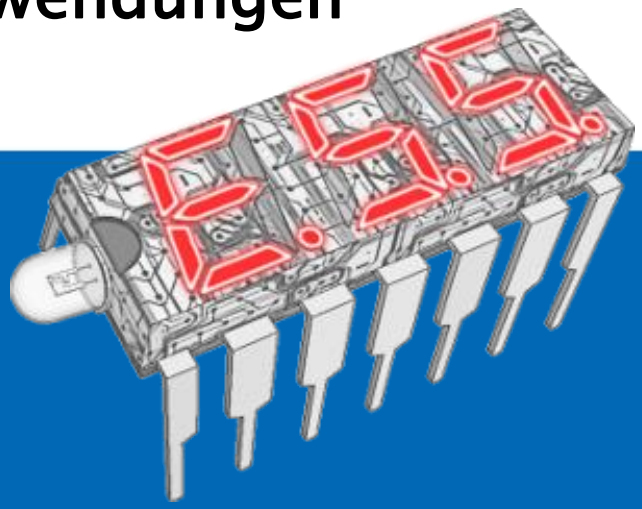
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Prinzipen und Komponenten Eingebetteter Systeme (PKES)

(9) Scheduling in eingebetteten Anwendungen

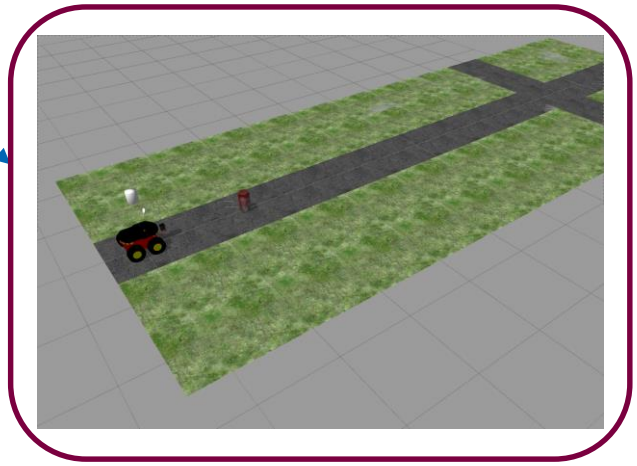


Sebastian Zug
Arbeitsgruppe: Embedded Smart Systems

Robotikaffine Studenten gesucht!

Autonomes Fahrrad

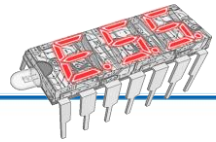
ROS basiertes Android Steuerelement



Gazebo Simulation

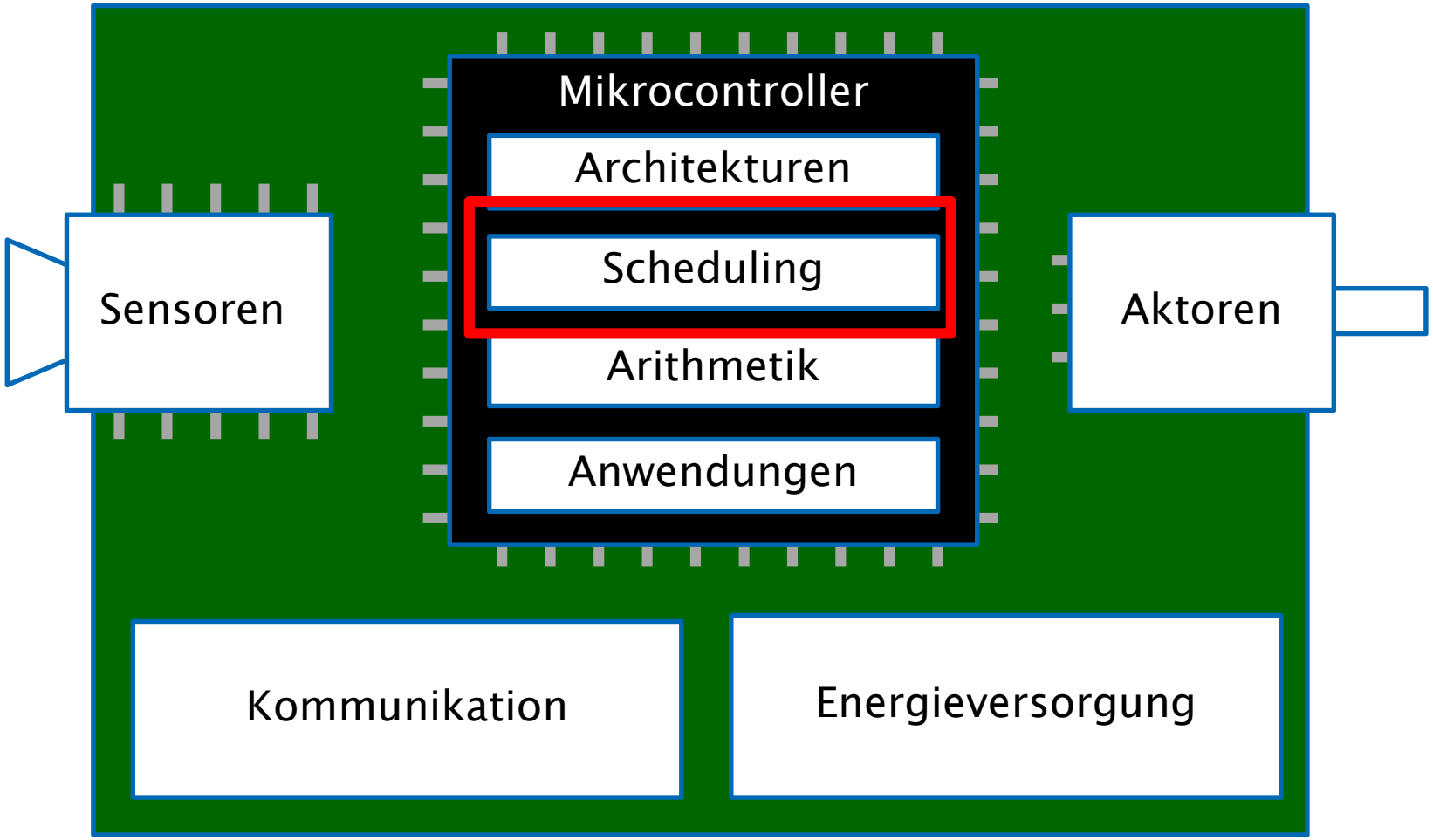
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



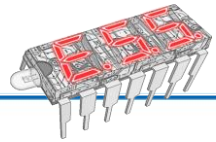
„Veranstaltungslandkarte“

Fehlertoleranz, Softwareentwicklung



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

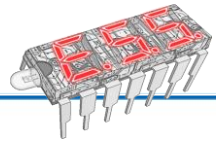


Fragestellungen dieser Vorlesung

1. Begründen Sie die Notwendigkeit des Schedulings anhand der auf der Motivationsfolie dargestellten automtiven Anwendungen.
2. Beschreiben Sie anhand des Taskmodells aus der Betriebssystemvorlesung die verschiedenen Level des Schedulings.
3. Welche Aspekte charakterisieren eine Taskmenge?
4. Wodurch zeichnen sich sporadische Tasks aus?
5. Definieren Sie harte und weiche Echtzeitfähigkeit!
6. Nennen Sie Kostenfunktionen für echtzeitfähiges Scheduling!
7. Erklären Sie die notwendigen/hinreichenden Kriterien für die Einplanbarkeitsanalyse?
8. Nach welchen Kriterien können Scheduling Ansätze strukturiert werden?

