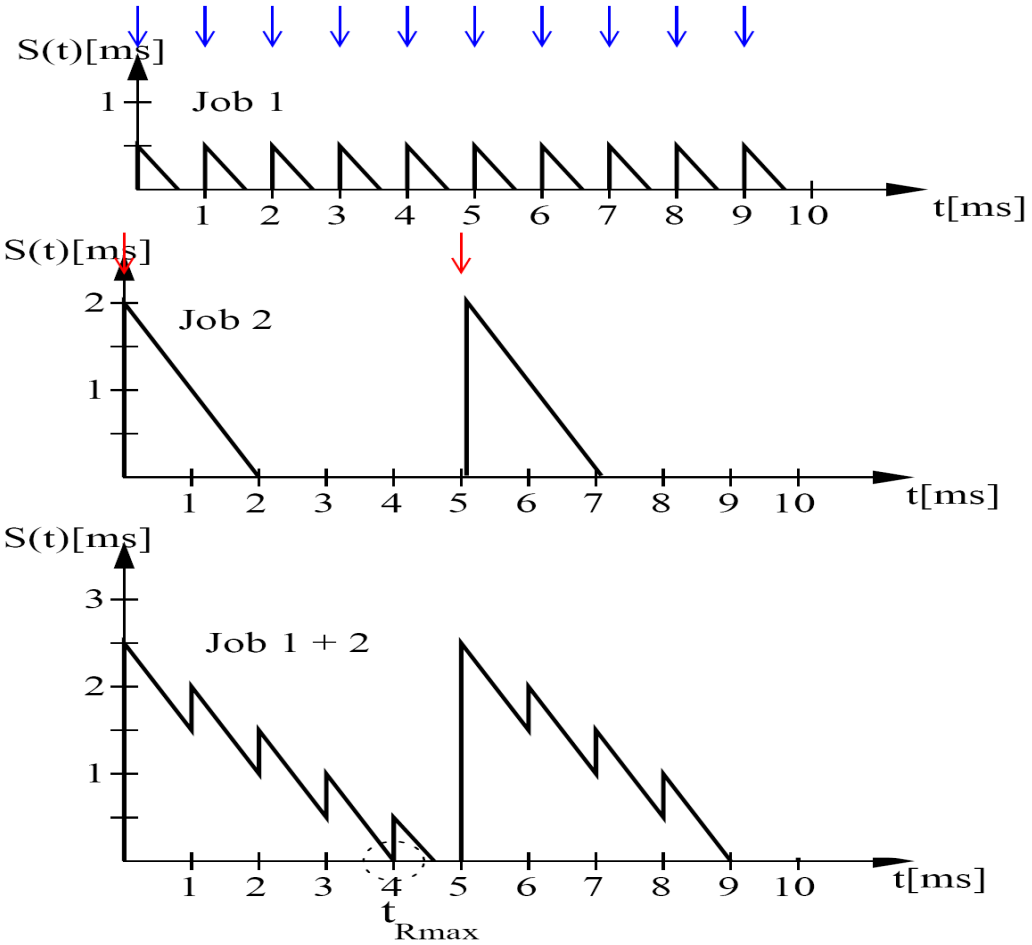
Echtzeitnachweis

Grafische Bestimmung der maximalen Reaktionszeit t_{Rmax}

Beispiel: 2 Jobs
Job 1: t_V = 0.5 ms, t_P = 1 ms
Job 2: t_V = 2ms, t_P = 5 ms

initiale Rechenzeitanforderung:
$R=t_{v1} + t_{v2}$ = 2.5ms

Job 1 Prio höher als Job 2 weil
$t_{P1} < t_{P2}$



Echtzeitnachweis

Grafische Bestimmung der maximalen Reaktionszeit t_{Rmax}

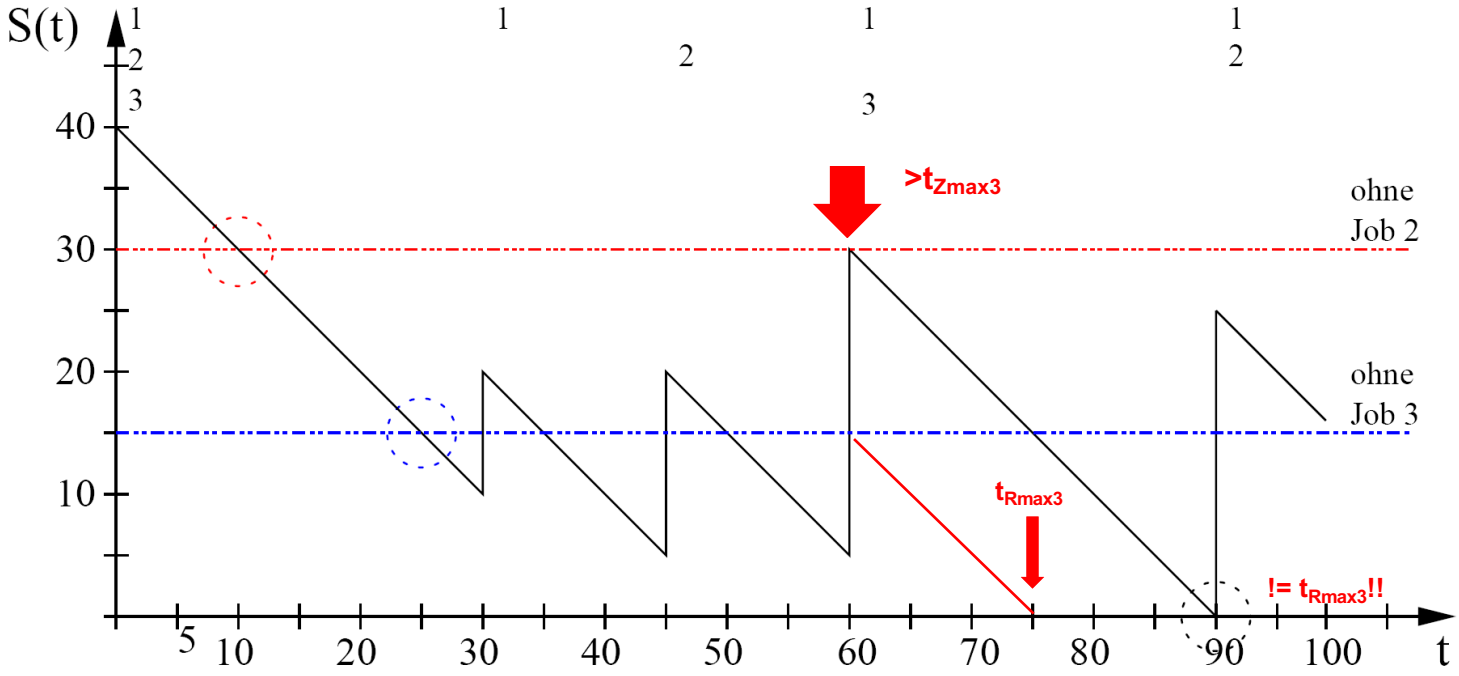
Anforderung	t_V	t_P	t_{Zmax}	t_{Zmin}
1	10 ms	30 ms	30 ms	0 ms
2	15 ms	45 ms	45 ms	0 ms
3	15 ms	60 ms	60 ms	0 ms

Echtzeitnachweis

Grafische Bestimmung der maximalen Reaktionszeit t_{Rmax}

Anforderung	t_V	t_P	t_{Zmax}	t_{Zmin}
1	10 ms	30 ms	30 ms	0 ms
2	15 ms	45 ms	45 ms	0 ms
3	15 ms	60 ms	60 ms	0 ms

Prio hoch
Prio mittel
Prio niedrig



Echtzeitnachweis

Grafische Bestimmung der maximalen Reaktionszeit t_{Rmax}

Als Übung

Wie werden folgende Tasks priorisiert?

