

# Echtzeitsysteme

## Grundlegende Abfertigung nicht-periodischer Echtzeitsysteme

**Peter Ulbrich**

Lehrstuhl für Verteilte Systeme und Betriebssysteme  
Friedrich-Alexander-Universität Erlangen-Nürnberg

<https://www4.cs.fau.de>

02. Dezember 2015



- 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
  - Zeitparameter nicht-periodischer Aufgaben
  - 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 Aufgabe (engl. *non-periodic task*)

Bieten deutlich weniger verwertbares Vorabwissen

## Nicht-periodische Aufgaben

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



## Nicht-periodische Aufgaben

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



Nicht-periodische Aufgabe  $T_i = (i_i, e_i, D_i)$

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

$$J_{i,j} = (r_{i,j}, e_{i,j}, d_{i,j})$$



# Nicht-periodische Aufgabe (engl. *non-periodic task*)

Bieten deutlich weniger verwertbares Vorabwissen

## Nicht-periodische Aufgaben

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



Nicht-periodische Aufgabe  $T_i = (i_i, e_i, D_i)$

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

$$J_{i,j} = (r_{i,j}, e_{i,j}, d_{i,j})$$

- **Weiche/feste** Termine  $\mapsto$  **Aperiodische Aufgabe** (engl. *aperiodic task*)
- **Harte Termine**  $\mapsto$  **Sporadische Aufgaben** (engl. *sporadic tasks*)



# Nicht-periodische Aufgabe (engl. *non-periodic task*)

Bieten deutlich weniger verwertbares Vorabwissen

## Nicht-periodische Aufgaben

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



Nicht-periodische Aufgabe  $T_i = (i_i, e_i, D_i)$

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

$$J_{i,j} = (r_{i,j}, e_{i,j}, d_{i,j})$$

- **Weiche/feste** Termine  $\mapsto$  **Aperiodische Aufgabe** (engl. *aperiodic task*)
- **Harte Termine**  $\mapsto$  **Sporadische Aufgaben** (engl. *sporadic tasks*)

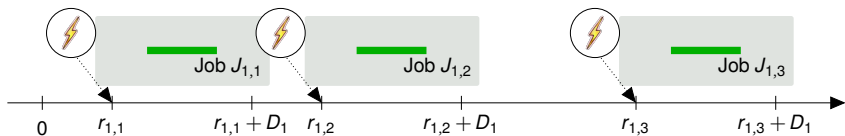


A-priori deutlich **weniger Wissen** als für periodische Aufgaben verfügbar

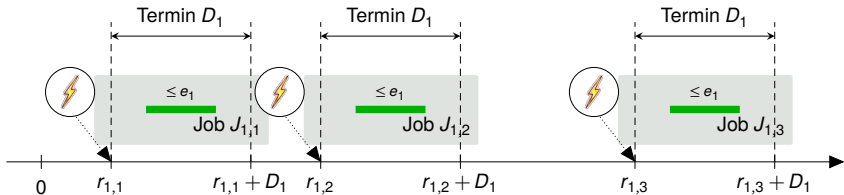
- Kaum Aussagen zum **Zeitpunkt ihrer Auslösung** möglich



# Nicht-periodische Aufgaben auf der Echtzeitachse



# Nicht-periodische Aufgaben auf der Echtzeitachse

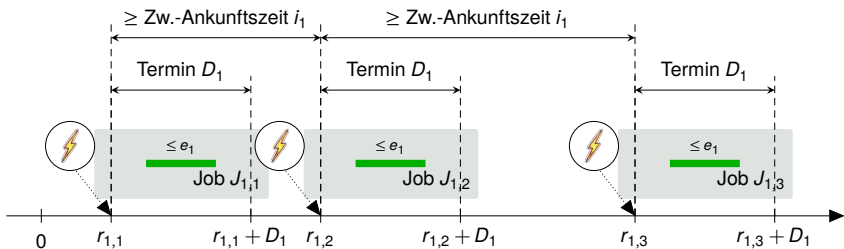


**WCET  $e_j$**  Maximale Ausführungszeit aller Aufträge  $J_{i,j}$  in  $T_i$

**relativer Termin  $D_j$**  Maximale Spanne zwischen Auslösezeit  $r_{i,j}$  und Fertigstellung  $\rightarrow$  absoluter Termin  $d_{i,j}$  von  $J_{i,j}$



# Nicht-periodische Aufgaben auf der Echtzeitachse



**WCET  $e_j$**  Maximale Ausführungszeit aller Aufträge  $J_{i,j}$  in  $T_i$

**relativer Termin  $D_j$**  Maximale Spanne zwischen Auslösezeit  $r_{i,j}$  und Fertigstellung  $\rightarrow$  absoluter Termin  $d_{i,j}$  von  $J_{i,j}$

