EMES: Eigenschaften mobiler und eingebetteter Systeme

Einführung

Dr. Felix Salfner, Dr. Siegmar Sommer

Wintersemester 2010/2011

Regularien

- EMES - Eigenschaften mobiler und eingebetteter Systeme
  - Dozenten: Dr. Felix Salfner, Dr. Siegmar Sommer
  - Webseite: http://www.rok.informatik.hu-berlin.de/rok/teaching/WS-1011/EMES/
- Ort & Zeit:
  - Mo, 11-13, Raum RUD26, 0'313
  - Fr, 11-13, Raum RUD26, 1'303
- Begleitendes Projekt
  - Einführung am 25.10.2010
- Kontakt
  - salfner@informatik.hu-berlin.de
  - sommer@informatik.hu-berlin.de

Über EMES

- EMES bildet bildet zusammen mit „Zuverlässige Systeme“ einen Kurs
- Wertung als Halbkurs in der Technischen Informatik
- Mündliche Prüfung, Voraussetzung für die Prüfungszulassung ist eine erfolgreiche Projektverteidigung

Mobile und eingebettete Systeme

- Prozessoren, die Informationen einlesen, verarbeiten und ausgeben
- Definierte Verarbeitungsaufgabe innerhalb einer gegebenen Umgebung
- Begriff intuitiv klar, aber schwer abgrenzbar
  - Consumer-Bereich: Multimedia, Heimkino, Haussteuerung, Fahrstuhl, PDAs, Mobiltelefone, Kameras, Spielzeug, Küchengeräte, ...
  - Büroanwendungen: Kopierer, Fax, PC-Hardware, Telefonanlage, Feuermeldetechnik, Lichtkontrolle, Klimaanlage, ...
  - Industrielle Anwendungen: Verteilte Kontrollsysteme, Fahrstuhl, Industrieroboter, Medizintechnik, Atomkraftwerk, ..
  - Verkehrstechnik: Bremssystem, Motorsteuerung, ABS, Bord-Computer, Regensensor, ...

Mobile und eingebettete Systeme

- Reaktives Verhalten
  - Berechnungen als Reaktion auf ein externes Ereignis
  - Korrektheit als Funktion der Verarbeitungszeit $\rightarrow$ Echtzeit
- Geringe Größe, geringes Gewicht
- Minimaler Stromverbrauch
- Harte Umgebungsbedingungen - Hitze, Vibration, Stromschwankungen, Interferenzen, Korrosion
- Minimale Kosten

Mobile und eingebettete Systeme

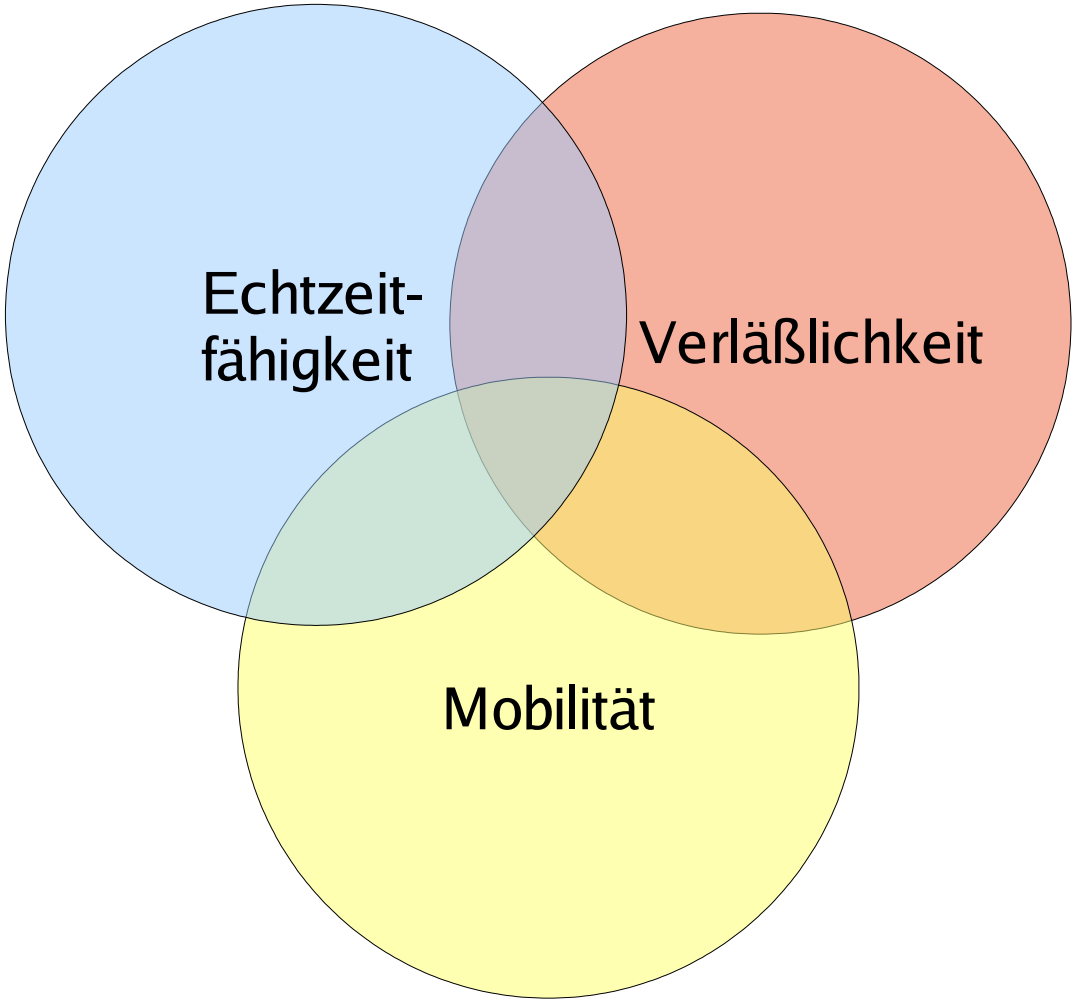
- Unterschied zu *commercial-of-the-shelf* (COTS) Systemen ?