1. Echtzeitbedingung

2. Echtzeitbedingung grafisch...

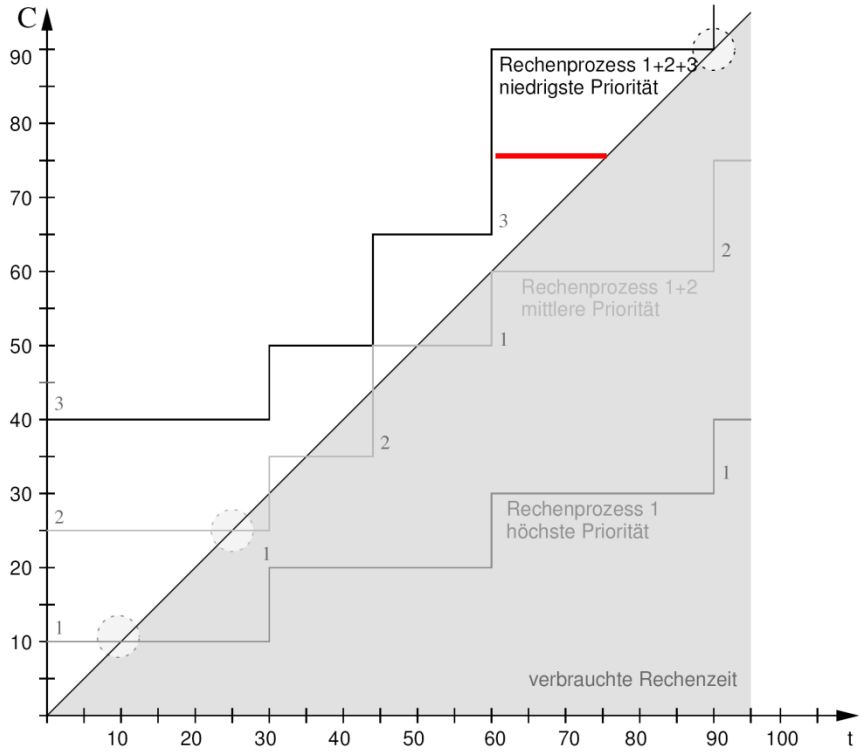
Prioritäten: 1…4 (4 am Höchsten):

J1	t_{p1}=18 msec	t_{v1}=2.5 msec
J2	t_{p2}=3 msec	t_{v2}=1 msec
J3	t_{p3}=2 msec	t_{v3}=0.5 msec
J4	t_{p4}=6 msec	t_{v4}=1.5 msec

Zusammenfassung

- **Echtzeit-Nachweis**
  - **Erforderliche Kenndaten**
  - **(grafisches) Werkzeug**

Echtzeitnachweis



Mathematischer Ansatz

Überprüfen, ob Rechtzeitigkeits-bedingung ($t_{Rmax} < t_{Zmax}$) erfüllt ist.

Voraussetzungen

- **Ereignisse sind periodisch**
- **Prioritätengesteuertes Scheduling**

»C(t)«: Rechenzeitanforderungsfunktion

Anschaulich:

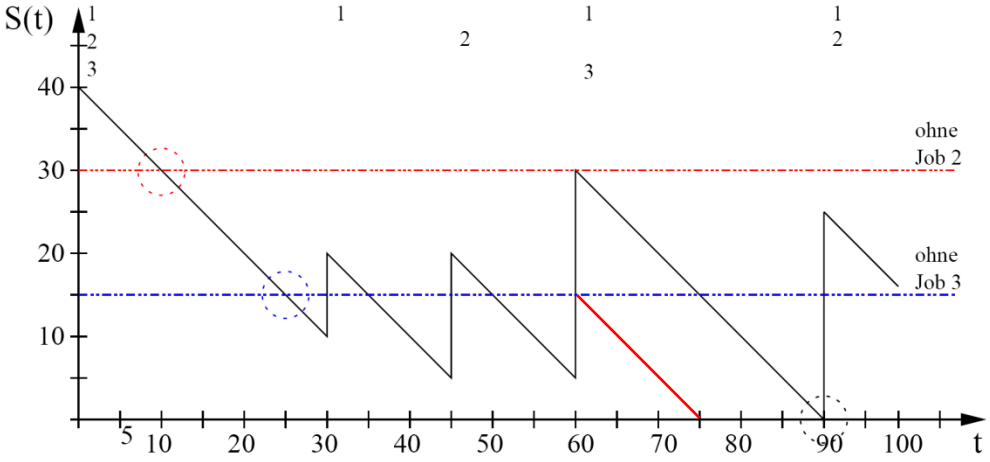
- **Auftragen der von den Jobs angeforderten Rechenzeit »C(t)« über der Zeit »t«.**
- **»C(t)« nimmt mit jeder Anforderung zu (monoton steigende Funktion).**
- **Einzeichnen der vom Rechner bis Zeitpunkt »t« zur Verfügung gestellten Rechenzeit »A(t)« (`A(t)=t, Winkelhalbierende`)**

