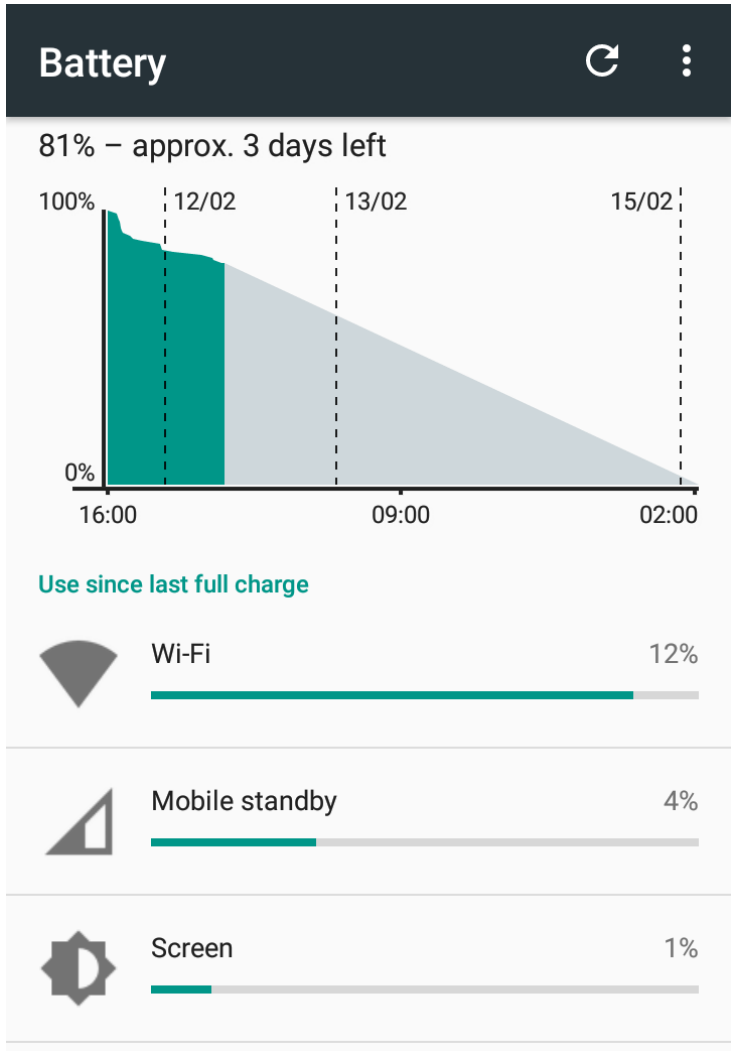
Energiemodelle für Komponenten vernetzter eingebetteter Systeme

Daniel Friesel
daniel.friesel@udo.edu

TU Dortmund, Informatik XII
Arbeitsgruppe Eingebettete Systemsoftware

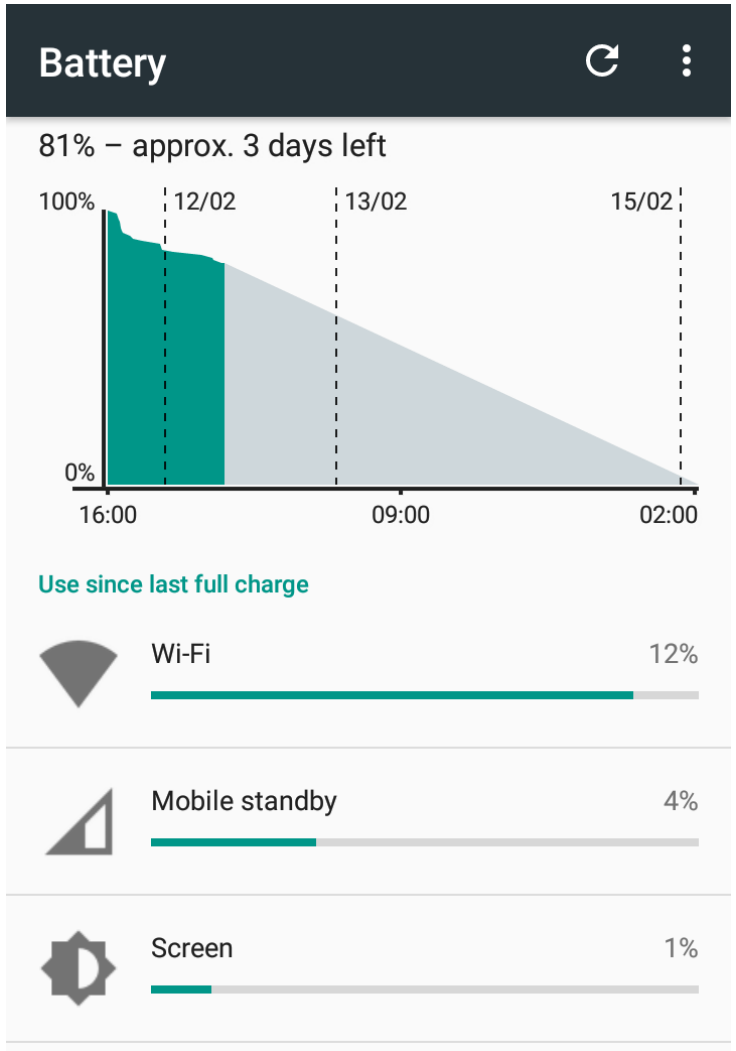
1. März 2018

Motivation



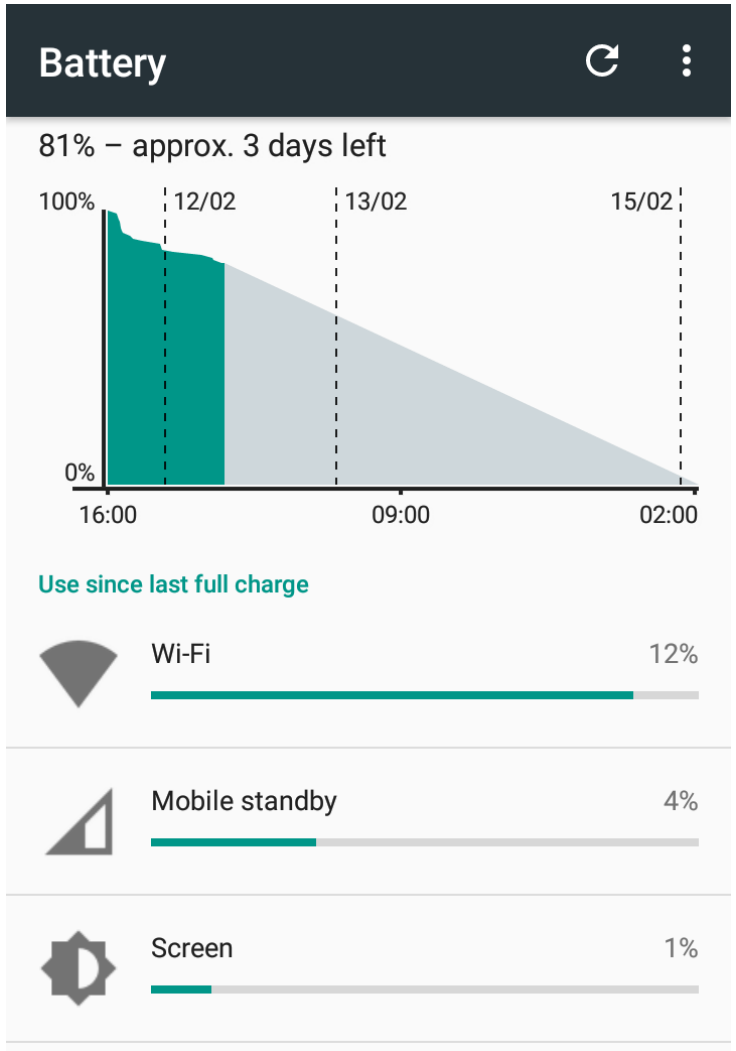
- Wie lange hält der Akku noch?
  - . . . und wieso ist er schon so leer?

Motivation



- Wie lange hält der Akku noch?
  - . . . und wieso ist er schon so leer?
- Standardlösung bei Smartphones und Laptops:
  - Ladungszähler im Akku → Ladungszustand
  - Performance Counter im System → Verbrauchsaufschlüsselung

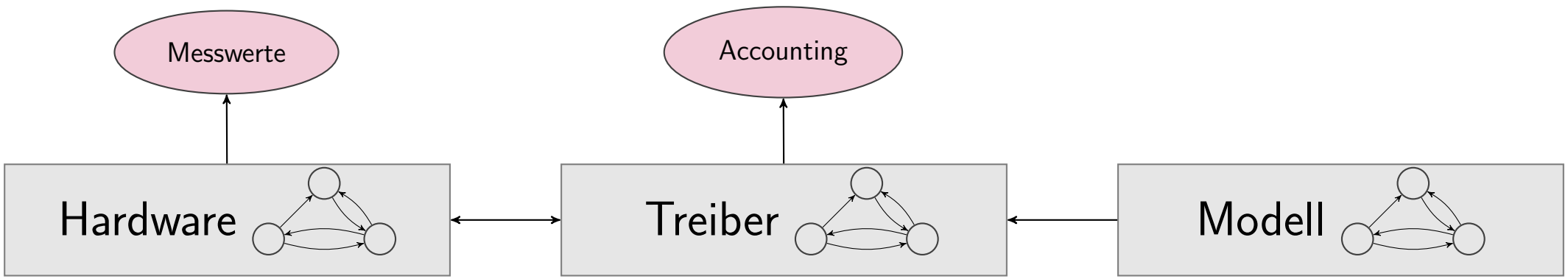
Motivation



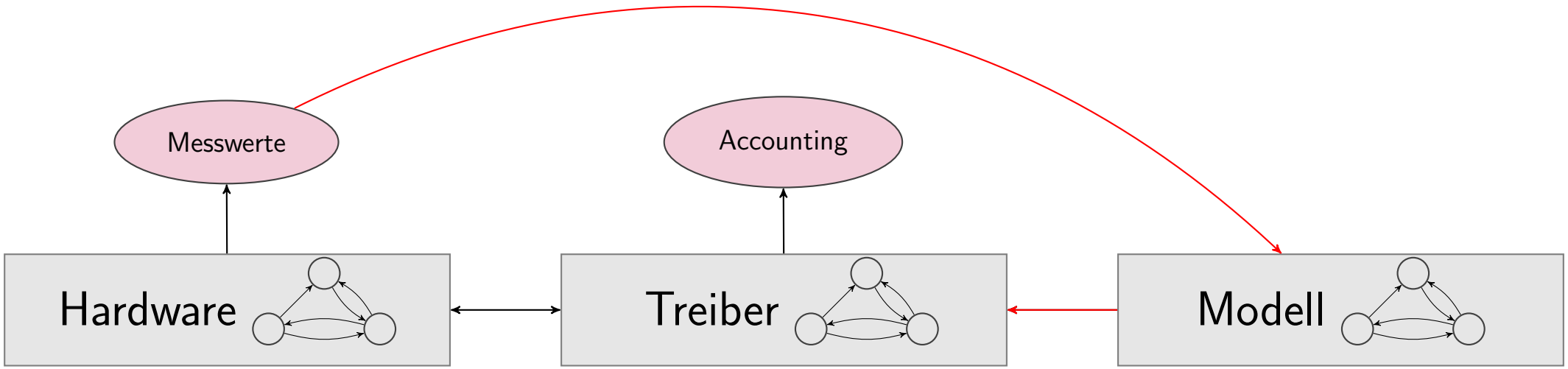
- Wie lange hält der Akku noch?
  - . . . und wieso ist er schon so leer?
- Standardlösung bei Smartphones und Laptops:
  - Ladungszähler im Akku → Ladungszustand
  - Performance Counter im System → Verbrauchsaufschlüsselung
- Kein einheitlicher Ansatz für IoT u.ä.
  - Modellierung oft Handarbeit
  - Variable Modellgüte und -komplexität
  - Kaum Betriebssystemunterstützung

Konzept

Konzept

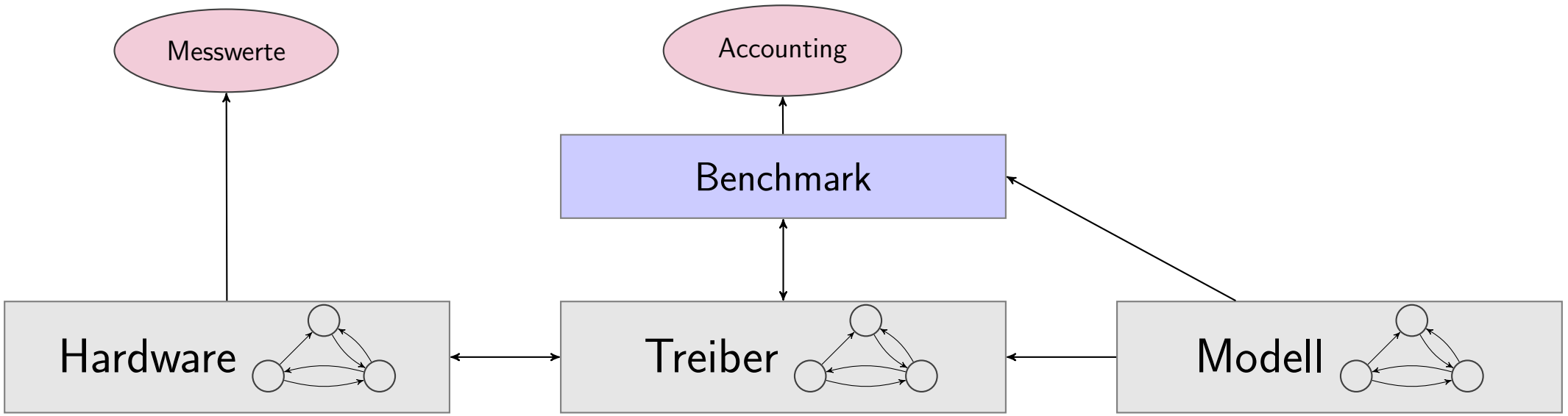


Konzept



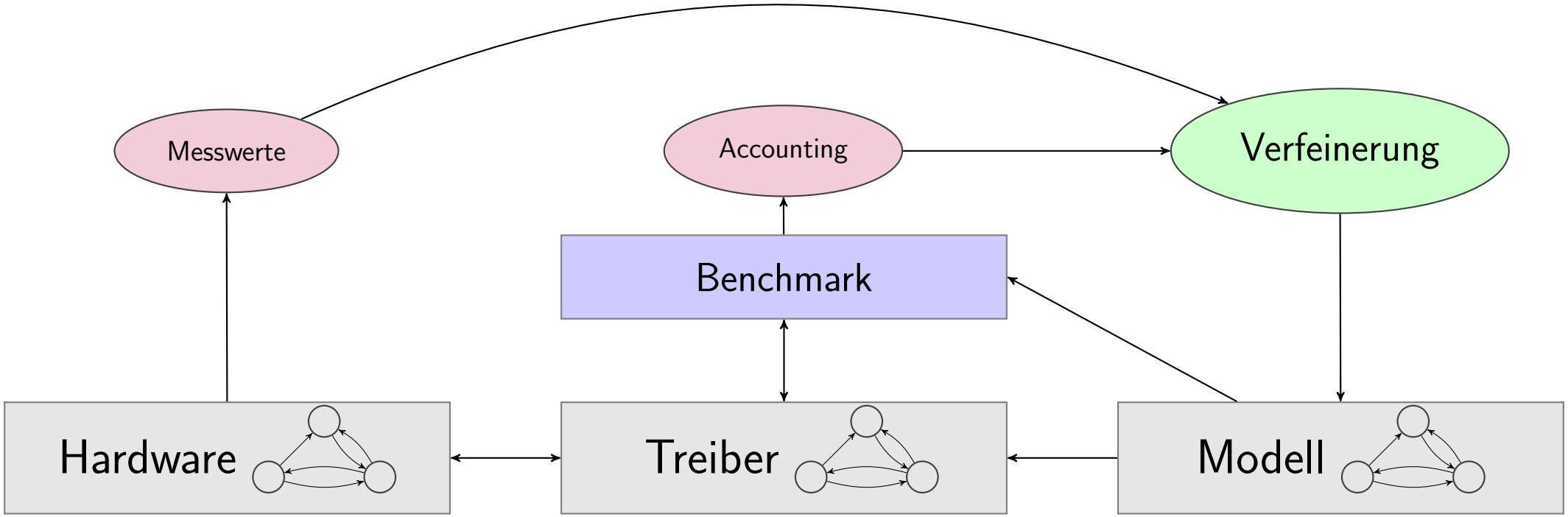
- Automatisierte Erstellung von Energiemodellen