**Zwischenankunftszeit  $i_j$**  Minimales Intervall  $[r_{i,j}; r_{i,j+1}]$  zwischen den Auslösezeiten der Aufträge in  $T_i$





**Erhaltung statischer Garantien** für periodische Arbeitsaufträge

- Einfluss nicht-periodischer Arbeitsaufträge begrenzen





**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 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

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*





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*
  - Gegenwärtige Ablaufplan muss **genügend Schlupf** aufweisen
    - Schlupf  $\sigma \geq$  WCET  $e$  des sporadischen Auftrags
    - Ausführungszeit ggf. erst zum Auslösezeitpunkt bekannt





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*
  - Gegenwärtige 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





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*
  - Gegenwärtige 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



# Zulassung sporadischer Arbeitsaufträge

Wann lassen sich die Zusicherungen einhalten?

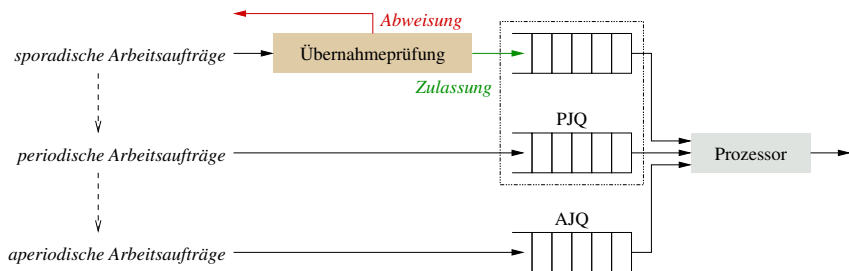


Übernahmeprüfung (engl. *acceptance test*) für sporadische Aufträge bei ereignisbedingter Auslösung

- Arbeitsauftrag wird **angenommen** falls gemeinsame Ausführung mit allen anderen Aufträgen machbar ist:
  - Test ist **gekoppelt mit der Einplanung**  $\mapsto$  *online*
  - Gegenwärtige 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
  - Zeitparameter nicht-periodischer Aufgaben
  - 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 takt- als auch für vorranggesteuerte 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 takt- als auch für vorranggesteuerte 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)



## Behandlung nicht-periodischer Aufgaben

Grundlegende Behandlungsmethoden für nicht-periodische Ereignisse lassen sich mit minimaler Unterstützung des Laufzeitsystems umsetzen. Sie sind sowohl für takt- als auch für vorranggesteuerte 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)



## Behandlung nicht-periodischer Aufgaben

Grundlegende Behandlungsmethoden für nicht-periodische Ereignisse lassen sich mit minimaler Unterstützung des Laufzeitsystems umsetzen. Sie sind sowohl für takt- als auch für vorranggesteuerte 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-periodische 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

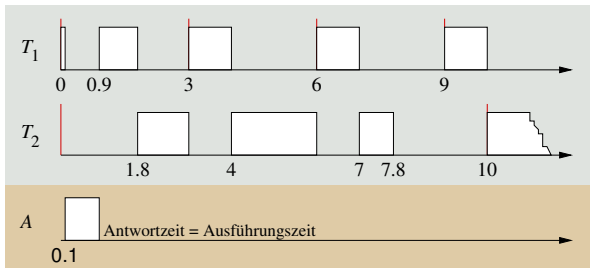


# Unterbrecherbetrieb

Antwortzeitminimierung – auf Kosten eines gut geordneten Ablaufplans



Nicht-periodische Arbeitsaufträge werden sofort ausgeführt



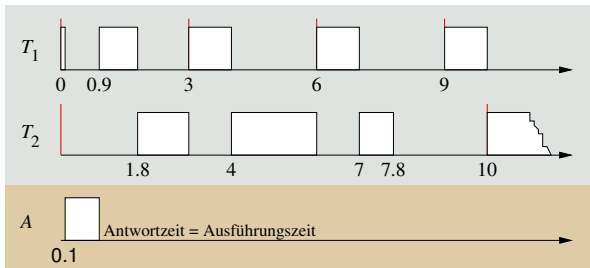
## ■ Beispiel:

- Periodische Aufgaben  $T_1 = (3, 1)$  und  $T_2 = (10, 4)$  (nach RM)
- Aperiodische Aufgabe  $A$  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 RM)
- Aperiodische Aufgabe  $A$  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





Bevorzugung nicht-periodischer Arbeitsaufträge

~> **Termineinhaltung** periodischer Aufgaben **nicht gesichert**



⚠ Bevorzugung nicht-periodischer Arbeitsaufträge

~> **Termineinhaltung** periodischer Aufgaben **nicht gesichert**

■ Beispiel (vgl. Folie 13):