	COTS	**MES**
Verläßlichkeit	Niedrig	Hoch
Zeitverhalten	Egal	Vorhersagbar
Umgebung	Standardisiert	Speziell
Nutzerkommunikation	Flexibel	Eingeschränkt
Ressourcen	Viel	Wenig
Gruppenbildung	Starr	Flexibel

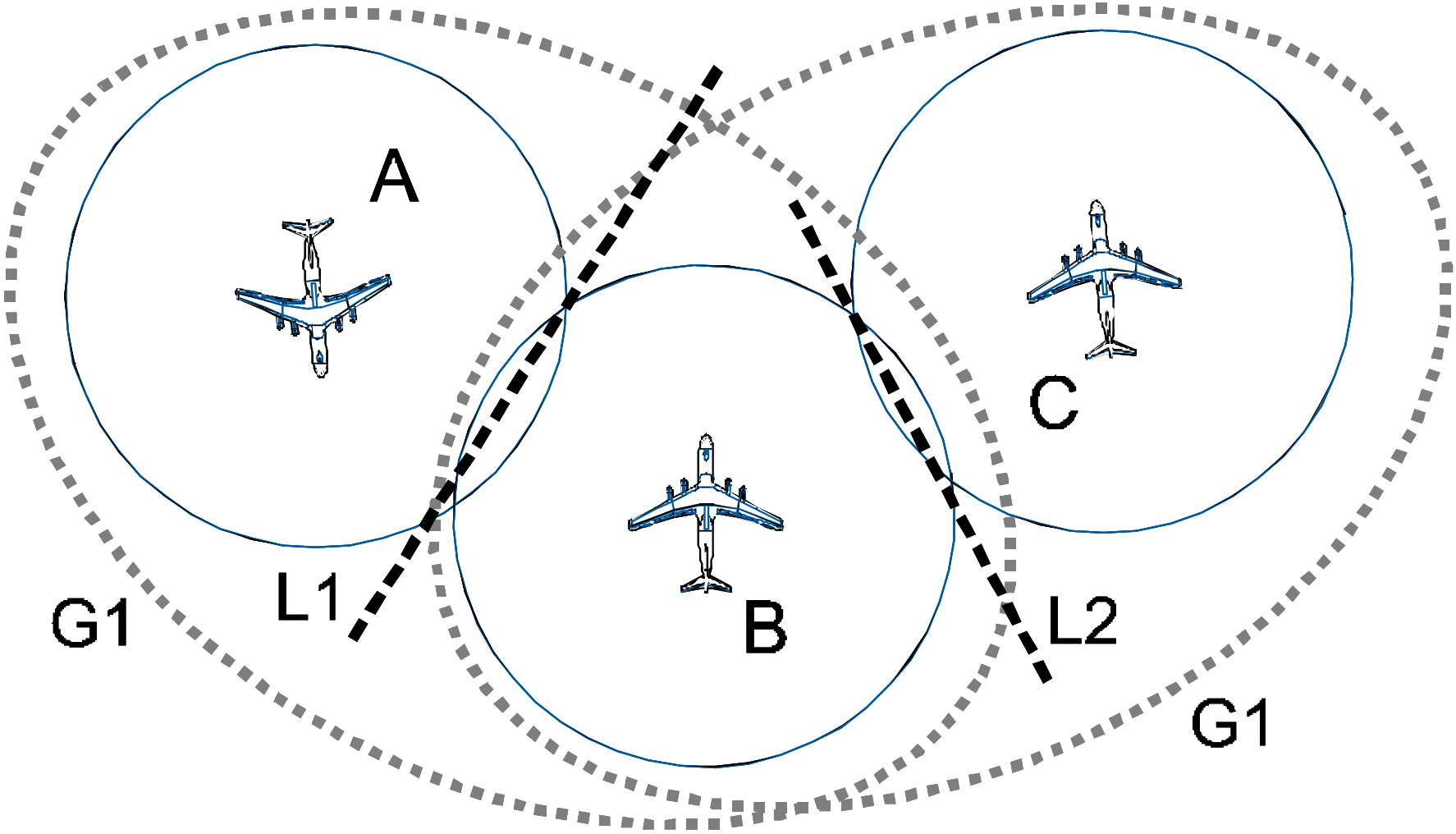
Mikroprozessor vs. Mikrocontroller

Mikroprozessor	**Mikrocontroller**
CPU separat von RAM, ROM, I/O	Ein Chip bzw. Modul
HW-Ressourcen erweiterbar	HW-Ressourcen fest
Teuer	Billig
GHz, GB	MHz, MB
Betriebssystem und Anwendungen auf der Festplatte	Anwendung im ROM / Flash-Speicher
C++, Java	Assembler, C

Drei Eigenschaften von MES



Beispiel



Funktionale vs. nichtfunktionale Eigenschaften

- Üblicherweise interessiert die eigentliche Funktion oder deren Ergebniswert ('Was')
  - Programmiersprachen
  - Interfacebeschreibungen
- Bei nichtfunktionalen Eigenschaften spielen „Randaspekte“ eine Rolle ('Wie')
  - Zeitverhalten !
  - Ressourcenverbrauch
  - Verlässlichkeitsparameter
  - Seiteneffekte
  - Sicherheit

Zeitverhalten

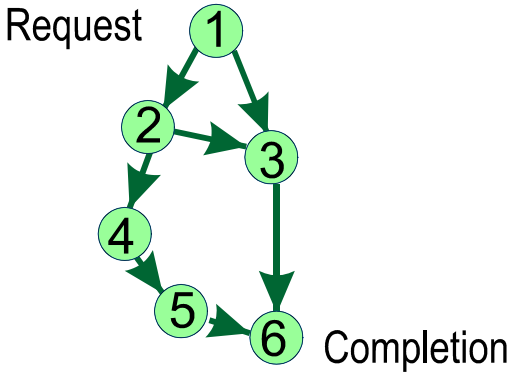
Sicherheit

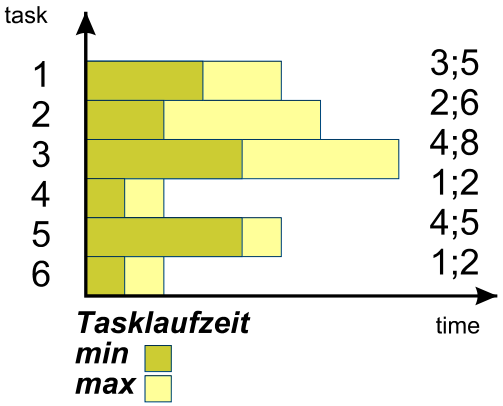
Nichtfunktionale Eigenschaften

- Definitionen und Maße sind schwierig
- Modularisierung funktioniert häufig nicht, weil verschiedene Eigenschaften Teilungen erfordern, die orthogonal zur funktionalen Teilung sind
- Abhängigkeit zwischen verschiedenen nichtfunktionalen Eigenschaften
- Vielfach probabilistische Abhängigkeiten
- Widersprüchliche Ziele
- Einordnung der Eigenschaften nicht immer klar und abhängig vom Kontext

