Designmigrationsstrategien von FlexRay nach Time-Triggered Ethernet

Hermand Dieumo Kenfack

Department Informatik
Hochschule für Angewandte Wissenschaften Hamburg

26. April 2012

Agenda

1. Einführung
2. Architektur und Konzept
3. Hintergrund
4. Migrationsstrategien
5. Zusammenfassung und Ausblick

Agenda

1. **Einführung**
   - Situation und Motivation
   - Ziel der Arbeit
2. **Architektur und Konzept**
3. **Hintergrund**
   - FlexRay
   - TTEthernet
   - Unterschied zwischen FlexRay und TTEthernet
   - Modellierung mit Task-Graphen
4. **Migrationsstrategien**
   - Migration des statischen Segments
   - Migration des dynamischen Segments
5. **Zusammenfassung und Ausblick**

Situation

- Automobil-Netzwerk, komplexes verteiltes System
  - Immer mehr E/E-Systeme (> 100)
  - Kommunikation über mehr als 2500 Signale

Anforderungen an einem Automobil-Netzwerk-Protokoll

- Determinismus
- Fehlertoleranz
- Hohe Bandbreite
- Event- und Time-Triggered Kommunikation

- Herkömmliche Protokolle (CAN) sind diese Anforderungen nicht gewachsen

Neue Ansätze werden untersucht

Zwei vielversprechende Protokolle

1. FlexRay
2. Time-Triggered Ethernet (TTEthernet)

Eigenschaften der beiden Protokolle

1. Deterministische Datenübertragung
2. Fehlertoleranz
3. Event- und Time-Triggered Kommunikation

TTEthernet zusätzliche Vorteile

1. Basiert auf dem Standard-Switch-Ethernet
2. Ethernet über 30 Jahre auf dem Mark
3. Mehr fachkundige Leute
4. Vorhandene Entwicklungs- und Diagnosewerkzeuge
5. Preiswerter
6. Deutlich höher Bandbreite $(100/1000 \frac{Mbit}{s})$

$\Rightarrow$ Migration von FlexRay nach TTEthernet lohnenswert

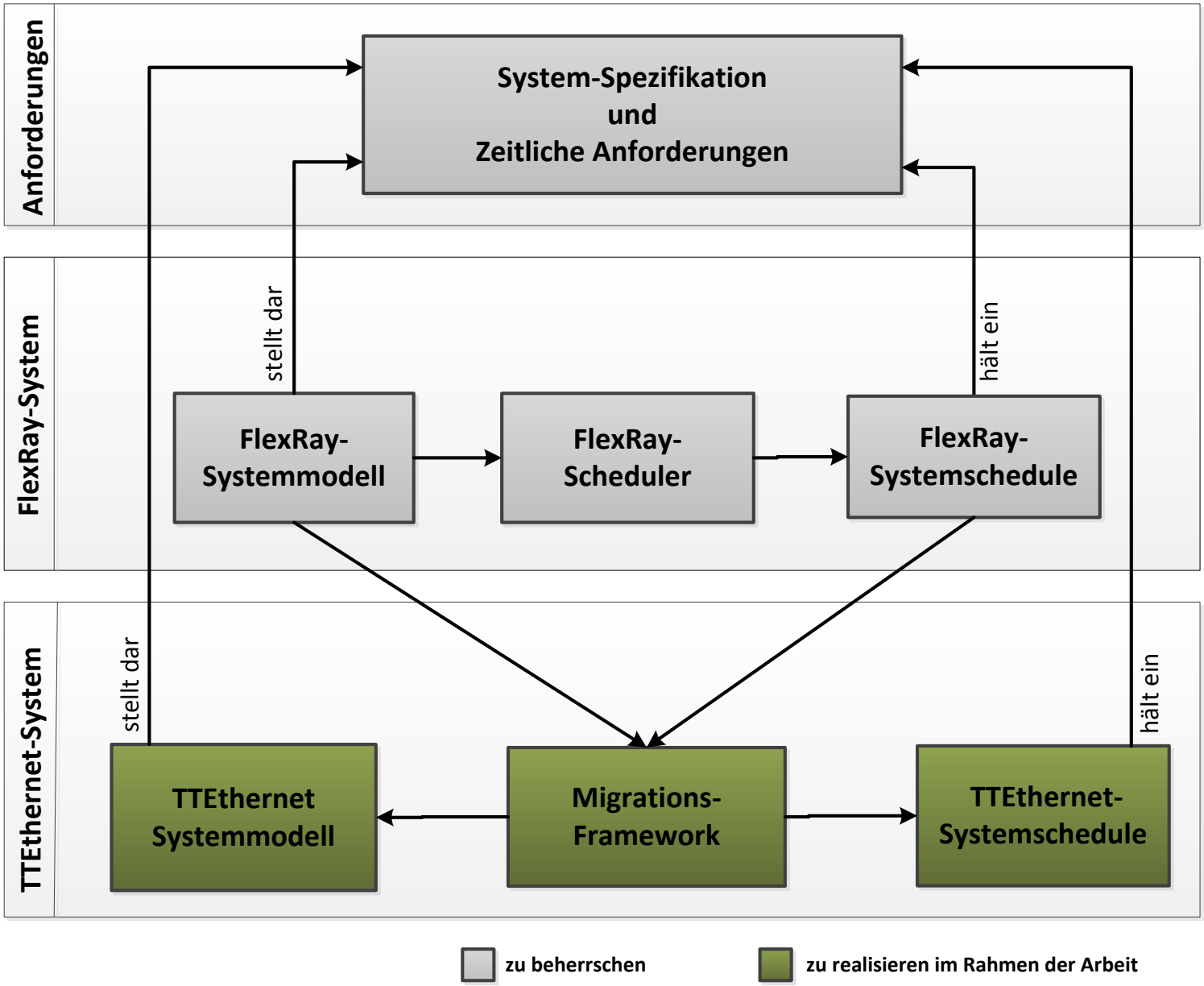
Ziel der Arbeit

- Entwicklung eines analytischen Frameworks zur Migration von FlexRay-Systemmodellen nach TTEthernet-Systemmodellen
- Fokus auf das Kommunikationsmodell
- Erarbeitung folgende Punkte
  1. Festlegung eines Frameworks zur Modellierung bzw. Beschreibung von FlexRay- und TTEthernet-Systemen
  2. Entwicklung der Migrationsstrategien
     - Statisches Segment
     - Dynamisches Segment
  3. Analyse und Validierung der Migrationsstrategien

Agenda

1. Einführung
   - Situation und Motivation
   - Ziel der Arbeit
2. **Architektur und Konzept**
3. Hintergrund
   - FlexRay
   - TTEthernet
   - Unterschied zwischen FlexRay und TTEthernet
   - Modellierung mit Task-Graphen
4. Migrationsstrategien
   - Migration des statischen Segments
   - Migration des dynamischen Segments
5. Zusammenfassung und Ausblick