Konzept



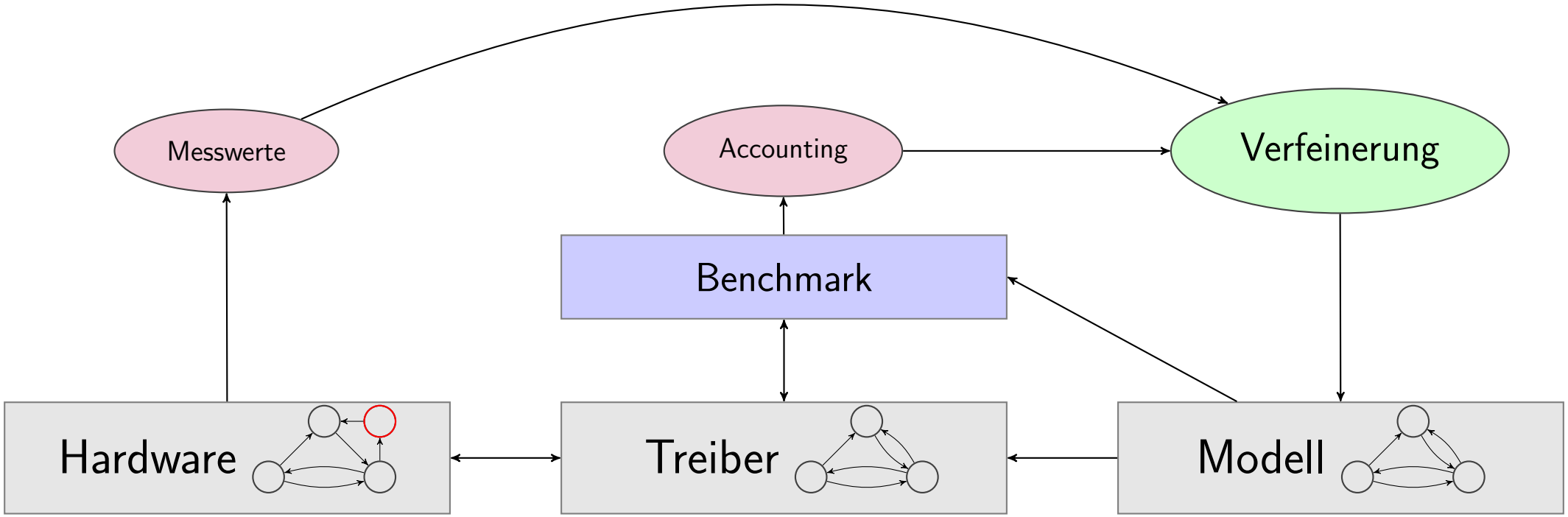
- Automatisierte Erstellung von Energiemodellen

Konzept



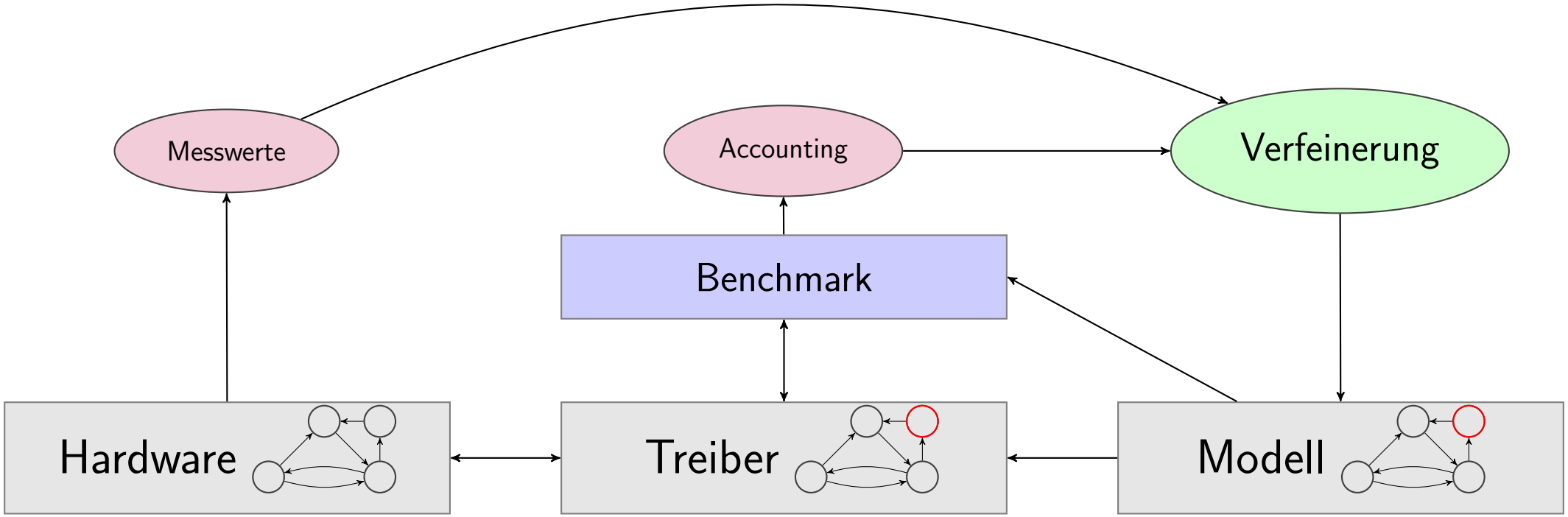
- Automatisierte Erstellung von Energiemodellen

Konzept



- Automatisierte Erstellung von Energiemodellen

Konzept



- Automatisierte Erstellung von Energiemodellen

Inhalt

Modelltypen

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

Modelltypen

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

  → Performance Counter: Zeit pro Zustand

Modelltypen

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

  → Performance Counter: Zeit pro Zustand

P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API

Modelltypen

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

  → Performance Counter: Zeit pro Zustand

P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API

  → Funktionsaufrufe zählen

Modelltypen

Analytisch

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

  → Performance Counter: Zeit pro Zustand

P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API

  → Funktionsaufrufe zählen

Automatenbasiert

Modelltypen

Analytisch

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

  → Performance Counter: Zeit pro Zustand

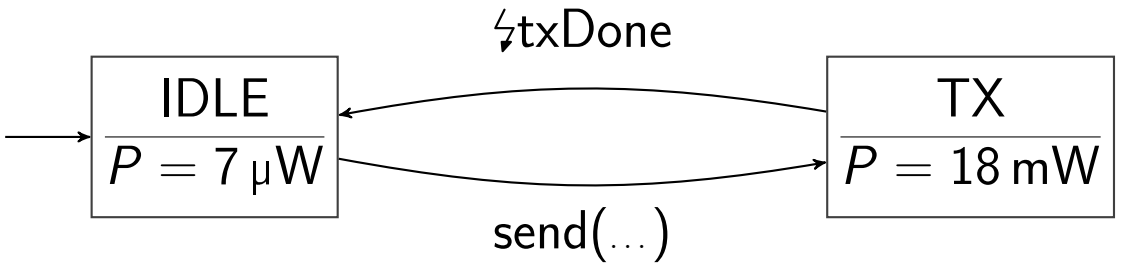
P_{base}	7 µW
send(...)	43 µJ

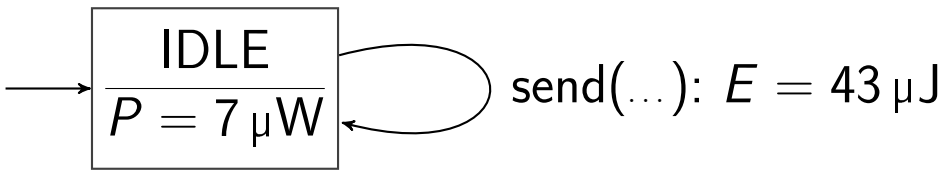
$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API

  → Funktionsaufrufe zählen

Automatenbasiert





Modelltypen

Analytisch

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

- Hardware-Zustände

  → Performance Counter: Zeit pro Zustand

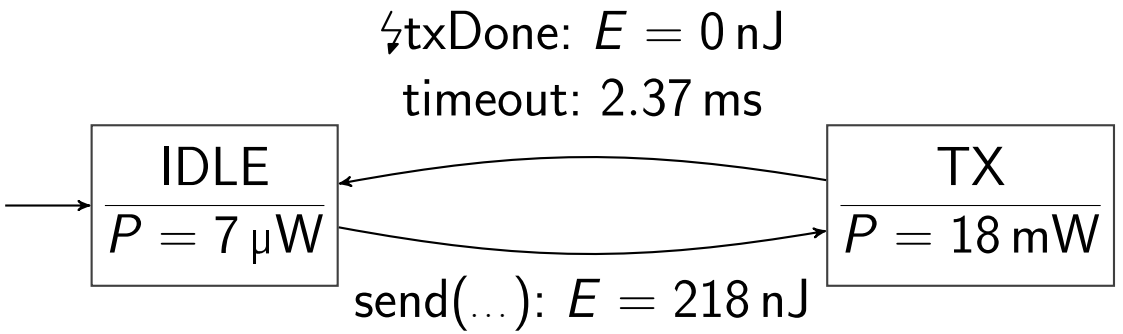
P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API

  → Funktionsaufrufe zählen

Automatenbasiert



Modelltypen

Analytisch

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

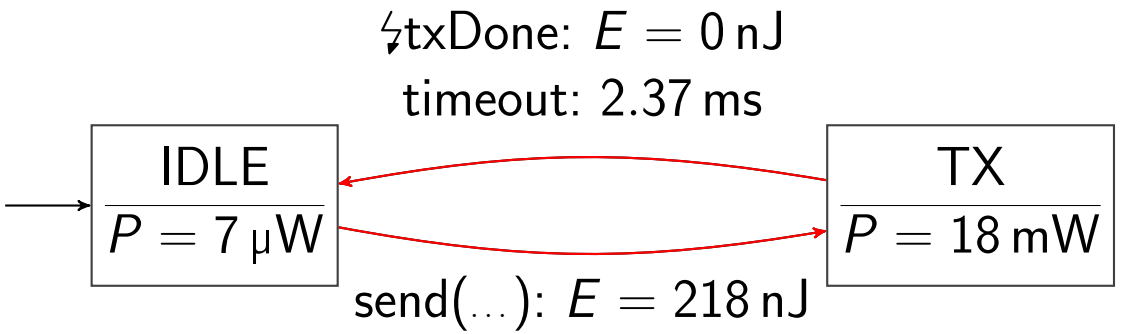
- Hardware-Zustände
  → Performance Counter: Zeit pro Zustand

P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API
  → Funktionsaufrufe zählen

Automatenbasiert



- Automat im Treiber abgebildet
  → Zustand vorhalten und ggf. aktualisieren

Modelltypen

Analytisch

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

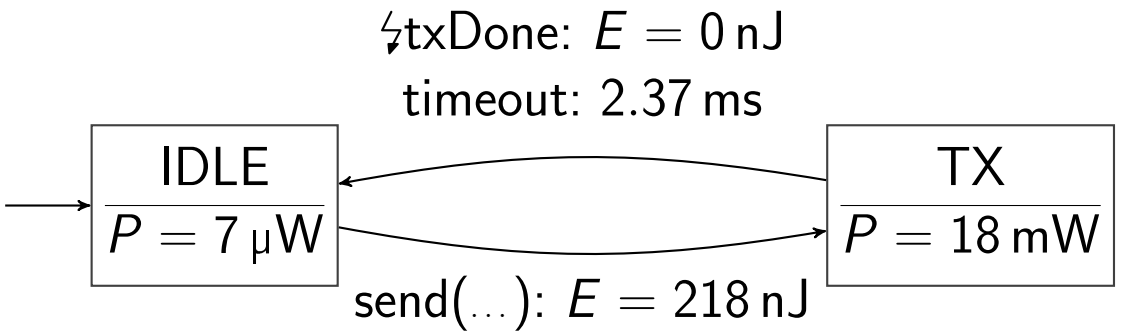
- Hardware-Zustände
  → Performance Counter: Zeit pro Zustand

P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API
  → Funktionsaufrufe zählen

Automatenbasiert



- Automat im Treiber abgebildet
  → Zustand vorhalten und ggf. aktualisieren
  - Transition $\hat{=}$ Treiberfunktion / Interrupt
  - Oder: Zeitabhängig („Tail States“)