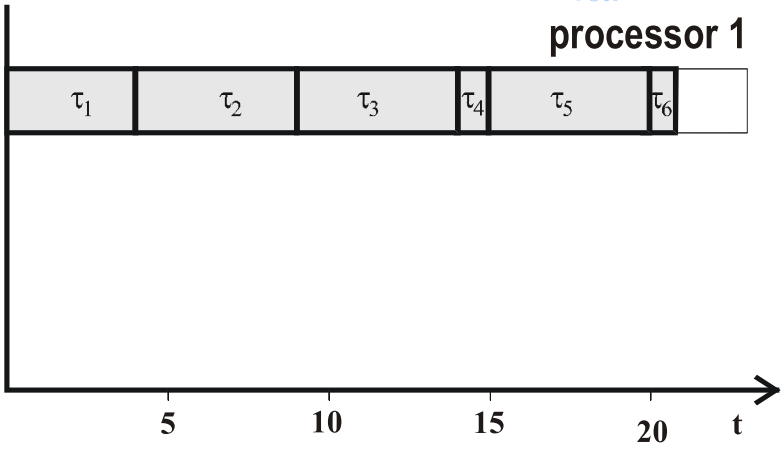
Sichtweisen

- Fallstudie
  - Dienst hängt von der Ausführung von 6 Tasks ab
  - Beendigungswahrscheinlichkeit jeder Task ist im Intervall gleichverteilt
  - ein oder zwei Prozessoren stehen zur Verfügung
  - Deadline 25 für Dienstfunktion
  - Prozessor-Fehlerrate λ=0,01; $R(t)=e^{-\lambda t}$
  - Sollen ein oder zwei Prozessoren eingesetzt werden?
  - Welche Schedulingstrategie soll genutzt werden?