Komponenten der Migration



Agenda

1 Einführung
- Situation und Motivation
- Ziel der Arbeit

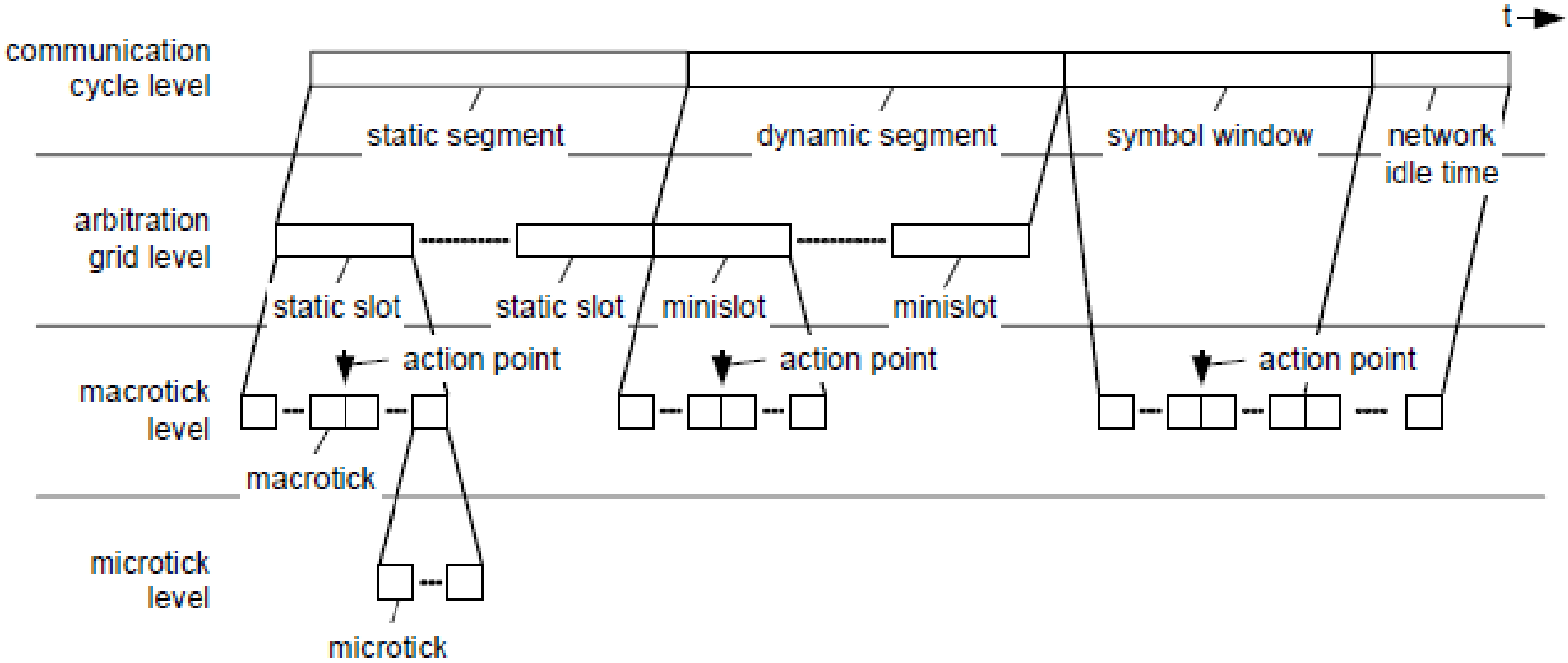
2 Architektur und Konzept

3 **Hintergrund**
- FlexRay
- TTEthernet
- Unterschied zwischen FlexRay und TTEthernet
- Modellierung mit Task-Graphen

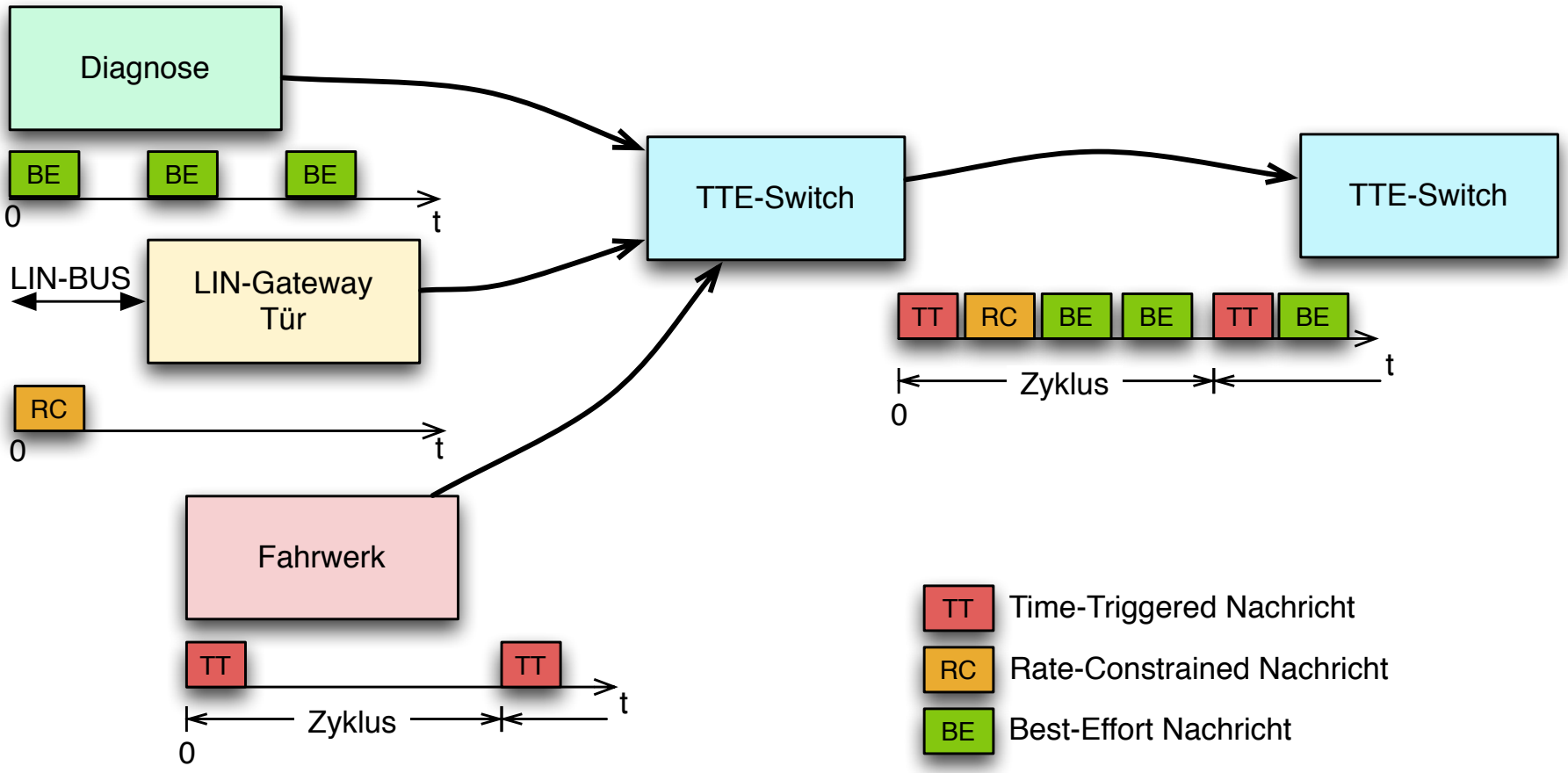
4 Migrationsstrategien
- Migration des statischen Segments
- Migration des dynamischen Segments