Modelltypen

Analytisch

IDLE	7 µW
TX	18 mW

$$E = \sum_{q \in Q} t_q \cdot P_q$$

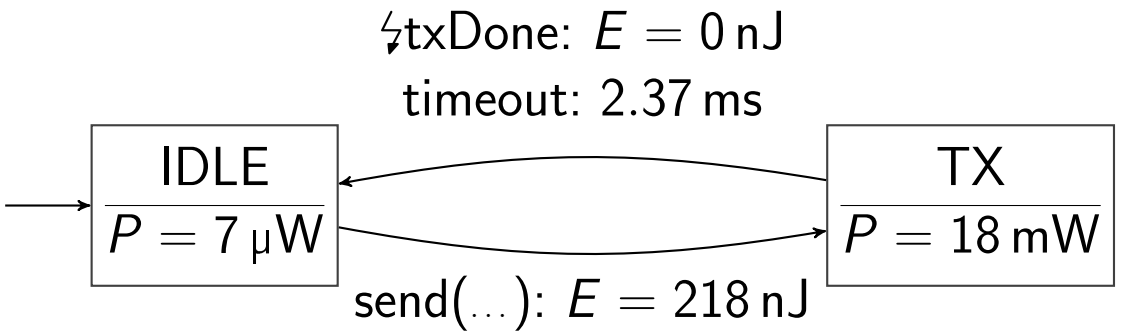
- Hardware-Zustände
  → Performance Counter: Zeit pro Zustand

P_{base}	7 µW
send(...)	43 µJ

$$E = t \cdot P_{base} + \sum_{f \in F} n_f \cdot E_f$$

- Treiber-API
  → Funktionsaufrufe zählen

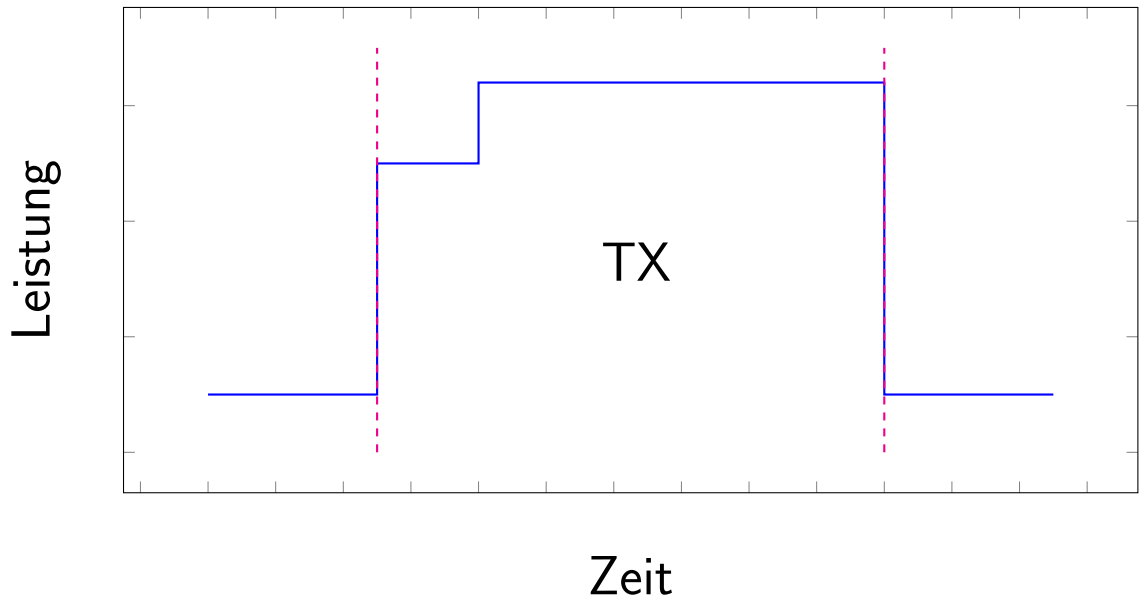
Automatenbasiert



- Automat im Treiber abgebildet
  → Zustand vorhalten und ggf. aktualisieren
  - Transition $\hat{=}$ Treiberfunktion / Interrupt
  - Oder: Zeitabhängig („Tail States“)
  - Ausdrucksstärker als übliche analytische Modelle [McC+11]

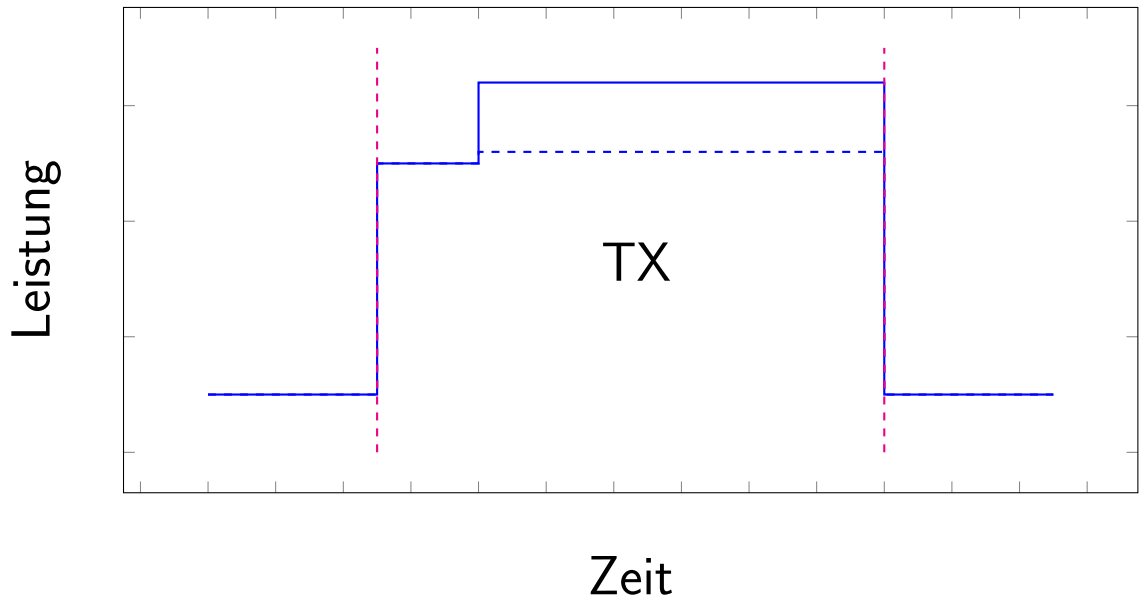
Parametergewahre Modelle

- Systemverhalten ist nicht statisch



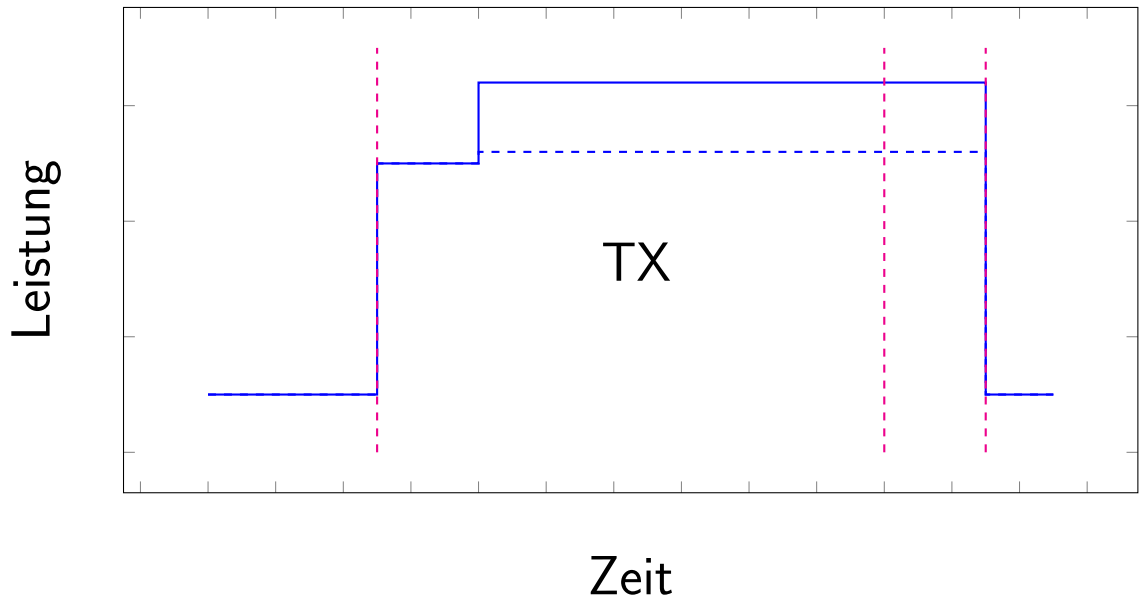
Parametergewahre Modelle

- Systemverhalten ist nicht statisch
  - Funkmodule: Konfigurierbare Sendeleistung



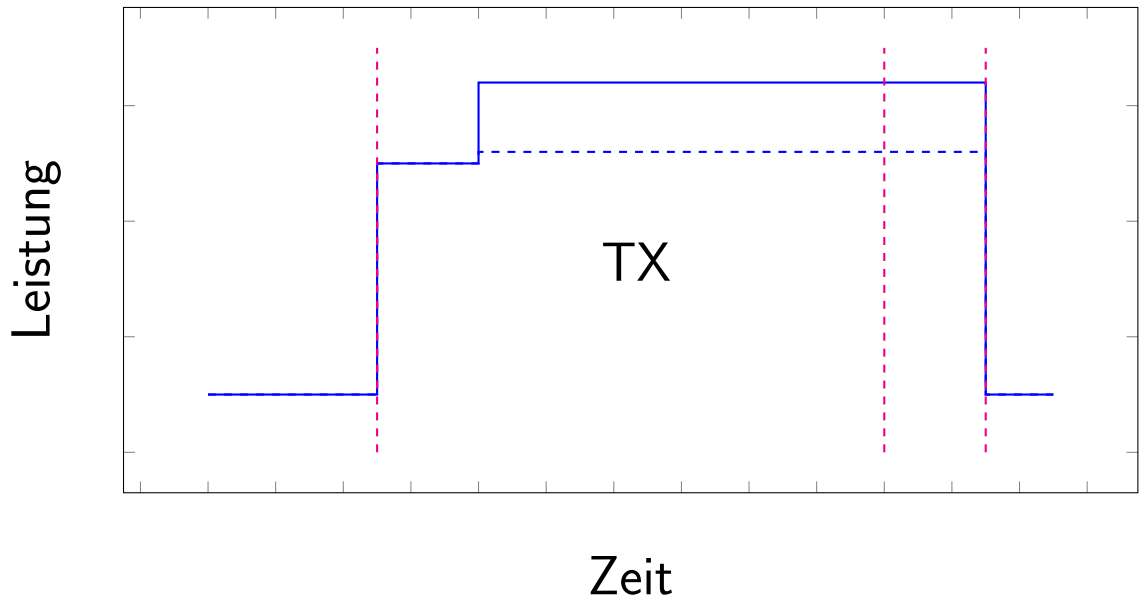
Parametergewahre Modelle

- Systemverhalten ist nicht statisch
  - Funkmodule: Konfigurierbare Sendeleistung, Bitrate, Paketlänge



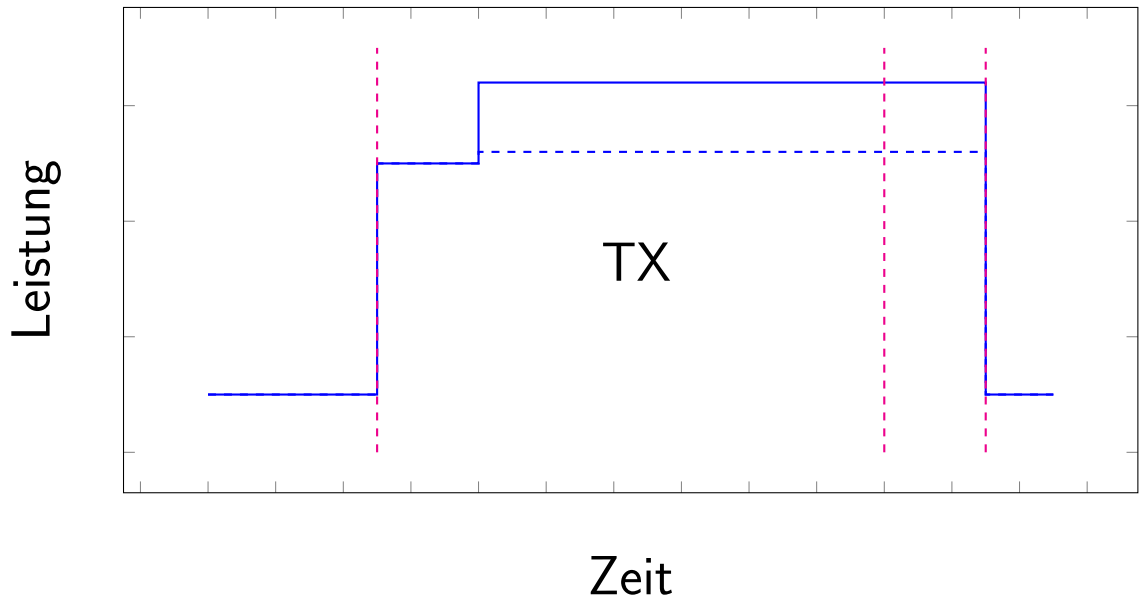
Parametergewahre Modelle

- Systemverhalten ist nicht statisch
  - Funkmodule: Konfigurierbare Sendeleistung, Bitrate, Paketlänge
  - Aber auch: Anzahl Null-Bits auf I^2C-Bus, ...