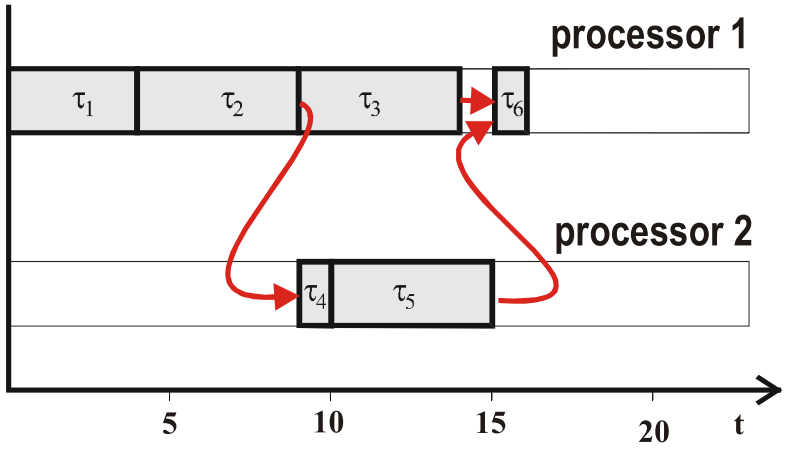




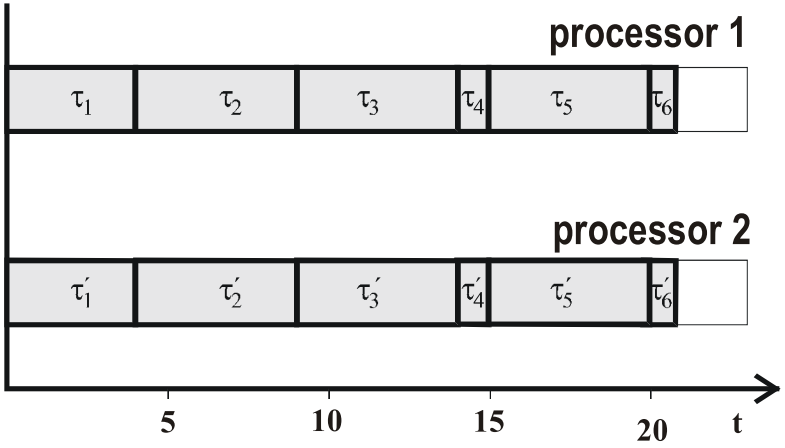
Alternative Designs



Einfacher Ansatz



