Echtzeitsysteme

Grundlegende Abfertigung nicht-periodischer Echtzeitsysteme

Peter Wägemann

Lehrstuhl für Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität Erlangen-Nürnberg

`https://sys.cs.fau.de/lehre/ss22/ezs/`

14. Juni 2022

Fragestellungen

- Unterschied periodische und nicht-periodische Aufgaben?
  - Wo spielen nicht-periodische Aufgaben eine Rolle?
  - Welche Herausforderungen ergeben sich für ihre Abfertigung?
- Basistechniken für ihre Abarbeitung?
  - Sind diese Techniken auf Anwendungsebene umsetzbar?
  - Benötigt man spezielle Unterstützung des Betriebssystems?
  - Welche Risiken, Vorteile und Nachteile beinhalten diese Techniken?
- Schlupfzeit – Verwendung für nicht-periodische Aufgaben?
  - Wie bestimmt man die Schlupfzeit?

Gliederung

1 Nicht-periodische Aufgaben
- Nicht-periodische Echtzeitanwendungen
- Zeitliche Parameter
- Herausforderung: Mischbetrieb

2 Erweiterte Behandlung nicht-periodischer Aufgaben
- Unterbrecherbetrieb
- Hintergrundbetrieb
- Periodischer Zusteller

3 Slack-Stealing
- Taktgesteuerte Systeme
- Ereignisgesteuerte Systeme

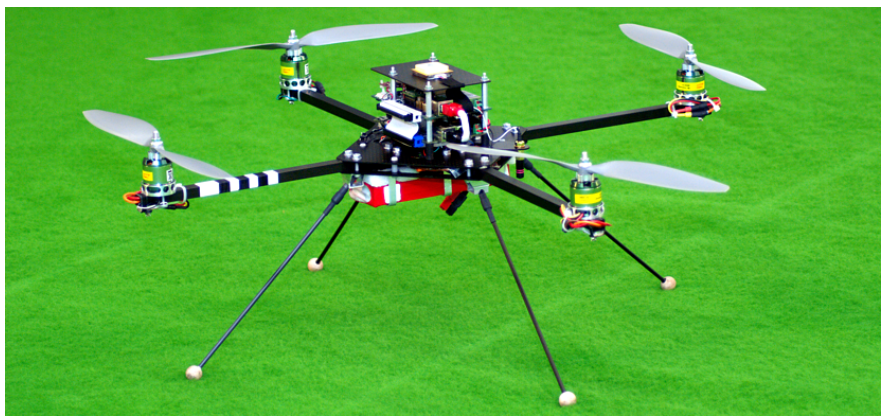
4 Zusammenfassung

Nicht-periodische Aufgaben in der Praxis

Quelle nicht-periodischer Aufgaben

Nicht-periodische Aufgaben behandeln Ereignisse, die sich aus Zustandsänderungen des zu kontrollierenden Systems ableiten (vgl. *event trigger*, Folie III-2/5).

- Beispiele für Zustandsänderungen:
  - Mensch-Maschine-Interaktion
    - ⚠ Menschliches Verhalten ist kaum quantifizierbar
  - Kommunikation
  - Fehlerbehandlung



Beispiel I4Copter:

- Steuerkommandos
  - Empfang über die Fernbedienung
  - Schlimmster Fall: Alle 100 ms
- Telemetriedaten-Übertragung
  - Füllen eines internen Puffers
  - Schlimmster Fall: Alle 9 ms

Restriktionen des periodischen Modells

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[1]

A6 Die Kosten durch Unterbrechungen, Ablaufplanung und Verdrängung sind vernachlässigbar

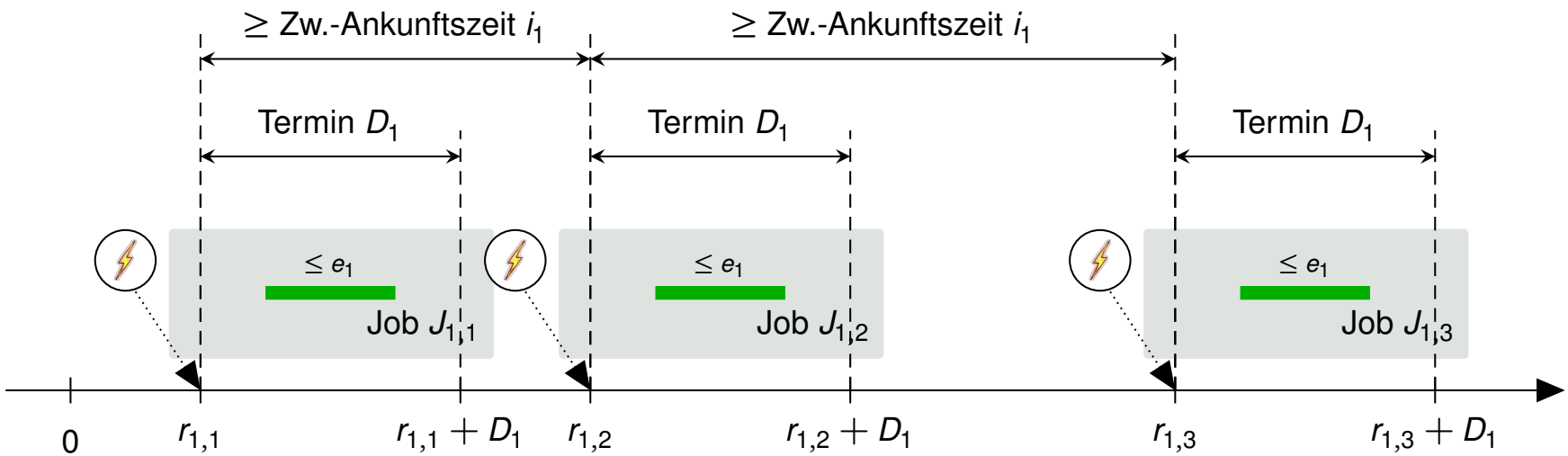
A7 Alle Aufgaben verhalten sich voll-präemptiv

[1] D.h. die einzige gemeinsame Ressource ist die CPU und es existieren keine Einschränkungen hinsichtlich der Auslösezeiten der Arbeitsaufträge voneinander.

Nicht-periodische Aufgaben

Erbringen in unregelmäßigen Zeitintervallen eine vorgegebene Systemfunktion. Jede nicht-periodische Aufgabe (T_i^S) ist eine Abfolge von Arbeitsaufträgen ($J_{i,j}$) mit vorgegebenen zeitlichen Eigenschaften.

- Weiche/feste Termine $\mapsto$ aperiodische Aufgabe (engl. *aperiodic task*)
- Harte Termine $\mapsto$ sporadische Aufgaben (engl. *sporadic tasks*)
- Besitzen minimale Zwischenankunftszeit (engl. *minimum interarrival time*) i_i mit $[r_{i,j}; r_{i,j+1}]$ zwischen den Auslösezeiten von T_i^S



Nicht-periodische Aufgaben (Forts.)

