

# Echtzeitsysteme

## Grundlegende Abfertigung nicht-periodischer Echtzeitsysteme

**Peter Wägemann**

Lehrstuhl für Systemsoftware

Friedrich-Alexander-Universität Erlangen-Nürnberg

<https://sys.cs.fau.de/lehre/ss24/ezs/>

5. Juni 2024



- 22.07. EZS-Prüfungen (1): 9:00, 9:45, 10:30, 12:00, 12:45, 13:30
- 25.07. EZS-Prüfungen (2): 9:00, 9:45, 10:30, 12:00, 12:45, 13:30
- 07.10. EZS-Prüfungen (4): 9:00, 9:45, 10:30, 12:00, 12:45, 13:30
- Prüfungsprotokolle: FSI Informatik
- Bitte Mail mit Terminwunsch an [i4ezs-owner@lists.cs.fau.de](mailto:i4ezs-owner@lists.cs.fau.de)

- Wiederholung **Mixed-Criticality-Scheduling** & **Moduswechsel**
- 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?



- 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



## 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).



## 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



## 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



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<sup>1</sup>

**A6** Die Kosten durch Unterbrechungen, Ablaufplanung und Verdrängung sind vernachlässigbar

**A7** Alle Aufgaben verhalten sich voll-präemptiv

---

<sup>1</sup>D.h. die einzige gemeinsame Ressource ist die CPU und es existieren keine Einschränkungen hinsichtlich der Auslösezeiten der Arbeitsaufträge voneinander.



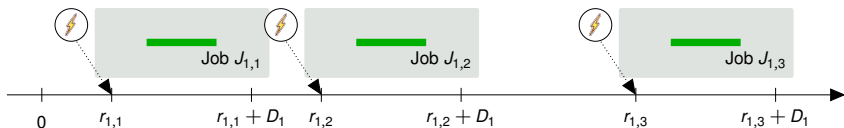




## Nicht-periodische Aufgaben (engl. *non-periodic tasks*)

### 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.



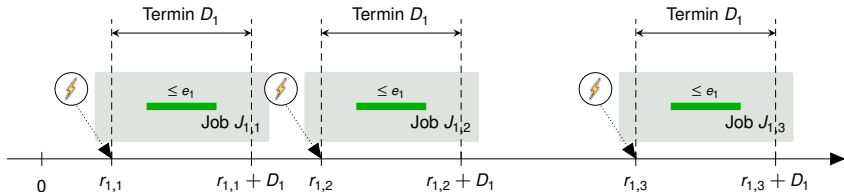


## Nicht-periodische Aufgaben (engl. *non-periodic tasks*)

### 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*)



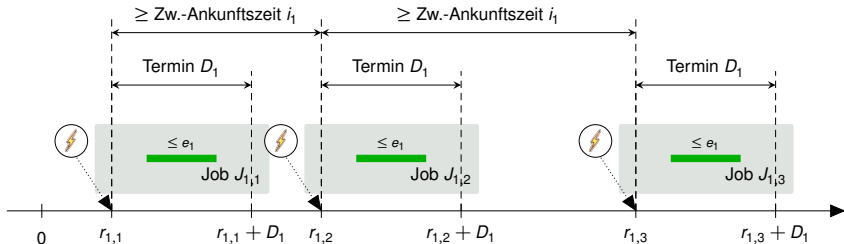


## Nicht-periodische Aufgaben (engl. *non-periodic tasks*)

### 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 inter-arrival time*)  $i_j$  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 inter-arrival time*)

$e_i$  maximale Ausführungszeit (WCET)

$D_i$  relativer Termin (engl. *deadline*), optional





## Nicht-periodische Aufgaben (Forts.)



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

$i_i$  minimale Zwischenankunftszeit (engl. *minimum inter-arrival 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 spätestem Auslösezeitpunkt sind Absolutwert und Intervallschreibweise *äquivalent*





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

$i_i$  minimale Zwischenankunftszeit (engl. *minimum inter-arrival 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 spätestem Auslösezeitpunkt sind Absolutwert und Intervallschreibweise *äquivalent*

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





## Nicht-periodische Aufgaben (Forts.)



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

$i_i$  minimale Zwischenankunftszeit (engl. *minimum inter-arrival 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 spätestem 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 $\leftrightarrow$ nicht-periodische

---



Erhaltung statischer Garantien für periodische Arbeitsaufträge

- Einfluss nicht-periodischer Arbeitsaufträge begrenzen







## 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





## 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 müssen erhalten bleiben

■ Sporadische Arbeitsaufträge → Termingarantie

- Einplanung unter Sicherstellung des vorgegebenen Termins
- Zusicherungen an periodische und bereits zugelassene sporadische Aufträge müssen erhalten bleiben

→ Zulassung oder Abweisung des sporadischen Arbeitsauftrags





## Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?