5 Zusammenfassung und Ausblick

FlexRay Kommunikationszyklus



TTEthernet Kommunikationszyklus



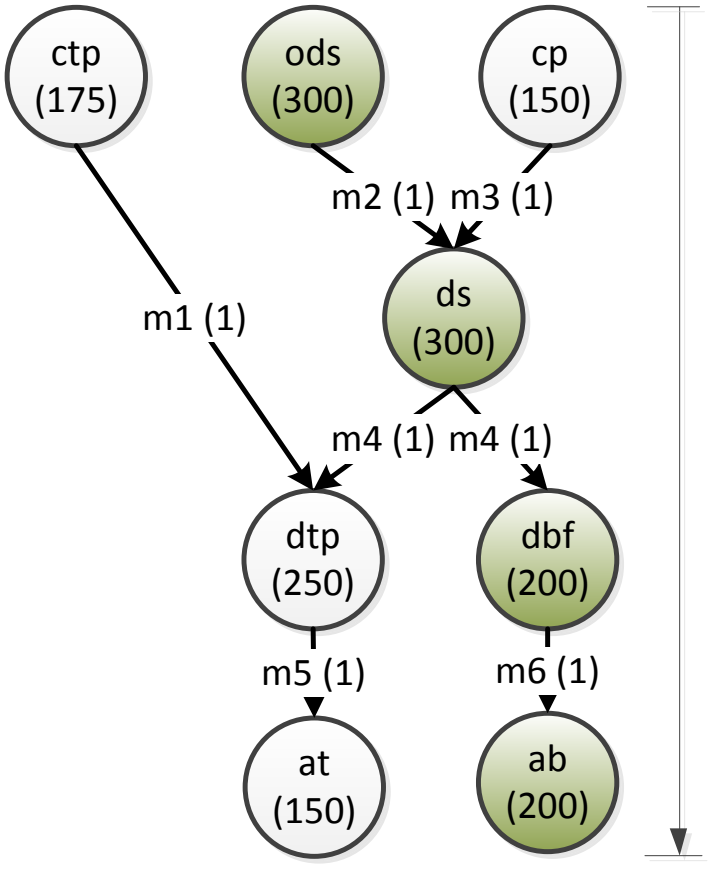
Unterschied zwischen FlexRay und TTEthernet

Eigenschaft	**FlexRay**	**TTEthernet**
Time-Triggered-Slot	Statische Länge	Dynamische Länge
Event-Triggered Kommunikation	Minisloting	RC und BE
Max. Zyklen	64	1
Payload-Bereich	$[1Byte - 254Byte]$	$[46Byte - 1500Byte]$
Topologie	Bus, Stern und Mix	Switch
Bandbreite	$10/20\frac{Mbit}{s}$	$100/1000\frac{Mbit}{s}$

Task-Graph (Funktionsmodell)

$$TG = (\mathbf{T}, \mathbf{E}, \mathbf{M}, s_m, t_W, c)$$

- Beispiel: Task-Graph einer ACC-Anwendung

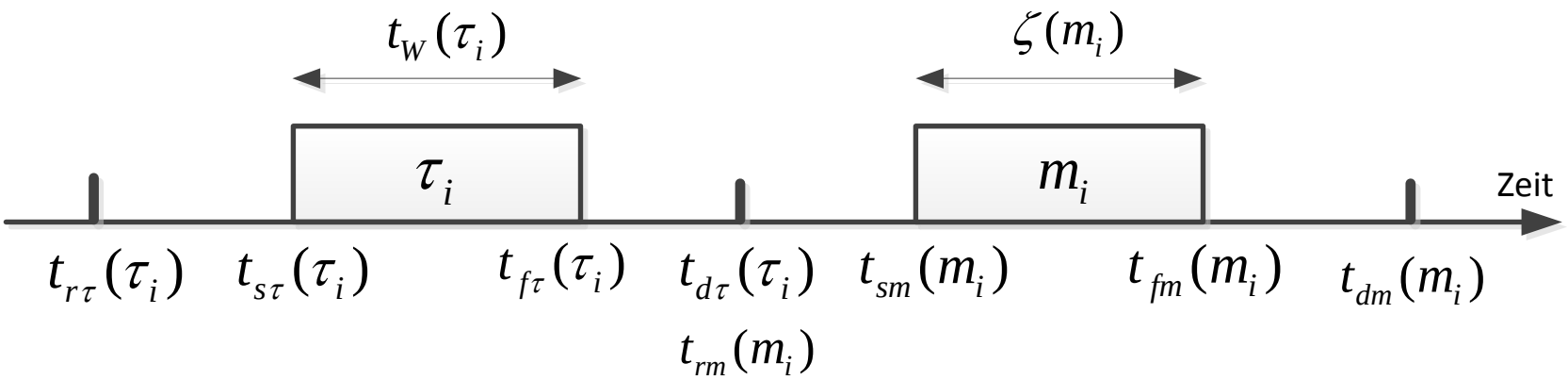


$$t_{sg}(TG_{ACC}) = 0us$$

ctp (Current throttle position)
ods (Object distance and speed)
cp (Current speed)
ds (Desired speed)
dtp (Desired throttle position)
dbf (Desired Braking force)
at (Actuate throttle)
ab (Actuate brakes)