Tupeldefinition einer nicht-periodischen Aufgabe $T_i^S = (i_i, e_i, D_i)$

- i_i Minimale Zwischenankunftszeit (engl. *minimum interarrival time*)
- e_i Maximale Ausführungszeit (WCET)
- D_i Relativer Termin (engl. *deadline*), optional

- Alternative Definition über mögliche Auslösezeitpunkte
  - Intervall $[r_i^{nach}; r_i^{vor}]$ (frühester/spätester Zeitpunkt)

→ $T_i^S = ([r_i^{nach}; r_i^{vor}], e_i, D_i)$

  - Beachte: $i_{i+1} = [r_{nach}; \infty[$

→ Bei unbekanntem spätesten Auslösezeitpunkt sind Absolutwert und Intervallschreibweise äquivalent

- Arbeitsaufträge der Aufgabe: $J_{i,j} = (r_{i,j}, e_{i,j}, d_{i,j})$

Aussagen zum Zeitpunkt ihrer Auslösung sind schwierig

→ A-priori deutlich weniger Wissen als für periodische Aufgaben verfügbar

Mischbetrieb: periodisch ⟷ nicht-periodische

Erhaltung statischer Garantien für periodische Arbeitsaufträge

- Einfluss nicht-periodischer Arbeitsaufträge begrenzen

Koexistenz periodischer und nicht-periodischer Arbeitsaufträge

- Einordnung in das (periodische) Prioritätsgefüge

→ Gekoppelte Einplanung nicht-periodischer Arbeitsaufträge

- Aperiodische Arbeitsaufträge ↦ Antwortzeitminimierung
  - Zusicherungen an periodische und bereits zugelassene sporadische Aufträge bleiben erhalten
- Sporadische Arbeitsaufträge ↦ Termingarantie
  - Einplanung unter Sicherstellung des vorgegebenen Termins
  - Zusicherungen an periodische und bereits zugelassene sporadische Aufträge bleiben erhalten

→ Zulassung oder Abweisung des sporadischen Arbeitsauftrags

Zulassung sporadischer Arbeitsaufträge

☞ Übernahmeprüfung (engl. *acceptance test*) für sporadische Aufträge bei ereignisbedingter Auslösung

- Arbeitsauftrag wird angenommen falls gemeinsame Ausführung mit allen anderen Aufträgen machbar ist:
  - Test ist gekoppelt mit der Einplanung $\mapsto$ *online*
  - Gegenwärtiger Ablaufplan muss genügend Schlupf aufweisen
    - Schlupf $\sigma \geq$ WCET e des sporadischen Auftrags
    - Ausführungszeit ggf. erst zum Auslösezeitpunkt bekannt

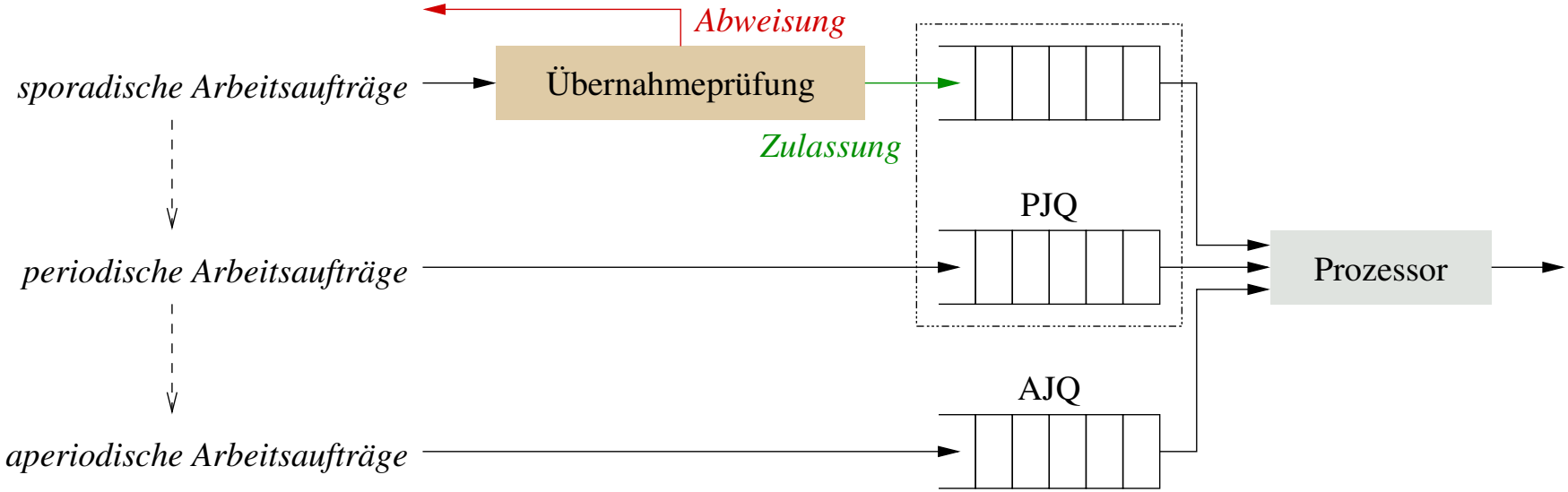
⚠ Nur Schlupf vor dem Termin ist von Relevanz

- Der sporadische Auftrag wird bei negativem Test abgewiesen
  - Anwendung wird eine schwerwiegende Ausnahmesituation angezeigt

→ Fehlerfall und Ausnahmebehandlung

⚠ Gleichzeitige sporadische Aufträge werden oft nach EDF getestet

Prioritätswarteschlangen im Betriebssystem



☞ Erweiterung des MLQ-Schedulers (vgl. IV-1/33)

- Periodische Arbeitsaufträge ↦ Periodic Job Queue
  - Implementierung durch Ablauftabelle oder -liste
- Sporadische Arbeitsaufträge ↦ Zweistufiger Prozess:
  1. Ausgelöste Arbeitsaufträge warten auf Übernahmeprüfung
  2. Zugelassene Arbeitsaufträge ↦ Eigene Queue oder PJQ
- Aperiodische Arbeitsaufträge ↦ Aperiodic Job Queue

Gliederung

1 Nicht-periodische Aufgaben
- Nicht-periodische Echtzeitanwendungen
- Zeitliche Parameter
- Herausforderung: Mischbetrieb

2 Erweiterte Behandlung nicht-periodischer Aufgaben
- Unterbrecherbetrieb
- Hintergrundbetrieb
- Periodischer Zusteller

3 Slack-Stealing
- Taktgesteuerte Systeme
- Ereignisgesteuerte Systeme

4 Zusammenfassung

Überblick

Behandlung nicht-periodischer Aufgaben

Grundlegende Behandlungsmethoden für nicht-periodische Ereignisse lassen sich mit minimaler Unterstützung des Laufzeitsystems umsetzen. Sie sind sowohl für zeit- als auch für ereignisgesteuerte Systeme geeignet und teilweise vollständig auf Anwendungsebene umsetzbar.