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





## Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?

- ☞ **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** ↪ *online*





# Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?



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  $\sigma \geq \text{WCET } e$  des sporadischen Auftrags
    - Ausführungszeit ggf. erst zum Auslösezeitpunkt bekannt





# Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?



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  $\sigma \geq \text{WCET } e$  des sporadischen Auftrags
    - Ausführungszeit ggf. erst zum Auslösezeitpunkt bekannt
- ⚠ Nur Schlupf **vor dem Termin** ist von Relevanz





# Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?



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  $\sigma \geq \text{WCET } e$  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





# Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?



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

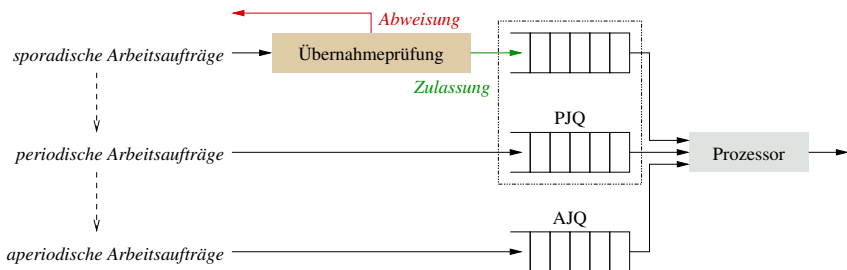


Gleichzeitige sporadische Aufträge werden oft nach EDF getestet





# Prioritätswarteschlangen im Betriebssystem



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

- Periodische Arbeitsaufträge → **periodic job queue**
  - Implementierung durch Ablaufabelle oder -liste
- Sporadische Arbeitsaufträge → **zweistufiger Prozess**:
  - 1 Ausgelöste Arbeitsaufträge warten auf Übernahmeprüfung
  - 2 Zugelassene Arbeitsaufträge → Eigene Queue oder PJQ
- Aperiodische Arbeitsaufträge → **aperiodic job queue**



- 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



## 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.



## 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**  $\rightsquigarrow$  **nicht-periodische Aufgaben haben Vorfahrt**
  - Ereignisbehandlung direkt in der Unterbrechungsbehandlung
  - Mittels **Ausnahmebehandlungen** (vgl. III-1/17 ff)



## 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**  $\rightsquigarrow$  **nicht-periodische Aufgaben haben Vorfahrt**
  - Ereignisbehandlung direkt in der Unterbrechungsbehandlung
  - Mittels **Ausnahmebehandlungen** (vgl. III-1/17 ff)
- **Hintergrundbetrieb**  $\rightsquigarrow$  **periodische Aufgaben haben Vorfahrt**
  - Phasen der Untätigkeit für nicht-periodische Aufgaben nutzen
  - Mittels **Verdrängung** (vgl. III-2/12 ff)



## 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



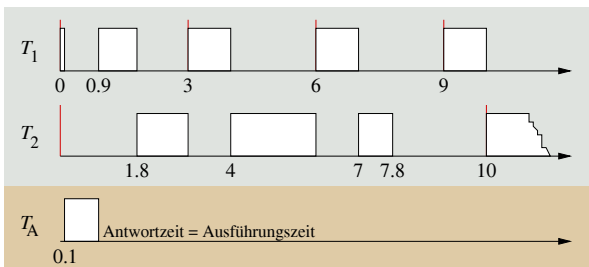


# Unterbrecherbetrieb

Antwortzeitminimierung – auf Kosten eines gut geordneten Ablaufplans



Nicht-periodische Arbeitsaufträge werden sofort ausgeführt



■ Beispiel:

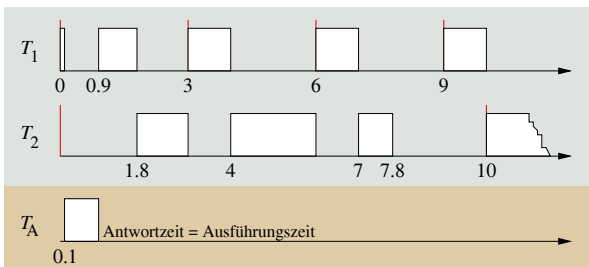
- Periodische Aufgaben  $T_1 = (3, 1)$  und  $T_2 = (10, 4)$  (nach RMA)
- Aperiodische Aufgabe  $T_A^S$  mit  $r_A = [0.1, \infty[$ ,  $e_A = 0.8$