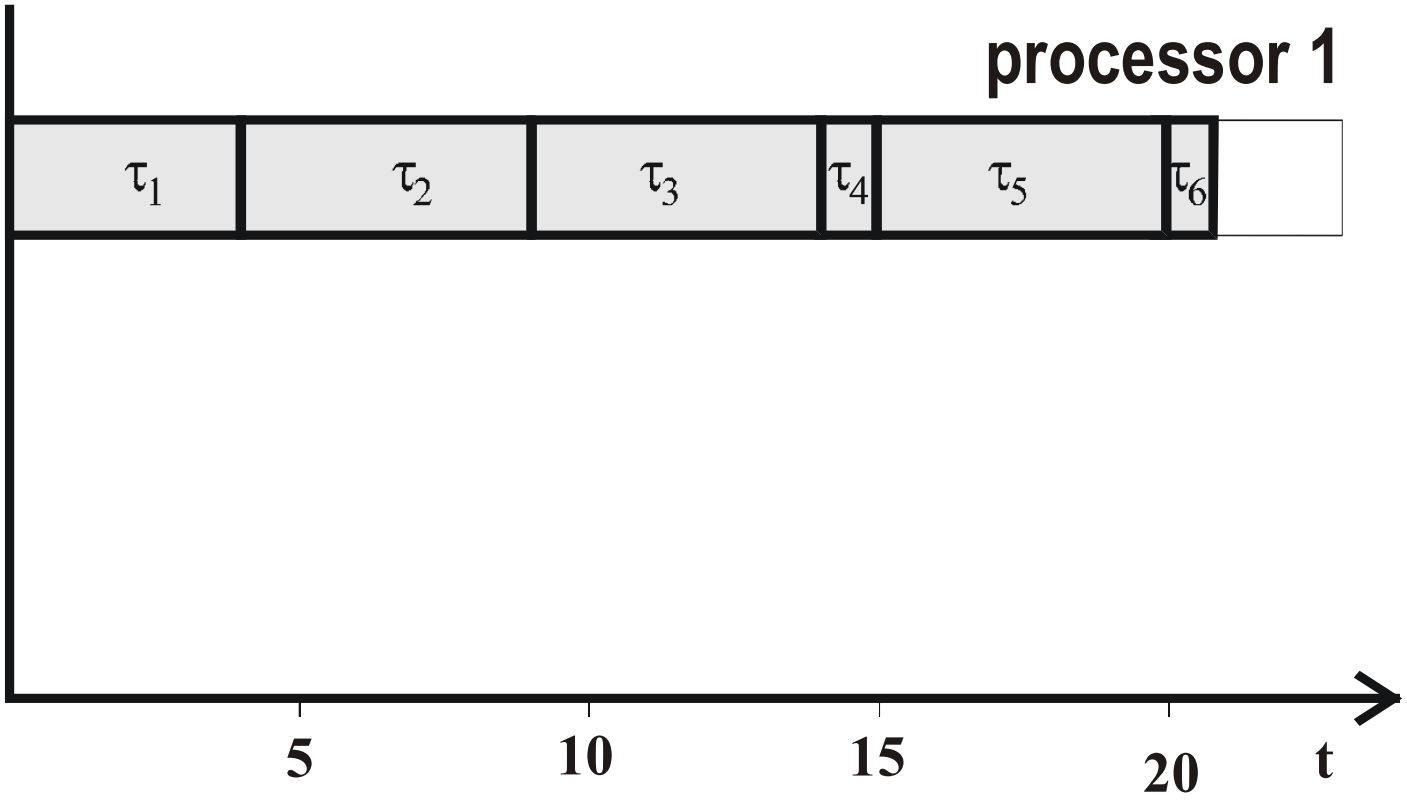
Echtzeit-Ansatz



Fehlertoleranter Ansatz

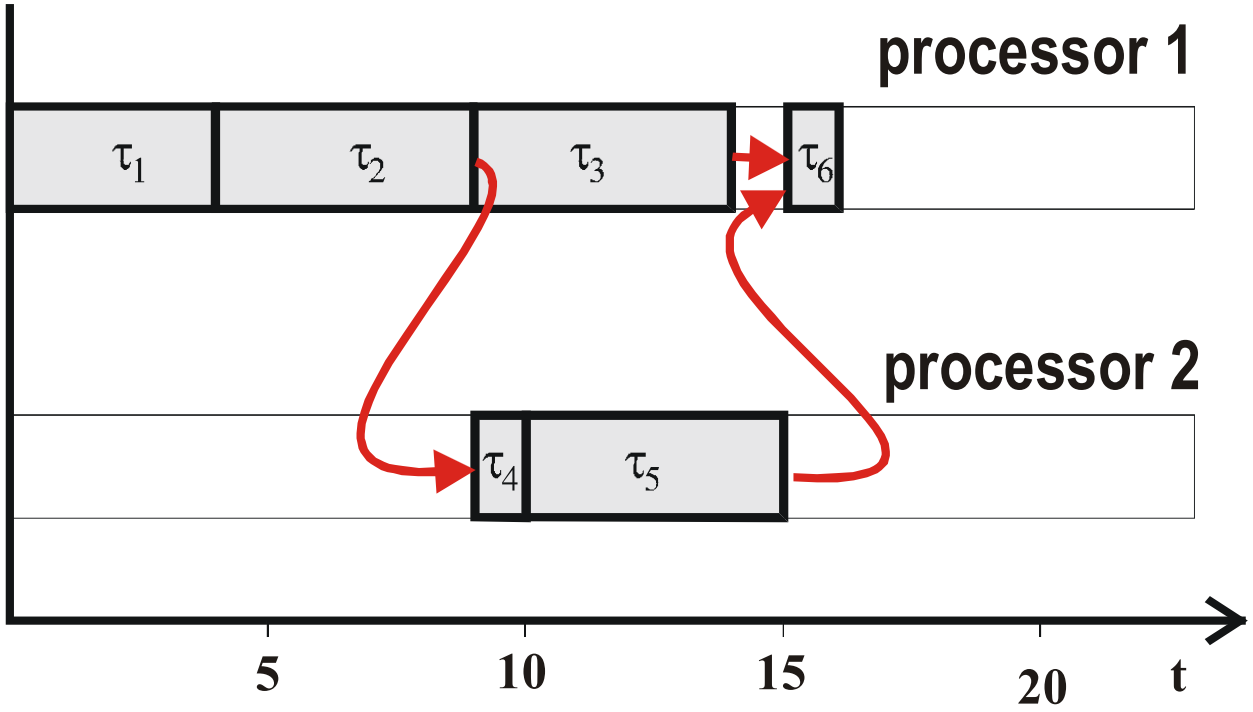
Adaptiver Ansatz

Design I: Einfacher Ansatz