- Unterbrecherbetrieb $\leadsto$ Nicht-periodische Aufgaben haben Vorfahrt
  - Ereignisbehandlung direkt in der Unterbrechungsbehandlung

  → Mittels Ausnahmebehandlungen (vgl. III-1/17 ff)
- Hintergrundbetrieb $\leadsto$ Periodische Aufgaben haben Vorfahrt
  - Phasen der Untätigkeit für nicht-periodische Aufgaben nutzen

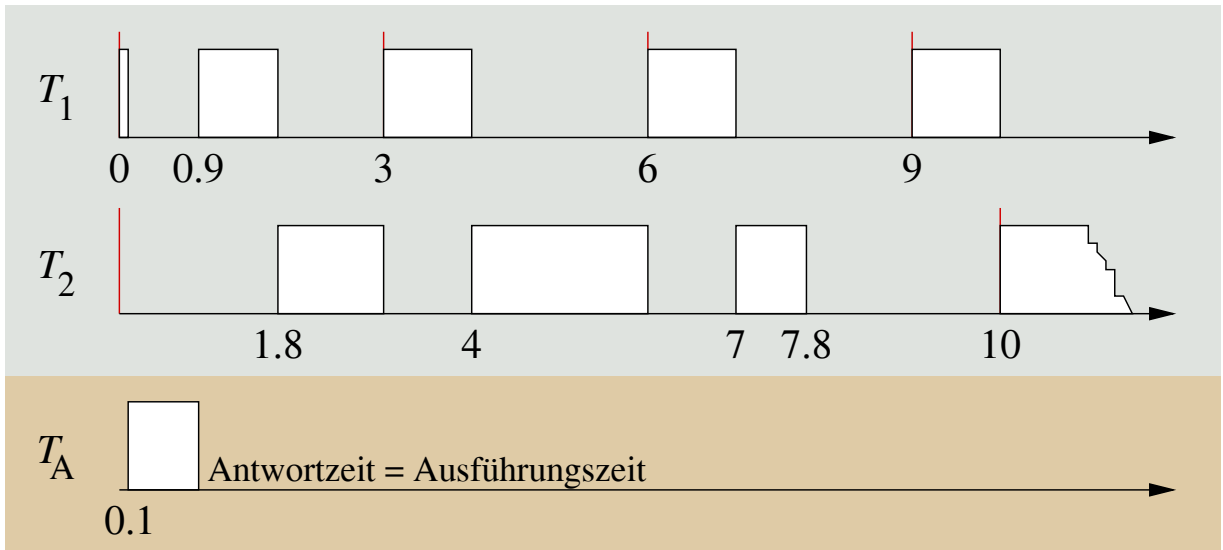
  → Mittels Verdrängung (vgl. III-2/12 ff)
- Periodischer Zusteller $\leadsto$ Alles ist eine periodische Aufgabe
  - Abfragen nicht-periodischer Ereignisse durch periodische Aufgaben

  → Einphasen nicht-periodischer Aufträge mit bekannten Mitteln

Unterbrecherbetrieb

Antwortzeitminimierung – auf Kosten eines gut geordneten Ablaufplans

Nicht-periodische Arbeitsaufträge werden sofort ausgeführt



- Beispiel:
  - Periodische Aufgaben $T_1 = (3, 1)$ und $T_2 = (10, 4)$ (nach RMA)
  - Aperiodische Aufgabe T_A^S mit $r_A = [0.1, \infty[$, $e_A = 0.8$

Verdrängung in Ausführung befindlicher periodischer Aufträge

- Erhöht das Risiko von Schwankungen (engl. *jitter*) im Ablauf periodischer Aufgaben

Unterbrecherbetrieb (Forts.)

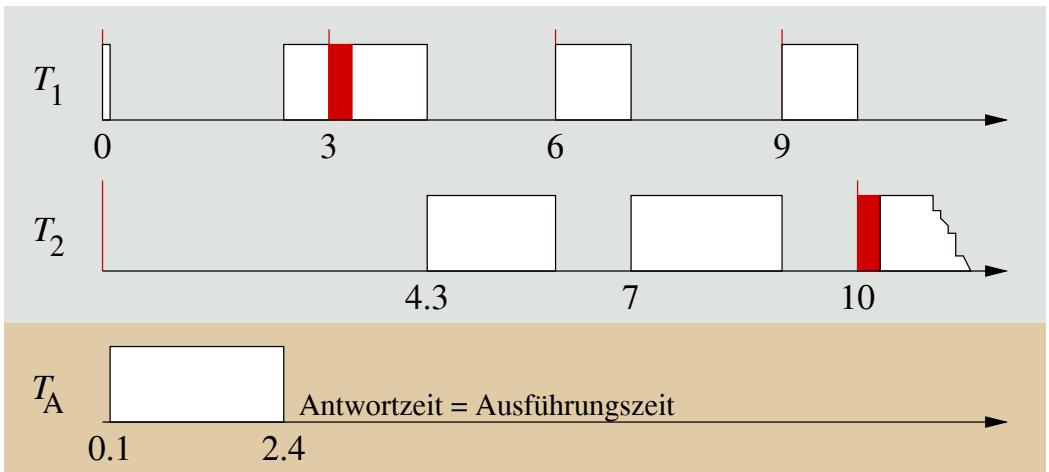
- Minimale Anforderungen an Laufzeitsystem: kooperative Ausführung + Interrupts möglich

⚠ Bevorzugung nicht-periodischer Arbeitsaufträge

⇝ Termineinhaltung periodischer Aufgaben nicht gesichert

- Beispiel (vgl. Folie 13):
  - WCET $e_A = 2.3$
  - *run to completion*

☞ periodische Tasks
  - Terminverletzung T_1
  - Terminverletzung T_2



⚠ Fortsetzung der unterbrochenen periodischen Aufgabe nach Ablauf der Schlupfzeit (vgl. III-2/27)

Apollo 11, Mondlandung

Folklore zum Bordcomputer der Landefähre [3, 7]

1. Rendezvousradar[2] wurde vor Beginn der Landung eingeschaltet
2. Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
   - Rendezvousradar erzeugte eine Flut von Scheinunterbrechungen
   - Dadurch wurden etwa 15 % an unerwarteter Rechenlast erzeugt

   → Verzögerung/Ausfall von Berechnungen zur Landungskontrolle
3. Landungskontrolle hatte minimalen Treibstoffverbrauch als Ziel
   - Kontrollprogramm erzeugt alle zwei Sekunden ein Stellwert
   - Stabilisierung durch Autopilot alle 100 ms
4. Landephase war mit einer Dauer von 11 Minuten geplant
   - Fehlerbedingt fielen gut eine Minute lang alle Kontrollkommandos aus
   - Erfolgreiche Landung durch Umschaltung auf manuelle Kontrolle

[2] Messung von Zeitintervallen zwischen bekannten Landmarken und Überprüfung von Position und Geschwindigkeit des Landemoduls relativ zum Kommandomodul.