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 (problematisch z.B. bei äquidistanter Abtastung)



- **Minimale Anforderungen** an Laufzeitsystem: kooperative Ausführung + Interrupts möglich

 **Bevorzugung nicht-periodischer Arbeitsaufträge**  
~> **Termineinhaltung** periodischer Aufgaben **nicht gesichert**



- **Minimale Anforderungen** an Laufzeitsystem: kooperative Ausführung + Interrupts möglich

⚠ **Bevorzugung nicht-periodischer Arbeitsaufträge**  
↪ **Termineinhaltung** periodischer Aufgaben **nicht gesichert**

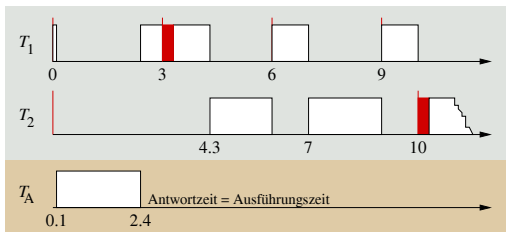
- **Beispiel** (vgl. Folie 14):

- WCET  $e_A = 2.3$
- *nicht verdrängbar*



periodische Tasks

- Terminverletzung  $T_1$
- Terminverletzung  $T_2$



- **Minimale Anforderungen** an Laufzeitsystem: kooperative Ausführung + Interrupts möglich

⚠ **Bevorzugung nicht-periodischer Arbeitsaufträge**  
↪ **Termineinhaltung** periodischer Aufgaben **nicht gesichert**

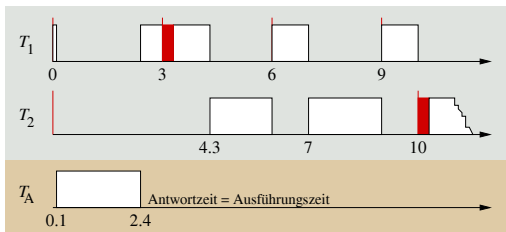
- **Beispiel** (vgl. Folie 14):

- WCET  $e_A = 2.3$
- *nicht verdrängbar*



periodische Tasks

- Terminverletzung  $T_1$
- Terminverletzung  $T_2$



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



- 1 Rendezvous-Radar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet

---

<sup>2</sup>Messung von Zeitintervallen zwischen bekannten Landmarken und Überprüfung von Position und Geschwindigkeit des Landemoduls relativ zum Kommandomodul.



- 1 Rendezvous-Radar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet
- 2 Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
  - Rendezvous-Radar erzeugte eine Flut von **Scheinunterbrechungen**
  - Dadurch wurden etwa 15 % an unerwarteter Rechenlast erzeugt→ Verzögerung/Ausfall von Berechnungen zur Landungskontrolle

---

<sup>2</sup>Messung von Zeitintervallen zwischen bekannten Landmarken und Überprüfung von Position und Geschwindigkeit des Landemoduls relativ zum Kommandomodul.



- 1 Rendezvous-Radar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet
- 2 Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
  - Rendezvous-Radar 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

---

<sup>2</sup>Messung von Zeitintervallen zwischen bekannten Landmarken und Überprüfung von Position und Geschwindigkeit des Landemoduls relativ zum Kommandomodul.



- 1 Rendezvous-Radar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet
- 2 Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
  - Rendezvous-Radar 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**

---

<sup>2</sup>Messung von Zeitintervallen zwischen bekannten Landmarken und Überprüfung von Position und Geschwindigkeit des Landemoduls relativ zum Kommandomodul.







**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**





**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]
<i>loser Draht</i>	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]





**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]
<i>loser Draht</i>	500
Kippschalter	1000
Wippschalter	1300
serielle Schnittstelle (115 kbps)	11500
Ethernet (10 Mbps)	14880
CAN-Bus	15000
I2C-Bus	50000
USB	90000
Ethernet (100 Mbps)	148800
Ethernet (1 Gbps)	1488000

- Selbst unscheinbare Komponenten können signifikante Last durch Unterbrechungen erzeugen
- Periodische Aufgaben stehen Unterbrechungen wehrlos gegenüber
- ⚠ Gefahr der **Überlast**

max. Raten verschiedener Unterbrechungsquellen [6]





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

- **Deutlich unter 100 %** CPU-Auslastung (vgl. IV-2/31)



Bestimmung der kritischen Situationen und freizuhaltende Kapazitäten ist eine große Herausforderung

- Unterbrechungsbedingte Verzögerungen und Last im Voraus einzuplanen benötigt fundiertes Expertenwissen
- Erfordert durchgehende **Anforderungsanalyse**<sup>3</sup>



Der **Fehlerfall ist dennoch nie auszuschließen**  
(Scheinunterbrechungen sind kaum vorherzusagen)

---

<sup>3</sup>(engl. *requirements engineering*), Fundament und Teilaktivität systematischer Softwareentwicklung – hier aber nicht nur Software.





**Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen





**Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen

1

Überwachung der **minimalen Zwischenankunftszeit**

- Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen (planmäßige Deaktivierung von Interrupts)





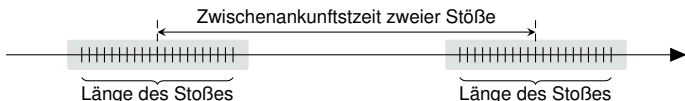
**Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen

## 1 Überwachung der **minimalen Zwischenankunftszeit**

- Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen (planmäßige Deaktivierung von Interrupts)

## 2 Überwachung von **Unterbrechungsstößen** (engl. *bursts*)

- Nach Unterbrechungsstoß werden Unterbrechungen eine Zeit lang abgeblockt





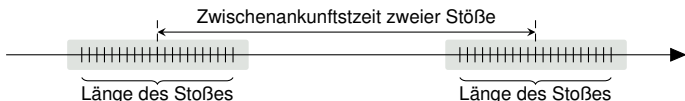
**Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen

## 1 Überwachung der **minimalen Zwischenankunftszeit**

- Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen (planmäßige Deaktivierung von Interrupts)

## 2 Überwachung von **Unterbrechungsstößen** (engl. *bursts*)

- Nach Unterbrechungsstoß werden Unterbrechungen eine Zeit lang abgeblockt



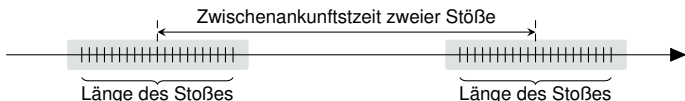
- **OSEKtime [5]** und **AUTOSAR OS [1]** schränken die minimale Zwischenankunftszeit ein oder überwachen sie (in Software)





➔ **Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen

- 1 Überwachung der **minimalen Zwischenankunftszeit**
  - Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen (planmäßige Deaktivierung von Interrupts)
- 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 (in Software)
- **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**



### 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



### 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



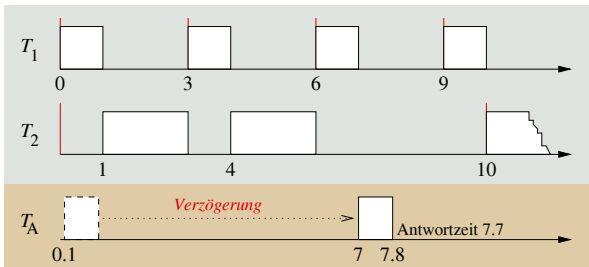


Aperiodische Arbeitsaufträge werden nur ausgeführt falls keine periodischen/sporadischen Aufträge zur Ausführung anstehen





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



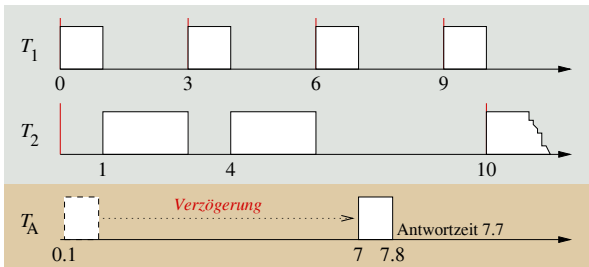
### ■ Beispiel:

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





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


~> schlechtes Antwortverhalten





# Periodischer Zusteller (engl. periodic server)

Periodische Abarbeitung aperiodischer Arbeitsaufträge

 **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








# Periodischer Zusteller (engl. periodic server)

Periodische Abarbeitung aperiodischer Arbeitsaufträge

 **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*)





# Periodischer Zusteller (engl. periodic server)

Periodische Abarbeitung aperiodischer Arbeitsaufträge

## 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*)






# Periodischer Zusteller (engl. periodic server)

Periodische Abarbeitung aperiodischer Arbeitsaufträge

 **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 Zeitintervall  $p_{srv}$  auf maximal  $e_{srv}$  Zeiteinheiten
- 
-  Verschiedene Varianten: abfragend, aufschiebbar, sporadisch





# Periodischer Zusteller (engl. periodic server)

Periodische Abarbeitung aperiodischer Arbeitsaufträge

 **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 Zeitintervall  $p_{srv}$  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:





Phasen der Auslösung, Bereitstellung und Ausführung eines Zustellers:

1

Untätig (engl. *idle*):

- Warteschlange des Zusteller ist leer





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





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





# 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  $T_s = (p_s, e_s)$



# 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



# 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

---

<sup>4</sup>Eintreffende aperiodische Arbeitsaufträge nachdem der Abfrager seine Untätigkeit festgestellt hat, kommen frühestens in der nächsten Abfrageperiode zum Zuge.