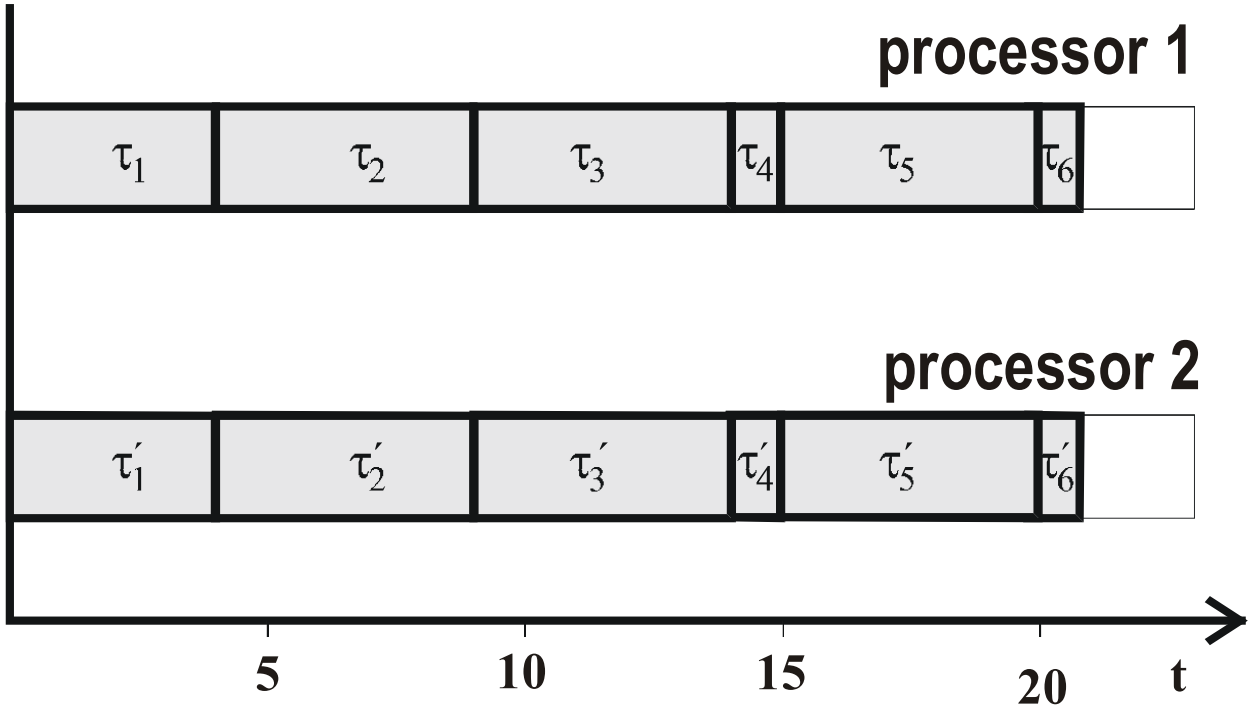
- Ein Prozessor, sequentielle Ausführung
- Keine Fehlertoleranz, keine Echtzeitgarantie