Maximale Frequenz ↦ Minimale Zwischenankunftszeit

Unterbrechungen erschweren eine deterministische Ausführung periodischer Aufgaben oder machen dies gar unmöglich:

- Zeitpunkt ihres Auftretens ist a-priori nicht bekannt
- Sie werden gegenüber aktuell ausgeführten Jobs bevorzugt

Quelle	**max. Frequenz** [in Hz]
loser Draht	500
Kippschalter	1000
Wippschalter	1300
serielle Schnittstelle (115 kpbs)	11500
Ethernet (10 Mbps)	14880
CAN-Bus	15000
I2C-Bus	50000
USB	90000
Ethernet (100 Mbps)	148800
Ethernet (1 Gbps)	1488000

max. Raten verschiedener Unterbrechungsquellen [6]

- Selbst unscheinbare Komponenten können signifikante Last durch Unterbrechungen erzeugen
- Periodische Aufgaben stehen Unterbrechungen wehrlos gegenüber

Gefahr der Überlast

Randnotiz: Unterlast

Maxime in Echtzeitsystemen: Kapazitäten gezielt frei lassen

Echtzeitrechensysteme sollten in kritischen Situation nur bis zu einem vorgegebenen Maximum belastet werden

- Deutlich unter 100 % CPU-Auslastung (vgl. IV-2/28)

Bestimmung der kritischen Situationen und freizuhaltende Kapazitäten ist eine große Herausforderung

- Unterbrechungsbedingte Verzögerungen und Last im Voraus einzuplanen benötigt fundiertes Expertenwissen
- Erfordert durchgehende Anforderungsanalyse[3]

Der Fehlerfall ist dennoch nie auszuschließen

(Scheinunterbrechungen sind kaum vorherzusagen)

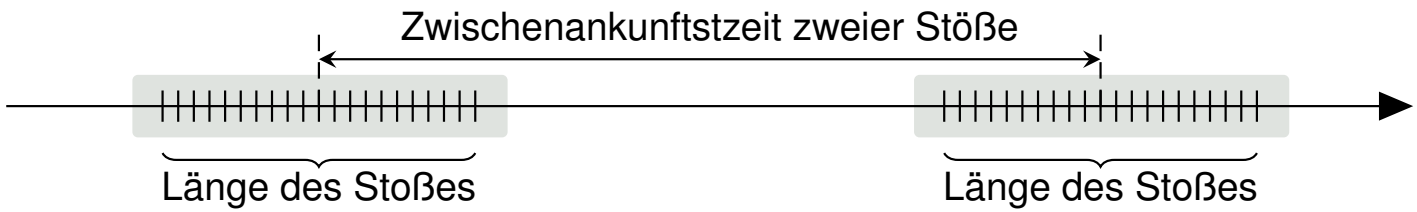
[3] (engl. *requirements engineering*), Fundament und Teilaktivität systematischer Softwareentwicklung – hier aber nicht nur Software.

Kontrollierter Unterbrecherbetrieb

Eine Überlastsicherung ist unumgänglich! [6]

☞ Ansatzpunkt: Beschränkung des Auftretens von Unterbrechungen

1. Überwachung der minimalen Zwischenankunftszeit
   - Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen
2. Überwachung von Unterbrechungsstößen (engl. *bursts*)
   - Nach Unterbrechungsstoß werden Unterbrechungen eine Zeit lang abgeblockt



- OSEKtime [5] und AUTOSAR OS [1] schränken die minimale Zwischenankunftszeit ein oder überwachen sie
- Hardware-Unterstützung hierfür notwendig (wie bei Infineon TriCore [2])

Den Schlupf periodischer Arbeitsaufträge nutzen

Nicht-periodische Arbeitsaufträge in deren Hintergrund ausführen

☞ Ausführung aperiodischer Aufträge im Hintergrund

→ Wenn keine periodischen Aufträge zur Ausführung anstehen

- Untätigkeit des Prozessors ausnutzen

- Schlupf auf der gesamten Echtzeitachse kann genutzt werden
  - Ruhephasen werden mit nicht-periodischen Aufträgen aufgefüllt
  - Verdrängung nicht-periodischer Aufträge
  - Einplanungsentscheidung erfolgt zur Laufzeit

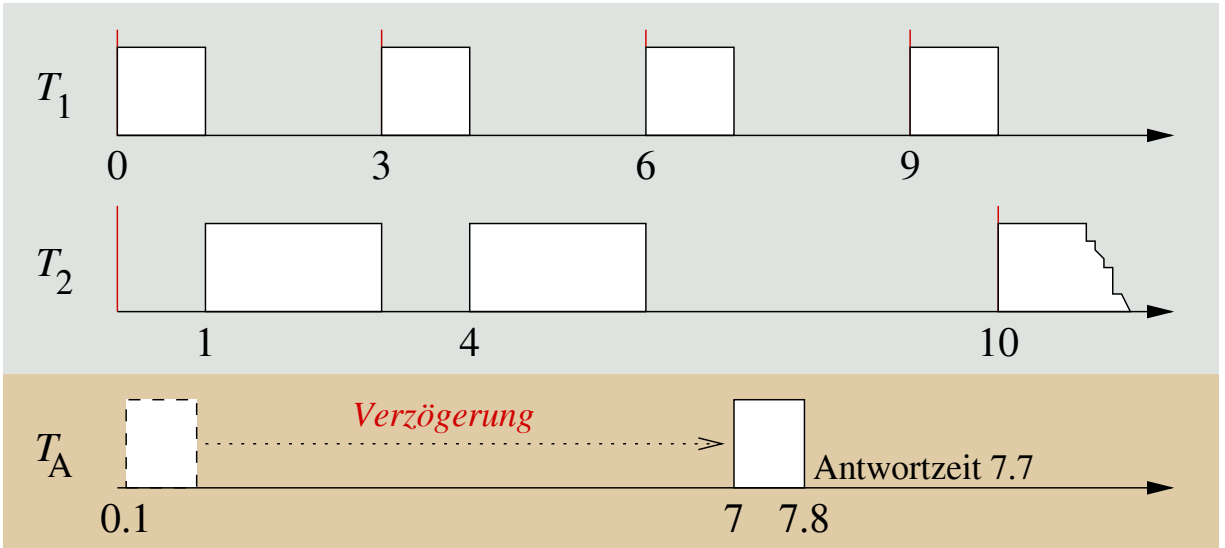
⚠ Nicht-periodische Aufträge zugunsten periodischer Aufträge verzögern