## 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<sup>4</sup>

---

<sup>4</sup>Eintreffende aperiodische Arbeitsaufträge nachdem der Abfrager seine Untätigkeit festgestellt hat, kommen frühestens in der nächsten Abfrageperiode zum Zuge.



## 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<sup>4</sup>

 Antwortzeiten nicht-periodischer Arbeitsaufträge unterliegen mitunter **starken Schwankungen**

- Abhängig vom Auslösezeitpunkt des Auftrag/Zustand des Abfragers

---

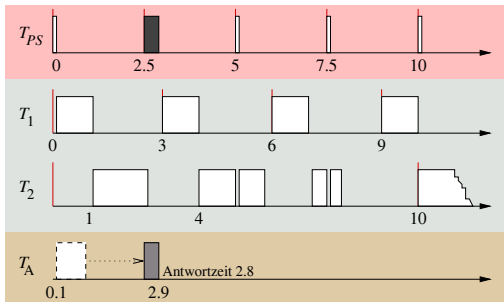
<sup>4</sup>Eintreffende aperiodische Arbeitsaufträge nachdem der Abfrager seine Untätigkeit festgestellt hat, kommen frühestens in der nächsten Abfrageperiode zum Zuge.



# 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
    - $T_A^S \mapsto ([0.1, \infty[, 0.4)$



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

### Aufgabensystem:

#### ■ periodische Aufgabe

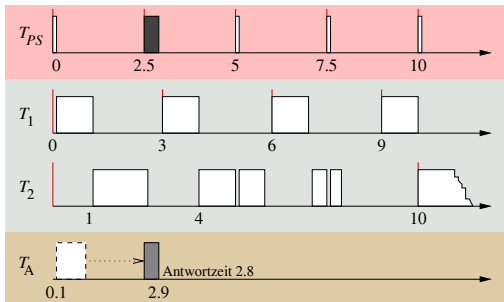
–  $T_{PS} = (2.5, 0.5)$

–  $T_1 = (3, 1)$

–  $T_2 = (10, 4)$

#### ■ aperiodischer Aufgabe

–  $T_A^S \mapsto ([0.1, \infty[, 0.4)$



1 Zusteller  $T_{PS}$  hat kürzeste Periode  $\leadsto$  höchste Priorität (RMA)





# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

### Aufgabensystem:

#### ■ periodische Aufgabe

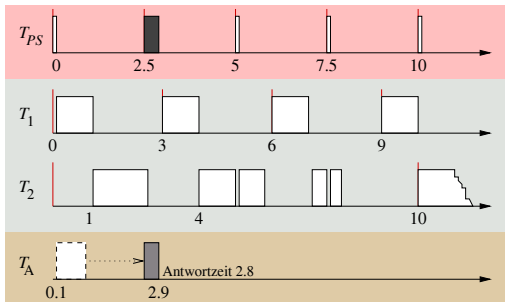
–  $T_{PS} = (2.5, 0.5)$

–  $T_1 = (3, 1)$

–  $T_2 = (10, 4)$

#### ■ aperiodischer Aufgabe

–  $T_A^S \mapsto ([0.1, \infty[, 0.4)$



1 Zusteller  $T_{PS}$  hat kürzeste Periode  $\leadsto$  höchste Priorität (RMA)

2 Zu Beginn der Abfrageperioden  $t_0$  ist die AJQ leer

→ Das Budget von  $T_{PS}$  **verfällt**



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

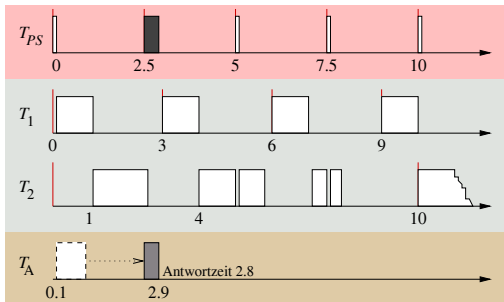
### Aufgabensystem:

#### ■ 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  $\leadsto$  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  $t_{2.5}$