■ WCET  $e_A = 2.3$

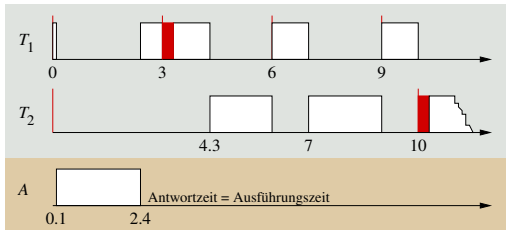
■ *run to completion*



periodische Tasks

■ Terminverletzung  $T_1$

■ Terminverletzung  $T_2$



⚠ Bevorzugung nicht-periodischer Arbeitsaufträge

→ **Termineinhaltung** periodischer Aufgaben **nicht gesichert**

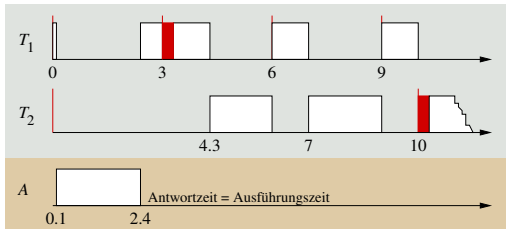
■ Beispiel (vgl. Folie 13):

- 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)



- 1 Rendezvousradar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet  
⚠ Falsche Vorgabe der Checkliste an die Astronauten

---

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



- 1 Rendezvousradar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet  
⚠ Falsche Vorgabe der Checkliste an die Astronauten
- 2 Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
  - Netzteile von Radar und Landeeinheit waren nicht synchronisiert
  - Rendezvousradar 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 Rendezvousradar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet  
⚠ Falsche Vorgabe der Checkliste an die Astronauten
- 2 Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
  - Netzteile von Radar und Landeeinheit waren nicht synchronisiert
  - 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

---

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



- 1 Rendezvousradar<sup>2</sup> wurde vor Beginn der Landung eingeschaltet  
⚠ Falsche Vorgabe der Checkliste an die Astronauten
- 2 Radarsteuerprogramm beanspruchte jedoch zu viel Rechenzeit
  - Netzteile von Radar und Landeeinheit waren nicht synchronisiert
  - 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

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



# Kontrollierter Unterbrecherbetrieb

Die Beeinflussung periodischer Aufgaben durch nicht-periodischer Aufgaben einschränken



**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**



# Kontrollierter Unterbrecherbetrieb

Die Beeinflussung periodischer Aufgaben durch nicht-periodischer Aufgaben einschränken



**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 |
|-----------------------------------|---------------|
| Messerschalter                    | 333           |
| 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 [5]



# Kontrollierter Unterbrecherbetrieb

Die Beeinflussung periodischer Aufgaben durch nicht-periodischer Aufgaben einschränken



**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 |
|-----------------------------------|---------------|
| Messerschalter                    | 333           |
| 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       |

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

⚠ Gefahr der **Überlast**

max. Raten verschiedener Unterbrechungsquellen [5]





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

- **Deutlich unter 100 %** CPU-Auslastung (vgl. IV-2/26)



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





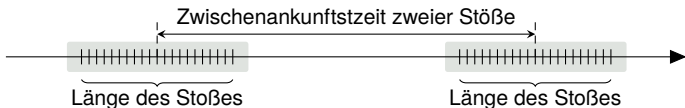
**Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen

**1** Überwachung der **minimalen Zwischenankunftszeit**

- Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen

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

- Nach einem Unterbrechungsstoß werden Unterbrechungen eine Zeit lang abgeblockt



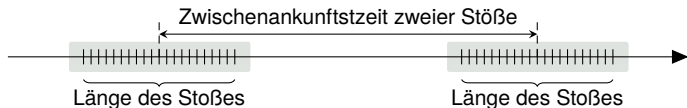
👉 **Ansatzpunkt:** Beschränkung des Auftretens von Unterbrechungen

## 1 Überwachung der minimalen Zwischenankunftszeit

- Nächste Unterbrechung wird erst nach Ablauf der minimalen Zwischenankunftszeit angenommen

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

- Nach einem Unterbrechungsstoß werden Unterbrechungen eine Zeit lang abgeblockt



- Einsatz in der Praxis: **OSEKtime** [4] und **AUTOSAR OS** [1] schränken die minimale Zwischenankunftszeit ein oder überwachen sie



# 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**



# 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



# 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 werden zugunsten periodischer Aufträge **verzögert**

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



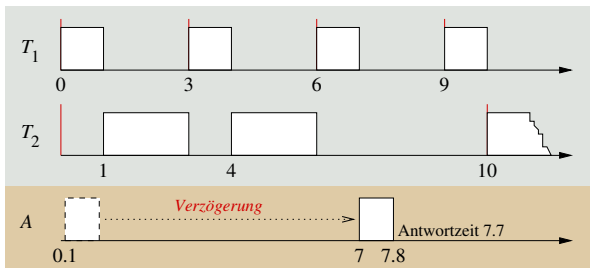


Aperiodische Arbeitsaufträge werden nur ausgeführt falls keine periodischen/sporadischen Aufträge zur Ausführung anstehen