- Ihre Antwortzeit verschlechtert sich
- Termineinhaltung bei sporadischen Aufgaben wird schwieriger
- Ansprechempfindlichkeit des Systems lässt nach

Hintergrundbetrieb

Korrekter Ablaufplan – auf Kosten des Antwortverhaltens

☞ Aperiodische Arbeitsaufträge werden nur ausgeführt falls keine periodischen/sporadischen Aufträge zur Ausführung anstehen



- Beispiel:
  - Periodische Aufgaben $T_1 = (3, 1)$ und $T_2 = (10, 4)$ (nach RM)
  - Aperiodische Aufgabe T_A^S mit $r_A = [0.1, \infty[$, $e_A = 0.8$

⚠ Antwortzeiten nicht-periodischer Arbeitsaufträge bleibt unberücksichtigt

$\rightsquigarrow$ schlechtes Antwortverhalten

Periodischer Zusteller (engl. periodic server)

Zusteller: Spezialisierung einer periodischen Aufgabe

- Definiert durch Periode p_{srv} und Ausführungszeit e_{srv}
  - Verhältnis $u_{srv} = e_{srv}/p_{srv} \mapsto$ Größe des Zustellers
- Ausführungsbudget (engl. *execution budget*) in Form der e_{srv}
  - Budget wird um bis zu e_{srv} Einheiten aufgefüllt (engl. *replenished*)
- Auffüllperiode p_{srv} (engl. *replenishment period*)
  - Das Budget des Zustellers wird regelmäßig erneuert

  → Auffüllzeit (engl. *replenishment time*)

- Begrenzung der Ausführungszeit nicht-periodischer Arbeitsaufträge im Zeitinterval p_{srv} auf maximal e_{srv} Zeiteinheiten

Verschiedene Varianten: abfragend, aufschiebbar, sporadisch

Ein periodischer Zusteller ist i.d.R. für die Ausführung der Aufträge mehrerer sporadischer/aperiodischer Aufgaben zuständig

Periodischer Zusteller – Phasen

☞ Phasen der Auslösung, Bereitstellung und Ausführung eines Zustellers:

1. Untätig (engl. *idle*):
   - Warteschlange des Zusteller ist leer
2. Zurückgestellt (engl. *backlogged*):
   - Mit Auslösung eines nicht-periodischen Arbeitsauftrags

   → Mindestens ein nicht-periodischer Auftrag ist ausführungsbereit
3. Kommt in Frage (engl. *is eligible*):
   - Ist zurückgestellt und weist Auftragsüberhang (engl. *backlog*) auf
   - Ausführungsbudget ist vorhanden
4. In Einplanung (engl. *scheduling*):
   - Teilnahme am Einplanungsverfahren periodischer Aufgaben

   → Reguläre periodische Aufgabe mit $T_s = (p_s, e_s)$
5. Verbraucht (engl. *consumes*):
   - Budget sinkt mit der Ausführung
6. Erschöpft (engl. *exhausted*):
   - Ausführungsbudget ist auf Null abgesunken

   → Wechsel zu zurückgestellt, warten auf Wiederauffüllung

Abfragender Zusteller (engl. *polling server*)

Einfachste Form eines periodischen Zustellers

Abfrager (engl. *poller*) $\mapsto T_{PS} = (p_{PS}, e_{PS})$

- Abfrageperiode (engl. *polling period*) p_{PS}
- Zyklisch bereitgestelltes Budget von e_{PS} Zeiteinheiten
- Schrittweise Abfertigung von nicht-periodischen Aufträgen

Ohne Auftragsüberhang verfällt das Budget unverzüglich

→ Sobald der Abfrager Untätigkeit feststellt

- Prüfung erfolgt nur am Anfang der Abfrageperiode[4]

Antwortzeiten nicht-periodischer Arbeitsaufträge unterliegen mitunter starken Schwankungen

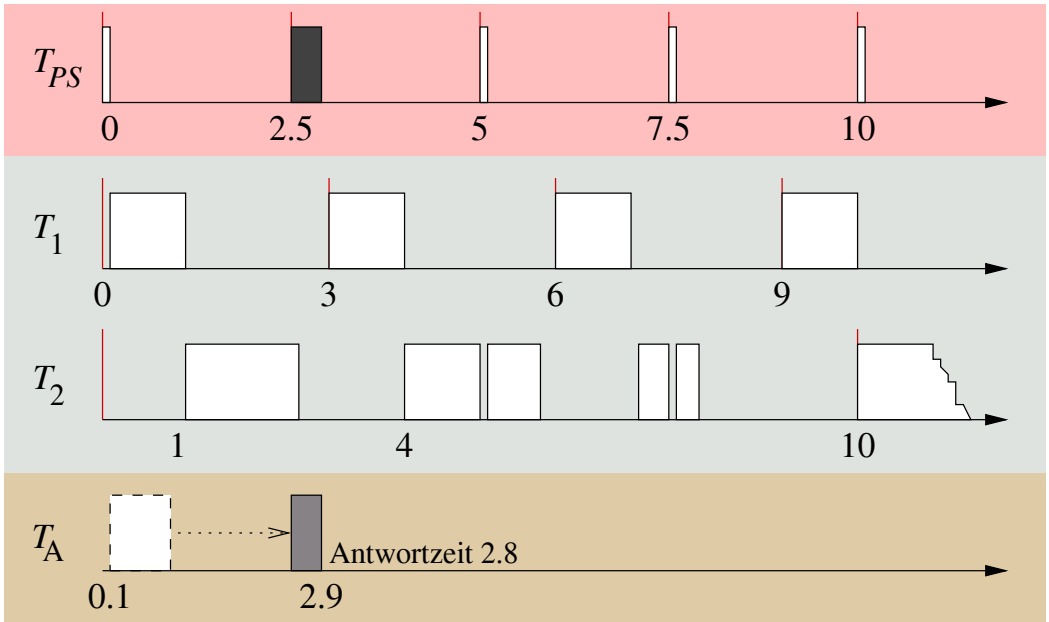
- Abhängig vom Auslösezeitpunkt des Auftrag/Zustand des Abfragers

[4] Eintreffende aperiodische Arbeitsaufträge nachdem der Abfrager seine Untätigkeit festgestellt hat, kommen frühestens in der nächsten Abfrageperiode zum Zuge.