Design II: Echtzeit-Ansatz



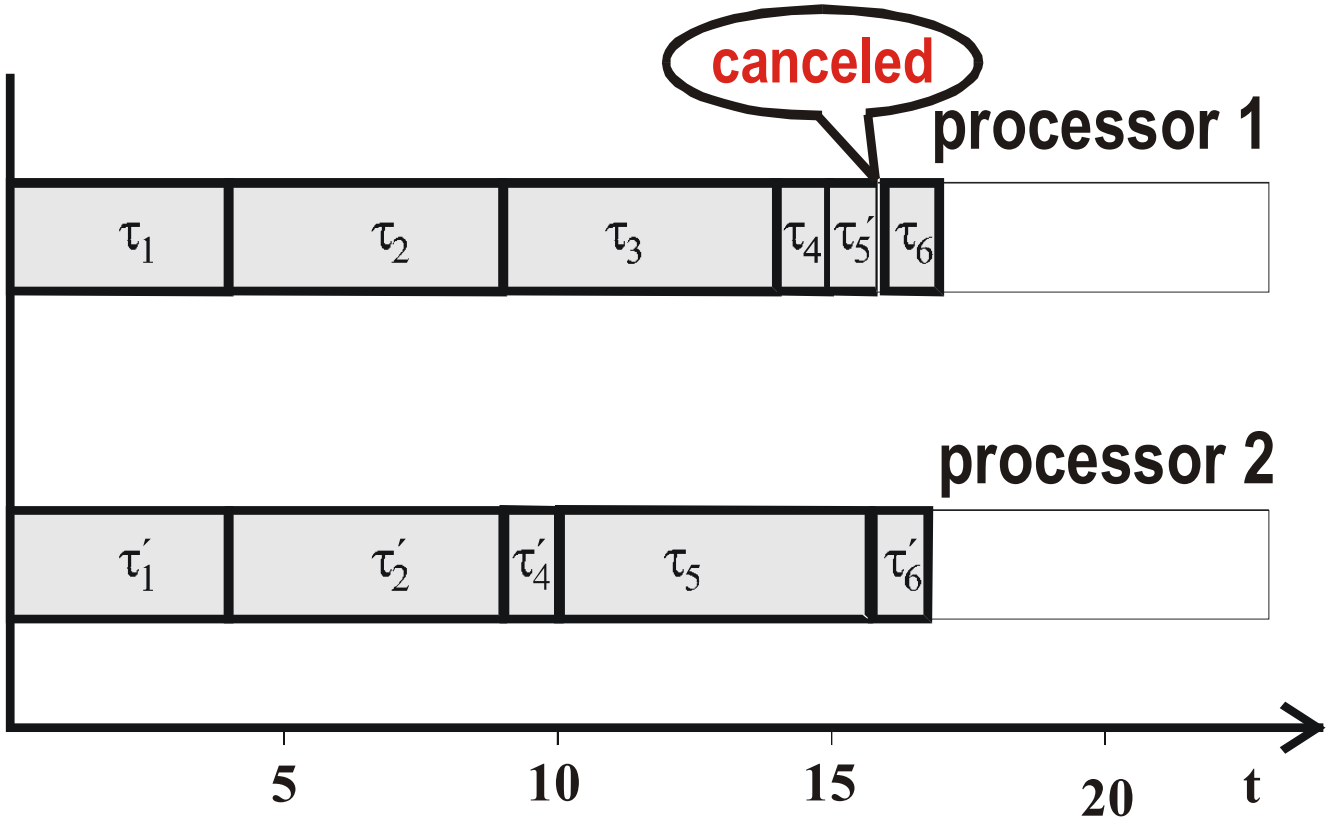
- Zwei Prozessoren, verteilte Ausführung
- Keine Fehlertoleranz, aber Echtzeit-Garantie (WCET=21)
- Vergrößertes Risiko eines Prozessorversagens

Design II: Fehlertoleranter Ansatz



- Zwei Prozessoren, redundante Ausführung
- Aspekte: Fehlerklasse, Unabhängigkeit
- Fehlertoleranz, aber keine Echtzeit-Garantie ($R_{ges} = 1 - (1 - R)^2$)

Design IV: Adaptiver Ansatz



- Zwei Prozessoren, dynamisches Eager-Scheduling
- Fehlertoleranz und Echtzeit je nach Situation

Eine mögliche Metrik: Responsivität

Erfolgswahrscheinlichkeit

$$\mathcal{R}(t) = P[t_{ready} \leq t] = \int_0^t (e^{-\lambda\tau}\epsilon_1(\tau)) * \cdots * (e^{-\lambda\tau}\epsilon_6(\tau))d\tau$$