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



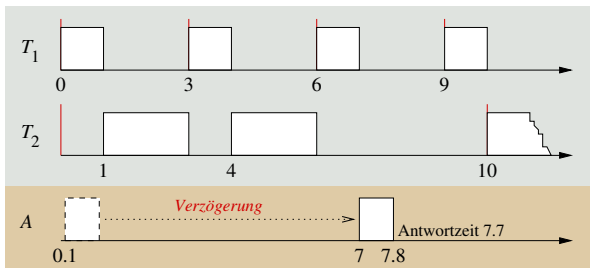
### ■ Beispiel:

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



Antwortzeiten nicht-periodischer Arbeitsaufträge bleibt unberücksichtigt  $\leadsto$  schlechtes Antwortverhalten





**Zusteller:** Spezialisierung einer periodischen Aufgabe

- Definiert durch **Periode**  $p_s$  und **Ausführungszeit**  $e_s$ 
  - Verhältnis  $u_s = e_s/p_s \mapsto$  Größe (engl. *size*) des Zustellers





### Zusteller: Spezialisierung einer periodischen Aufgabe

- Definiert durch **Periode**  $p_s$  und **Ausführungszeit**  $e_s$ 
  - Verhältnis  $u_s = e_s/p_s \mapsto$  Größe (engl. *size*) des Zustellers
- **Ausführungsbudget** (engl. *execution budget*) in Form der  $e_s$ 
  - Budget wird um bis zu  $e_s$  Einheiten aufgefüllt (engl. *replenished*)





### Zusteller: Spezialisierung einer periodischen Aufgabe

- Definiert durch **Periode**  $p_s$  und **Ausführungszeit**  $e_s$ 
  - Verhältnis  $u_s = e_s/p_s \mapsto$  Größe (engl. *size*) des Zustellers
- **Ausführungsbudget** (engl. *execution budget*) in Form der  $e_s$ 
  - Budget wird um bis zu  $e_s$  Einheiten aufgefüllt (engl. *replenished*)
- **Auffüllperiode**  $p_s$  (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**  $p_s$  und **Ausführungszeit**  $e_s$ 
    - Verhältnis  $u_s = e_s/p_s \mapsto$  Größe (engl. *size*) des Zustellers
  - **Ausführungsbudget** (engl. *execution budget*) in Form der  $e_s$ 
    - Budget wird um bis zu  $e_s$  Einheiten aufgefüllt (engl. *replenished*)
  - **Auffüllperiode**  $p_s$  (engl. *replenishment period*)
    - Das Budget des Zustellers wird regelmäßig erneuert
    - **Auffüllzeit** (engl. *replenishment time*)
  - Begrenzung der Ausführungszeit nicht-periodischer Arbeitsaufträge im Zeitintervall  $p_s$  auf maximal  $e_s$  Zeiteinheiten
-  Verschiedene Varianten: abfragend, aufschiebbar, sporadisch



# Periodischer Zusteller (engl. periodic server)

## Periodische Abarbeitung aperiodischer Arbeitsaufträge



**Zusteller:** Spezialisierung einer periodischen Aufgabe

- Definiert durch **Periode**  $p_s$  und **Ausführungszeit**  $e_s$ 
  - Verhältnis  $u_s = e_s/p_s \mapsto$  Größe (engl. *size*) des Zustellers
- **Ausführungsbudget** (engl. *execution budget*) in Form der  $e_s$ 
  - Budget wird um bis zu  $e_s$  Einheiten aufgefüllt (engl. *replenished*)
- **Auffüllperiode**  $p_s$  (engl. *replenishment period*)
  - Das Budget des Zustellers wird regelmäßig erneuert
  - **Auffüllzeit** (engl. *replenishment time*)

- Begrenzung der Ausführungszeit nicht-periodischer Arbeitsaufträge im Zeitintervall  $p_s$  auf maximal  $e_s$  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



# 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



# 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



# 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*)  $\mapsto T_P = (p_P, e_P)$

- Abfrageperiode (engl. *polling period*)  $p_P$
- Zyklisch bereitgestelltes Budget von  $e_P$  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*)  $\mapsto T_P = (p_P, e_P)$

- Abfrageperiode (engl. *polling period*)  $p_P$
- Zyklisch bereitgestelltes Budget von  $e_P$  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*)  $\mapsto T_P = (p_P, e_P)$

- **Abfrageperiode** (engl. *polling period*)  $p_P$
- Zyklisch bereitgestelltes Budget von  $e_P$  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

---

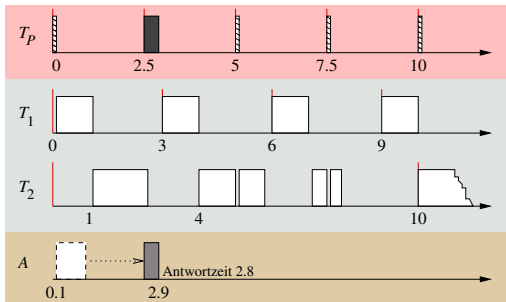
<sup>4</sup>Eintreffende aperiodische Arbeitsaufträge nachdem der Abfrager seine Untätigkeit festgestellt hat, kommen frühestens in der nächsten Abfrageperiode zum Zuge.