Abfragender Zusteller – Beispiel (1)

Grundlegende Funktionsweise abfragender Zusteller

- Aufgabensystem:
  - periodische Aufgabe
    - $T_{PS} = (2.5, 0.5)$
    - $T_1 = (3, 1)$
    - $T_2 = (10, 4)$
  - aperiodischer Aufgabe
    - $T_A^S \mapsto ([0.1, \infty[, 0.4)$



1. Zusteller T_{PS} hat kürzeste Periode $\rightsquigarrow$ höchste Priorität (RMA)
2. Zu Beginn der Abfrageperioden t_0 ist die AJQ leer
   → Das Budget von T_{PS} verfällt
3. Auslösezeitpunkt r_A ist $t_{0.1}$
   - Kurz nach dem Abfragezeitpunkt t_0

   → Ausführung von T_A^S erfolgt in Abfrageperiode $t_{2.5}$

- Hier kommt das Laufzeitsystem ohne Verdrängung aus
  - Budget erlaubt dem Zusteller vollständige Abarbeitung von T_A^S

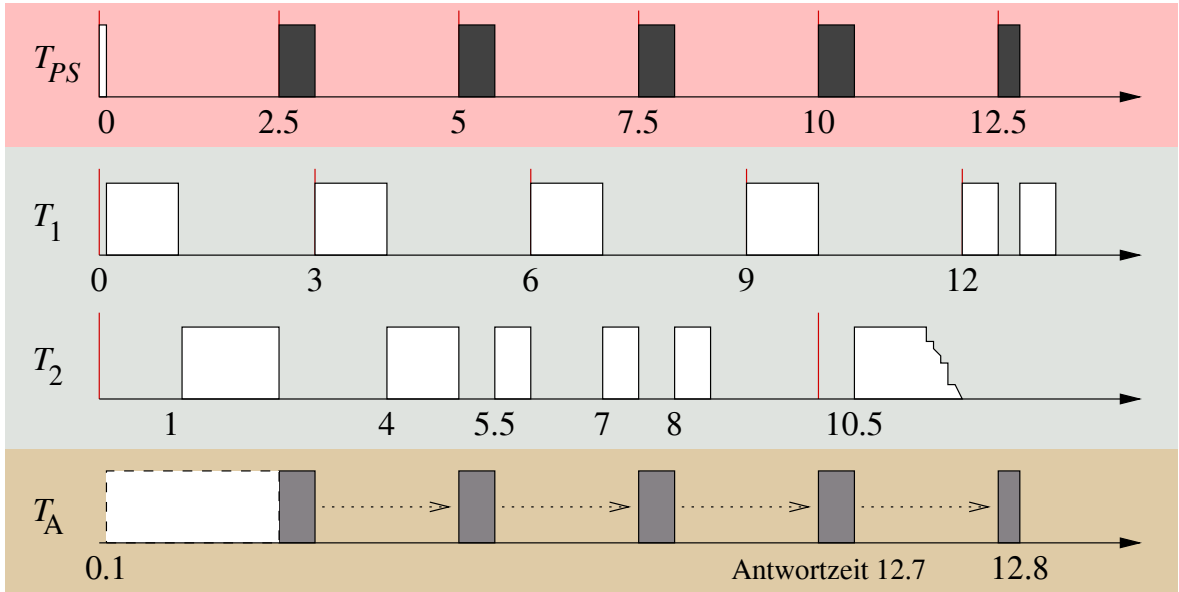
Abfragender Zusteller – Beispiel (2)

Aufteilung eines aperiodischen Arbeitsauftrags auf mehrere Auffüllperioden

- Aufgabensystem:

  T_{PS}, T_1, T_2 } wie gehabt

  $T_A^S \mapsto ([0.1, \infty[, 2.3)$



☞ Ausführung von A benötigt (mindestens) fünf Abfrageperioden:

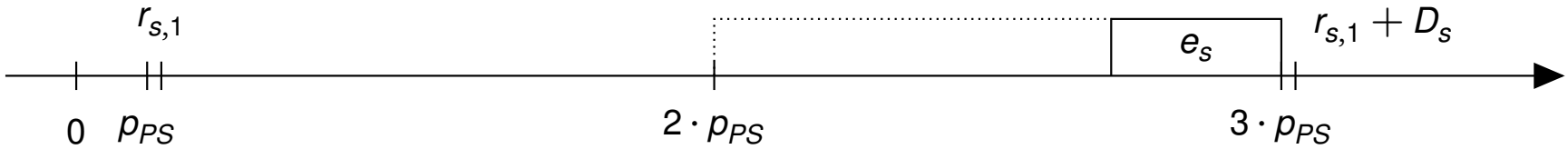
- 4×0.5 Zeiteinheiten (dem Budget von T_{PS})
  - Abfrageperioden $t_{2.5}$, t_5, $t_{7.5}$ und t_{10}
- 1×0.3 Zeiteinheiten (bis A beendet ist)
  - Abfrageperiode $t_{12.5} \rightsquigarrow$ AJQ ist leer, T_{PS} wird untätig

⚠ Das Laufzeitsystem muss Verdrängung unterstützen

Abfragende Zusteller und sporadische Aufgaben

Wenn der Abfragebetrieb in der Ereignissteuerung an seine Grenzen stößt...

- Periodische Zusteller $T_{PS} = (p_{PS}, e_{PS})$ behandelt sporadische Aufgabe $T_s^S = (i_s, e_s, D_s)$ mit Auslösezeitpunkt $r_{s,1}$ von $J_{s,1}$



- Schlimmstenfalls wird der erste Abtastzeitpunkt verpasst: $r_{s,1} > p_{PS}$

→ Abarbeitung von $J_{s,1}$ beginnt mit der zweiten Abfrageperiode

⚠ T_{PS} kann durch prioritätsorientierte Einplanung verzögert werden

- Fertigstellung zur 3. Abfrageperiode

☞ Termin von T_s^S muss aber gehalten werden: $2 \cdot p_{PS} \leq r_{s,1} + D_s$

- Termin D_s begrenzt die Auffüllperiode: $p_{PS} \leq D_s/2$

  → $D_s \leq i_s$, falls $J_{s,1}$ vor $r_{s,2}$ fertiggestellt sein muss

⚠ Gefahr der Überlast – normalerweise gilt: $i_s \ll \overline{r_{s,i+1} - r_{s,i}}$

- Minimale Zwischenankunftszeiten sind u.U. sehr kurz! (vgl. Folie 16)

Nachteile des Abfragebetriebs

Verfall des noch nicht vollständig ausgeschöpften Ausführungsbudgets eines untätigen Abfragers

→ Längere Antwortzeiten im Falle aperiodischer Aufgaben

→ Überlast im Fall sporadischer Aufgaben

- Sporadische Ereignisse müssen sehr hochfrequent abgefragt werden

- Eintreffende nicht-periodische Arbeitsaufträge bleiben in der laufenden Abfrageperiode unberücksichtigt
  - Behandlung verspäteter Aufträge frühestens zur nächsten Abfrageperiode
  - Vergleiche Beispiel Folie 24

→ Ansammlung in der Warteschlange, der Abfrager wird zurückgestellt

Restbudget eines Abfragers müsste bewahrt werden

Qual der Wahl. . .