$$t_{dg}(TG_{ACC}) = 3000us$$

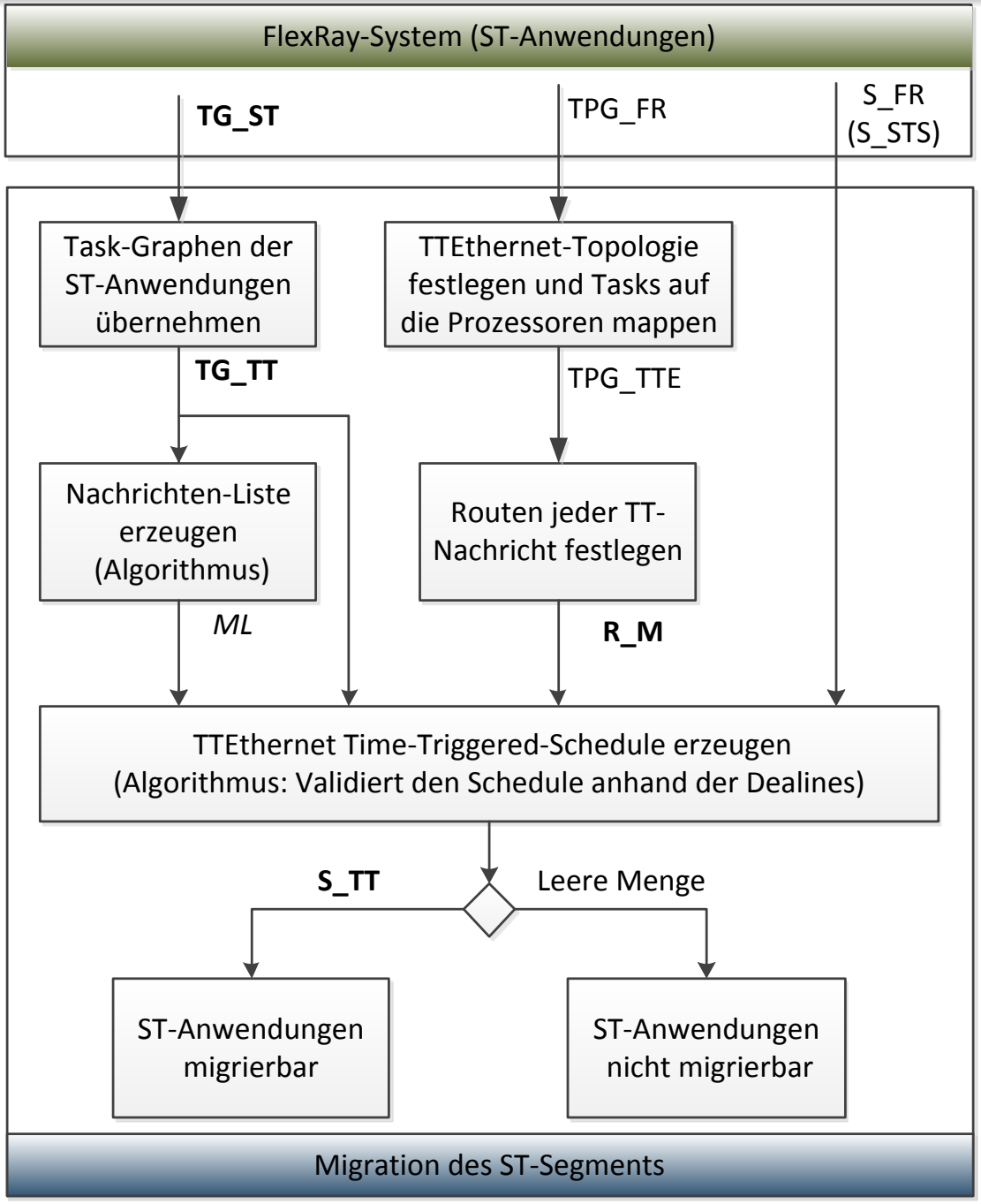
Zeitliche Eigenschaften eines Tasks und einer Nachricht



Agenda

1. Einführung
   - Situation und Motivation
   - Ziel der Arbeit
2. Architektur und Konzept
3. Hintergrund
   - FlexRay
   - TTEthernet
   - Unterschied zwischen FlexRay und TTEthernet
   - Modellierung mit Task-Graphen
4. **Migrationsstrategien**
   - Migration des statischen Segments
   - Migration des dynamischen Segments
5. Zusammenfassung und Ausblick

Prozess Migration des statischen Segments



Schedule einer TT-Nachricht

$$\mathcal{S}_{TT}(m) = (\mathbf{conf}(m), t_{sm}(m), t_{fm}(m), t_{pm}(m), N_{rep}(m))$$

TT-Nachrichten Scheduling(1)

1: **procedure** TTSchldErzeugen

Require: $\mathbf{TG_{TT}} \subseteq \mathbf{TG}m,\ TPG,\ \mathbf{R_M},\ \mathcal{ML},\ \mathcal{S}_{FR},\ \mathcal{S}_{\mathbf{TT}}$

Ensure: $\mathcal{S}_{\mathbf{TT}}$

2: $T_{CC} \leftarrow \mathcal{S}_{FR}.T_{CC} \times \mathcal{S}_{FR}.N_{CC}$

3: Erstelle eine leere Menge $\mathcal{S}_{\mathbf{TT}}$ der TT-Nachrichten-Schedules

4: **while** $\mathcal{ML} \neq \emptyset$ **do**

5: sei m_i die erste Nachricht aus $\mathcal{ML}$

6: $t_{pm}(m_i) \leftarrow t_{dg}(TG_{m_i}), TG_{m_i} \in \mathbf{TG_{TT}} \wedge m_i \in TG_{m_i}$

7: $N_{rep}(m_i) \leftarrow \left\lfloor \frac{T_{CC}}{t_{pm}(m_i)} \right\rfloor$

8: Ermittele die Konflikt-Menge $\mathbf{conf}(m_i)$ der Nachricht m_i

9: $\Delta_{Slot} \leftarrow \zeta_{SW}(m_i) + (2 \times T_{cp})$ * *Slot-Länge berechnen* *\

10: **if** Existiert ein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{rm}(m_i) \leq t_a) \wedge (t_b \leq t_{dm}(m_i)) \wedge (\forall m_k \in \mathbf{conf}(m_i), (t_b \leq t_{sm}(m_k)) \vee (t_a \geq t_{fm}(m_k)))$ **then**

11: $t_{sm}(m_i) \leftarrow t_a$

12: $t_{fm}(m_i) \leftarrow t_b$

13: **else**

14: **return** $\emptyset$ * *Schedule nicht ausführbar* *\

15: **end if**

TT-Nachrichten Scheduling(1)

1: **procedure** TTSchldErzeugen

Require: $\mathbf{TG_{TT}} \subseteq \mathbf{TG}m,\ TPG,\ \mathbf{R_M},\ \mathcal{ML},\ \mathcal{S}_{FR},\ \mathcal{S}_{\mathbf{TT}}$

Ensure: $\mathcal{S}_{\mathbf{TT}}$

2: $T_{CC} \leftarrow \mathcal{S}_{FR}.T_{CC} \times \mathcal{S}_{FR}.N_{CC}$

3: Erstelle eine leere Menge $\mathcal{S}_{\mathbf{TT}}$ der TT-Nachrichten-Schedules

4: **while** $\mathcal{ML} \neq \emptyset$ **do**

5: sei m_i die erste Nachricht aus $\mathcal{ML}$

6: $t_{pm}(m_i) \leftarrow t_{dg}(TG_{m_i}), TG_{m_i} \in \mathbf{TG_{TT}} \wedge m_i \in TG_{m_i}$

7: $N_{rep}(m_i) \leftarrow \left\lfloor \frac{T_{CC}}{t_{pm}(m_i)} \right\rfloor$

8: Ermittele die Konflikt-Menge $\mathbf{conf}(m_i)$ der Nachricht m_i

9: $\Delta_{Slot} \leftarrow \zeta_{SW}(m_i) + (2 \times T_{cp})$ * *Slot-Länge berechnen* *\

10: **if** Existiert ein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{rm}(m_i) \leq t_a) \wedge (t_b \leq t_{dm}(m_i)) \wedge (\forall m_k \in \mathbf{conf}(m_i), (t_b \leq t_{sm}(m_k)) \vee (t_a \geq t_{fm}(m_k)))$ **then**

11: $t_{sm}(m_i) \leftarrow t_a$

12: $t_{fm}(m_i) \leftarrow t_b$

13: **else**

14: **return** $\emptyset$ * *Schedule nicht ausführbar* *\

15: **end if**

TT-Nachrichten Scheduling(1)