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



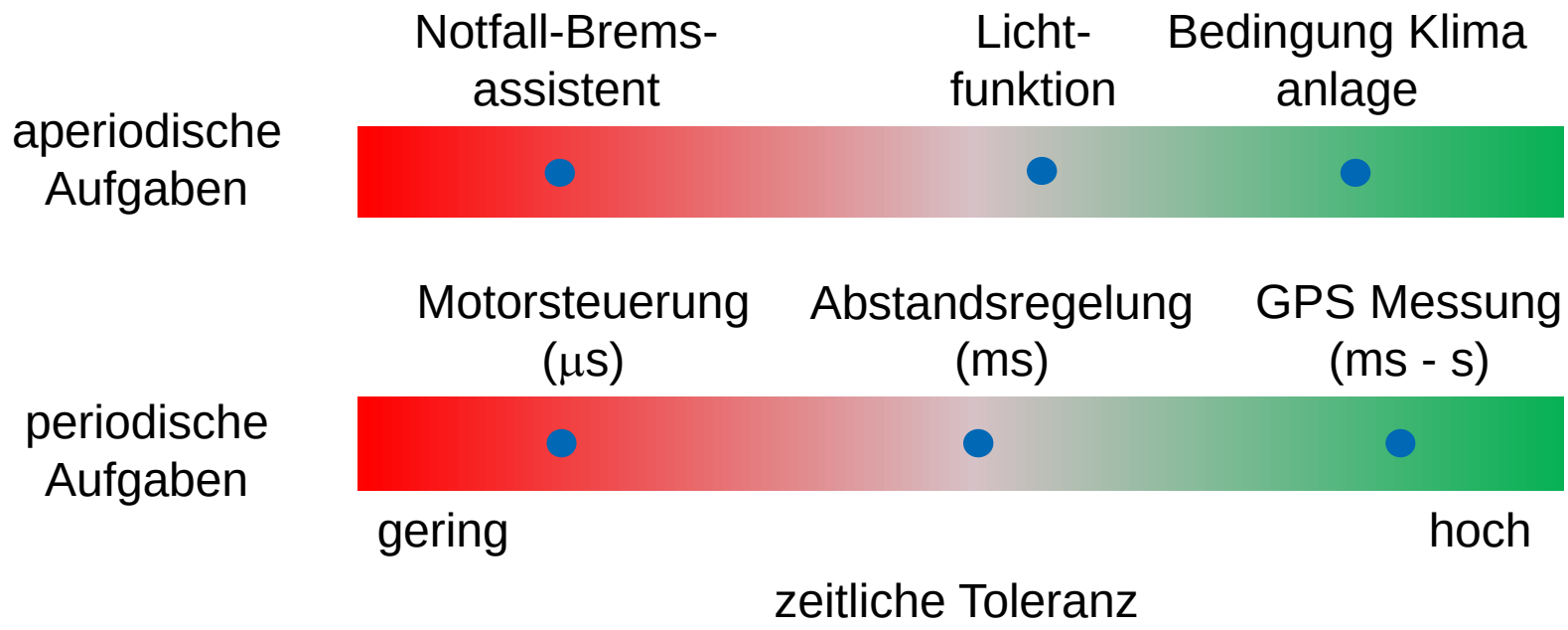
Literaturhinweise

- Peter Marwedel
  Eingebettete Systeme
  Springer Lehrbuch, 2008
- Dieter Zöbel
  Echtzeitsysteme – Grundlagen der Planung
  Springer Lehrbuch, 2008

Warum überhaupt?

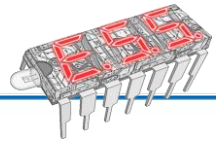
Beschränkte Ressourcen werden mit einer Vielzahl von Aufgaben unterschiedlicher Priorität konfrontiert.

Source: Autobild.de



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Echtzeit

- Echtzeitbetrieb nach DIN 44300 … Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse **innerhalb einer vorgegebenen Zeitspanne** verfügbar sind. Die Daten können je nach Anwendungsfall nach einer **zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen**.

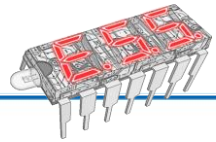
- Harte Echtzeit (Rechtzeitigkeit – timeliness)
  = die Abarbeitung einer Anwendung wird innerhalb eines bestimmten Zeithorizontes umgesetzt

$$A \equiv r + \Delta e \leq d$$

- Weiche Echtzeit
  = es genügt, die Zeitbedingungen für den überwiegenden Teil der Fälle zu erfüllen, geringfügige Überschreitungen der Zeitbedingungen sind erlaubt

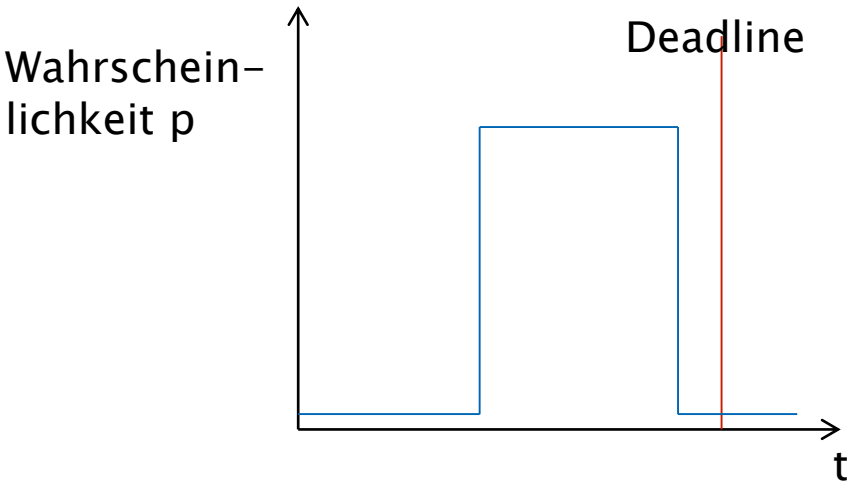
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

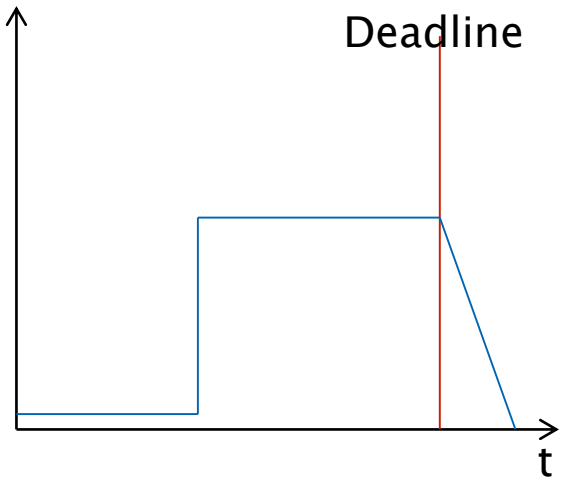


Graphische Darstellung Echtzeitbedingung

Harte Echtzeit



Weiche Echtzeit



$$A \equiv r + \Delta e \leq d$$

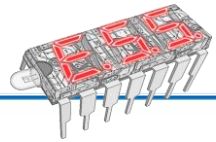
$$P(A \mid B) = 1$$

$$P(A \mid B) < 1$$

B=Kontext der Anwendung

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Intuitive Lösung

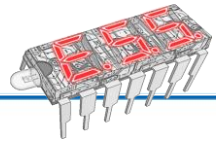
Echtzeitimplementierung als „nanokernel“

- Ausrichtung an einer minimalen festen Periode p
- Evaluation des Laufzeitverhaltens notwendig
- keine Schutzfunktionen des Speichers
- Polling als einzige Zugriffsfunktion auf die Hardware

```
switch off interrupts
setup timer
c = 0;
while (1) {
   suspend until timer expires
   c++;
   Task0();    //do tasks due every cycle
   if (((c+0) % 2) == 0)
      Task1(); //do tasks due every 2nd cycle
   if (((c+1) % 3) == 0)
      Task2(); //do tasks due every 3rd cycle, with phase 1
...
}
```

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Intuitive Lösung

Vorteile

✓ Einfache Umsetzbarkeit auf einem Mikrocontroller
✓ Vereinfachter Hardwarezugriff

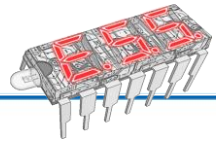
Nachteil

– keine Prioritäten für die einzelnen Anwendungen
– keine Adaption zur Laufzeit
– keine Interrupts
– keine Unterbrechung
– beschränkte Wiederverwendbarkeit des Codes
– nur für einfache Systeme

Systematisches Scheduling erforderlich

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

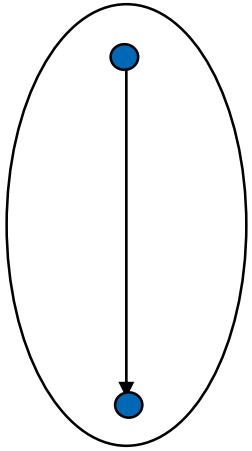
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Taskmodell

Eine Task ist die Ausführung eines sequentiellen Programms auf einem Prozessor in seiner spezifischen Umgebung (Kontext).