Parametergewahre Modelle

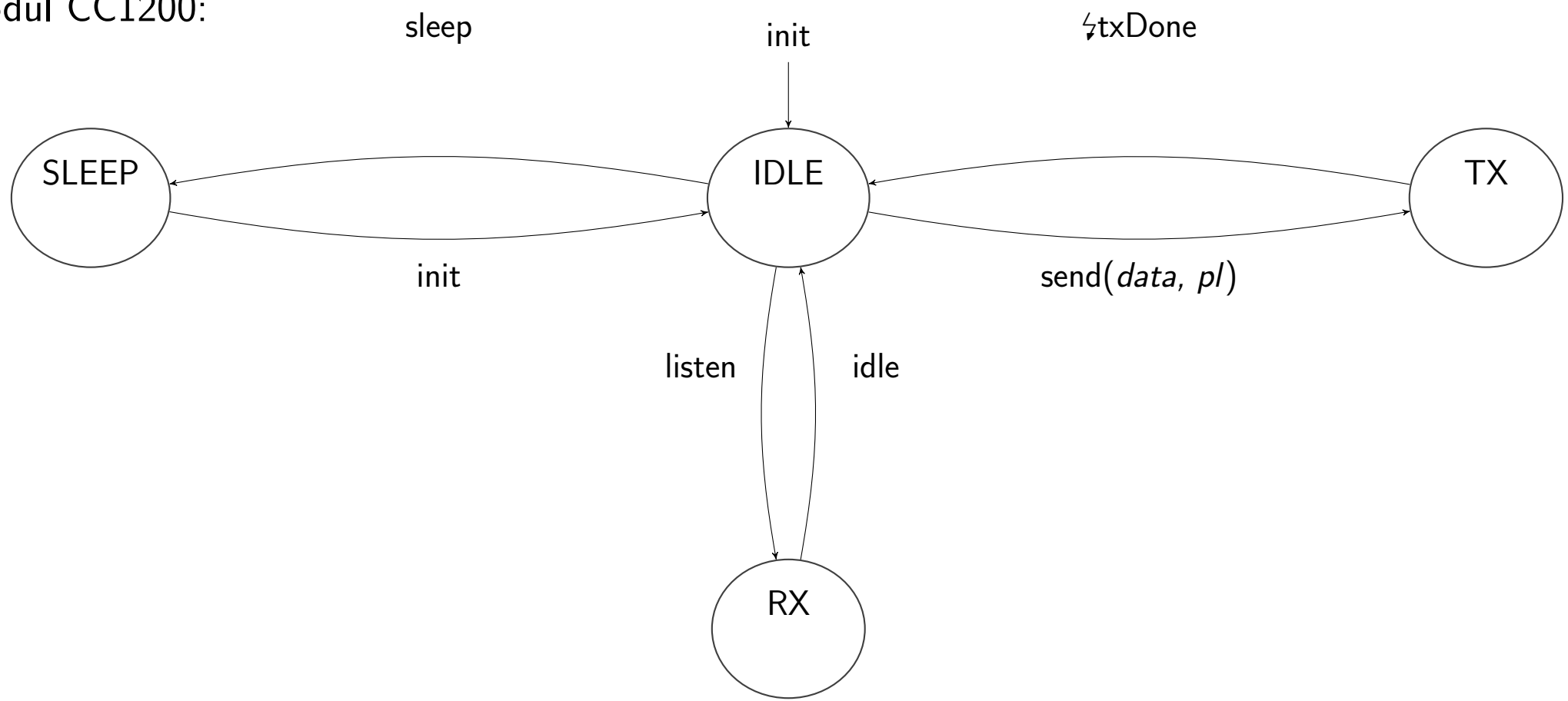
- Systemverhalten ist nicht statisch
  - Funkmodule: Konfigurierbare Sendeleistung, Bitrate, Paketlänge
  - Aber auch: Anzahl Null-Bits auf I^2C-Bus, ...



→ Funktionen statt statischer Energiewerte

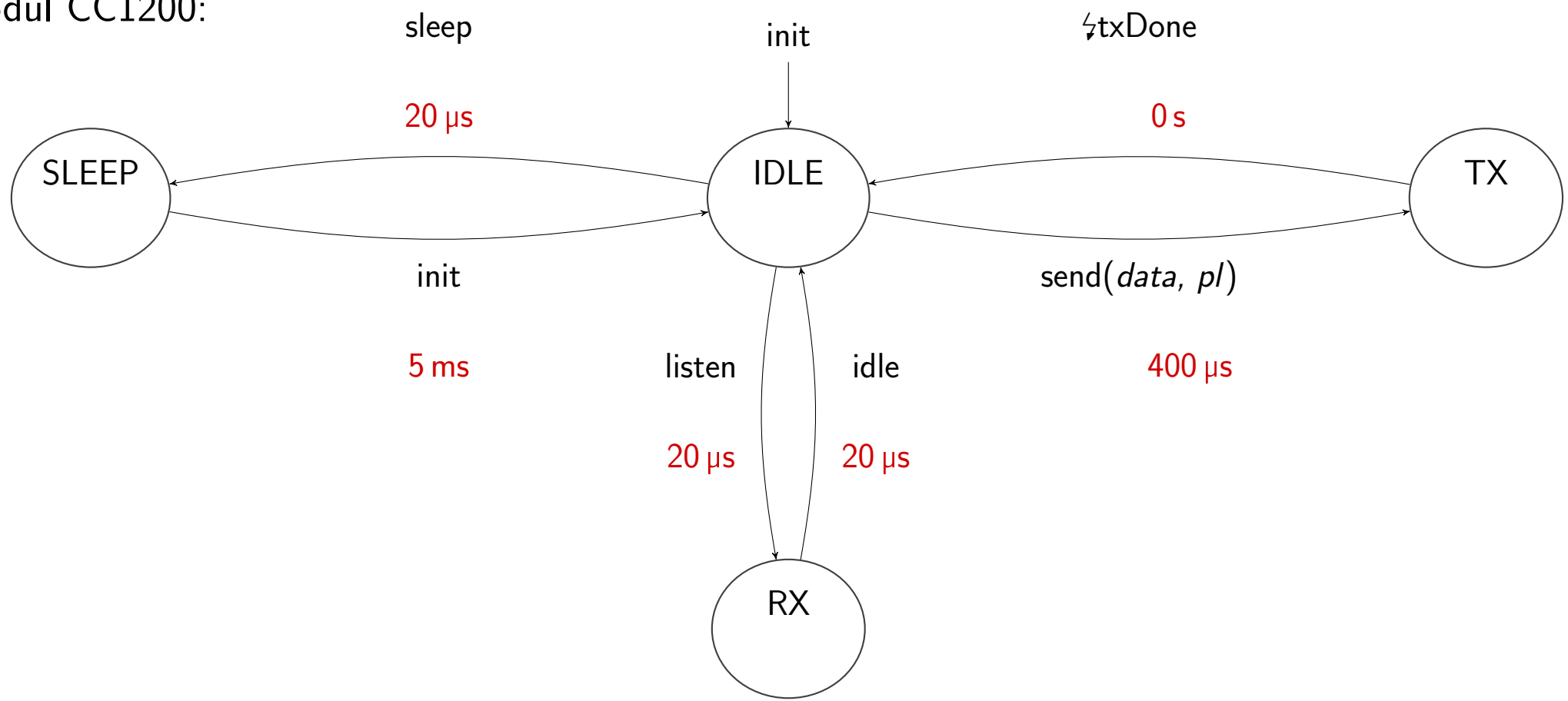
Parametergewahre Priced Timed Automata (PTA)

Funkmodul CC1200:



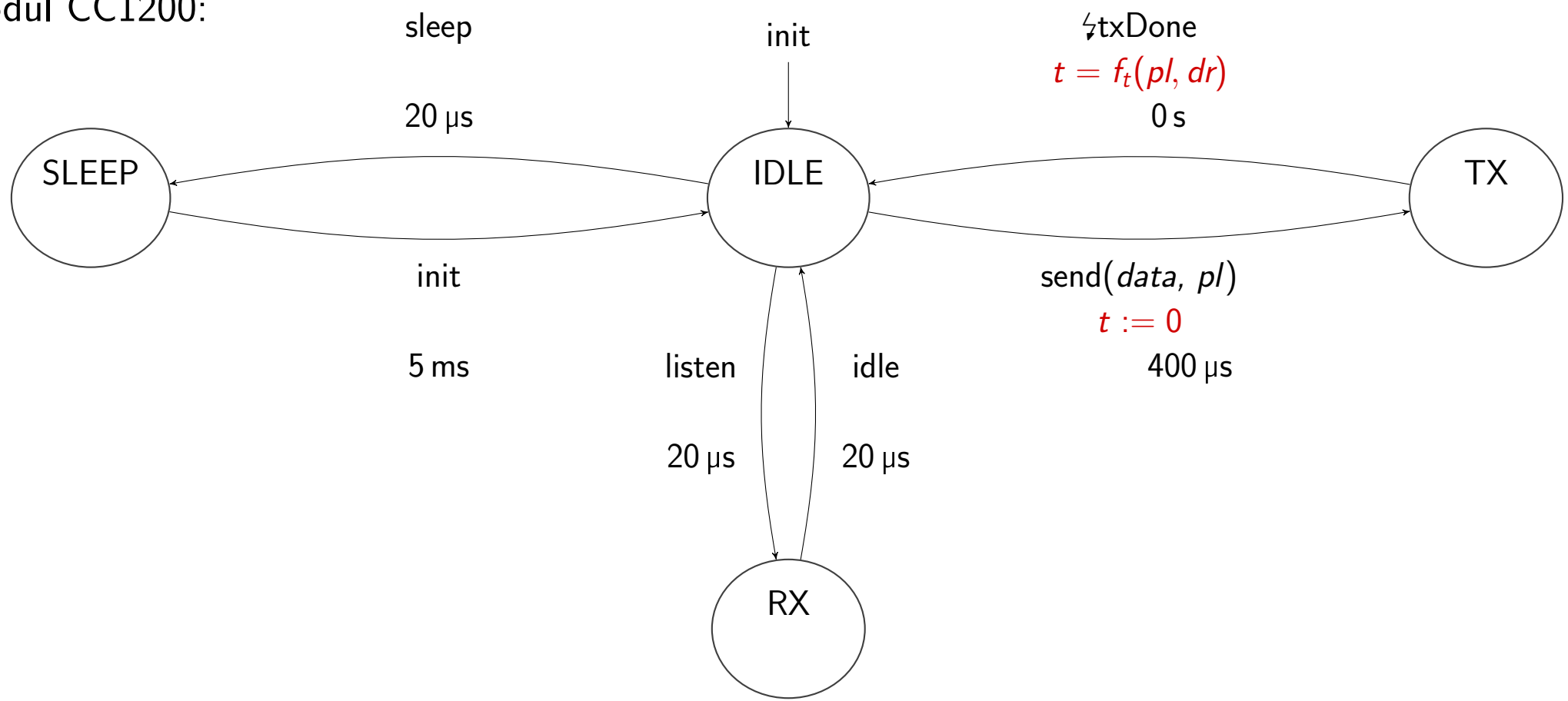
Parametergewahre Priced Timed Automata (PTA)

Funkmodul CC1200:



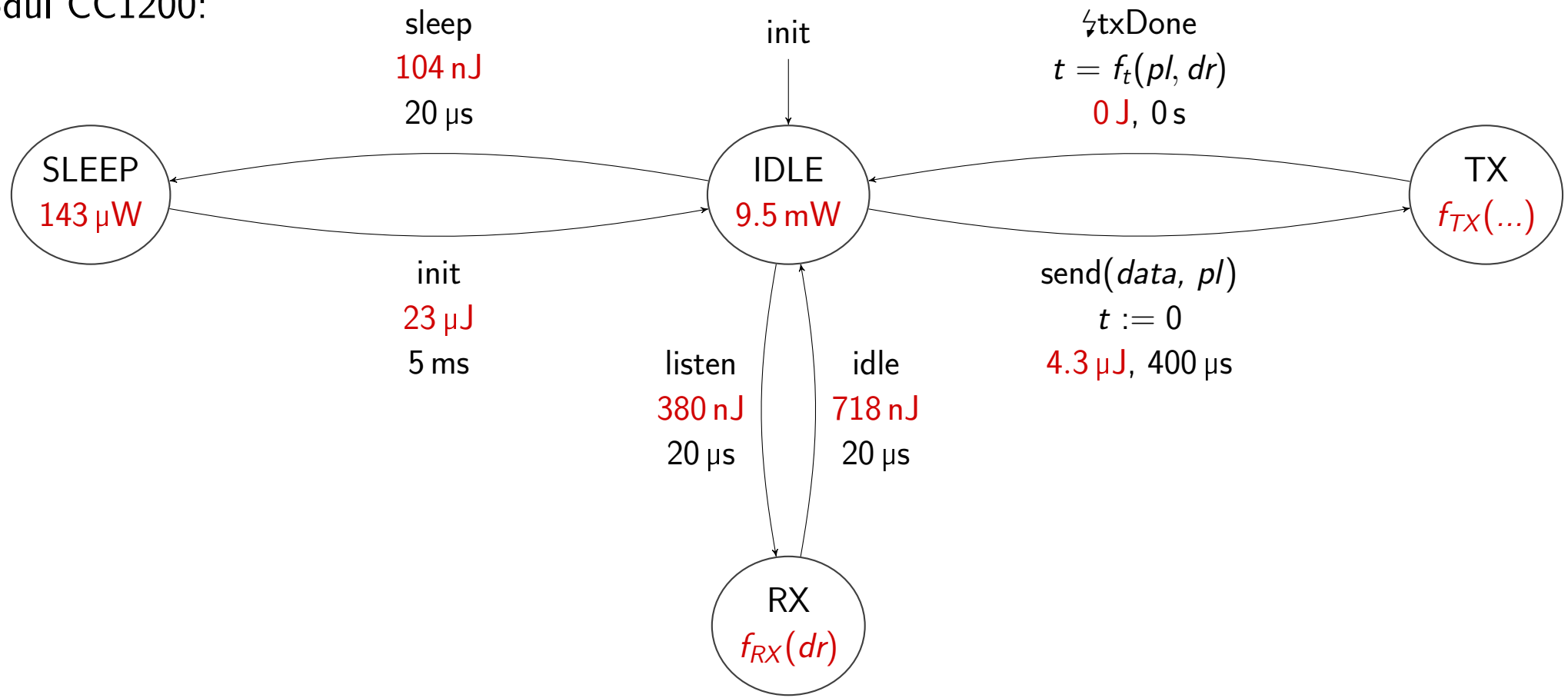
Parametergewahre Priced Timed Automata (PTA)

Funkmodul CC1200:



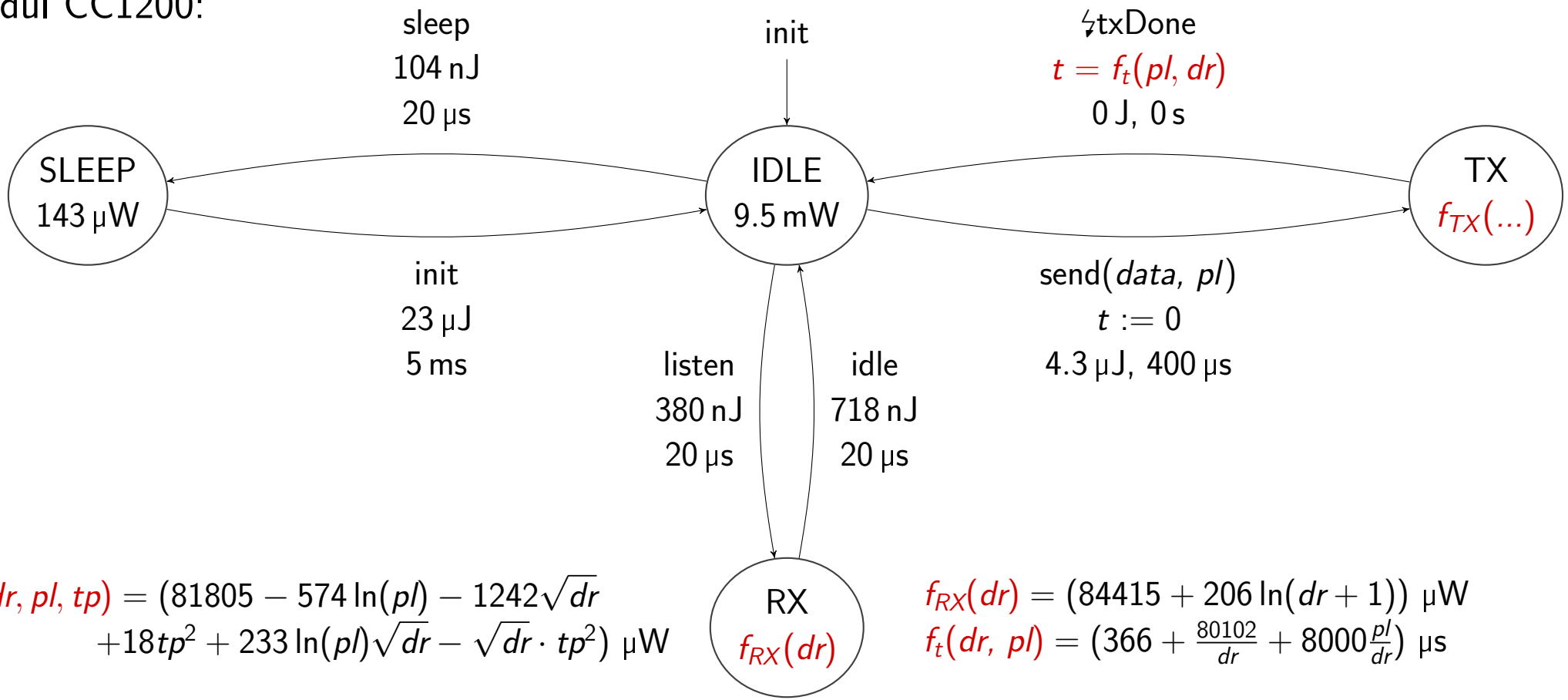
Parametergewahre Priced Timed Automata (PTA)

Funkmodul CC1200:



Parametergewahre Priced Timed Automata (PTA)

Funkmodul CC1200:



$$f_{TX}(dr, pl, tp) = (81805 - 574\ln(pl) - 1242\sqrt{dr} + 18tp^2 + 233\ln(pl)\sqrt{dr} - \sqrt{dr} \cdot tp^2)\ \mu W$$

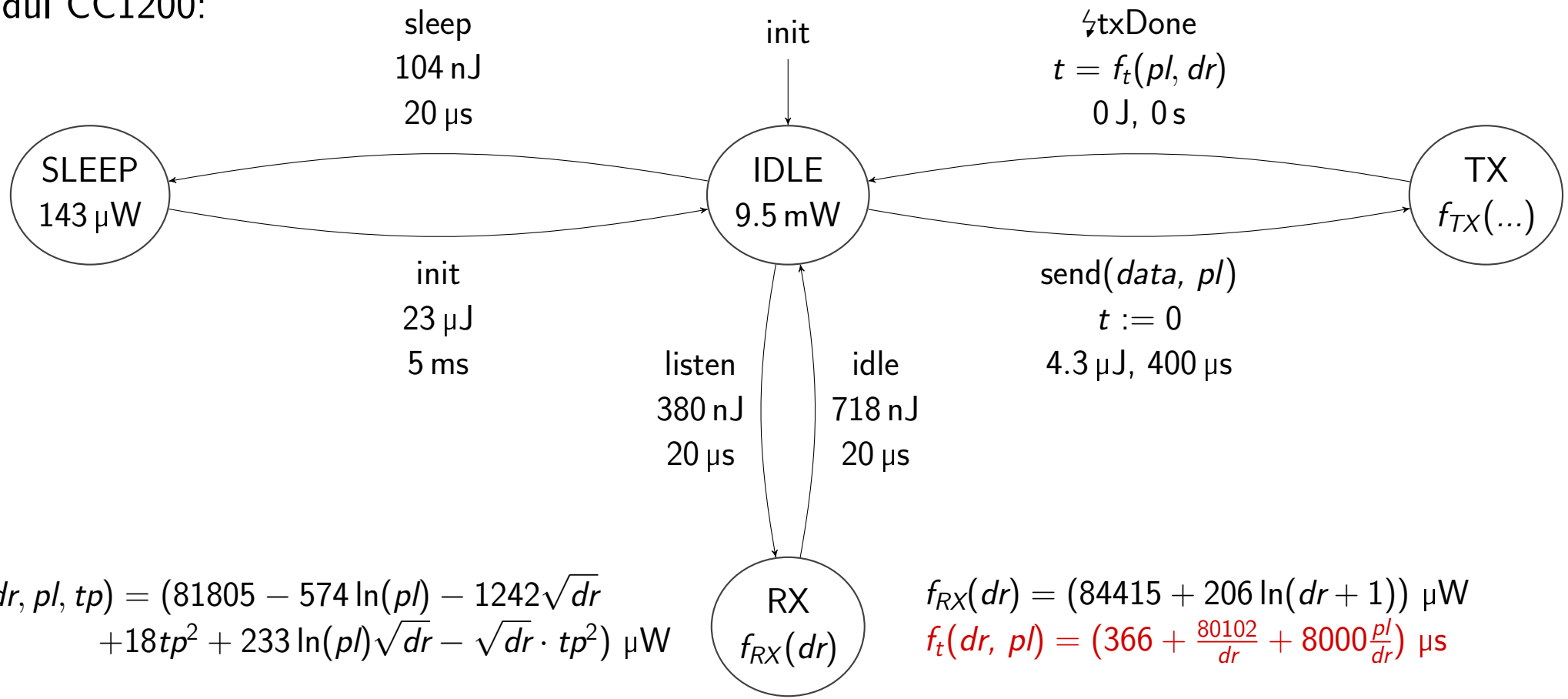
$$f_{RX}(dr) = (84415 + 206\ln(dr+1))\ \mu W$$

$$f_t(dr, pl) = (366 + \tfrac{80102}{dr} + 8000\tfrac{pl}{dr})\ \mu s$$

Parameter: Paketlänge pl, Bitrate dr, Sendeleistung tp

Parametergewahre Priced Timed Automata (PTA)

Funkmodul CC1200:



$$f_{TX}(dr, pl, tp) = (81805 - 574\ln(pl) - 1242\sqrt{dr} + 18tp^2 + 233\ln(pl)\sqrt{dr} - \sqrt{dr} \cdot tp^2)\ \mu W$$

$$f_{RX}(dr) = (84415 + 206\ln(dr+1))\ \mu W$$

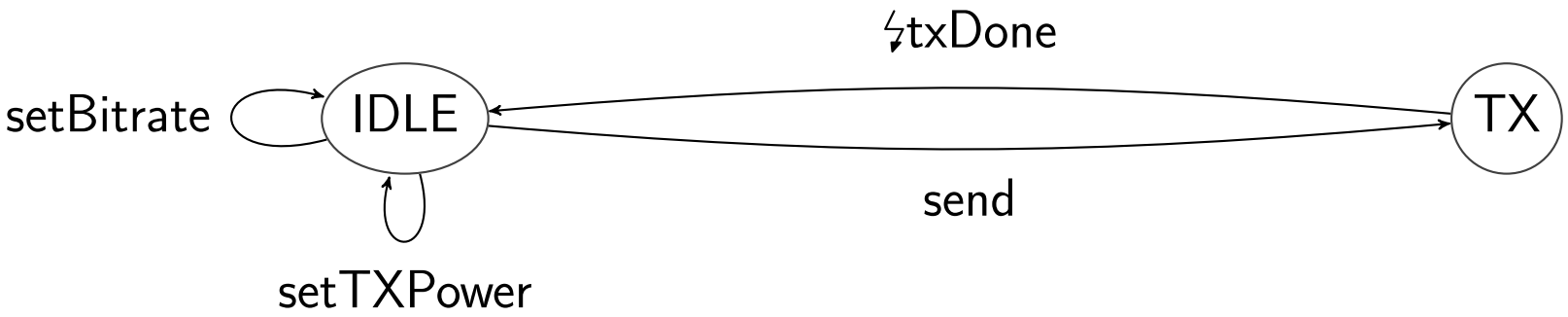
$$f_t(dr, pl) = (366 + \tfrac{80102}{dr} + 8000\tfrac{pl}{dr})\ \mu s$$

Parameter: Paketlänge pl, Bitrate dr, Sendeleistung tp

Inhalt

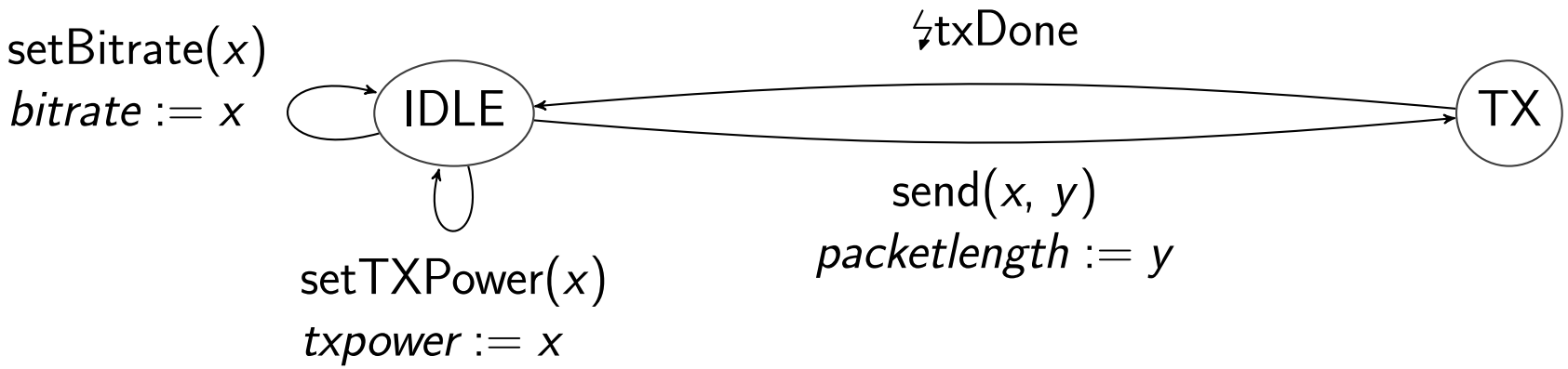
Benchmark-Generierung

- Hardwarekomponenten verhalten sich wie ein DFA
  - Zustände und Transitionen meist vom Hersteller dokumentiert



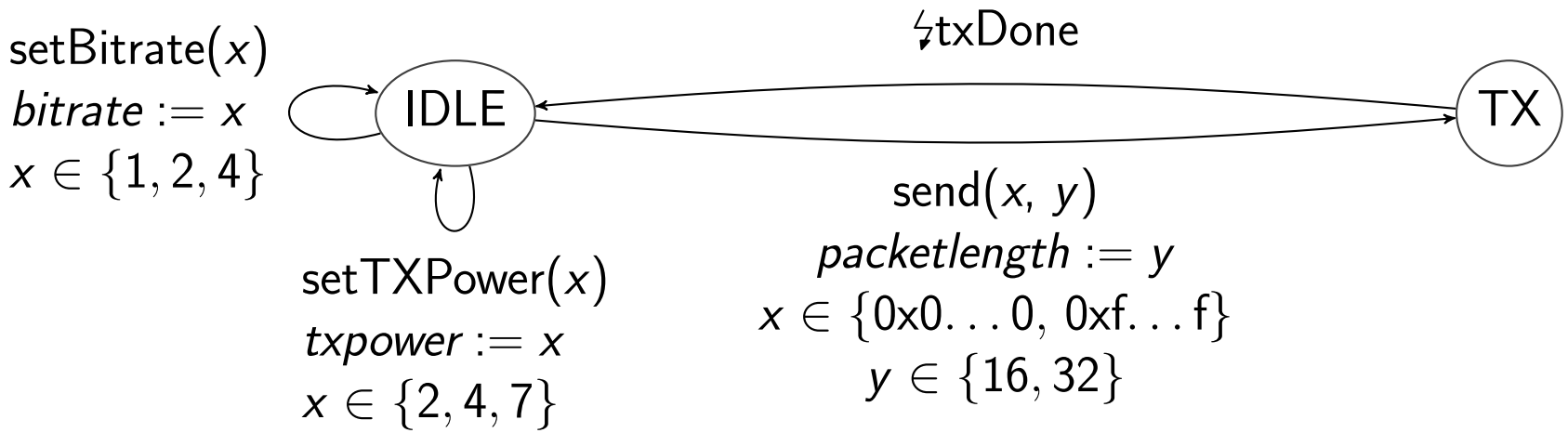
Benchmark-Generierung

- Hardwarekomponenten verhalten sich wie ein DFA
  - Zustände und Transitionen meist vom Hersteller dokumentiert
- Konfigurierbare Parameter und Funktionsargumente bekannt