1: **procedure** TTSchldErzeugen
Require: $\mathbf{TG_{TT}} \subseteq \mathbf{TG}m,\ TPG,\ \mathbf{R_M},\ \mathcal{ML},\ \mathcal{S}_{FR},\ \mathcal{S}_{\mathbf{TT}}$
Ensure: $\mathcal{S}_{\mathbf{TT}}$
2: $T_{CC} \leftarrow \mathcal{S}_{FR}.T_{CC} \times \mathcal{S}_{FR}.N_{CC}$
3: Erstelle eine leere Menge $\mathcal{S}_{\mathbf{TT}}$ der TT-Nachrichten-Schedules
4: **while** $\mathcal{ML} \neq \emptyset$ **do**
5: sei m_i die erste Nachricht aus $\mathcal{ML}$
6: $t_{pm}(m_i) \leftarrow t_{dg}(TG_{m_i}), TG_{m_i} \in \mathbf{TG_{TT}} \wedge m_i \in TG_{m_i}$
7: $N_{rep}(m_i) \leftarrow \left\lfloor \frac{T_{CC}}{t_{pm}(m_i)} \right\rfloor$
8: Ermittele die Konflikt-Menge **conf**(m_i) der Nachricht m_i
9: $\Delta_{Slot} \leftarrow \zeta_{SW}(m_i) + (2 \times T_{cp})$ * *Slot-Länge berechnen* *\
10: **if** Existiert ein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{rm}(m_i) \leq t_a) \wedge (t_b \leq t_{dm}(m_i)) \wedge (\forall m_k \in \mathbf{conf}(m_i), (t_b \leq t_{sm}(m_k)) \vee (t_a \geq t_{fm}(m_k)))$ **then**
11: $t_{sm}(m_i) \leftarrow t_a$
12: $t_{fm}(m_i) \leftarrow t_b$
13: **else**
14: **return** $\emptyset$ * *Schedule nicht ausführbar* *\
15: **end if**

TT-Nachrichten Scheduling(1)

1: **procedure** TTSchldErzeugen

Require: $\mathbf{TG_{TT}} \subseteq \mathbf{TG}m,\ TPG,\ \mathbf{R_M},\ \mathcal{ML},\ \mathcal{S}_{FR},\ \mathcal{S}_{\mathbf{TT}}$

Ensure: $\mathcal{S}_{\mathbf{TT}}$

2: $T_{CC} \leftarrow \mathcal{S}_{FR}.T_{CC} \times \mathcal{S}_{FR}.N_{CC}$

3: Erstelle eine leere Menge $\mathcal{S}_{\mathbf{TT}}$ der TT-Nachrichten-Schedules

4: **while** $\mathcal{ML} \neq \emptyset$ **do**

5: sei m_i die erste Nachricht aus $\mathcal{ML}$

6: $t_{pm}(m_i) \leftarrow t_{dg}(TG_{m_i}), TG_{m_i} \in \mathbf{TG_{TT}} \wedge m_i \in TG_{m_i}$

7: $N_{rep}(m_i) \leftarrow \left\lfloor \frac{T_{CC}}{t_{pm}(m_i)} \right\rfloor$

8: Ermittele die Konflikt-Menge $\mathbf{conf}(m_i)$ der Nachricht m_i

9: $\Delta_{Slot} \leftarrow \zeta_{SW}(m_i) + (2 \times T_{cp})$ * *Slot-Länge berechnen* *\

10: **if** Existiert ein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{rm}(m_i) \leq t_a) \wedge (t_b \leq t_{dm}(m_i)) \wedge (\forall m_k \in \mathbf{conf}(m_i), (t_b \leq t_{sm}(m_k)) \vee (t_a \geq t_{fm}(m_k)))$ **then**

11: $t_{sm}(m_i) \leftarrow t_a$

12: $t_{fm}(m_i) \leftarrow t_b$

13: **else**

14: **return** $\emptyset$ * *Schedule nicht ausführbar* *\

15: **end if**

TT-Nachrichten Scheduling(1)

1: **procedure** TTSchldErzeugen

Require: $\mathbf{TG_{TT}} \subseteq \mathbf{TG}m,\ TPG,\ \mathbf{R_M},\ \mathcal{ML},\ \mathcal{S}_{FR},\ \mathcal{S}_{\mathbf{TT}}$

Ensure: $\mathcal{S}_{\mathbf{TT}}$

2: $T_{CC} \leftarrow \mathcal{S}_{FR}.T_{CC} \times \mathcal{S}_{FR}.N_{CC}$

3: Erstelle eine leere Menge $\mathcal{S}_{\mathbf{TT}}$ der TT-Nachrichten-Schedules

4: **while** $\mathcal{ML} \neq \emptyset$ **do**

5: sei m_i die erste Nachricht aus $\mathcal{ML}$

6: $t_{pm}(m_i) \leftarrow t_{dg}(TG_{m_i}), TG_{m_i} \in \mathbf{TG_{TT}} \wedge m_i \in TG_{m_i}$

7: $N_{rep}(m_i) \leftarrow \left\lfloor \frac{T_{CC}}{t_{pm}(m_i)} \right\rfloor$

8: Ermittele die Konflikt-Menge $\mathbf{conf}(m_i)$ der Nachricht m_i

9: $\Delta_{Slot} \leftarrow \zeta_{SW}(m_i) + (2 \times T_{cp})$ * *Slot-Länge berechnen* *\

10: **if** Existiert ein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{rm}(m_i) \leq t_a) \wedge (t_b \leq t_{dm}(m_i)) \wedge (\forall m_k \in \mathbf{conf}(m_i), (t_b \leq t_{sm}(m_k)) \vee (t_a \geq t_{fm}(m_k)))$ **then**

11: $t_{sm}(m_i) \leftarrow t_a$

12: $t_{fm}(m_i) \leftarrow t_b$

13: **else**

14: **return** $\emptyset$ * *Schedule nicht ausführbar* *\

15: **end if**

TT-Nachrichten Scheduling(2)

16: * *Die Intervalle für die Wiederholungen sollen auch frei sein.* *\
17: **if** $N_{rep}(m_i) > 0$ **then**
18: $k \leftarrow 1$
19: **while** $k \leq N_{rep}(m_i)$ **do**
20: $T_\theta = t_{pm}(m_i) \times k$
21: **if** Existiert kein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{sm}(m_i) + T_\theta \leq t_a) \wedge (t_b \leq t_{fm}(m_i) + T_\theta)$ **then**
22: **return** $\emptyset$ * *Schedule nicht ausführbar* *\
23: **end if**
24: $k \leftarrow k + 1$
25: **end while**
26: **end if**
27: $\mathcal{S}_{\mathbf{TT}} \leftarrow \mathcal{S}_{\mathbf{TT}} \cup \{\mathbf{conf}(m_i), t_{sm}(m_i), t_{fm}(m_i), t_{pm}(m_i), N_{rep}(m_i)\}$
28: $\mathcal{ML} \leftarrow \mathcal{ML} - m_i$ * *Lösche* m_i *von* $\mathcal{ML}$ *\
29: **end while**
30: **return** $\mathcal{S}_{\mathbf{TT}}$
31: **end procedure**