Abfragender Zusteller $\Longleftrightarrow$ Unterbrecherbetrieb $\Longleftrightarrow$ Hintergrundbetrieb

- Unterbrecherbetrieb:
  - \+ Sehr gute Antwortzeiten für nicht-periodische Arbeitsaufträge
  - – Erfordert die Behandlung von Unterbrechungen
  - – Verzögert periodische Arbeitsaufträge $\rightsquigarrow$ Überlastgefahr
- Hintergrundbetrieb:
  - \+ Liefert korrekte Ablaufpläne[5]
  - – Benötigt Verdrängung
  - – Lange Antwortzeiten für nicht-periodische Arbeitsaufträge
- Abfragender Zusteller:
  - \+ Einfache Implementierung auf Anwendungsebene
    - – Nicht-periodischer Arbeitsaufträge $\mapsto$ periodische Aufgabe
  - \+ Liefert korrekte Ablaufpläne[5]
  - – Lange Antwortzeiten durch Aufgabe des Ausführungsbudgets
  - – Hoher Overhead durch den Abfragebetrieb

[5] Aus Sicht der periodischen Aufgaben und deren mathematischen Annahmen.

Gliederung

1 Nicht-periodische Aufgaben
- Nicht-periodische Echtzeitanwendungen
- Zeitliche Parameter
- Herausforderung: Mischbetrieb

2 Erweiterte Behandlung nicht-periodischer Aufgaben
- Unterbrecherbetrieb
- Hintergrundbetrieb
- Periodischer Zusteller

3 Slack-Stealing
- Taktgesteuerte Systeme
- Ereignisgesteuerte Systeme

4 Zusammenfassung

Slack-Stealing

Die Abfertigung nicht-periodischer Jobs durch Nutzung der Schlupfzeit optimieren

- Echtzeitbetrieb bedeutet Rechtzeitigkeit (vgl. Folie II/11)

☞ Fertigstellung von Arbeitsaufträgen vor ihrem Termin unnötig

  - Kurz vor oder genau zum Termin ausreichend

- Verschiebung periodischer Aufträge um ihre Schlupfzeit
  - Ausführung nicht-periodischer Aufträge in den entstehenden Lücken

→ Schlupfdieb (engl. *slack stealing*)

⚠ Termine periodischer Aufträge dürfen nicht gefährdet werden

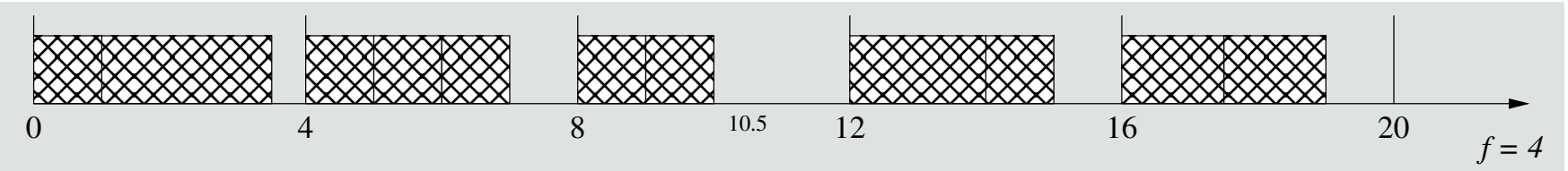
  - Nach dem Aufbrauchen des Schlupfs
    - Suspendierung des gerade ausgeführten nicht-periodischen Auftrags
    - Direkte Einlastung des verzögerten periodischen Auftrags

- Slack-Stealing existiert für takt- und für ereignisgesteuerte Systeme

Ausgangspunkt: Taktsteuerung

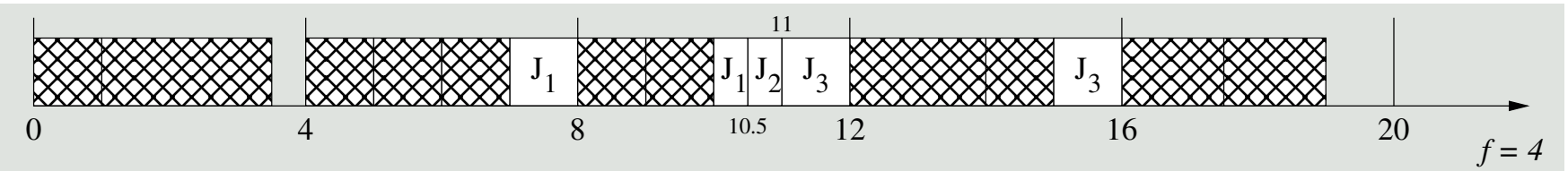
Beispiel: *Major Cycle* eines zyklischen Ablaufplans (vgl. IV-3/10)

- Beispiel: Fünf Schlupfbereiche im große Durchlauf



  - Schraffierte Bereiche $\mapsto$ statisch eingeplante periodische Aufträge

- Jobs $J_1 \mapsto ([4, \infty[, 1.5)$, $J_2 \mapsto ([9.5, \infty[, 0.5)$, $J_3 \mapsto ([10.5, \infty[, 2)$



  - Ausführungszeiten 1.5, 0.5 und 2
  - Zulässige Ausführungsintervalle $[earliest, latest]$
  - Jobs haben keinen harten Termin

- Mittlere Antwortzeit: $((10.5-4)+(11-9.5)+(16-10.5))/3 = 4.5$

Taktsteuerung und Slack-Stealing

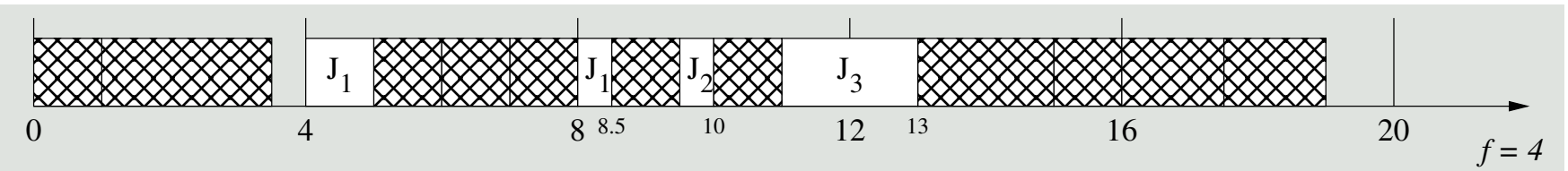
- Schlupf in Rahmen k ist die Zeitspanne $f - e_P$, mit e_P Zeiteinheiten für periodische Aufträge in k