Eine Task ist:
- erfüllt eine von Programm spezifizierte Aufgabe
- Träger der Aktivität = Abstraktion der Rechenzeit
- die kleinste planbare Einheit



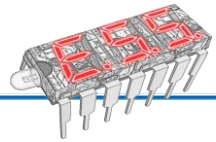
Start

Task als sequentielle und endliche Ausführungsfolge von Instruktionen

Abschluß

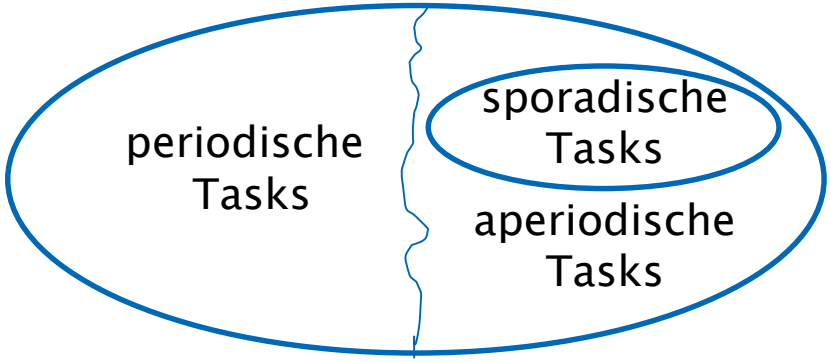
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

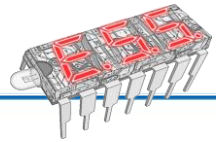


Charakterisierung von Tasks – Zeitverhalten

- Periodische Tasks … werden mit einer bestimmten Frequenz *f* regelmäßig aktiviert.
  - Durchlaufen einer Regelschleife
  - Pollendes Abfragen eines Sensors
- Aperiodische Tasks … lassen sich nicht auf ein zeitlich wiederkehrendes Muster abbilden.
  - Tastendruck auf einem Bedienfeld
- Sporadische Tasks … treten nicht regulär auf. Man nimmt aber eine obere Schranke bzgl. der Häufigkeit ihres Aufrufs an.
  - Fahrradtacho (obere Schranke = Geschwindigkeit)



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



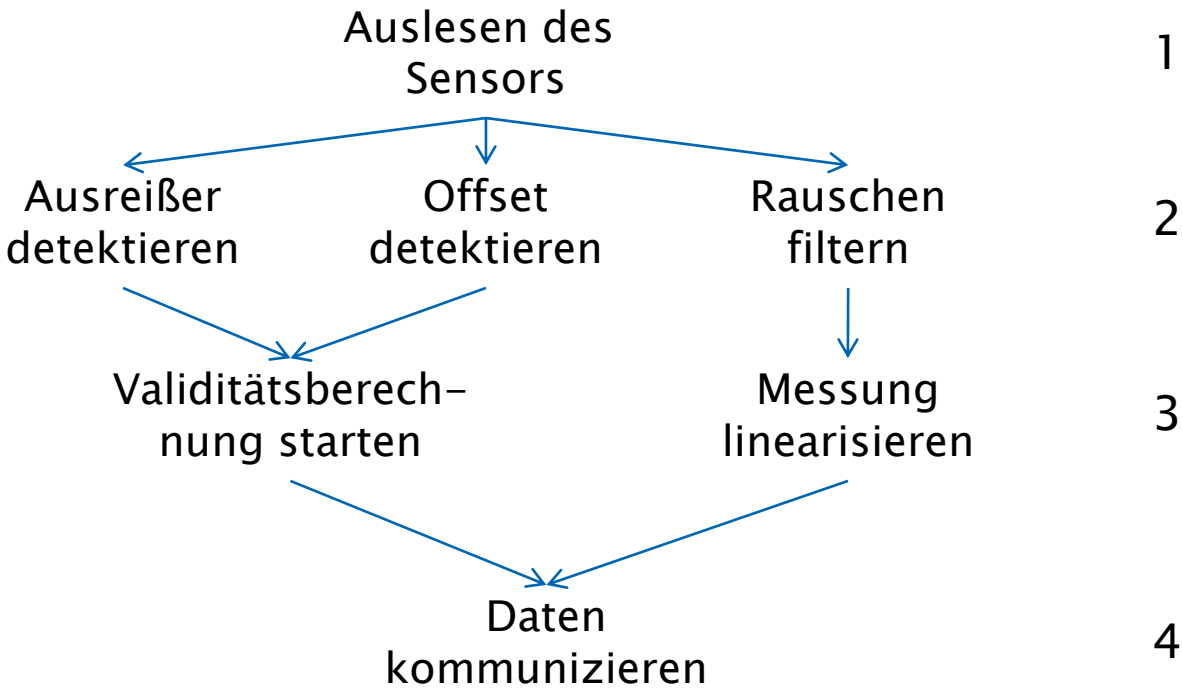
Charakterisierung von Tasks – Abhängigkeiten

unabhängige Tasks

- können in jeder beliebigen Reihenfolge ausgeführt werden

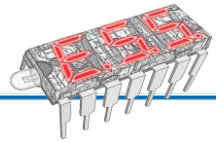
abhängige Tasks

- Abhängigkeiten sind in einem Precedencegraphen darstellbar als “vorher”-Relation



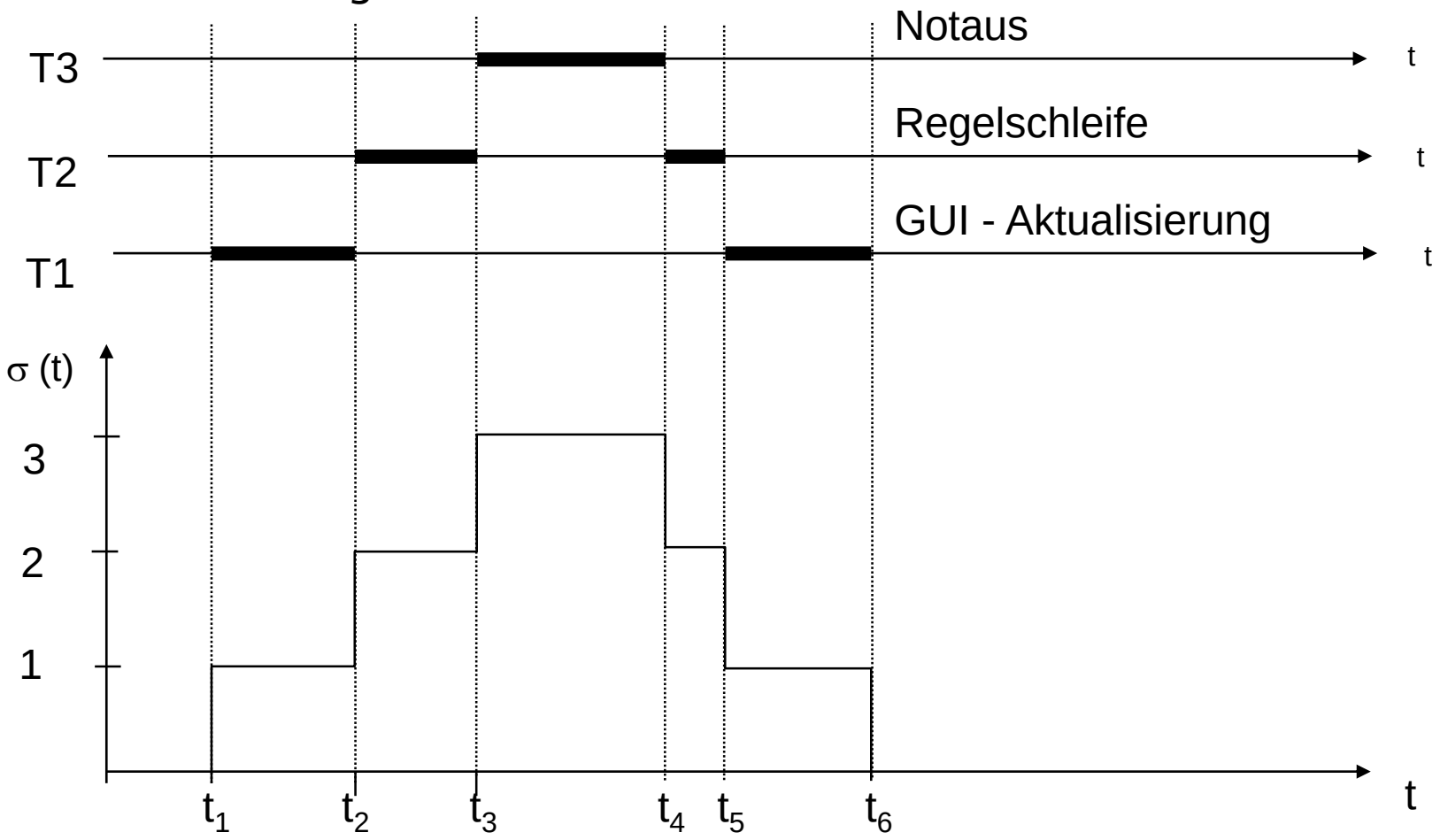
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



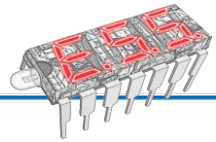
Charakterisierung von Tasks – Unterbrechbarkeit

- Darf die Abarbeitung der Task unterbrochen werden?