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

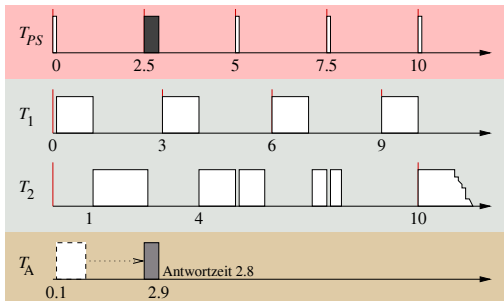
### Aufgabensystem:

#### ■ 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  $\leadsto$  höchste Priorität (RMA)

2 Zu Beginn der Abfrageperioden  $t_0$  ist die AJQ leer

$\rightarrow$  Das Budget von  $T_{PS}$  **verfällt**

3 Auslösezeitpunkt  $r_A$  ist  $t_{0,1}$

■ Kurz nach dem Abfragezeitpunkt  $t_0$

$\rightarrow$  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  $T_A^S$



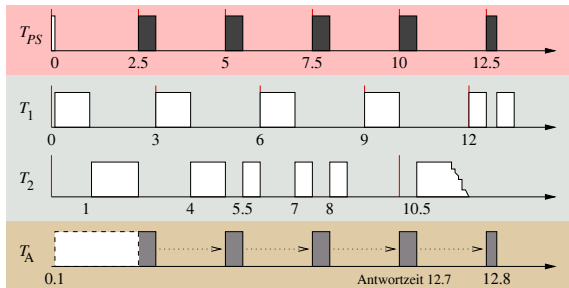
# Abfragender Zusteller – Beispiel (2)

Aufteilung eines aperiodischen Arbeitsauftrags auf mehrere Auffüllperioden

■ Aufgabensystem:

$\left. \begin{matrix} T_{PS} \\ T_1 \\ T_2 \end{matrix} \right\} \text{ wie gehabt}$

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

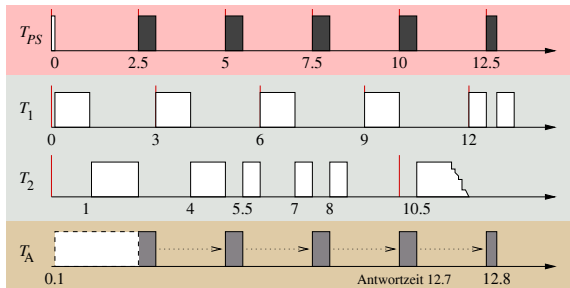


# Abfragender Zusteller – Beispiel (2)

Aufteilung eines aperiodischen Arbeitsauftrags auf mehrere Auffüllperioden

## Aufgabensystem:

$$\left. \begin{array}{l} T_{PS} \\ T_1 \\ T_2 \end{array} \right\} \text{wie gehabt}$$
$$T_A^S \mapsto ([0.1, \infty[, 2.3)$$



☞ Ausführung von  $A$  benötigt (mindestens) fünf Abfrageperioden:

$4 \times 0.5$  Zeiteinheiten (dem Budget von  $T_{PS}$ )

- Abfrageperioden  $t_{2.5}$ ,  $t_5$ ,  $t_{7.5}$  und  $t_{10}$

$1 \times 0.3$  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  $T_s^S = (j_s, e_s, D_s)$  mit Auslösezeitpunkt  $r_{s,1}$  von  $J_{s,1}$



# 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  $T_s^S = (j_s, e_s, D_s)$  mit Auslösezeitpunkt  $r_{s,1}$  von  $J_{s,1}$



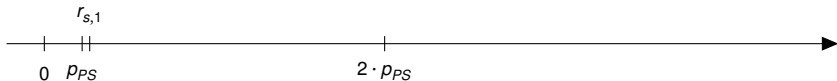
- Schlimmstenfalls wird der **erste Abtastzeitpunkt** verpasst:  $r_{s,1} > p_{PS}$



# 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  $T_s^S = (j_s, e_s, D_s)$  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

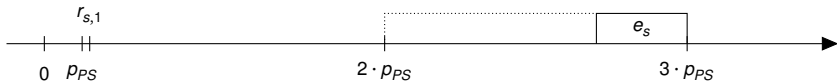




# Abfragende Zusteller und sporadische Aufgaben

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

- Periodische Zusteller  $T_{PS} = (p_{PS}, e_{PS})$  behandelt sporadische Aufgabe  $T_s^S = (j_s, e_s, D_s)$  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**

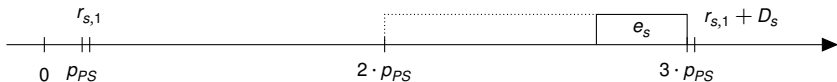
👉 Termin von  $T_s^S$  muss aber gehalten werden:  $2 \cdot p_{PS} \leq r_{s,1} + D_s$



# 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  $T_s^S = (j_s, e_s, D_s)$  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:  $p_{PS} \leq D_s/2$

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

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

- Minimale Zwischenankunftszeiten sind u.U. sehr kurz! (vgl. Folie 17)





**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





**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 25
  - Ansammlung in der Warteschlange, der Abfrager wird zurückgestellt





**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 25
- 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  $\leadsto$  Überlastgefahr

<sup>5</sup>Aus Sicht der periodischen Aufgaben und deren mathematischen Annahmen.