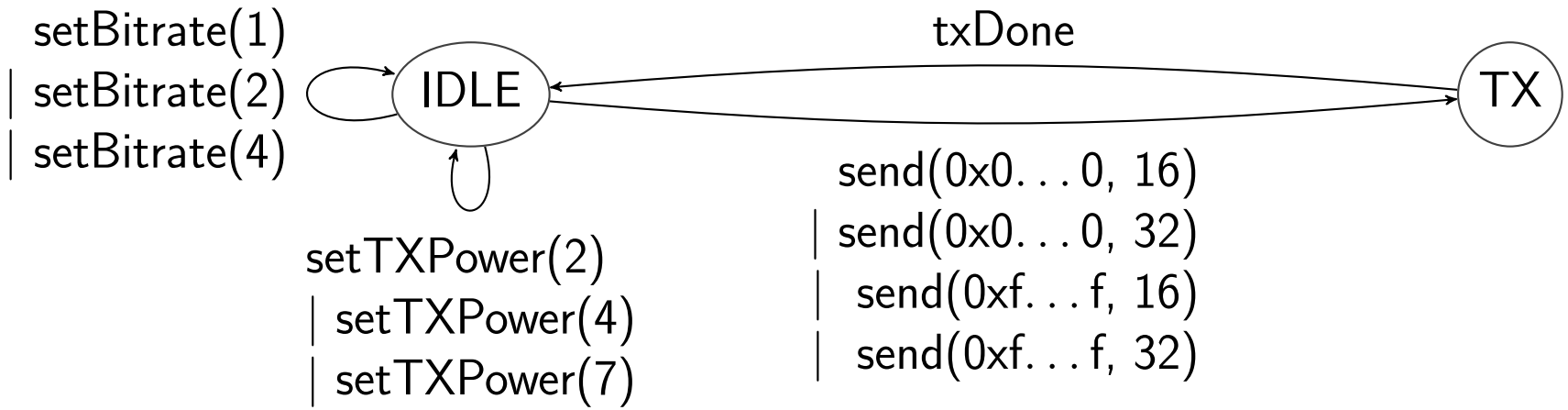
Benchmark-Generierung

- Hardwarekomponenten verhalten sich wie ein DFA
  - Zustände und Transitionen meist vom Hersteller dokumentiert
- Konfigurierbare Parameter und Funktionsargumente bekannt
  - Erlaubte Werte ebenfalls



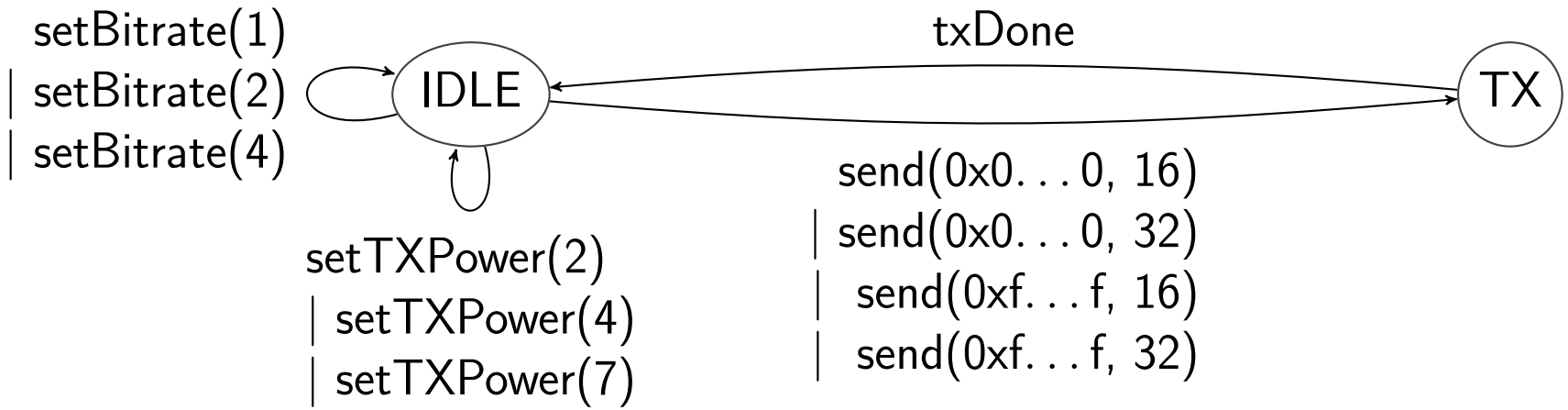
Benchmark-Generierung

- Hardwarekomponenten verhalten sich wie ein DFA
  - Zustände und Transitionen meist vom Hersteller dokumentiert
- Konfigurierbare Parameter und Funktionsargumente bekannt
  - Erlaubte Werte ebenfalls



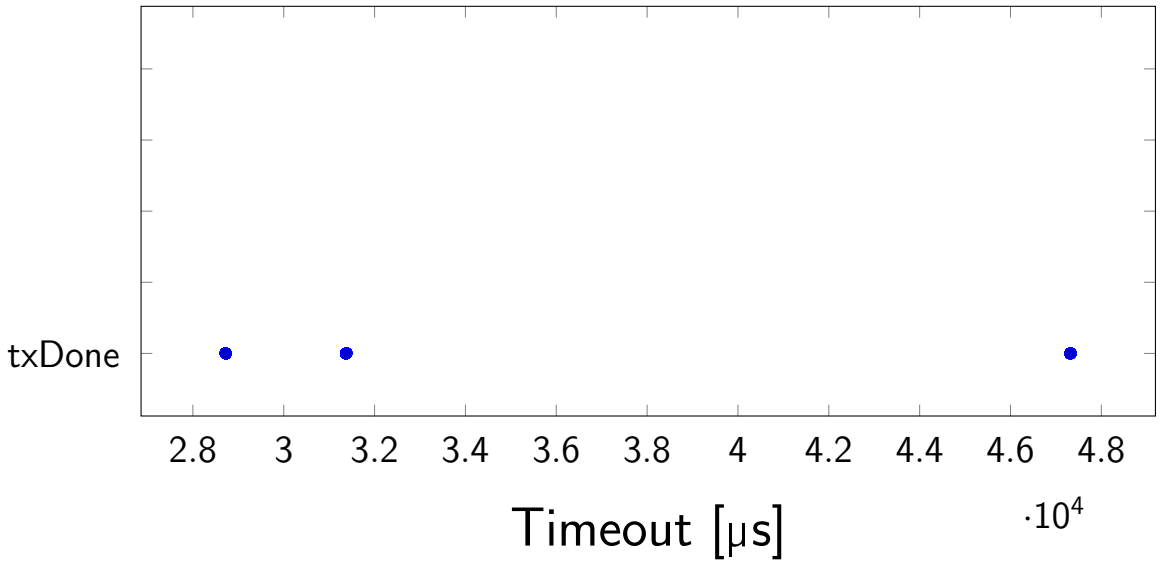
Benchmark-Generierung

- Hardwarekomponenten verhalten sich wie ein DFA
  - Zustände und Transitionen meist vom Hersteller dokumentiert
- Konfigurierbare Parameter und Funktionsargumente bekannt
  - Erlaubte Werte ebenfalls

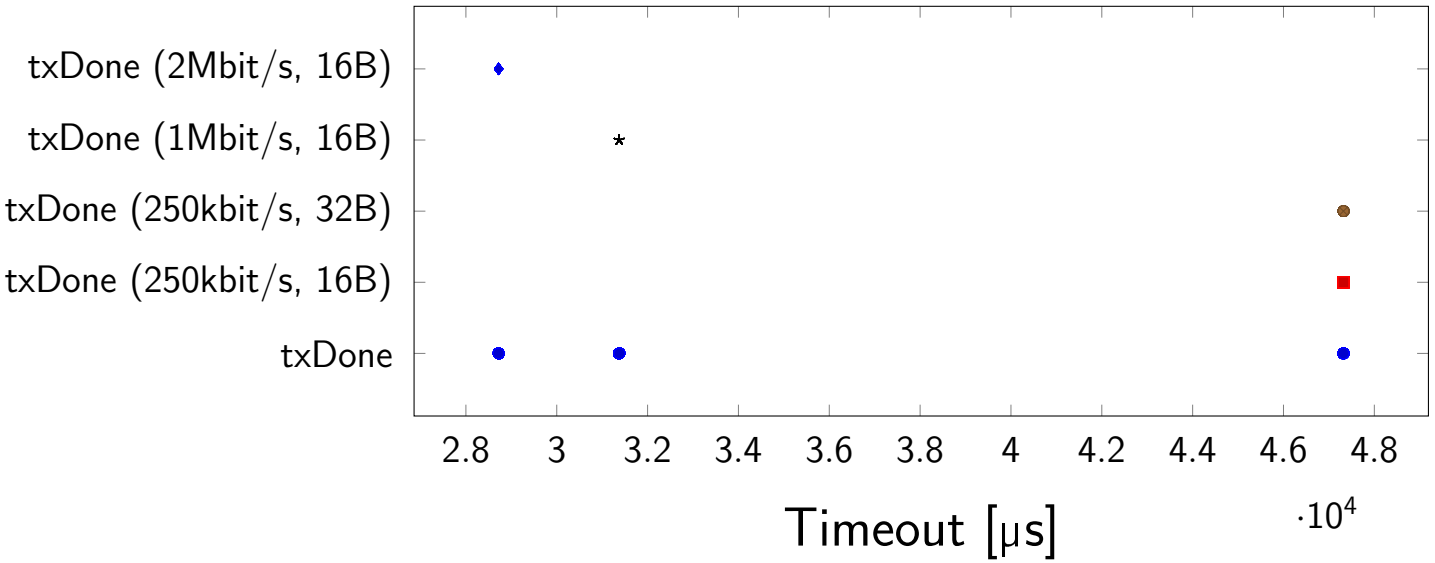


- Testprogramm: DFA-Transitionen und Zustände ablaufen
  - UART und GPIO zur Synchronisierung mit Energiemessung
  - Können z.B. per AspectC++ an Treiberfunktionen gewoben werden