TT-Nachrichten Scheduling(2)

* *Die Intervalle für die Wiederholungen sollen auch frei sein.* *\
if $N_{rep}(m_i) > 0$ **then**
  $k \leftarrow 1$
  while $k \leq N_{rep}(m_i)$ **do**
    $T_\theta = t_{pm}(m_i) \times k$
    if Existiert kein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{sm}(m_i) + T_\theta \leq t_a) \wedge (t_b \leq t_{fm}(m_i) + T_\theta)$ **then**
      return $\emptyset$ * *Schedule nicht ausführbar* *\
    end if
    $k \leftarrow k + 1$
  end while
end if
$\mathcal{S}_{\mathbf{TT}} \leftarrow \mathcal{S}_{\mathbf{TT}} \cup \{\mathbf{conf}(m_i), t_{sm}(m_i), t_{fm}(m_i), t_{pm}(m_i), N_{rep}(m_i)\}$
$\mathcal{ML} \leftarrow \mathcal{ML} - m_i$ * *Lösche* m_i *von* $\mathcal{ML}$ *\

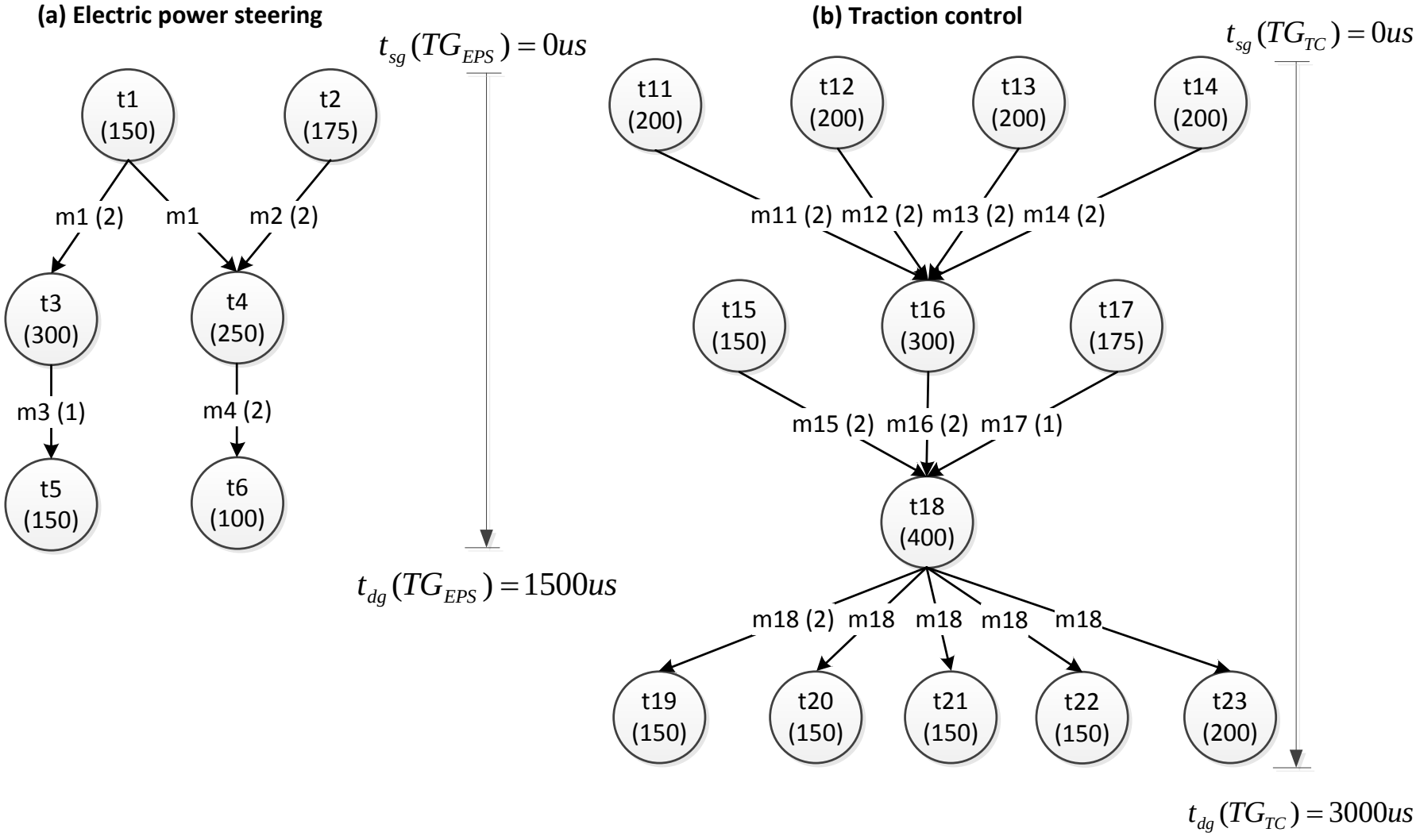
return $\mathcal{S}_{\mathbf{TT}}$

TT-Nachrichten Scheduling(2)