## 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  $\leadsto$  Überlastgefahr

### ■ Hintergrundbetrieb:

- + Liefert korrekte Ablaufpläne<sup>5</sup>
- Benötigt Verdrängung
- Lange Antwortzeiten für nicht-periodische Arbeitsaufträge

<sup>5</sup>Aus Sicht der periodischen Aufgaben und deren mathematischen Annahmen.





# Qual der Wahl. . .

Abfragender Zusteller  $\leftrightarrow$  Unterbrecherbetrieb  $\leftrightarrow$  Hintergrundbetrieb

## ■ Unterbrecherbetrieb:

- + Sehr gute Antwortzeiten für nicht-periodische Arbeitsaufträge
- Erfordert die Behandlung von Unterbrechungen
- Verzögert periodische Arbeitsaufträge  $\leadsto$  Überlastgefahr

## ■ Hintergrundbetrieb:

- + Liefert korrekte Ablaufpläne<sup>5</sup>
- Benötigt Verdrängung
- Lange Antwortzeiten für nicht-periodische Arbeitsaufträge

## ■ Abfragender Zusteller:

- + Einfache Implementierung auf Anwendungsebene
  - Nicht-periodischer Arbeitsaufträge  $\mapsto$  periodische Aufgabe
- + Liefert korrekte Ablaufpläne<sup>5</sup>
- Lange Antwortzeiten durch Aufgabe des Ausführungsbudgets
- Hoher Overhead durch den Abfragebetrieb

<sup>5</sup>Aus Sicht der periodischen Aufgaben und deren mathematischen Annahmen.





- 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



- Echtzeitbetrieb bedeutet **Rechtzeitigkeit** (vgl. Folie II/11)
- ☞ Fertigstellung von Arbeitsaufträgen **vor ihrem Termin** unnötig
  - Kurz vor oder genau zum Termin ausreichend



- **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

Die Abfertigung nicht-periodischer Jobs durch Nutzung der Schlupfzeit optimieren

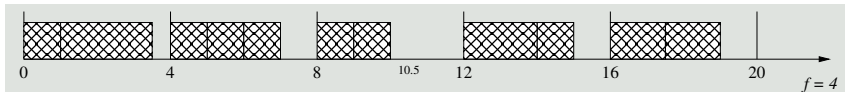
- **Echtzeitbetrieb** bedeutet **Rechtzeitigkeit** (vgl. Folie II/11)
- ☞ Fertigstellung von Arbeitsaufträgen **vor ihrem Termin** unnötig
  - Kurz vor oder genau zum Termin ausreichend
- Verschiebung periodischer Aufträge um ihre Schlupfzeit
  - Ausführung nicht-periodischer Aufträge in den entstehenden 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



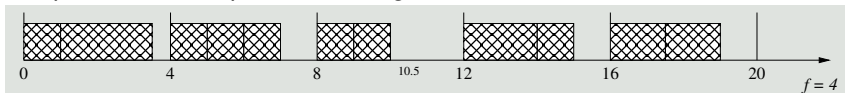
- Schraffierte Bereiche  $\mapsto$  statisch eingeplante **periodische Aufträge**



# Ausgangspunkt: Taktsteuerung

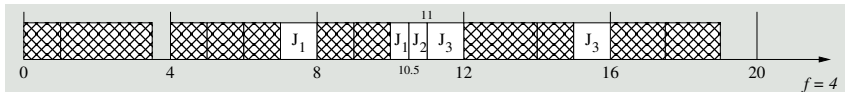
Beispiel: *Major Cycle* eines zyklischen Ablaufplans (vgl. IV-3/10)

## ■ Beispiel: Fünf Schlupfbereiche im große Durchlauf



■ Schraffierte Bereiche  $\mapsto$  statisch eingeplante **periodische Aufträge**

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



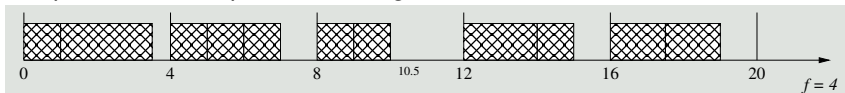
- Ausführungszeiten 1.5, 0.5 und 2
- Zulässige Ausführungsintervalle [*earliest*, *latest*]
- Jobs haben keinen harten Termin