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

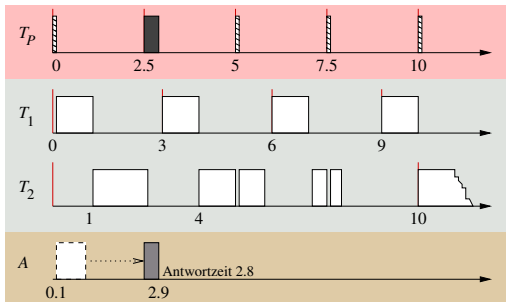
- Aufgabensystem:
  - periodische Tasks
    - $T_p = (2.5, 0.5)$
    - $T_1 = (3, 1)$
    - $T_2 = (10, 4)$
    - RM
  - aperiodischer Job
    - $A \mapsto (0.4, [0.1, \infty])$



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

- Aufgabensystem:
  - periodische Tasks
    - $T_p = (2.5, 0.5)$
    - $T_1 = (3, 1)$
    - $T_2 = (10, 4)$
    - RM
  - aperiodischer Job
    - $A \mapsto (0.4, [0.1, \infty])$



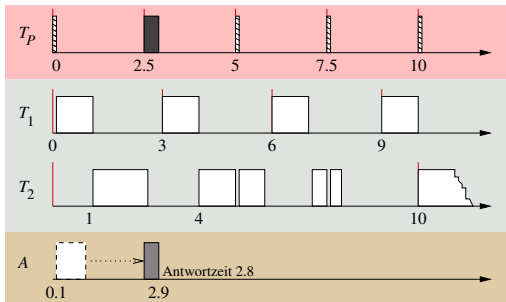
- 1 Zusteller  $T_p$  hat kürzeste Periode  $\leadsto$  höchste Priorität (RM)



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

- Aufgabensystem:
  - periodische Tasks
    - $T_p = (2.5, 0.5)$
    - $T_1 = (3, 1)$
    - $T_2 = (10, 4)$
    - RM
  - aperiodischer Job
    - $A \mapsto (0.4, [0.1, \infty])$



1 Zusteller  $T_p$  hat kürzeste Periode  $\leadsto$  höchste Priorität (RM)

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

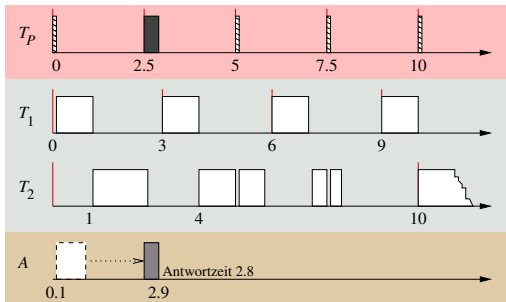
$\rightarrow$  Das Budget von  $T_p$  **verfällt**



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

- Aufgabensystem:
  - periodische Tasks
    - $T_p = (2.5, 0.5)$
    - $T_1 = (3, 1)$
    - $T_2 = (10, 4)$
    - RM
  - aperiodischer Job
    - $A \mapsto (0.4, [0.1, \infty])$



1 Zusteller  $T_p$  hat kürzeste Periode  $\leadsto$  höchste Priorität (RM)

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

→ Das Budget von  $T_p$  **verfällt**

3 Auslösezeitpunkt von  $A$  ist  $t_{0,1}$

■ Kurz nach dem Abfragezeitpunkt  $t_0$

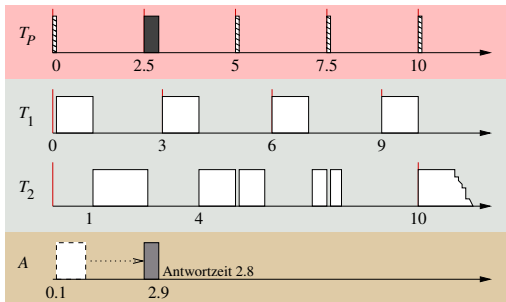
→ Ausführung von  $A$  erfolgt in Abfrageperiode  $t_{2,5}$



# Abfragender Zusteller – Beispiel (1)

## Grundlegende Funktionsweise abfragender Zusteller

- **Aufgabensystem:**
  - **periodische Tasks**
    - $T_p = (2.5, 0.5)$
    - $T_1 = (3, 1)$
    - $T_2 = (10, 4)$
    - RM
  - **aperiodischer Job**
    - $A \mapsto (0.4, [0.1, \infty])$