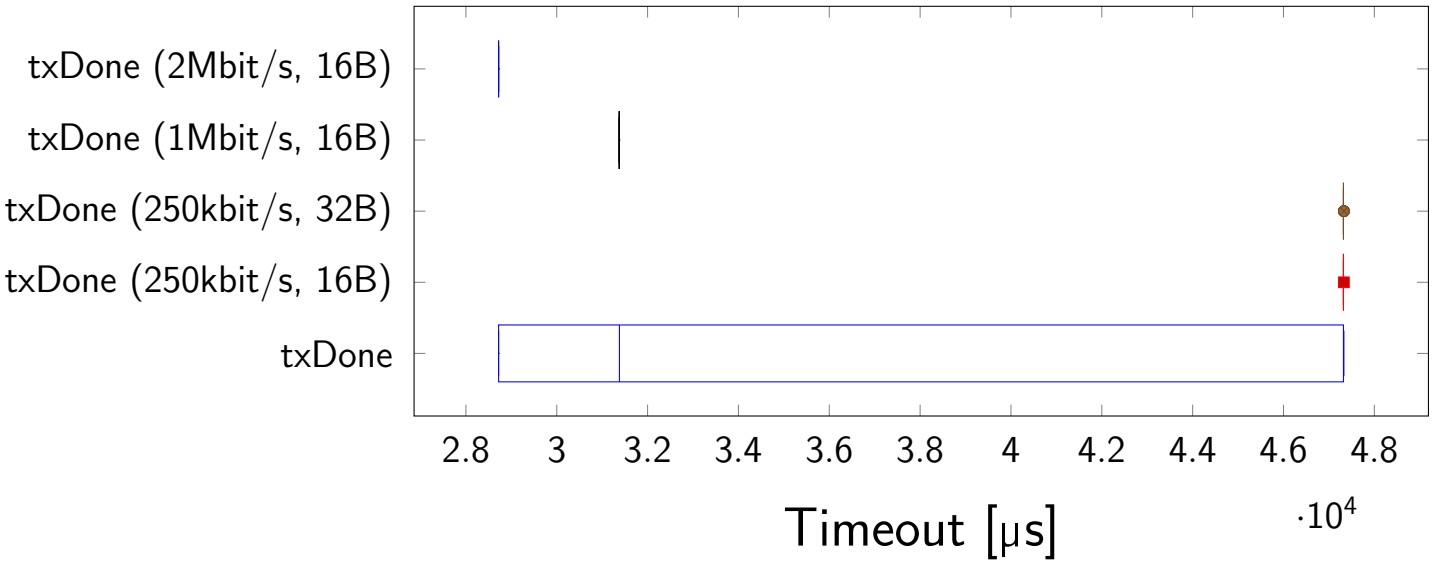
Heuristik für Parameter-Abhängigkeiten



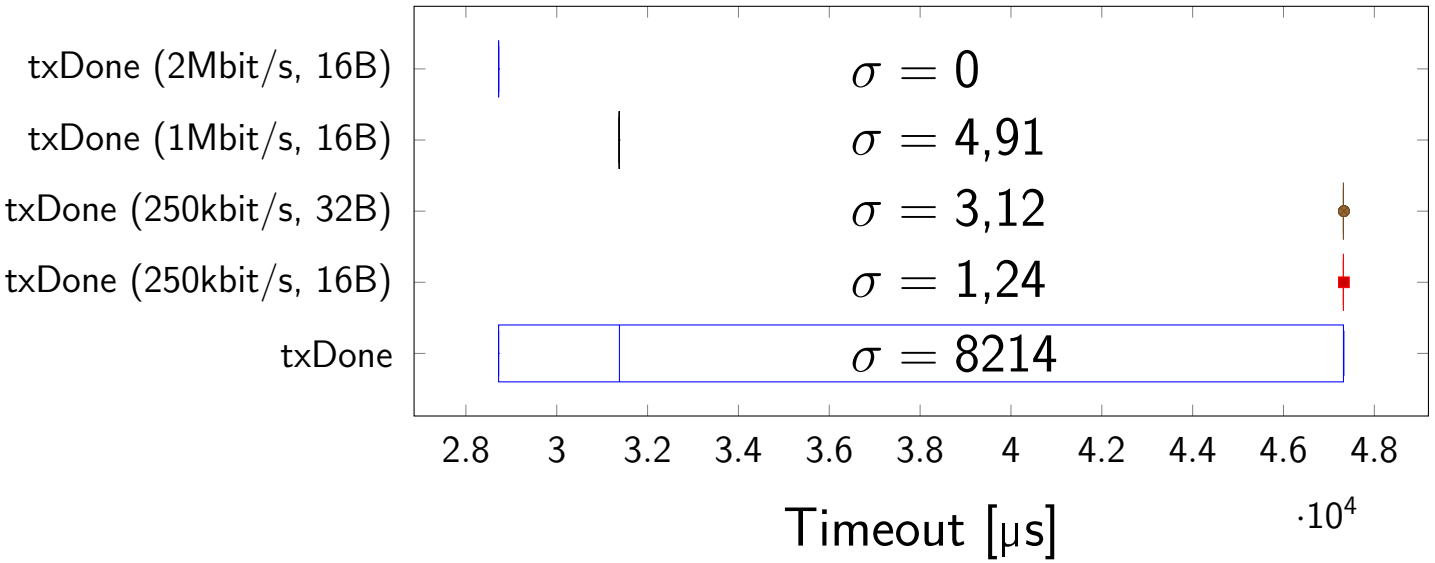
Heuristik für Parameter-Abhängigkeiten



Heuristik für Parameter-Abhängigkeiten

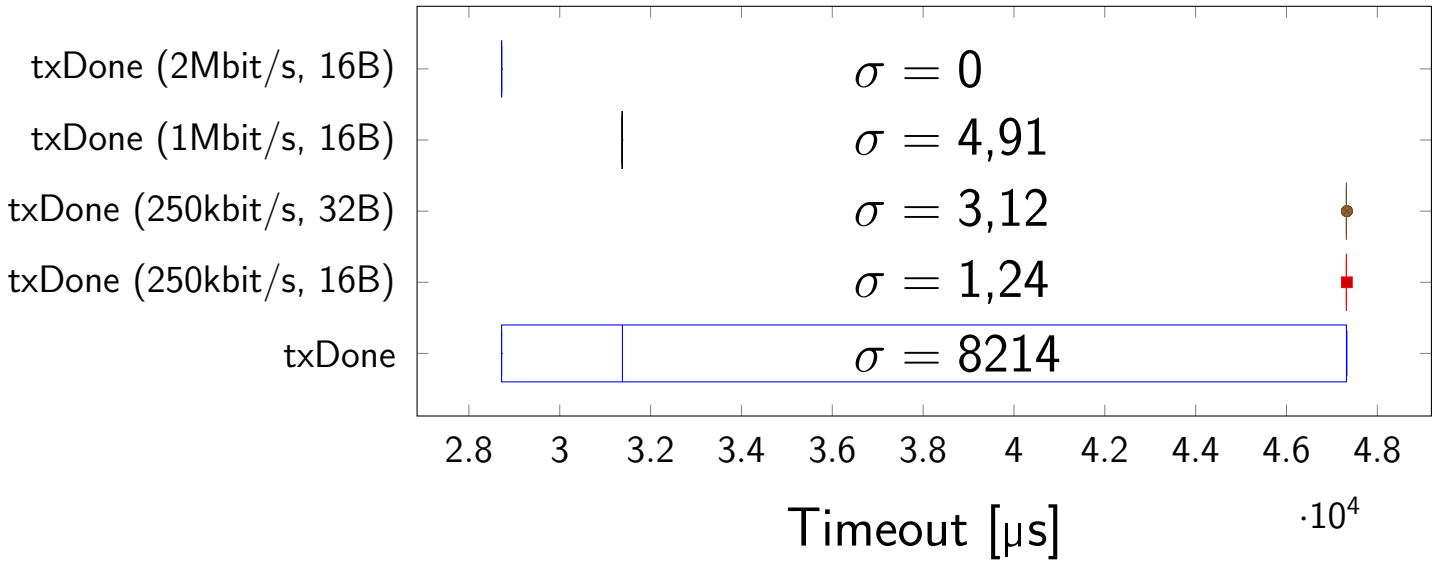


Heuristik für Parameter-Abhängigkeiten



- Vergleich der Streuungsmaße
  - Alle Parameter variabel: txDone $\rightarrow \sigma_X$
  - Alle Parameter fest: z.B. txDone (1Mbit/s, 16B) $\rightarrow \sigma_{X,\vec{p}}$

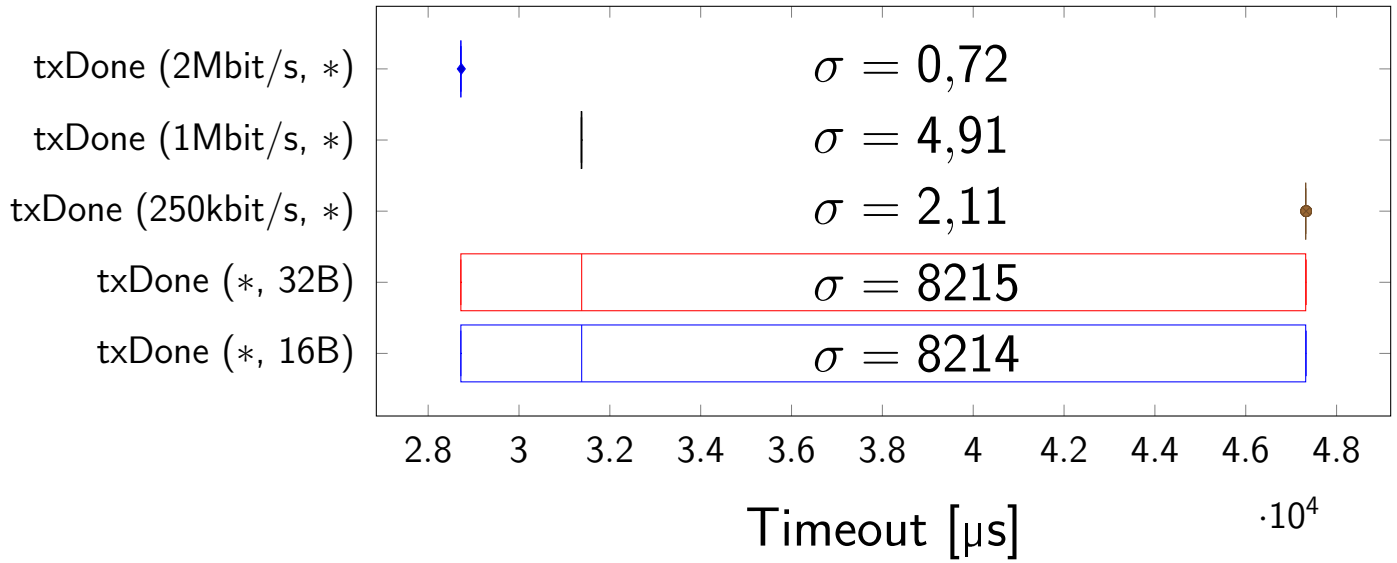
Heuristik für Parameter-Abhängigkeiten



- Vergleich der Streuungsmaße
  - Alle Parameter variabel: txDone $\rightarrow \sigma_X$
  - Alle Parameter fest: z.B. txDone (1Mbit/s, 16B) $\rightarrow \sigma_{X,\vec{p}}$

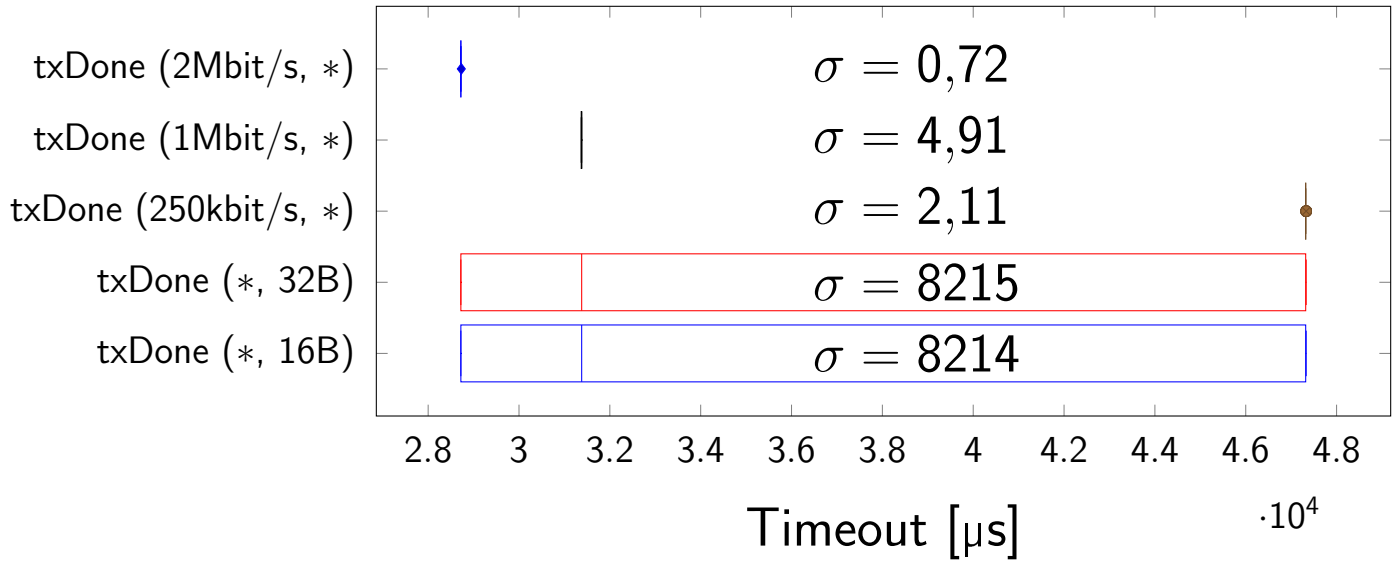
- $\sigma_{X,\vec{p}} \ll \sigma_X$ $\Rightarrow$ parameterabhängig

Heuristik für Parameter-Abhängigkeiten



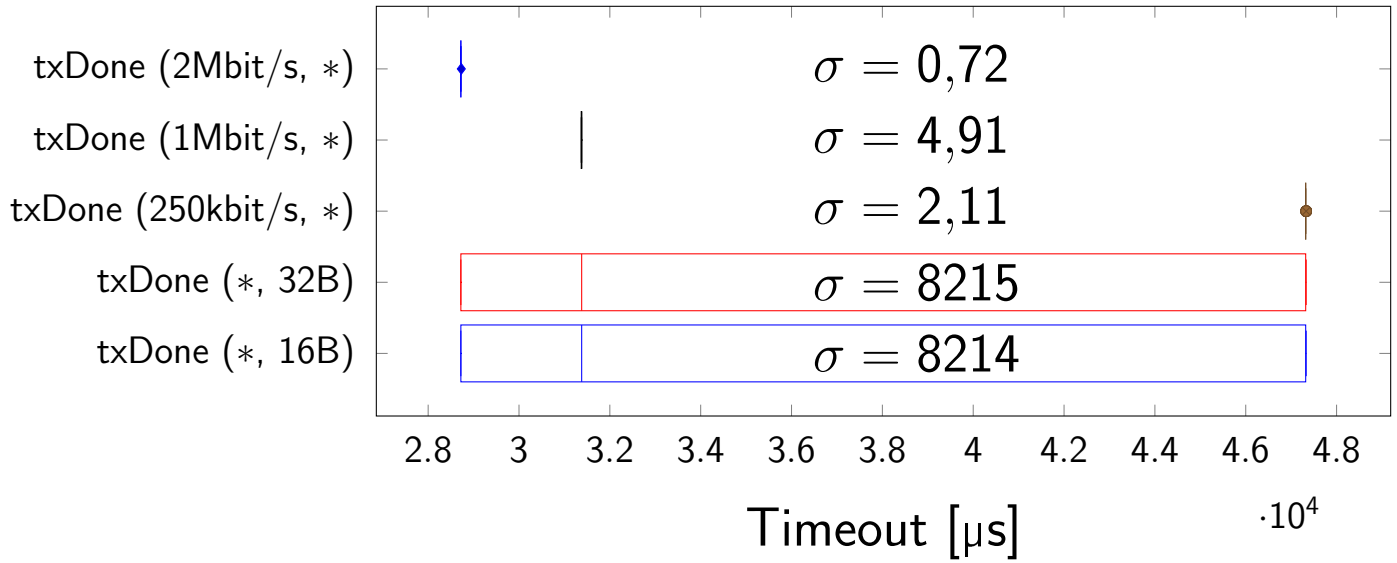
- Vergleich der Streuungsmaße
  - Alle Parameter variabel: txDone $\rightarrow \sigma_X$
  - Alle Parameter fest: z.B. txDone (1Mbit/s, 16B) $\rightarrow \sigma_{X,\vec{p}}$
  - Ein Parameter variabel: z.B. txDone (1Mbit/s, $*$) $\rightarrow \sigma_{X,\vec{p}\setminus i}$
- $\sigma_{X,\vec{p}} \ll \sigma_X$ $\Rightarrow$ parameterabhängig

Heuristik für Parameter-Abhängigkeiten



- Vergleich der Streuungsmaße
  - Alle Parameter variabel: txDone $\rightarrow \sigma_X$
  - Alle Parameter fest: z.B. txDone (1Mbit/s, 16B) $\rightarrow \sigma_{X,\vec{p}}$
  - Ein Parameter variabel: z.B. txDone (1Mbit/s, $*$) $\rightarrow \sigma_{X,\vec{p}\setminus i}$
- $\sigma_{X,\vec{p}} \ll \sigma_X$ $\Rightarrow$ parameterabhängig
- $\sigma_{X,\vec{p}} \ll \sigma_{X,\vec{p}\setminus i}$ $\Rightarrow$ abhängig von Parameter i