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

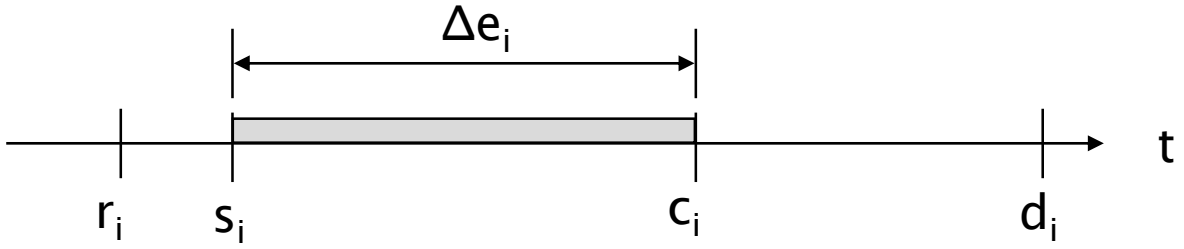
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



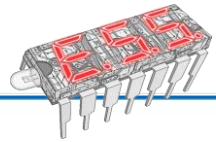
Parameter einer Task

- T …Tasktyp (Abfragen des Temperaturfühlers)
- T_i … i-te Instanz des Tasktyp (Taskobjekt)
- T_i^j … j-te Ausführung des Taskobjektes T_i

T_i	Instanz i der Task T
r_i	Bereitzeit (ready time)
Δe_i	Ausführungszeit (execution time)
s_i	Startzeit (starting time)
c_i	Abschlußzeit (completion time)
d_i	Frist (deadline)



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



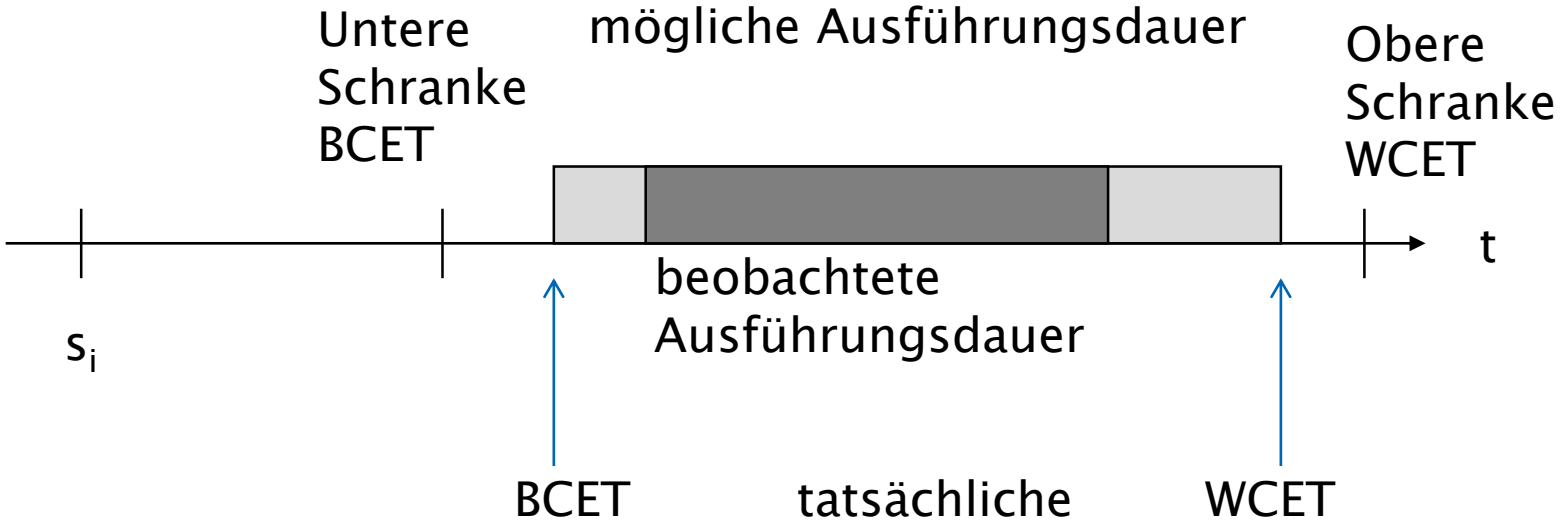
Exkurs

Herausforderung Ausführungsdauer Δe_i

Definiert über

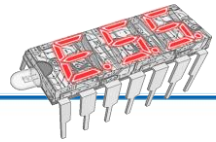
- die Programmlogik (Kontrollflussgraph),
- die Eingabedaten (Auswirkungen auf Schleifendurchlaufzahlen etc.),
- den Compiler (Optimierungsstufe) und
- die Architektur und Taktfrequenz des Ausführungsrechners (Cache- und Pipelining-Effekten)

Lösung: Annahme einer Worst-Case-Execution-Time



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

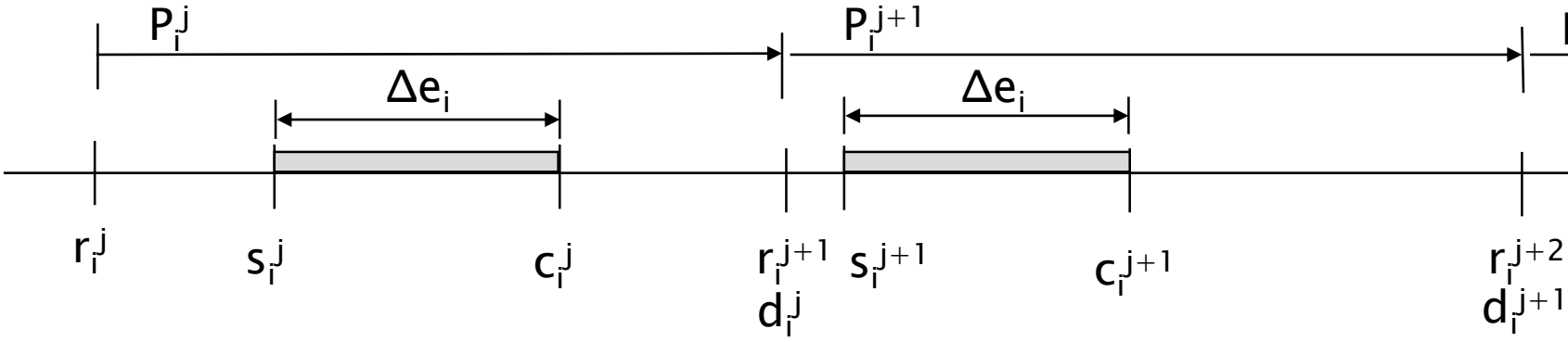
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Erweiterung auf periodische Tasks

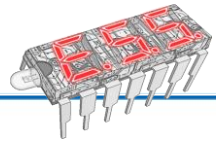
- T …Tasktyp (Abfragen des Temperaturfühlers)
- T_i … i-te Instanz des Tasktyp (Taskobjekt)
- T_{ij} … j-te Ausführung des Taskobjektes T_i

T_i	Instanz i der Task T
r_i	Bereitzeit (ready time)
Δe_i	Ausführungszeit (execution time)
s_i	Startzeit (starting time)
c_i	Abschlußzeit (completion time)
d_i	Frist (deadline)
Δp_i	Periode
Δj_i	Zeitspanne zwischen zwei Startzeiten (Jitter)



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Scheduling

Allgemeine Formulierung des Schedulingproblems:

Gegeben seien:

Eine Menge von Tasks	$T = \{T_1, T_2, ..., T_n\}$
Eine Menge von Prozessoren	$P = \{P_1, P_2, ..., P_m\}$
Eine Menge von Ressourcen	$R = \{R_1, R_2, ..., R_s\}$

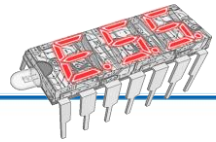
Scheduling bedeutet die Zuordnung von Prozessoren und Ressourcen zu Tasks, so dass alle für individuelle Tasks definierten Beschränkungen eingehalten werden.

In seiner allgemeinen Form ist das Scheduling-Problem NP-vollständig.

Einschränkungen: Anzahl der betrachteten Prozessoren/Ressourcen,
nur periodische Tasks,
gleiche Bereitzeiten aller Tasks,
keine Anhängigkeiten,
nur feste Prioritäten, ...