- 1 Zusteller  $T_p$  hat kürzeste Periode  $\leadsto$  höchste Priorität (RM)
  - 2 Zu Beginn der Abfrageperioden  $t_0$  ist die AJQ leer
    - Das Budget von  $T_p$  **verfällt**
  - 3 Auslösezeitpunkt von  $A$  ist  $t_{0,1}$ 
    - Kurz nach dem Abfragezeitpunkt  $t_0$
    - Ausführung von  $A$  erfolgt in Abfrageperiode  $t_{2,5}$
- Hier kommt das Laufzeitsystem ohne Verdrängung aus
- Budget erlaubt dem Zusteller vollständige Abarbeitung von  $A$



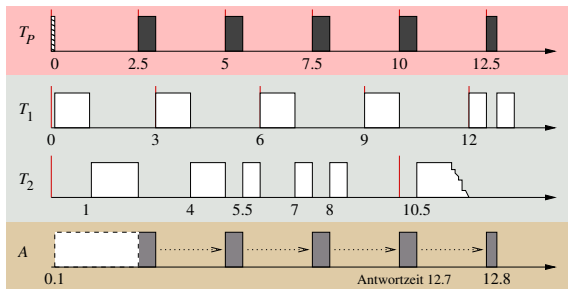
# Abfragender Zusteller – Beispiel (2)

Aufteilung eines aperiodischen Arbeitsauftrags auf mehrere Auffüllperioden

## Aufgabensystem:

$\left. \begin{array}{l} T_P \\ T_1 \\ T_2 \end{array} \right\} \text{ wie gehabt}$

$A \mapsto (2.3, [0.1, \infty])$



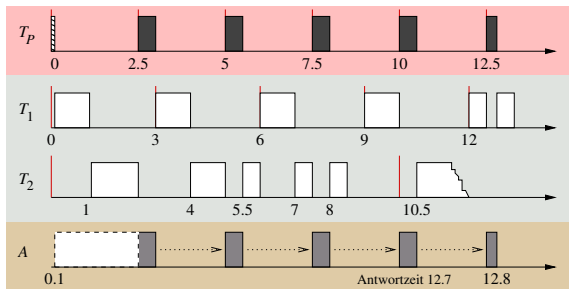
# Abfragender Zusteller – Beispiel (2)

Aufteilung eines aperiodischen Arbeitsauftrags auf mehrere Auffüllperioden

## Aufgabensystem:

$$\left. \begin{array}{l} T_P \\ T_1 \\ T_2 \end{array} \right\} \text{ wie gehabt}$$

$$A \mapsto (2.3, [0.1, \infty))$$



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

$4 \times 0.5$  Zeiteinheiten (dem Budget von  $T_P$ )

- 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,  $T_P$  wird **untätig**

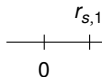
⚠ Das Laufzeitsystem muss Verdrängung unterstützen



# Abfragende Zusteller und sporadische Aufgaben

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

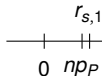
- Periodische Zusteller  $T_P = (p_P, e_P)$  behandelt sporadische Aufgabe  $T_S = (i_S, e_S, D_S)$  mit Auslösezeitpunkt  $r_{S,1}$  von  $J_{S,1}$



# Abfragende Zusteller und sporadische Aufgaben

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

- Periodische Zusteller  $T_P = (p_P, e_P)$  behandelt sporadische Aufgabe  $T_S = (i_S, e_S, D_S)$  mit Auslösezeitpunkt  $r_{S,1}$  von  $J_{S,1}$



- Schlimmstenfalls wird der **Abtastzeitpunkt**  $np_P$  verpasst:  $r_{S,1} > np_P$



# Abfragende Zusteller und sporadische Aufgaben

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

- Periodische Zusteller  $T_P = (p_P, e_P)$  behandelt sporadische Aufgabe  $T_S = (i_S, e_S, D_S)$  mit Auslösezeitpunkt  $r_{S,1}$  von  $J_{S,1}$



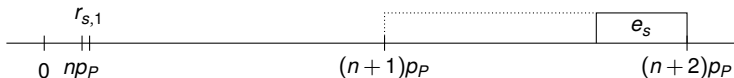
- Schlimmstenfalls wird der **Abtastzeitpunkt**  $np_P$  verpasst:  $r_{S,1} > np_P$   
→ Abarbeitung von  $J_{S,1}$  beginnt mit der zweiten Abfrageperiode



# Abfragende Zusteller und sporadische Aufgaben

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

- Periodische Zusteller  $T_P = (p_P, e_P)$  behandelt sporadische Aufgabe  $T_S = (i_S, e_S, D_S)$  mit Auslösezeitpunkt  $r_{S,1}$  von  $J_{S,1}$