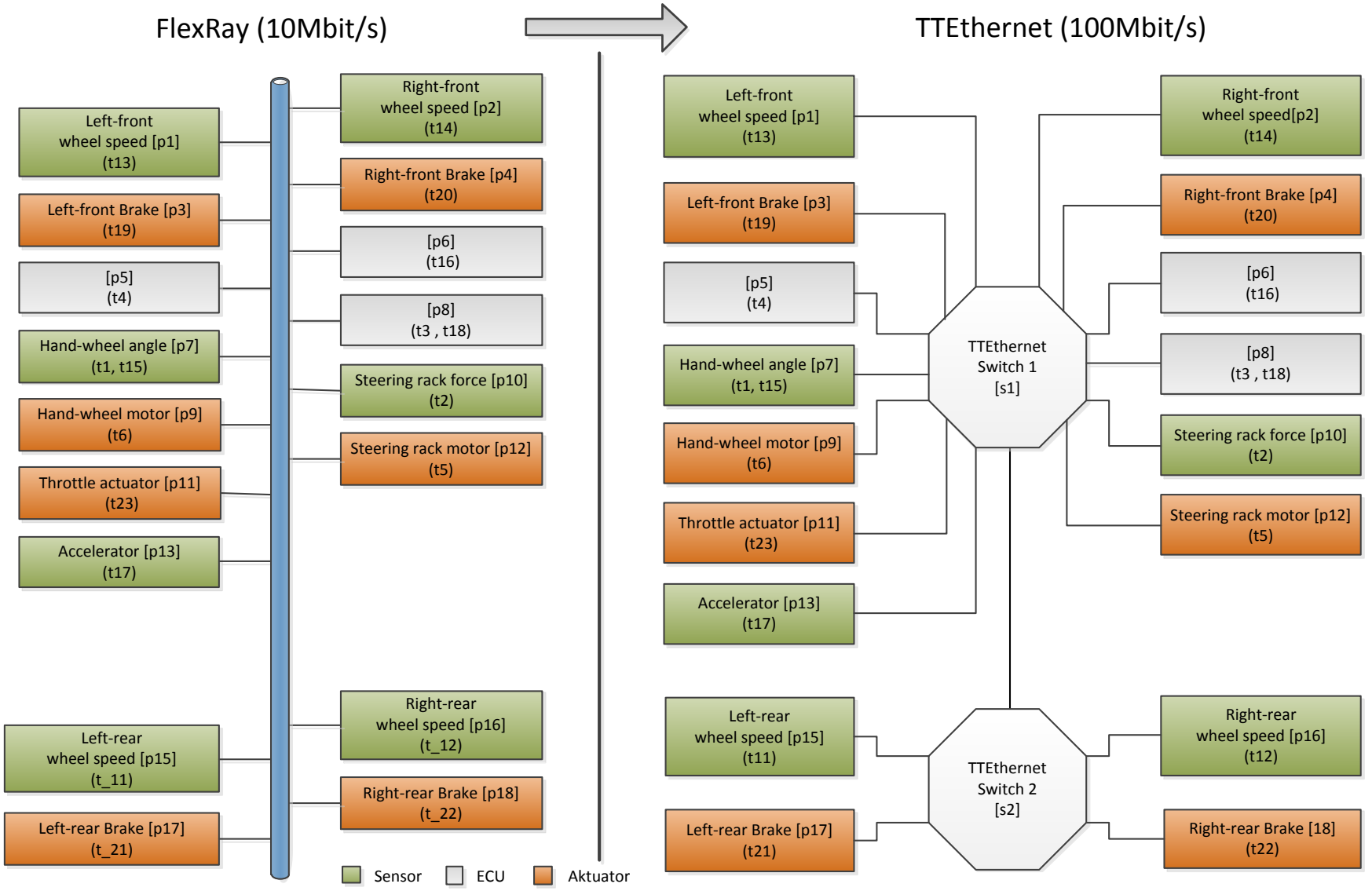
* *Die Intervalle für die Wiederholungen sollen auch frei sein.* *\
if $N_{rep}(m_i) > 0$ **then**
  $k \leftarrow 1$
  while $k \leq N_{rep}(m_i)$ **do**
    $T_\theta = t_{pm}(m_i) \times k$
    if Existiert kein Zeitslot $[t_a, t_b = t_a + \Delta_{Slot})] : (t_{sm}(m_i) + T_\theta \leq t_a) \wedge (t_b \leq t_{fm}(m_i) + T_\theta)$ **then**
      return $\emptyset$ * *Schedule nicht ausführbar* *\
    end if
    $k \leftarrow k + 1$
  end while
end if
$\mathcal{S}_{\mathbf{TT}} \leftarrow \mathcal{S}_{\mathbf{TT}} \cup \{\mathbf{conf}(m_i), t_{sm}(m_i), t_{fm}(m_i), t_{pm}(m_i), N_{rep}(m_i)\}$
$\mathcal{ML} \leftarrow \mathcal{ML} - m_i$ * *Lösche* m_i *von* $\mathcal{ML}$ *\

return $\mathcal{S}_{\mathbf{TT}}$

Fallstudie: Migration EPS- und TC-Anwendung: Task-Graphen



Fallstudie: Migration EPS- und TC-Anwendung: Topologien



Fallstudie: Migration EPS- und TC-Anwendung: Routen und Konflikt-Nachrichten

m_i	$\mathbf{R_{m_i}}$	$\mathbf{conf}(m_i)$	$\zeta_{SW}(m_i)$
m_1	$\{\{l_{[p_7,s_1]}, l_{[s_1,p_5]}\}, \{l_{[p_7,s_1]}, l_{[s_1,p_8]}\}\}$	$\{m_2, m_{15}\}$	15
m_2	$\{l_{[p_{10},s_1]}, l_{[s_1,p_5]}\}$	$\{m_1\}$	15
m_{11}	$\{l_{[p_{15},s_2]}, l_{[s_2,s_1]}, l_{[s_1,p_6]}\}$	$\{m_{12}, m_{13}, m_{14}\}$	25
m_{12}	$\{l_{[p_{16},s_2]}, l_{[s_2,s_1]}, l_{[s_1,p_6]}\}$	$\{m_{11}, m_{13}, m_{14}\}$	25
m_{13}	$\{l_{[p_1,s_1]}, l_{[s_1,p_6]}\}$	$\{m_{12}, m_{11}, m_{14}\}$	15
m_{14}	$\{l_{[p_2,s_1]}, l_{[s_1,p_6]}\}$		
m_3	$\{l_{[p_8,s_1]}, l_{[s_1,p_{12}]}\}$	$\{m_{18}\}$	15
m_4	$\{l_{[p_5,s_1]}, l_{[s_1,p_9]}\}$		
m_{15}	$\{l_{[p_7,s_1]}, l_{[s_1,p_8]}\}$	$\{m_1, m_{16}, m_{17}\}$	15
m_{16}	$\{l_{[p_6,s_1]}, l_{[s_1,p_8]}\}$	$\{m_1, m_{15}, m_{17}\}$	15
m_{17}	$\{l_{[p_{13},s_1]}, l_{[s_1,p_8]}\}$	$\{m_1, m_{15}, m_{16}\}$	15
m_{18}	$\{\{l_{[p_8,s_1]}, l_{[s_1,p_3]}\}, \{l_{[p_8,s_1]}, l_{[s_1,p_4]}\}, \{l_{[p_8,s_1]}, l_{[s_1,s_2]}, l_{[s_2,p_{18}]}\}, \{l_{[p_8,s_1]}, l_{[s_1,p_{11}]}\}\}$	$\{m_3\}$	25

Fallstudie: Migration EPS- und TC-Anwendung: Anforderungen und TT-Schedules

m_i	$t_{rm}(m_i)$	$t_{dm}(m_i)$	$t_{pm}(m_i)$	$N_{rep}(m_i)$	$t_{ms}(m_i)$	$t_{mf}(m_i)$
m_1	150	600	1500	1	150(1650)	185(1685)
m_2	175	600	1500	1	185(1685)	220(1720)
m_{11}	200	833	0	0	200	245
m_{12}	200	833	0	0	245	290
m_{13}	200	833	0	0	290	325
m_{14}	200	833	0	0	200	235
m_3	900	1350	1500	1	900(2400)	935(2435)
m_4	850	1400	1500	1	850(2350)	885(2385)
m_{15}	150	1766	0	0	185	220
m_{16}	1133	1766	0	0	1133	1168
m_{17}	175	1766	0	0	220	255
m_{18}	2166	2799	0	0	2166	2211

Migration des dynamischen Segments

Zwei Strategien

1. Abbildung von DYN-Nachrichten auf TT-Nachrichten
2. Abbildung von DYN-Nachrichten auf prioritätsbasierte RC-Nachrichten
   - Validierung anhand der WCRTs der Nachrichten

- Es wird benötigt:
  - Beschreibung des Schedules einer DYN- und RC-Nachrichten
  - WCRTs von DYN-Nachrichten ($\Re_{DYN}(m)$)
  - WCRTs von RC-Nachrichten ($\Re_{RC}(m)$)
  - Nachrichten-Liste ($\mathcal{ML}$)