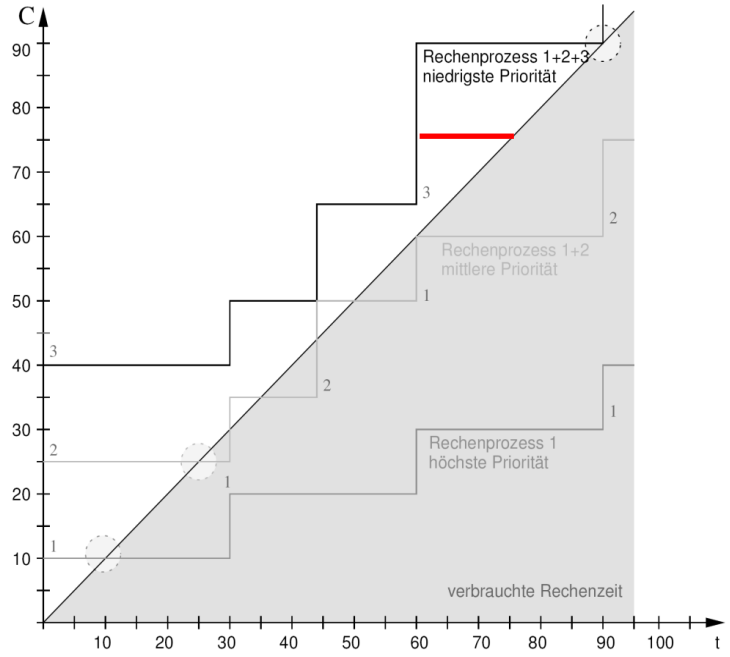
Der Zeitpunkt t, zu dem sich C(t) und A(t) schneiden, ist die maximale Reaktionszeit des niedrigpriorsten Prozesses, (nur) wenn $t_{Rmax} < t_{Zmax}$:

$C(t)=A(t)$; $C(t_{Rmax})=A(t_{Rmax})$; $C(t_{Rmax})=t_{Rmax}$;