☞ Periodischen Aufträgen Zeitpuffer am Rahmenende entziehen

  - Berechnung des Schlupfes geschieht einmal vor der Laufzeit und hängt nur vom aktuellen Rahmen ab

→ Periodische Aufträge werden ans Ende ihres Rahmens geschoben

- Jobs J_1, J_2 und J_3, wie gehabt (vgl. Folie 31):



  - J_1 wird sofort eingelastet, muss jedoch verdrängt werden
  - J_2 wird ebenso behandelt, kann aber komplett durchlaufen
  - J_3 wird verzögert bis der laufende periodische Auftrag fertig ist

- Mittlere Antwortzeit: $((8.5-4)+(10-9.5)+(13-10.5))/3 = 2.5$

Ereignissteuerung und Slack-Stealing

- Konzeptionell ist Slack-Stealing auch hier einfach
  - Schlupfzeit-Dieb (engl. *slack-stealer*) arbeitet anstehende nicht-periodische Arbeitsaufträge ab, auf:

    höchster Priorität, wenn Schlupfzeit vorhanden ist, und auf
    niedrigster Priorität, wenn keine Schlupfzeit vorhanden ist

⚠ Aufwändige Berechnung der Schlupfzeit [4, S. 233 ff.]

- EDF mit dynamischen Prioritäten
  - Statische vorberechnete Schlupfzeiten
    ↦ Berücksichtigung der kompletten Hyperperiode
  - Hyperperiode ist ein *gleitendes* Zeitfenster bei nicht-periodischen Systemen
  - Keine exakten Start-/Endzeitpunkte der Hyperperioden (nur $kgV(i_i)$)
  - Dynamischen Schlupfberechnung
    ↦ Betrachtung des aktuellen Tätigkeitsintervalls[6]
  - Buchführung über Untätigkeit, gestohlenen Schlupf und bereits verbrauchte Rechenzeit der periodischen Aufträge notwendig
- RM mit statischen Prioritäten
  - Schlupfzeit hängt von ihrem Verwendungszeitpunkt ab
  - Schlupfzeit-Dieb darf daher nicht gierig (engl. *greedy*) sein

[6] Dessen Länge zunächst mit Hilfe der Zeitbedarfsanalyse (vgl. IV-2/27 ff) bestimmt werden muss.

Gliederung

1 Nicht-periodische Aufgaben
- Nicht-periodische Echtzeitanwendungen
- Zeitliche Parameter
- Herausforderung: Mischbetrieb

2 Erweiterte Behandlung nicht-periodischer Aufgaben
- Unterbrecherbetrieb
- Hintergrundbetrieb
- Periodischer Zusteller

3 Slack-Stealing
- Taktgesteuerte Systeme
- Ereignisgesteuerte Systeme

4 Zusammenfassung

Resümee

Nicht-periodische Aufgaben werden ereignisgesteuert ausgelöst

- Harte o. feste/weiche Termine (sporadische/aperiodische Aufgaben)
- Mischbetrieb ist eine Herausforderung

Unterbrecherbetrieb bevorzugt nicht-periodische Aufgaben

- Sehr gute Antwortzeit, anfällig für Überlast
- Gefährdet statische Garantien $\rightsquigarrow$ kontrollierter Unterbrecherbetrieb

Hintergrundbetrieb stellt nicht-periodische Aufgaben hinten an

- Antwortzeiten hängen von der Last periodischer Aufgaben ab

Abfragende Zusteller konvertieren sie in periodische Aufgaben

- Schlechte Antwortzeiten, Ausführungsbudget, Auffüllperiode

Slack-Stealing ist ein guter Kompromiss

- Einfache Umsetzung in gut strukturierten, zeitgesteuerten Systemen
- Nicht praktikabel in vorranggesteuerten Systemen

Literaturverzeichnis

[1] AUTOSAR:
Specification of Operating System (Version 4.0.0) / Automotive Open System Architecture GbR.
2009. –
Forschungsbericht

[2] Infineon Technologies AG (Hrsg.):
TC1796 User's Manual (V2.0).
St.-Martin-Str. 53, 81669 München, Germany: Infineon Technologies AG, Jul. 2007

[3] Jr., S. R. M.:
My Fascinating Interview with Allan Klumpp.
`http://www.unt.edu/UNT/departments/CC/Benchmarks/benchmarks_html/sepoct95/lunar.htm`, 1995

[4] Liu, J. W. S.:
Real-Time Systems.
Englewood Cliffs, NJ, USA : Prentice Hall PTR, 2000. –
ISBN 0–13–099651–3

[5] OSEK/VDX Group:
Time Triggered Operating System Specification 1.0 / OSEK/VDX Group.
2001. –
Forschungsbericht. –
`http://portal.osek-vdx.org/files/pdf/specs/ttos10.pdf`

Literaturverzeichnis (Forts.)

[6] Regehr, J. ; Duongsaa, U. :
Preventing interrupt overload.
In: *Proceedings of the 2005 ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems (LCTES '05)*.
New York, NY, USA : ACM Press, 2005. –
ISBN 1–59593–018–3, S. 50–58

[7] Zühlsdorf, R. :
Protokoll des Funkverkehrs bei der ersten Landung auf dem Mond.
`http://members.fortunecity.de/rogerzuehlsdorf/Ap11d.htm`, 1999

EZS – Cheat Sheet

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
ω_i Antwortzeit (engl. response time)
σ_i Schlupf (engl. slack)

Periodische Aufgaben

p_i Periode (engl. period)
ϕ_i Phase (engl. phase)

Nicht-Periodische Aufgaben

i_i Minimale Zwischenankunftszeit (engl. minimal interarrival-time)

Aufgaben – Tupel

$T_p = (p, e, D, \phi)$ Periodische Aufgabe ohne Priorität (zeitgesteuert oder dynamische Taskpriorität), $D = p$ und $\phi = 0$ sind der Reihe nach optional
$T_i^S = (i_i, e_i, D_i)$ Nicht-periodische Aufgabe (Schreibweise mit i_i)
$T_i^S = ([r_i^{nach}; r_i^{vor}], e_i, D_i)$ Nicht-periodische Aufgabe (Schreibweise mit Auslöseintervall)
$J_{i,j} = (r_{i,j}, e_{i,j}, d_{i,j})$ Arbeitsauftrag

Ablaufplanung

P_i Priorität (engl. priority) der Aufgabe T_i
Ω_i Prioritätsebenen (engl. number of priorities)
$h_{\Delta t}$ Rechenzeitbedarf (engl. demand)
$u_{\Delta t}$ CPU-Auslastung (engl. utilisation)
U Absolute CPU-Auslastung
H Hyperperiode (großer Durchlauf, engl. major cycle)
f Rahmenlänge (kleiner Durchlauf, engl. minor cycle)
e_i^f WCET aller Aufträge im Rahmen i

Zusteller

T_{PS} Abfragender Zusteller (engl. polling server)