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Übliche (nicht Echtzeit) Ziele

Mittlere Antwortzeit:
(avarage response time)

$$t_r = 1/n \sum_{(i=1, \ldots, n)} (c_i - r_i)$$

Zeit bis zum vollständigen Abschluß:
(total completion time)

$$t_c = \max_i (c_i) - \min_i (r_i)$$

Gewichtete Summe der Abschlußzeiten:

$$t_w = \sum_{i\ (i=1, \ldots, n)} w_i c_i$$

Hohe Auslastung des Systems

$$r = (\sum_{(i=1, \ldots, n)} \Delta e_i)/t$$

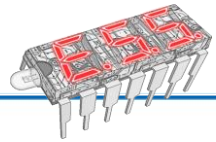
nicht Echtzeit

Verfahren des Nicht-Echtzeitscheduling können nicht übernommen werden:

- Keine Deadlines (kein *d*)
- Keine unterschiedlichen Prioritäten der Tasks (Fairness)
- Kurze Reaktionszeiten genügen nicht, Zeiten müssen garantiert sein
- Berücksichtigung weiterer Parameter:
  - Periode
  - Abhängigkeiten von anderen Tasks (Taskgraph)

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Echtzeit Kostenfunktionen

Maximale Zahl verspäteter Tasks:

$$N_{late} = \sum_{(i=1, \ldots, n)} miss\ (c_i)$$

mit:

$$miss\ (c_i) = \begin{cases} 0 & if\ c_i \leq d_i \\ 1 & sonst \end{cases}$$

Maximale Verspätung:
(maximum lateness)

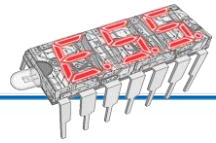
$$L_{max} = \max_i (c_i - d_i)$$

Für Prozesse mit harten Zeitbedingungen gilt dabei:

$$L_{max} \leq 0$$

Die vorgegebene Deadline muss immer eingehalten werden.

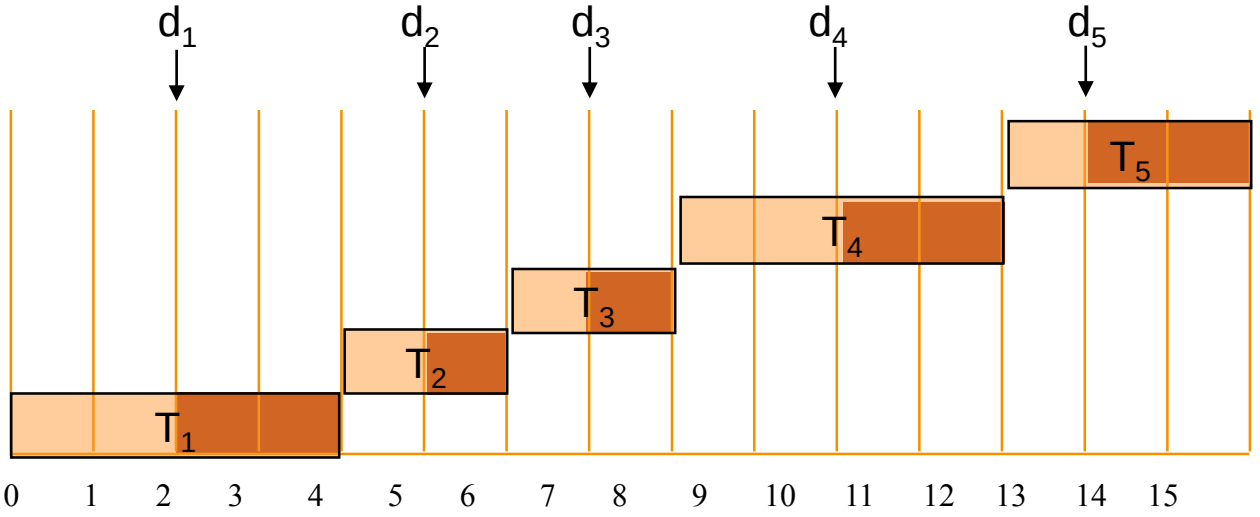
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



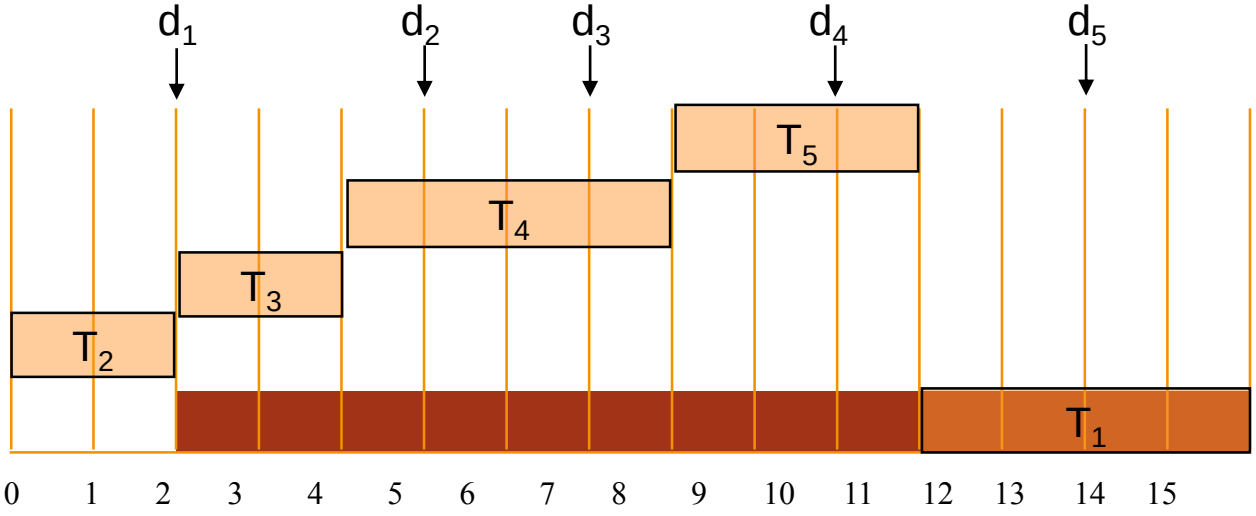
Mögliche Kostenkonflikte

Bereitzeit $r_i = 0$

$L_{max} = L_1 = L_4 = L_5 = 2$
$N_{late} = 5$

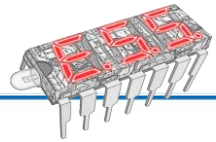


$L_{max} = L_1 = 13$
$N_{late} = 1$



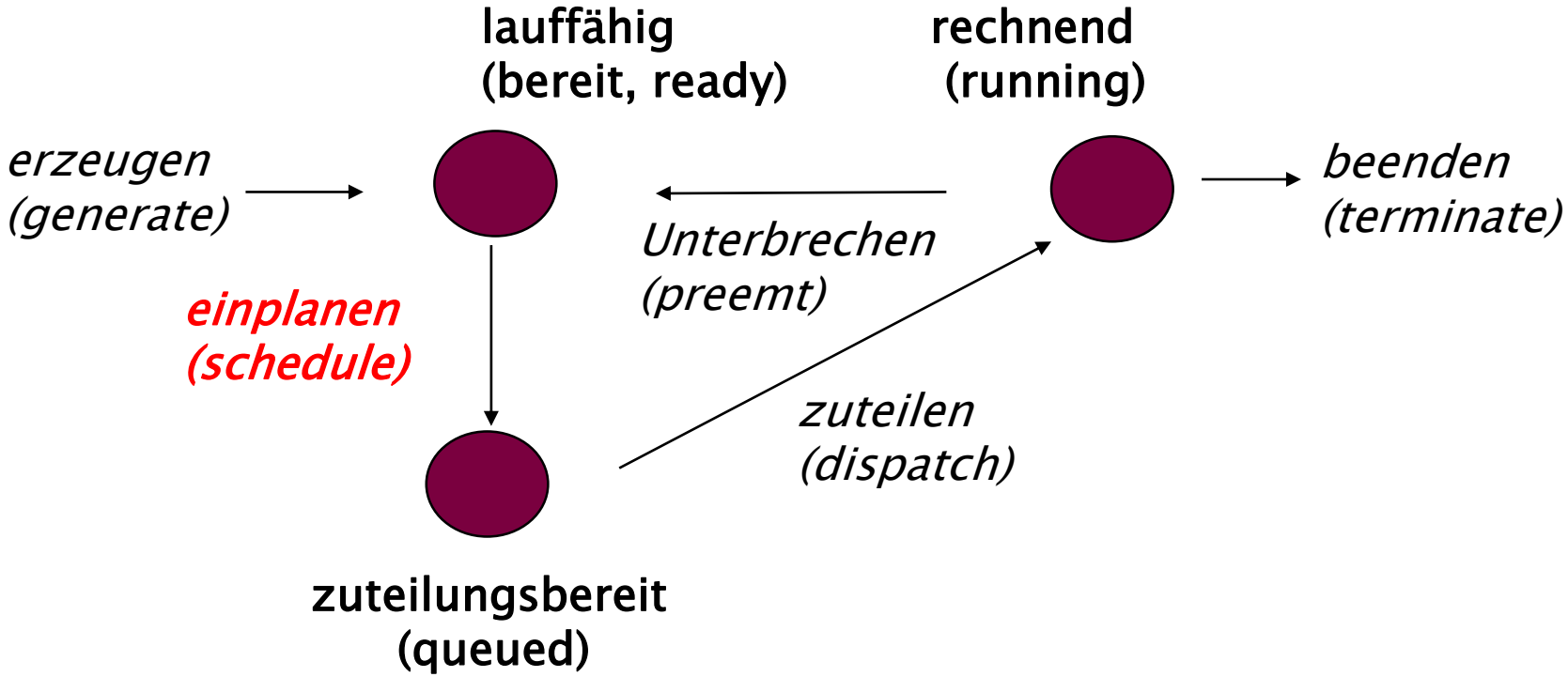
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