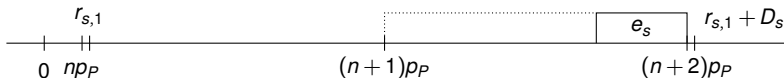
- Schlimmstenfalls wird der **Abtastzeitpunkt  $np_P$**  verpasst:  $r_{S,1} > np_P$   
→ Abarbeitung von  $J_{S,1}$  beginnt mit der zweiten Abfrageperiode
- ⚠  $T_P$  kann durch ereignisgesteuerte Einplanung verzögert werden
- Fertigstellung zur 3. Abfrageperiode  $(n+2)p_P$
- 👉 Termin von  $T_S$  muss aber gehalten werden:  $(n+2)p_P \leq r_{S,1} + D_S$



# Abfragende Zusteller und sporadische Aufgaben

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

- Periodische Zusteller  $T_P = (p_P, e_P)$  behandelt sporadische Aufgabe  $T_S = (i_S, e_S, D_S)$  mit Auslösezeitpunkt  $r_{s,1}$  von  $J_{s,1}$



- Schlimmstenfalls wird der **Abtastzeitpunkt**  $np_P$  verpasst:  $r_{s,1} > np_P$   
→ Abarbeitung von  $J_{s,1}$  beginnt mit der zweiten Abfrageperiode
- ⚠  $T_P$  kann durch ereignisgesteuerte Einplanung verzögert werden
- Fertigstellung zur 3. Abfrageperiode  $(n+2)p_P$
- 👉 Termin von  $T_S$  muss aber gehalten werden:  $(n+2)p_P \leq r_{s,1} + D_S$

- Termin  $D_S$  begrenzt die Auffüllperiode  $p_P$ :  $p_P \leq D_S/2$

→  $D_S \leq i_S$ , 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!





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
  - Zu späte Aufträge werden frühestens in der nächsten Abfrageperiode behandelt
  - Vergleiche Beispiel Folie 24
  - 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

- Zu späte Aufträge werden frühestens in der nächsten Abfrageperiode behandelt
- Vergleiche Beispiel Folie 24
- Ansammlung in der Warteschlange, der Abfrager wird zurückgestellt



**Restbudget** eines Abfragers müsste bewahrt werden

- Dazu mehr in der nächsten Woche 😊



# 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**

<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

<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
  - Zeitparameter nicht-periodischer Aufgaben
  - 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/10)
- ☞ Fertigstellung von Arbeitsaufträgen **vor ihrem Termin** unnötig
  - Kurz vor oder genau zum Termin ausreichend



# Slack-Stealing

Die Abfertigung nicht-periodischer Jobs durch Nutzung der Schlupfzeit optimieren

- Echtzeitbetrieb bedeutet **Rechtzeitigkeit** (vgl. Folie II/10)
- ☞ 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 den 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

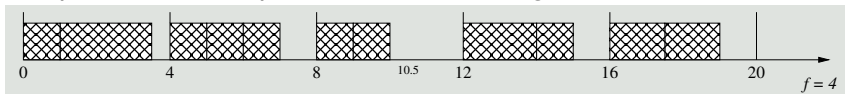
- **Echtzeitbetrieb** bedeutet **Rechtzeitigkeit** (vgl. Folie II/10)
- ☞ Fertigstellung von Arbeitsaufträgen **vor ihrem Termin** unnötig
  - Kurz vor oder genau zum Termin ausreichend
- Verschiebung periodischer Aufträge um ihre Schlupfzeit
  - Ausführung nicht-periodischer Aufträge in den entstehenden Lücken
  - **Schlupfdieb** (engl. *slack stealing*)
- ⚠ Termine periodischer Aufträge dürfen nicht gefährdet werden
  - Nach dem aufbrauchen den 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 Schrupfbereiche im ersten große Durchlauf



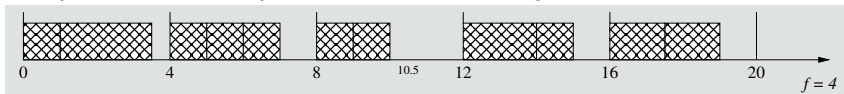
- Schraffierte Bereiche  $\mapsto$  statisch eingeplante periodische Aufträge



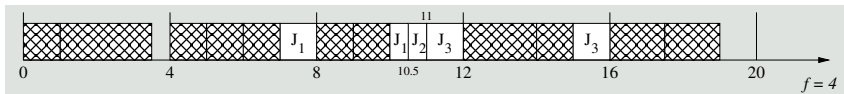
# Ausgangspunkt: Taktsteuerung

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

- Beispiel: Fünf Schrupfbereiche im ersten große Durchlauf



- Schraffierte Bereiche  $\mapsto$  statisch eingeplante periodische Aufträge
- Aperiod. Jobs  $J_1 \mapsto 1.5(4, \infty]$ ,  $J_2 \mapsto 0.5(9.5, \infty]$ ,  $J_3 \mapsto 2(10.5, \infty]$