Migration des DYN-Segments: Strategie 2

1. RC-Schedule jede DYN-Nachricht berechnen (Algorithmus)

Schedule einer DYN-Nachricht

$$\mathcal{S}_{DYN}(m) = (mSlot(m), bCC(m), CCRep(m), ltMSlot(m))$$

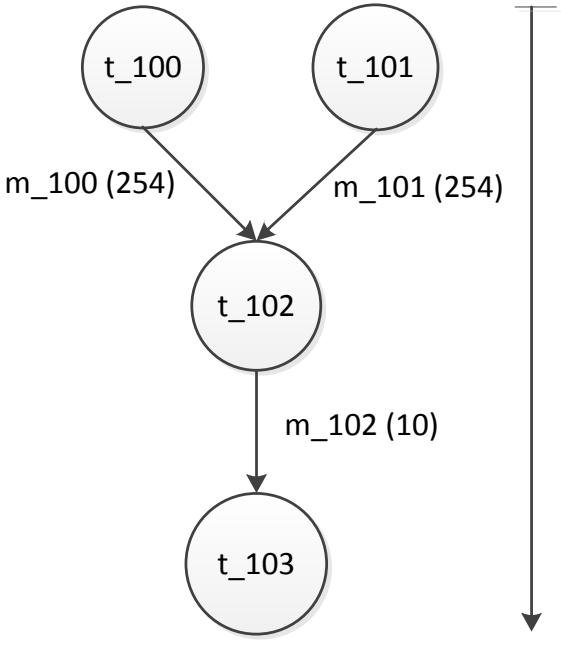
Schedule einer RC-Nachricht

$$\mathcal{S}_{RC}(m) = (CtId, T_{BAG}, D_{max})$$

2. WCRTs der DYN- und RC-Nachrichten berechnen (Gleichungen)
3. Ist die WCRT jeder RC-Nachricht kleiner als die ursprunliche DYN-Nachricht, dann sind die Anwendungen Migrierbar, sonst nicht.

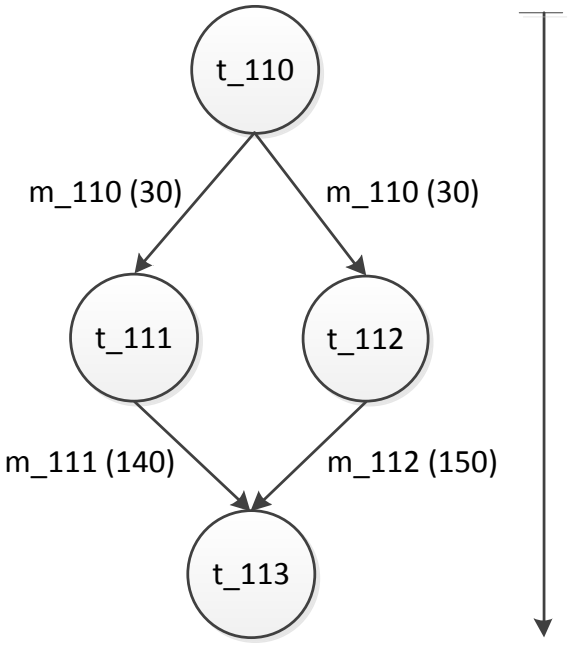
Fallstudie: Migration DYN-Anwendungen

(a) Driver Assistance (DA)



$t_{dg}(TG_{DA}) = 50ms$

(b) Diagnose (Diag)



$t_{dg}(TG_{Diag}) = 50ms$

Task-Function (DA)
t_100 (Send image)
t_101 (Send image)
t_102 (Interprete image)
t_103 (Actuator)

Task-Function (Diag)
t_110 (Request diagnose data)
t_111 (Send diagnose data)
t_112 (Send diagnose data)
t_112 (Interprete diagnose data)

Fallstudie: Migration DYN-Anwendungen

DYN-Segment-Konfiguration

T_{DYNS}	T_{MSlot}	N_{MSlot}
$600\mu s$	$6\mu s$	100

Datenmengen und Übertragungszeiten

m_i	$d(m_i)[Byte]$	$\zeta_B(m_i)[\mu s]$
m_{100}	262	262
m_{101}	262	262
m_{102}	18	18
m_{110}	38	38
m_{111}	148	148
m_{112}	158	158

DYN-Nachrichten-Schedules

m_i	$mSlot(m_i)$	$bCC(m_i)$	$CCRep(m_i)$	$ltMSlot(m_i)$
m_{100}	1	1	1	56
m_{101}	2	1	1	56
m_{102}	3	1	1	97
m_{110}	4	1	1	93
m_{111}	5	1	1	75
m_{112}	6	1	1	73

Fallstudie: Migration DYN-Anwendungen: Ergebnis erste Strategie

TT-Schedule

m_i	**conf**(m_i)	$t_{sm}(m_i)[\mu s]$	$t_{fm}(m_i)[\mu s]$	$N_{rep}(m_i)$
m_{100}	$\{m_{101}\}$	0	48	1
m_{101}	$\{m_{100}\}$	48	96	1
m_{102}	$\{m_{110}\}$	0	23	1
m_{110}	$\{m_{102}\}$	23	46	1
m_{111}	$\{m_{112}\}$	0	30	1
m_{112}	$\{m_{111}\}$	30	78	1

Fallstudie: Migration DYN-Anwendungen: Ergebnis zweite Strategie

RC-Schedule

m_i	$CtID$	$T_{BAG}[\mu s]$	D_{max}
m_{100}	100	2000	280
m_{101}	101	2000	280
m_{102}	102	2000	72
m_{110}	110	2000	72
m_{111}	111	2000	166
m_{112}	112	2000	176

Worst-Case-Response-Times

m_i	$\Re_{DYN}(m)[\mu s]$	$\Re_{RC}(m)[\mu s]$
m_{100}	2262	1136
m_{101}	2518	1136
m_{102}	2530	1035
m_{110}	> 35000	1035
m_{111}	> 35000	1045
m_{112}	> 35000	1062

Agenda

1. Einführung
   - Situation und Motivation
   - Ziel der Arbeit
2. Architektur und Konzept
3. Hintergrund
   - FlexRay
   - TTEthernet
   - Unterschied zwischen FlexRay und TTEthernet
   - Modellierung mit Task-Graphen
4. Migrationsstrategien
   - Migration des statischen Segments
   - Migration des dynamischen Segments
5. **Zusammenfassung und Ausblick**

Zusammenfassung

- Ziel: Entwicklung eines analytischen Frameworks für die Migration eines FlexRay-Systemmodells nach einem TTEthernet-Systemmodells mit dem Fokus auf dem Kommunikationsmodell
- Migration des statischen Segments
- Migration des dynamischen Segments
- Fallstudien lieferte gute Ergebnisse
- Migration von FlexRay-Anwendungen nach TTEthernet-Anwendungen möglich

Ausblick

- Solide Basis für weiterführende Arbeiten
- Tool-Chain für die entwickelten Methoden
- Migration von anderen Protokollen, wie CAN und MOST, nach TTEthernet
- TTEthernet als Backbone

Vielen Dank für die Aufmerksamkeit!