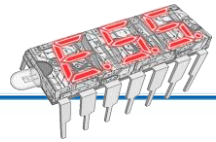


Vorgehen

1. Einplanbarkeitsanalyse (*feasibility check*) — *Strategisches Scheduling*
2. Planerstellung (*schedule construction*) — *Operatives Scheduling*
3. Prozessorzuteilung (*dispatching*) — *Dispatching*



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

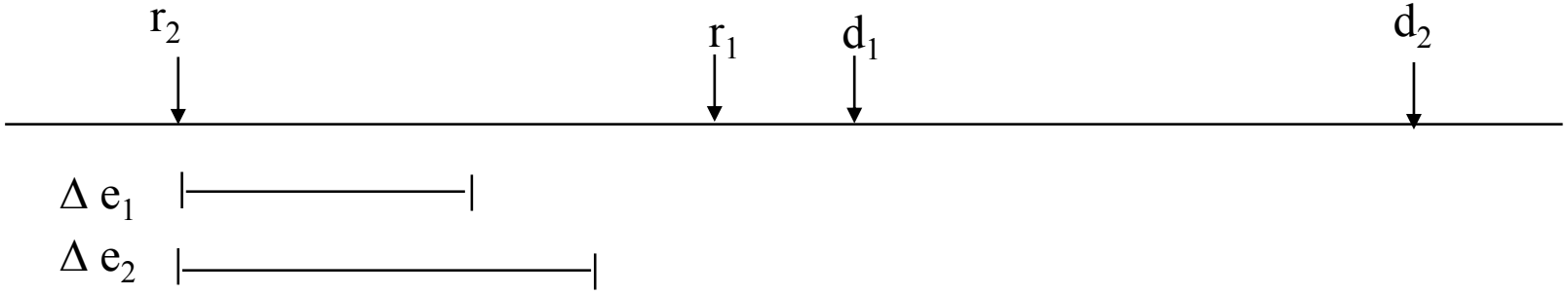


Einplanbarkeitsanalyse I

1. Korrekte Konfiguration der Tasks

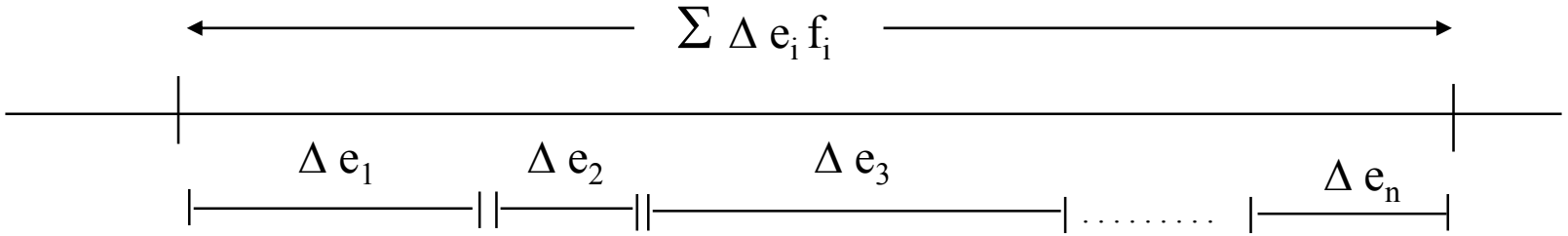
Die Deadline muss hinreichend weit von der Startzeit entfernt sein

$$\forall i \; : \Delta e_i < (d_i - r_i) < \Delta p_i \quad (\Delta p_i = 1/f_i)$$



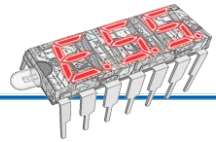
2. verfügbare (Prozessor-) Zeit

$$\sum \Delta e_i f_i \leq d_{max}$$



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



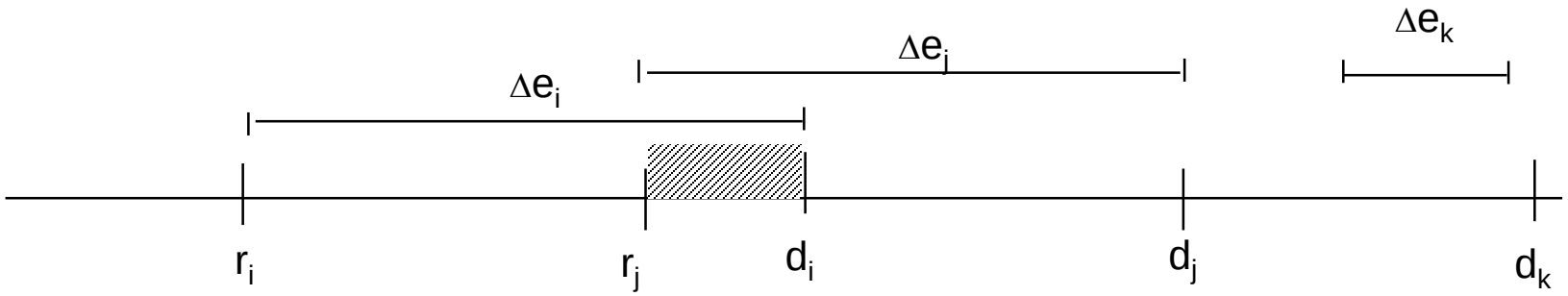
Einplanbarkeitsanalyse II

3. Überlappende Tasks

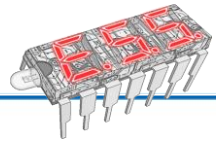
Die letztmögliche Ausführungszeit zweier Tasks überlappt nicht.

$$\forall i,j\ ,\ d_i < d_j : \quad d_i \ \leq\ d_j \ -\ \Delta e_j$$

Achtung: Das Scheitern des Tests schließt die Existenz eines gültigen Schedules nicht aus!



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



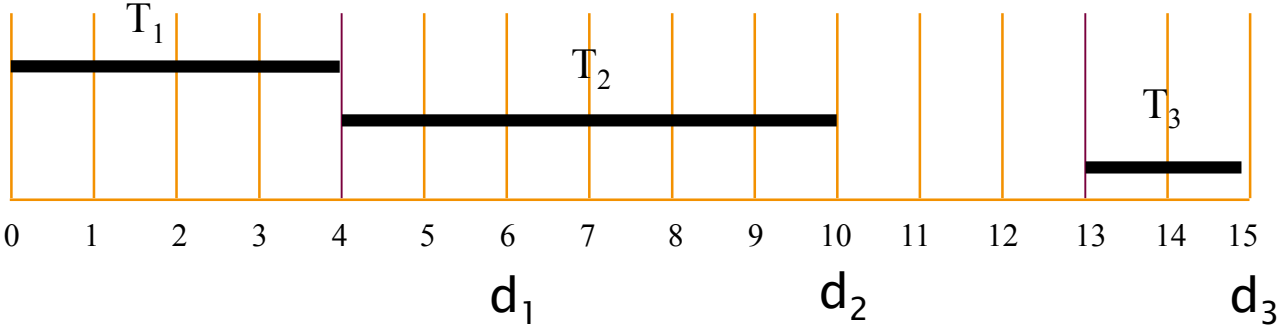
Beispiel

$\forall i,j\ ,\ d_i < d_j :\quad d_i \ \leq\ d_j \ -\ \Delta e_j$

In beiden Fällen wird die Bedingung nicht erfüllt !

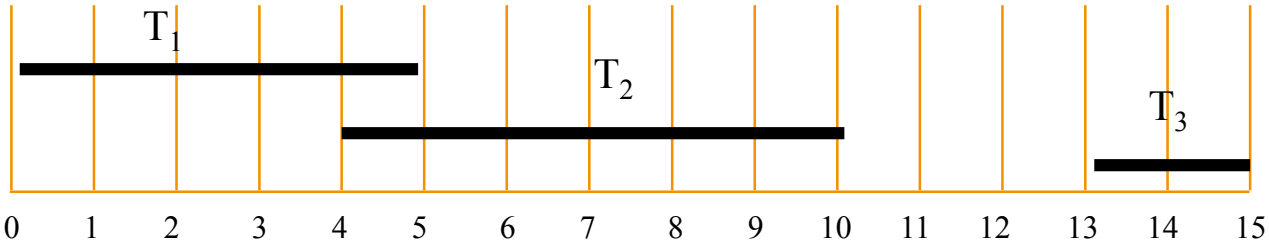
$T_1 : \Delta e_1 = 4, r_1 = 0, d_1 = 6$
$T_2 : \Delta e_2 = 6, r_2 = 4, d_2 = 10$
$T_3 : \Delta e_3 = 2, r_3 = 13, d_3 = 15$

➡ planbar !!