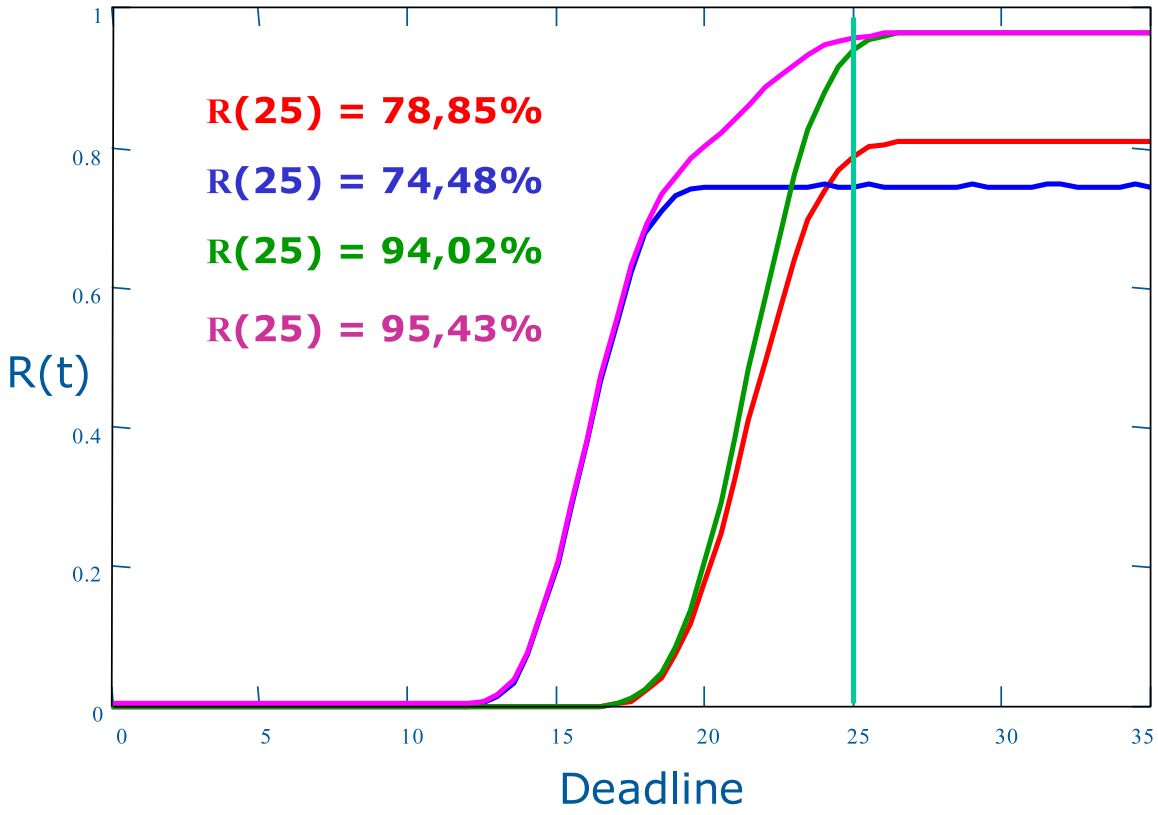
mit

$e^{-\lambda t}$: Zuverlässigkeit (Wahrscheinlichkeit für korrekte Funktion im Intervall $[0, t]$)

$\epsilon(t)$: Wahrscheinlichkeitsdichtfunktion für Beendigung zum Zeitpunkt t bei fehlerfreien Betrieb, Ermittlung durch Faltung der Einzelfunktionen:
$\epsilon_{gesamt} = \epsilon_1(t) * \epsilon_2(t) * ... * \epsilon_6(t)$

Vergleich

Wahrscheinlichkeit, dass der Dienst sein Ergebnis rechtzeitig liefert



- Teilweise Optimierung kann Gesamtergebnis verschlechtern

Sequentiell, ein Prozessor
Verteilt, zwei Prozessoren
Redundant, zwei Prozessoren
Eager Scheduling, zwei Prozessoren

Konzept von EMES

- JoJo-Ansatz:
  - Theoretische Probleme und technische Anwendungen im Wechsel
  - Praktische Erfahrungen durch Projektarbeit
- Drei Module (z.T. verschränkt):
  - Echtzeit
  - Kommunikation
  - Systeme & Fallstudien
- Aspekte der Verlässlichkeit werden im Halbkurs „Zuverlässige Systeme“ behandelt
- Aspekte der mobilen Kommunikation werden im Halbkurs „Rechnerkommunikation“ über die hier vorgenommene Einführung hinaus behandelt

(Vorläufige) Themenplanung

- Grundlagen Echtzeit: Harte/weiche, Scheduling, Abhängigkeiten, Taskmodell
- RMA/EDF
- RT-Anwendungen
- Drahtlose Kommunikation
- Standards der mobilen Kommunikation (GSM, UMTS, WLAN, Bluetooth, etc.)
- Feldbusse (CAN, TTP)
- Echtzeitkommunikation
- Gruppenkommunikation
- Uhrensynchronisation
- Embedded & RT-Betriebssysteme
- Architekturen (TTA)
- Umgang mit Energie
- Prozessoren für eingebettete Systeme

Literatur

- Jane W.S. Liu: *Real-Time Systems*, Prentice Hall, 2000
- C.M. Krishna, Kang G. Shin: *Real-Time Systems*, McGraw-Hill, 1997
- H. Kopetz: *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Kluwer Academic, 1997
- G.T. Coulouris, J. Dollimore, T. Kindberg: *Distributed Systems*, Addison-Wesley, 1994
- Foliensatz
- Veröffentlichungen
  - auf Konferenzen
  - in wissenschaftlichen Zeitschriften