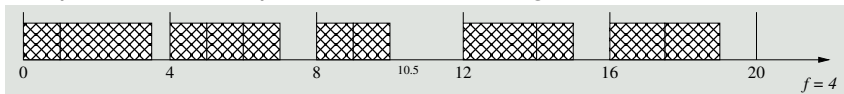
- Ausführungszeiten 1.5, 0.5 und 2
- zulässige Ausführungsintervalle (*earliest, latest*)
- $\infty$  meint: der Job hat keinen, einen weichen oder festen Termin



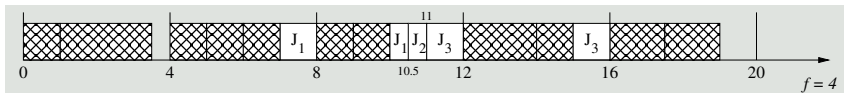
# Ausgangspunkt: Taktsteuerung

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

- Beispiel: Fünf Schrupfbereiche im ersten große Durchlauf



- Schraffierte Bereiche  $\mapsto$  statisch eingeplante periodische Aufträge
- Aperiod. Jobs  $J_1 \mapsto 1.5(4, \infty]$ ,  $J_2 \mapsto 0.5(9.5, \infty]$ ,  $J_3 \mapsto 2(10.5, \infty]$



- Ausführungszeiten 1.5, 0.5 und 2
- zulässige Ausführungsintervalle (*earliest, latest*)
- $\infty$  meint: der Job hat keinen, einen weichen oder festen Termin
- Mittlere Antwortzeit:  $((10.5 - 4) + (11 - 9.5) + (16 - 10.5))/3 = 4.5$

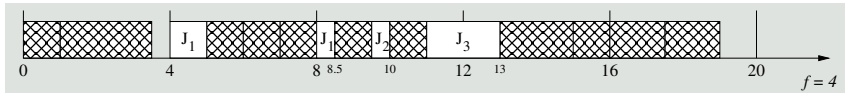


- Schlupf in Rahmen  $k$  ist die **Zeitspanne**  $f - e_p$ , mit  $e_p$  Zeiteinheiten für periodische Aufträge in  $k$
- ☞ Periodischen Aufträgen **Zeitpuffer am Rahmenende entziehen**
  - Berechnung des Schlupfes geschieht einmal vor der Laufzeit und hängt nur vom aktuellen Rahmen ab
  - Periodische 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. V-1/31):

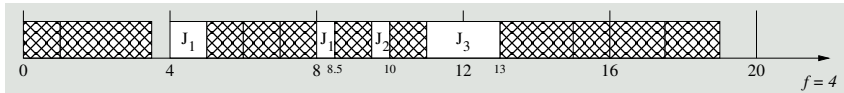


- $J_1$  wird sofort eingelastet, muss jedoch verdrängt werden
- $J_2$  wird ebenso behandelt, kann aber komplett durchlaufen
- $J_3$  wird verzögert bis der laufende periodische Auftrag fertig ist



# 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. V-1/31):



- $J_1$  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

---

<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** [3, 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** [3, S. 233 ff.]

- EDF mit **dynamischen Prioritäten**
  - **Statische vorberechnete Schlupfzeiten**
    - Berücksichtigung der **kompletten Hyperperiode**
  - **Dynamischen Berechnung**
    - 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** [3, S. 233 ff.]

- EDF mit **dynamischen Prioritäten**
  - Statische vorberechnete Schlupfzeiten
    - ↳ Berücksichtigung der **kompletten Hyperperiode**
  - Dynamischen Berechnung
    - ↳ 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
  - Zeitparameter nicht-periodischer Aufgaben
  - 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 gut Antwortzeiten, anfällig für **Überlast**
- **Gefährdet statische Garantien**  $\rightsquigarrow$  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 gut Antwortzeiten, anfällig für **Überlast**
- **Gefährdet statische Garantien**  $\leadsto$  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 gut Antwortzeiten, 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/apperiodische Aufgaben)
- **Mischbetrieb** ist eine Herausforderung

**Unterbrecherbetrieb** bevorzugt nicht-periodische Aufgaben

- Sehr gut Antwortzeiten, 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 vorranggesteuerten Systemen



- [1] AUTOSAR:  
Specification of Operating System (Version 4.0.0) / Automotive Open System  
Architecture GbR.  
2009. –  
Forschungsbericht
- [2] 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
- [3] Liu, J. W. S.:  
*Real-Time Systems.*  
Englewood Cliffs, NJ, USA : Prentice Hall PTR, 2000. –  
ISBN 0–13–099651–3
- [4] 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>



- [5] 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
- [6] Zühlsdorf, R. :  
*Protokoll des Funkverkehrs bei der ersten Landung auf dem Mond*.  
<http://members.fortunecity.de/rogerzuehlsdorf/Ap11d.htm>, 1999