Echtzeitnachweis

Mathematischer Ansatz



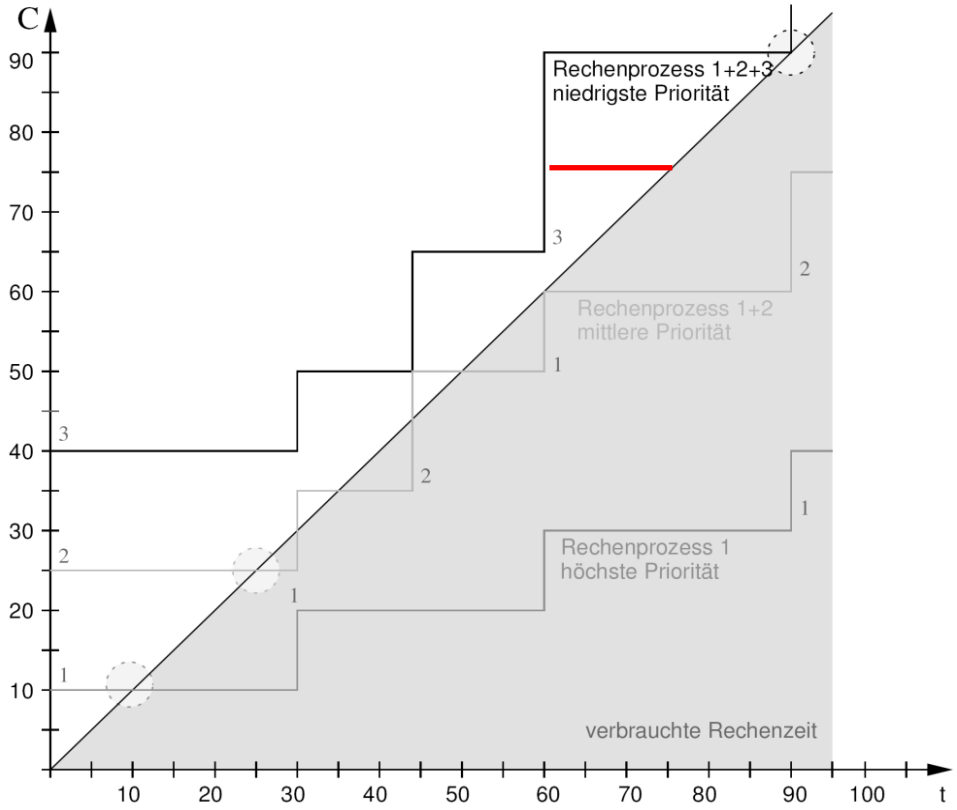


Basis: Rechenzeitanforderungsfunktion C(t)
Zusammenhang zwischen S(t) und C(t): S(t) = C(t) - A(t) = C(t) - t;

Echtzeitnachweis

Mathematischer Ansatz

Anforderung	t_V	t_P	t_{Zmax}	t_{Zmin}
1	10 ms	30 ms	30 ms	0 ms
2	15 ms	45 ms	45 ms	0 ms
3	15 ms	60 ms	60 ms	0 ms



Vorgehen:

- **Im Koordinatensystem werden Rechenzeitanforderungen gemäß ihrer Priorität aufsummiert**
- **a) Zuerst Anforderungen des höchstprioren Rechenprozesse eintragen.**
- **Schnittpunkt mit der Winkelhalbierenden: maximale Reaktionszeit des jeweils niedrigpriorsten Jobs wenn $t_{Rmax} < t_{Zmax}$ ansonsten nur der Zeitpunkt an dem bisherige Anforderungen erstmalig abgearbeitet sind → Nachweis Auslastungsbedingung <100%**
- **Ermittelung t_{Rmax} bei > t_{Zmax} : gerade betrachteter niedrigpriorste Rechenprozess wird nur einmal! aktiviert; alle höherprioren Rechenprozesse werden so oft sie kommen unterbrechen. Schnittpunkt mit Winkelhalbierender ist t_{Rmax}**
- **Wiederhole Prozedur a) mit nächst niedrigpriorem Rechenprozess**

Echtzeitnachweis

Mathematischer Ansatz

Gauss-Klammer $\lceil \rceil$ entspricht Funktion `ceil` (`ceil(2.2) = 3`)

Hyperperiode eines Satzes zyklischer Tasks: Kleinstes gemeinsames Vielfaches aller Prozesszeiten („Ablauf beginnt von vorne“)

Formal wenn $t_{Rmax} < t_{Zmax}$:

- maximale Reaktionszeit t_{Rmax} : kleinste Lösung der Gleichung »C(t)-t = 0« (s.u.)

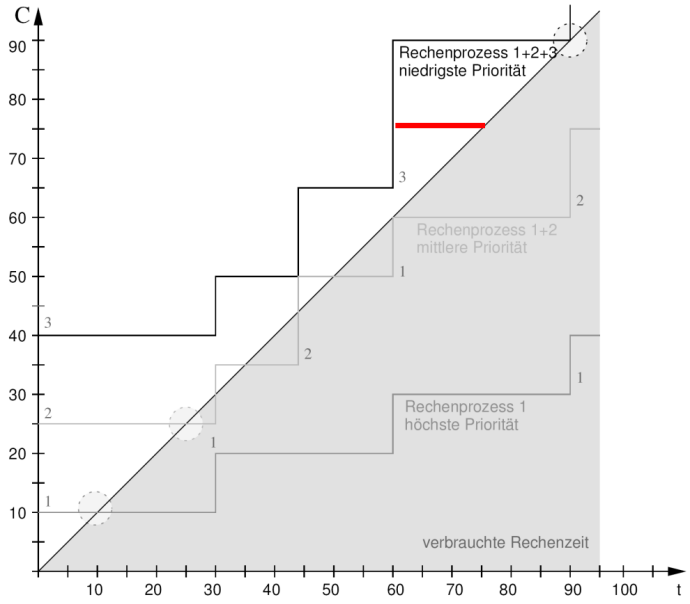
oder:

- An allen Sprungstellen einer Hyperperiode (Vielfache aller Prozesszeiten) , höchstens aber bis t_{Rmax} testen, ob $C(t) \leq t$

Berechnung von C(t):