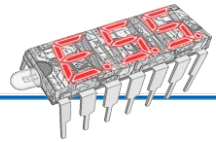
$T_1 : \Delta e_1 = 5, r_1 = 0, d_1 = 6$
$T_2 : \Delta e_2 = 6, r_2 = 4, d_2 = 10$
$T_3 : \Delta e_3 = 2, r_3 = 13, d_3 = 15$

➡ nicht planbar !!



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Zusammenfassung Einplanbarkeitskriterien

Def.: Ein Plan heißt brauchbar (valid, feasible) für eine Taskmenge $\{ T_1, T_2, \ldots, T_n \}$, falls bei vorgegebenem r_i, Δe_i, d_i und Δp_i die Startzeit s_i und die Abschlußzeit c_i so gewählt sind, dass:

1.) Alle Zeitbedingungen (für eine einzelne Task) eingehalten werden

$\forall i : \Delta e_i < d_i - r_i < \Delta p_i$ (notwendig)

2.) genügend Ressourcen vorhanden sind

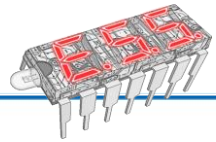
$\sum \Delta e_i f_i <$ verfügbare Prozessorzeit (notwendig)

3.) keine überlappenden Ausführungszeiten entstehen

$\forall i,j , d_i < d_j : \quad d_i \leq d_j - \Delta e_j$ (hinreichend)

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Klassifizierung nach dem Zeitpunkt der Planung I

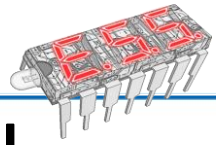
Statische Verfahren (offline-scheduling):

- Analyse der Durchführbarkeit zur Entwicklungszeit
- Fester Plan, wann welche Task beginnt (Task-Beschreibungs-Liste, TDL)
- Planung für periodische Tasks basierend auf Superperiode
- unflexibel gegenüber Änderungen
- maximale Auslastung stets erreichbar
- kaum Aufwand zur Laufzeit

ZEIT	Aktion	WCET
10	starte T1	16
15	sende M5	
26	stoppe T1	
33	starte T2	37
...	...	
...		

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Klassifizierung nach dem Zeitpunkt der Planung II

Dynamische Verfahren:

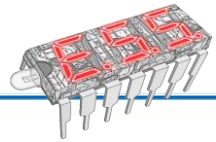
- Nicht adaptive Verfahren
- Offline Analyse der Task-Laufzeit
- Online Analyse der Durchführbarkeit von Scheduling-Plänen
- bei Ankunft einer Task wird ihr basierend auf ihren Eigenschaften eine Priorität zugewiesen

- Adaptive Verfahren
- Offline Analyse der Task-Laufzeit; Schätzung der durchschn. Ausführungszeit
- Online Analyse der Durchführbarkeit von Scheduling-Plänen
- Fehlertoleranz notwendig, da keine Garantie gegeben werden kann

Schedulingverfahren	Planung	Laufzeitanalyse
Statisch	offline	offline
Dynamisch, nicht adaptiv	online	offline
Dynamisch, adaptiv	online	online

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

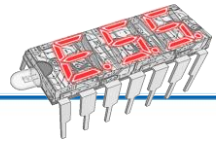


Zusammenfassung

Anforderung der Anwendung	harte Echtzeit	weiche Echtzeit
	synchron bereit	asynchron bereit
Taskmodell	periodisch	aperiodisch / sporadisch
	präemptiv	nicht-präemptiv
	unabhängig	abhängig
Scheduler	statisch	dynamisch

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



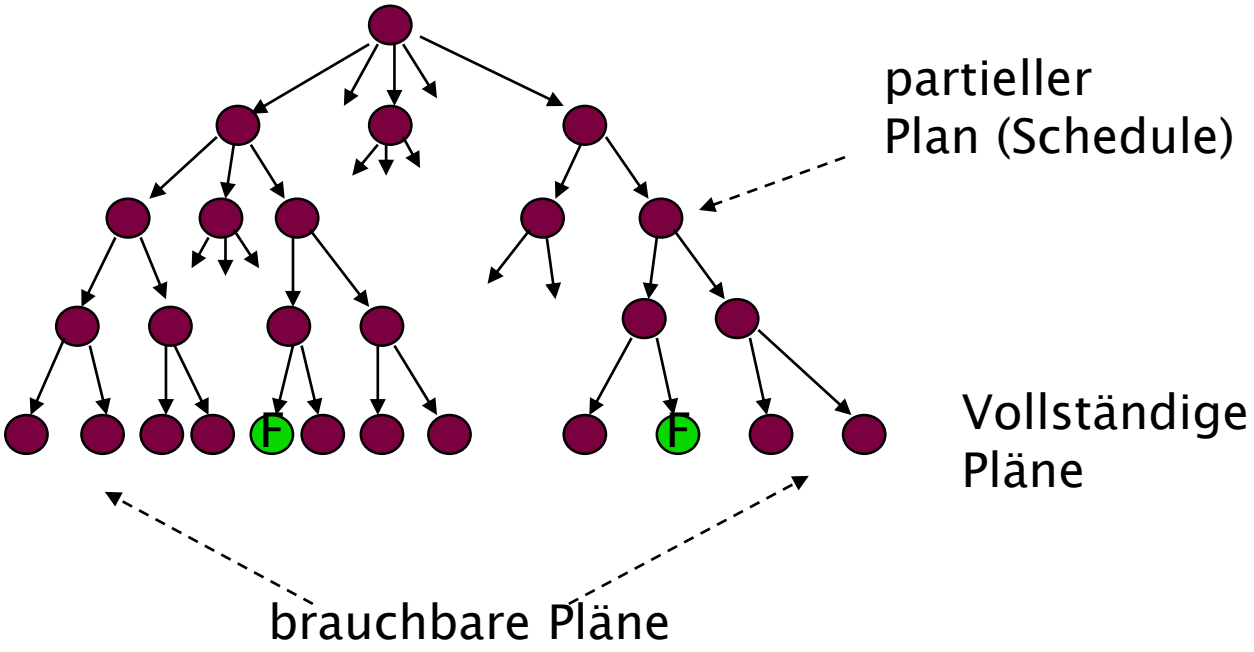
Basiskonzepte des Scheduling – Durchsuchen

Gegeben : Menge T nicht unterbrechbarer Tasks mit

$$|T| = n, \quad r_i , \Delta e_i , c_i .$$

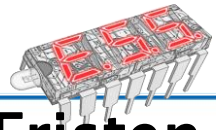
Ein statisches Planungsverfahren soll untersuchen, ob ein Plan existiert. Das Verfahren soll einen Plan in Form einer Folge von Tupeln (i, s_i) mit i : Tasknr. und s_i : Startzeit von T_i generieren.

Erzeugung des gesamten Suchraumes



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Planung unter Einbeziehung von Breitzeiten und Fristen

Bratley`s Algorithmus (P. Bratley; M. Florian, P. Robillard: “Scheduling with earliest start and due date contstraints”, Naval Research Quarterly, 18 (4), 1971

feasible PL (PL_k, i) : Funktion, die testet, ob Task T_i in den bisherigen Plan PL_k integriert werden kann, d.h. ob überhaupt noch eine einplanbare Lücke für Task T_i existiert.