# Ausgangspunkt: Taktsteuerung

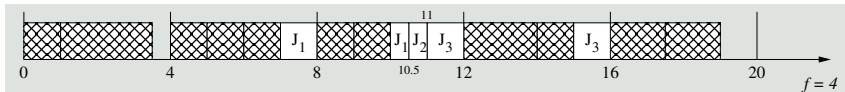
Beispiel: *Major Cycle* eines zyklischen Ablaufplans (vgl. IV-3/10)

## ■ 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: $((10.5 - 4) + (11 - 9.5) + (16 - 10.5))/3 = 4.5$



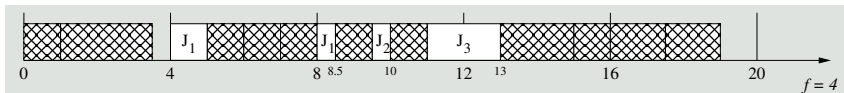
- Schlupf in Rahmen  $k$  ist die **Zeitspanne**  $f - e_p$ , mit  $e_p$  Zeiteinheiten für periodische Aufträge in  $k$
- ☞ Periodischen Aufträgen **Zeitpuffer am Rahmenende entziehen**
  - Berechnung des Schlupfes geschieht einmal vor der Laufzeit und hängt nur vom aktuellen Rahmen ab
  - Periodische Aufträge werden ans Ende ihres Rahmens geschoben





# 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 32):

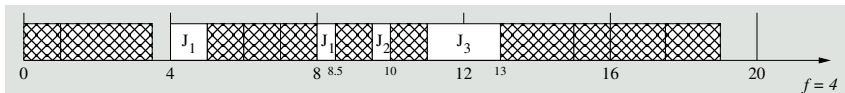


- $J_1$  wird sofort eingelastet, muss jedoch verdrängt werden
- $J_2$  wird ebenso behandelt, kann aber komplett durchlaufen
- $J_3$  wird verzögert bis der laufende periodische Auftrag fertig ist



# 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 32):



- $J_1$  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


---

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



# 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.]

---

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



# 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**  $\rightarrow$  Berücksichtigung der **Hyperperiode**
  - **Hyperperiode ist ein gleitendes Zeitfenster (engl. sliding window)** bei nicht-periodischen Systemen
  - Keine exakten Start-/Endzeitpunkte der Hyperperioden (nur  $kgV(i_j)$ )
  - **Dynamischen Schlupfberechnung**  $\rightarrow$  Betrachtung des **aktuellen Tätigkeitsintervalls**<sup>6</sup>
  - Buchführung über **Untätigkeit, gestohlenen Schlupf** und **bereits verbrauchte Rechenzeit** der periodischen Aufträge notwendig

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



# 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  $\mapsto$  Berücksichtigung der Hyperperiode
  - Hyperperiode ist ein *gleitendes Zeitfenster* (engl. *sliding window*) bei nicht-periodischen Systemen
  - Keine exakten Start-/Endzeitpunkte der Hyperperioden (nur  $kgV(i_j)$ )
  - Dynamischen Schlupfberechnung  $\mapsto$  Betrachtung des aktuellen Tätigkeitsintervalls<sup>6</sup>
  - 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

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



- 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 werden ereignisgesteuert ausgelöst

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





**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**  $\leadsto$  kontrollierter Unterbrecherbetrieb



**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



**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**  $\leadsto$  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



**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**  $\leadsto$  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 vorrangesteuerten Systemen



- [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](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>

- [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



## 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  
 $\omega_i$  Antwortzeit (engl. response time)  
 $\sigma_i$  Schlupf (engl. slack)

Periodische Aufgaben

$p_i$  Periode (engl. period)  
 $\phi_i$  Phase (engl. phase)

Nicht-Periodische Aufgaben

$i_j$  Minimale Zwischenankunftszeit (engl. minimal inter-arrival 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

$T_i^S = (i_j, e_j, D_j)$  Nicht-periodische Aufgabe (Schreibweise mit  $i_j$ )

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

$\Omega_i$  Prioritätsebenen (engl. number of priorities)

$h_{\Delta_i}$  Rechenzeitbedarf (engl. demand)

$u_{\Delta_i}$  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  $i$

$LO$  niedrigstes Kritikalitätslevel (MCS)

$HI$  höchstes Kritikalitätslevel (MCS)

## Zusteller

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