- eigene Rechenzeitanforderung zum Zeitpunkt t und die Anforderungen aller übrigen höher- oder gleichprioren Jobs aufsummieren

$$C_i(t) = \lceil \frac{t}{t_{p,i}} \rceil \cdot t_{v,i} + \sum_j \lceil \frac{t}{t_{p,j}} \rceil \cdot t_{v,j}$$

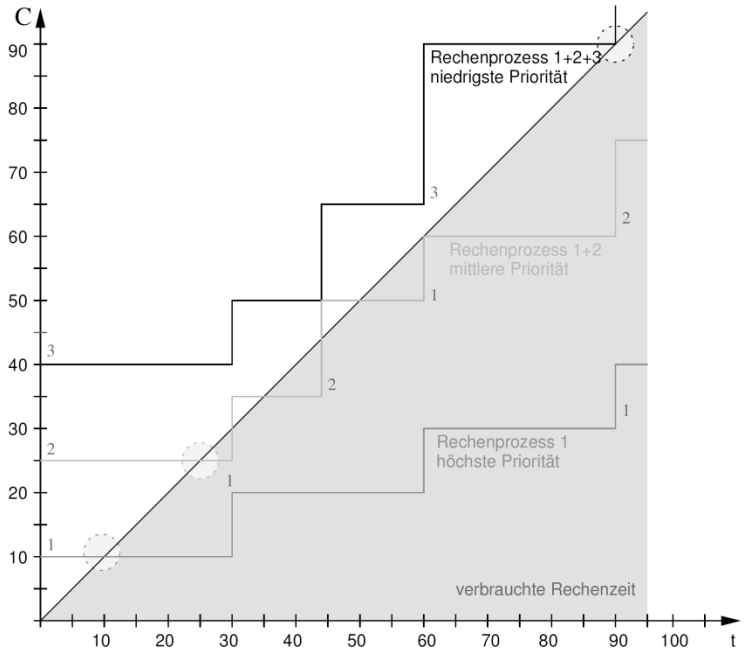


Echtzeitnachweis

Mathematischer Ansatz

Beispiel

Anforderung	t_V	t_P	t_{Zmax}	t_{Zmin}
1	10 ms	30 ms	30 ms	0 ms
2	15 ms	45 ms	45 ms	0 ms
3	15 ms	60 ms	60 ms	0 ms



$$C_i(t) = \lceil \frac{t}{t_{p,i}} \rceil \cdot t_{v,i} + \sum_j \lceil \frac{t}{t_{p,j}} \rceil \cdot t_{v,j}$$

J1: $C_1(t) = \lceil \frac{t}{30ms} \rceil \cdot 10ms$

J2: $C_2(t) = \lceil \frac{t}{45ms} \rceil \cdot 15ms + \lceil \frac{t}{30ms} \rceil \cdot 10ms$

J3: $C_3(t) = \lceil \frac{t}{60ms} \rceil \cdot 15ms + \lceil \frac{t}{30ms} \rceil \cdot 10ms + \lceil \frac{t}{45ms} \rceil \cdot 15ms$

Echtzeitnachweis

Gesucht: $t_{Rmaxi} = C(t_{Rmaxi})$

- ***Worst*-case Annahme, alle Tasks stehen jetzt t=0 gleichzeitig zur Bearbeitung an!**
- **Die erreichte Reaktionszeit für Task i sei $t_{Ri} = t_{vi} + t_{wi}$, wobei t_{vi} vorgegeben und t_{wi} wird von allen höherprioren Tasks j verursacht**
- **Prozesszeit (Periodendauer) des zu Task i dazugehörigen Prozesssignals sei t_{Pi},**
- **Annahme: Task j sei höherprior als Task i und unterbricht diese**
- **Während t_{Ri} wird Task i von Task j genau $[t_{Ri} / t_{Pj}]$ mal für die Dauer von t_{Vj} unterbrochen, also: $t_{Wi} = [t_{Ri}/t_{Pj}]*t_{Vj}$**
- **Gibt es mehr als eine Task j, die höherprior sind (Taskmenge HPRIO), so summieren sich die Unterbrechungszeiten verursacht durch jede Task j :**

$$t_{Wi} = \sum_{\forall j \in HPRIO} \left(\lceil t_{Ri} / t_{Pj} \rceil * t_{Vj} \right)$$