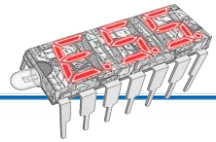
earliest PL (PL_k, i) : Funktion, die bezogen auf den Plan PL_k die Task T_i in die früheste Lücke einplant, die

1. aufgrund der Bereitzeit r_i von T_i
2. aufgrund des bereits bisher erstellten Plans PL_k möglich ist.

schedule (PL_k, X_k) :
FORALL (i IN $T \setminus X_k$) AND *feasible PL* (PL_k, i)
schedule (*earliest PL* (PL_k, i), $X_k \cup \{ i \}$)

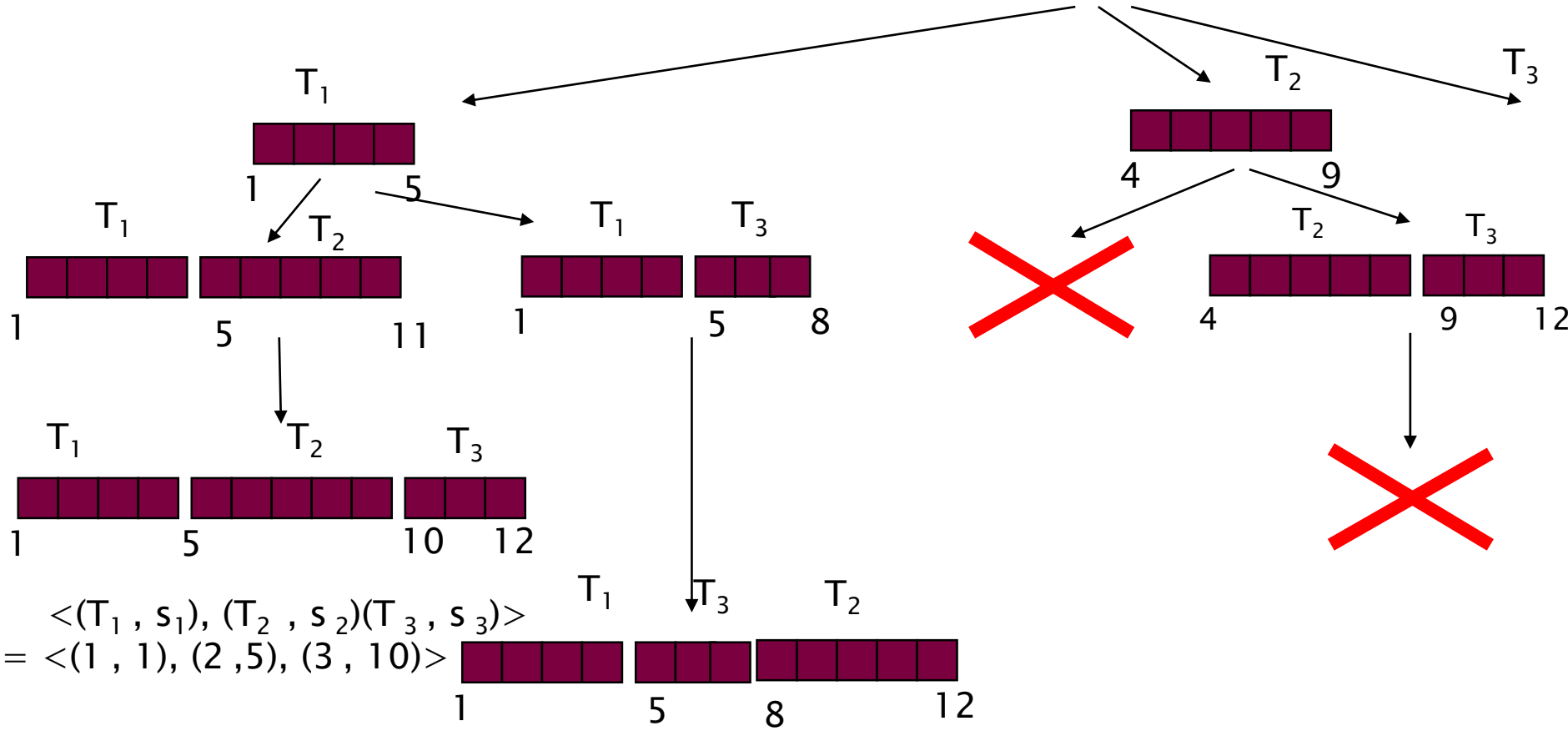
(T: Menge der Tasks, X: Menge der Tasks, die schon eingeplant sind)

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug



Beispiel

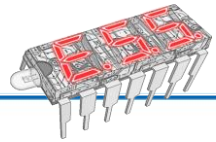
T_1 , r_1 , d_1 , Δe_1	1, 1, 7, 4
T_2 , r_2 , d_2 , Δe_2	2, 4, 12, 5
T_3 , r_3 , d_3 , Δe_3	3, 0, 14, 3



$<(T_1 , s_1), (T_2 , s_2)(T_3 , s_3)>$
$= <(1 , 1), (2 ,5), (3 , 10)>$

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

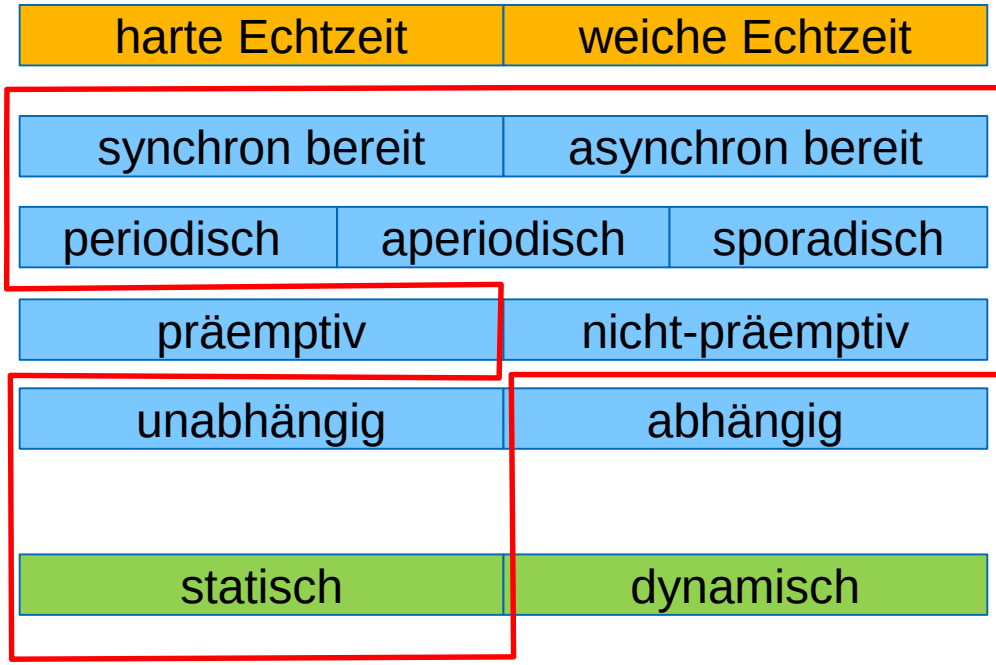


Probleme des Suchansatzes

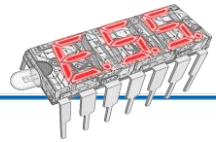
- Die kombinatorische Erzeugung aller Pläne benötigt n! Planungsschritte bei n Tasks.
- Tatsächlich ist das Suchverfahren NP-vollständig

Schlechte Nachricht:

Falls die Bereitzeiten nicht alle gleich sind und nicht-unterbrechbare Tasks vorliegen, gibt es kein anderes Verfahren das optimal ist.



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK
Arbeitsgruppe ESS
Jun.-Prof. Dr. Sebastian Zug

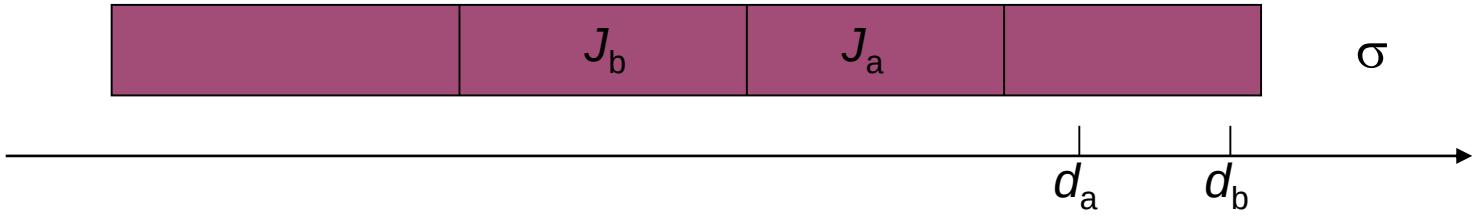


Earliest Due Date

Scheduling auf **einem** Prozessor. Alle *n* Tasks sind unabhängig voneinander und können zur gleichen Zeit begonnen werden (zum Zeitpunkt 0).

EDD: Earliest Due Date (Jackson, 1955)
Jeder Algorithmus, der die Tasks in der Reihenfolge nicht abnehmender Deadlines ausführt, ist optimal bzgl. der Minimierung der maximalen Verspätung.

Beweis nach (Buttazzo, 2002): Sei A ein Algorithmus, der verschieden von EDD ist. Dann gibt es zwei Tasks J_a und J_b in dem von A erzeugten Schedule σ, so dass in σ J_b unmittelbar vor J_a steht, aber $d_a \leq d_b$ ist:



Bis zur nächsten Woche ...

www.ovgu.de