- **Die Taskreaktionszeit $t_{Ri} = C(t_{Ri})$ der Task i ergibt sich dann zu:**

$$t_{Ri} = t_{Vi} + \sum_{\forall j \in HPRIO} \left(\lceil t_{Ri} / t_{Pj} \rceil * t_{Vj} \right)$$

- **Suche Lösung der Gleichung (Fixpunktsuche) für Task i durch Iteration für n=0,... n aus den natürlichen Zahlen,mit dem Startwert $t^0_{Ri}=t_{Vi}$:**

$$t^{n+1}_{Ri} = t_{Vi} + \sum_{\forall j \in HPRIO} \left(\lceil t^{n}_{Ri} / t_{Pj} \rceil * t_{Vj} \right)$$

Echtzeitnachweis

Beispiel: Hyperperiode= je 600ms treten P1,P2,P3 gleichzeitig auf

→ $t_{Rmaxi} < t_{Zmaxi} < 600ms$

Prozess p_i	***p_{pimin}***	***t_{Zmaxi}***	***t_{vimax}***
P1	*150 ms*	*150 ms*	*30 ms*
P2	*100 ms*	*100 ms*	*10 ms*
P3	*200 ms*	*200 ms*	*100 ms*

$$t_{Ri}^{n+1} = t_{Vi} + \sum_{\forall j \in HPRIO} \left(\left\lceil t_{Ri}^{n} / t_{Pj} \right\rceil * t_{Vj} \right)$$

- **Task Prioritäten: P2 hoch, P1 mittel und P3 niedrig.**
- **Task P3 mit $t^0_{R3}=t_{V3}$= 100 (alles in ms): ok!** ([100/100] ==> 1; [140/100]==>2)

t^{n+1}_{R3}	$t^n_{R3}/t_{P1} * t_{V1}$ +	$t^n_{R3}/t_{P2} * t_{V2}$ +	t_{V3}
140	[100/150] * 30 +	[100/100] * 10+	100
150	[140/150] * 30+	[140/100] * 10+	100
150	[150/150] * 30+	[150/100] * 10+	100

- **Task P1 mit $t^0_{R1}=t_{V1}$= 30 (alles in ms): ok!**

t^{n+1}_{R1}	$t^n_{R1}/t_{P2} * t_{V2}$ +	t_{V1}
40	[30/100] * 10+	30
40	[40/100] * 10+	30

- **Task P2 mit $t^{n+1}_{R2}= t^n_{R2}= t^0_{R2}=t_{V2}$= 10,0 ms ok!**

Echtzeitnachweis

Hinreichende Schedulingbedingung

Voraussetzungen:

- **Scheduling mit statischen Prioritäten (Rate Monotonic, ...)**
- **Zyklische Task k ohne Abhängigkeiten ➔worst case: t_{Vmaxk},t_{Pmink}**
- **t_{Zmax}= t_{Pmin}; Trick: Maximal zulässige Reaktionszeiten durch entsprechende Prozesszeiten modellieren (min($t_{Zmax,k}$, $t_{Pmin,k}$))**

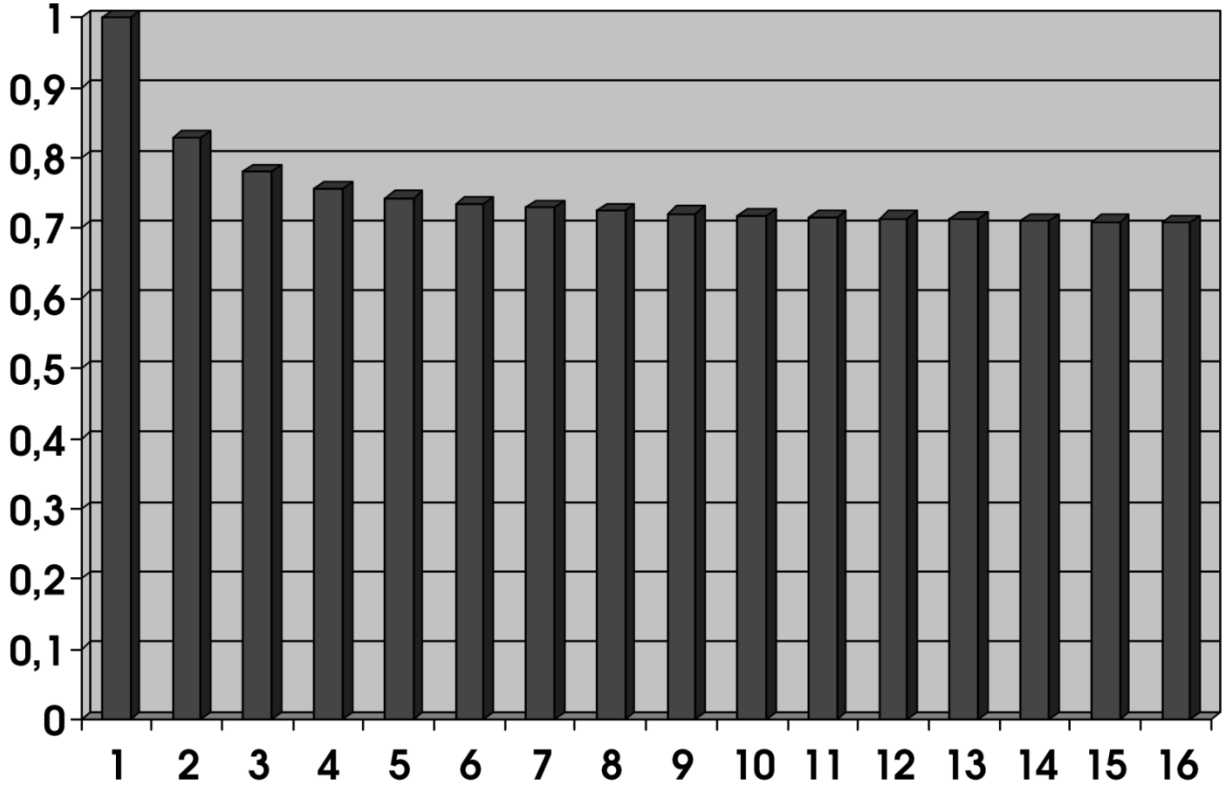
$$\mu = \sum_{k=1}^{i} \frac{t_{Vmax,k}}{\min(t_{Zmax,k},\ t_{Pmin,k})} \leq i(2^{1/i} - 1)$$

i	Auslastungs-grenze (in %)
1	100
2	82,8
3	78
4	75,7
5	74,3
10	71,8
$n \to \infty$	69,3

Aussage: ➔falls Bedingung für Auslastung erfüllt, ist auch Rechtzeitigkeitsbedingung erfüllt!!!
➔Umkehrung gilt nicht ➔falls nicht kann trotzdem erfüllt sein ➔ graph. Verfahren

Hinreichende Schedulingbedingung

Auslastung bei n Tasks < $U(n) = n(2^{1/n} - 1)$



Echtzeitnachweis

Ein Übungsbeispiel...

Wie werden folgende Tasks priorisiert?

1. Echtzeitbedingung

Hinreichende Scheduling-Bedingung

$$\mu = \sum_{k=1}^{i} \frac{t_{V,k}}{\min(t_{Zmax,k},\ t_{P,k})} \leq i(2^{1/i} - 1)$$

2. Echtzeitbedingung mit RZ-Anforderungsfunktion (grafisch)

Anforderung	t_V	t_P	t_{Zmax}
1	1 ms	6 ms	6 ms
2	1 ms	3 ms	3 ms
3	1.5 ms	15 ms	10 ms
4	0.5 ms	5 ms	5 ms

i	Auslastungs-grenze (in %)
1	100
2	82,8
3	78
4	75,7
5	74,3
10	71,8
$n \to \infty$	69,3

1. Ohne t_{Zmax}

2. Mit t_{Zmax} aus Tabelle

3. Mit t_{Zmax} aus Tabelle, aber → $t_{v3*} = n * t_{v3}$; t_{v3*} Maximum möglich ohne Echtzeitverletzung?

Echtzeitnachweis

Noch ein Übungsbeispiel zu Hause!...

Wie werden folgende Tasks priorisiert?

1. Echtzeitbedingung

Hinreichende Scheduling-Bedingung

2. Echtzeitbedingung mit RZ-Anforderungsfunktion (mathematisch und grafisch)

Anforderung	t_V	t_P	t_{Zmax}
1	1 ms	5 ms	5 ms
2	3 ms	10 ms	10 ms
3	3 ms	15 ms	9 ms