Heuristik für Parameter-Abhängigkeiten



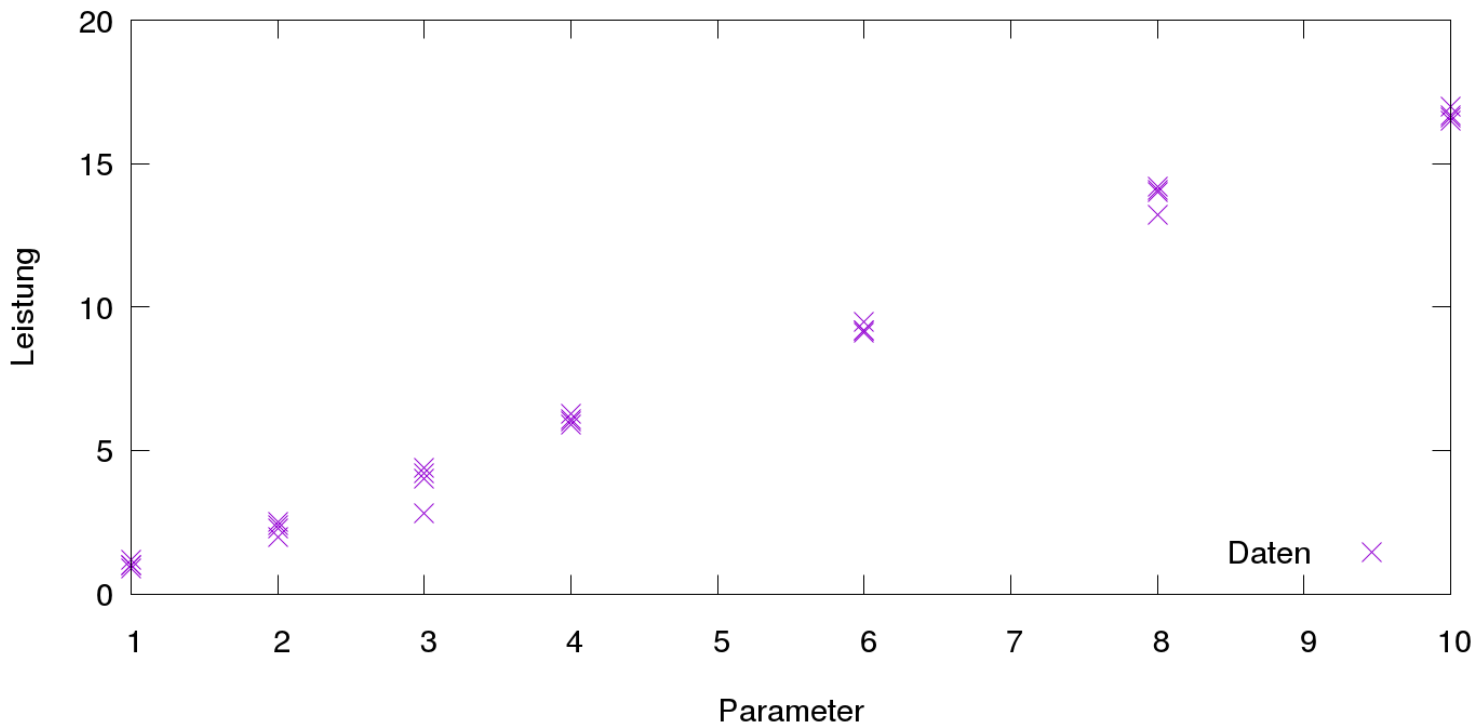
- Vergleich der Streuungsmaße
  - Alle Parameter variabel: txDone $\rightarrow \sigma_X$
  - Alle Parameter fest: z.B. txDone (1Mbit/s, 16B) $\rightarrow \sigma_{X,\vec{p}} \rightarrow \overline{\sigma_X}$
  - Ein Parameter variabel: z.B. txDone (1Mbit/s, $*$) $\rightarrow \sigma_{X,\vec{p}\setminus i} \rightarrow \overline{\sigma_{X,i}}$
- $\sigma_{X,\vec{p}} \ll \sigma_X$ bzw. $\frac{\overline{\sigma_X}}{\sigma_X} < \frac{1}{2} \Rightarrow$ parameterabhängig
- $\sigma_{X,\vec{p}} \ll \sigma_{X,\vec{p}\setminus i}$ bzw. $\frac{\overline{\sigma_X}}{\overline{\sigma_{X,i}}} < \frac{1}{2} \Rightarrow$ abhängig von Parameter i

Modellierung

- Parameter-unabhängige Eigenschaften: Median
- Sonst: Funktionsbestimmung per Regressionsanalyse

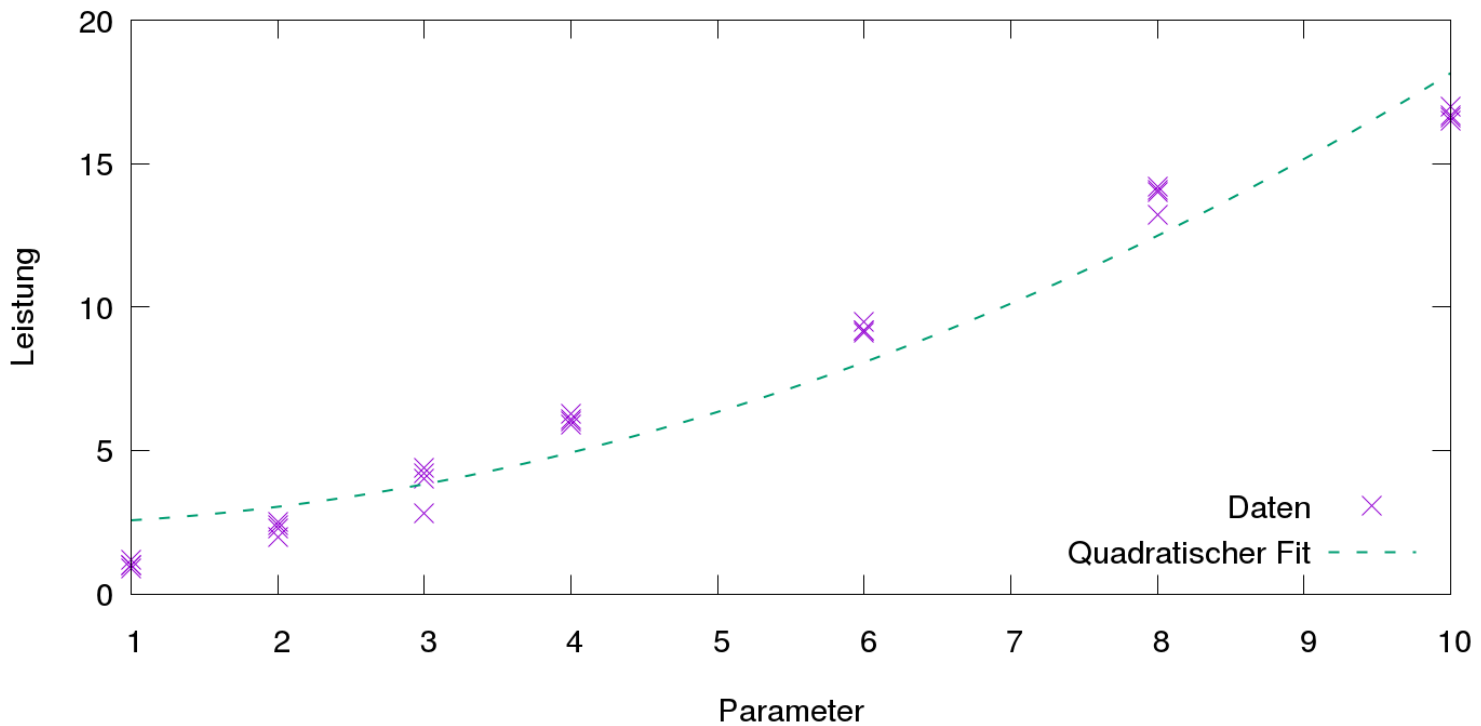
Modellierung

- Parameter-unabhängige Eigenschaften: Median
- Sonst: Funktionsbestimmung per Regressionsanalyse
  - Fitting domänenspezifischer Funktionen (x, x^2, e^x, $\log x$, ...) für jeden Parameter
  - Auswahl der Funktion mit niedrigster mittlerer MSD



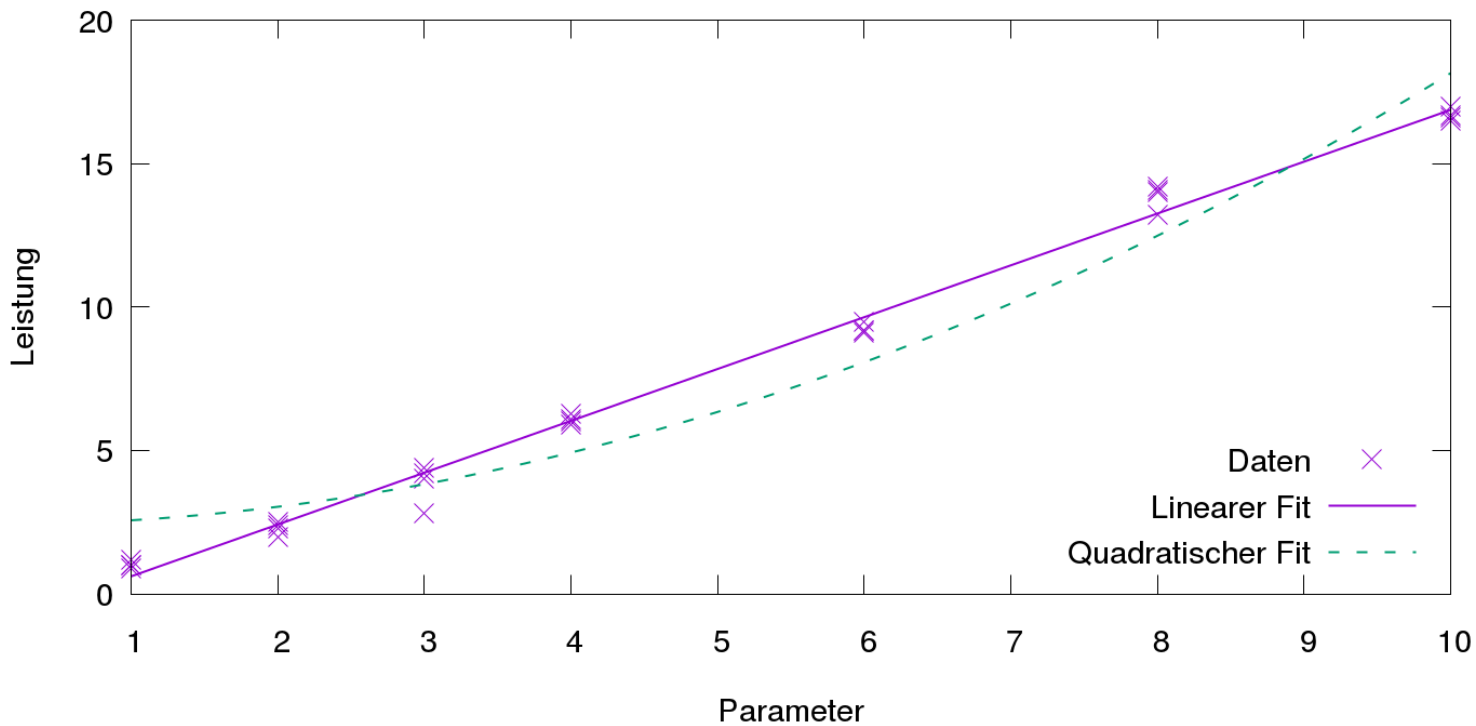
Modellierung

- Parameter-unabhängige Eigenschaften: Median
- Sonst: Funktionsbestimmung per Regressionsanalyse
  - Fitting domänenspezifischer Funktionen (x, x^2, e^x, $\log x$, ...) für jeden Parameter
  - Auswahl der Funktion mit niedrigster mittlerer MSD



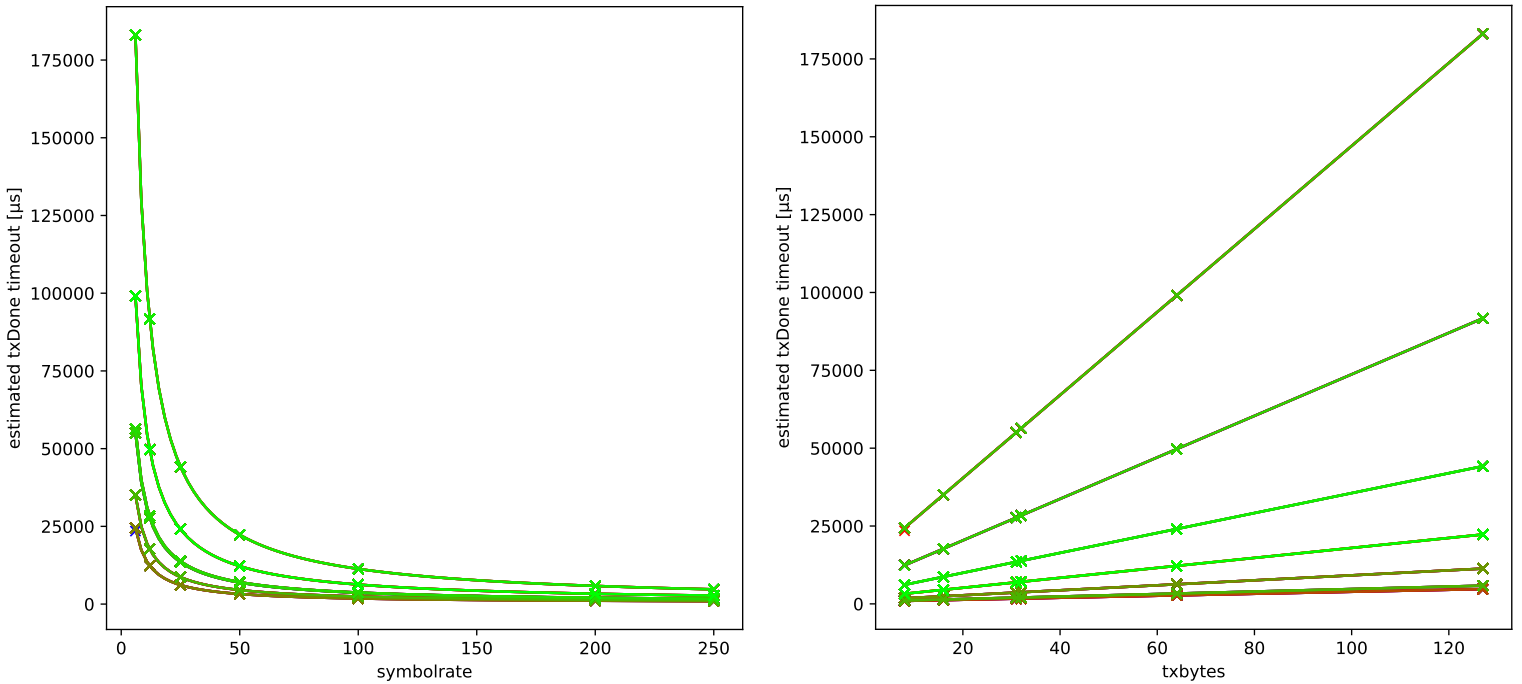
Modellierung

- Parameter-unabhängige Eigenschaften: Median
- Sonst: Funktionsbestimmung per Regressionsanalyse
  - Fitting domänenspezifischer Funktionen (x, x^2, e^x, $\log x$, ...) für jeden Parameter
  - Auswahl der Funktion mit niedrigster mittlerer MSD



Inhalt

Ergebnisse: Übertragungsdauer



$$f_t(dr,\ pl) = (366 + \frac{80102}{dr} + 8000\frac{pl}{dr})$$

Ergebnisse: Modellgüte

Eigenschaft	Bitrate	Länge	Sendeleistung	Abweichung	Median	Minimum
CC1200 TX power	✓	✓	✓	0.8 %	12 %	0.2 %
CC1200 TX time	✓	✓	–	0.1 %	87 %	0.1 %
CC1200 RX power	✓	–	–	<0.1 %	0.3 %	<0.1 %
nRF24 TX power	✓	–	✓	1.2 %	34 %	<0.1 %
nRF24 TX time	✓	–	–	<0.1 %	16 %	<0.1 %
nRF24 RX power	✓	–	–	<0.1 %	2.2 %	<0.1 %

- Nach Monte Carlo-Kreuzvalidierung ($\frac{2}{3}$ Training, $\frac{1}{3}$ Validierung)

Ergebnisse: Modellgüte

Eigenschaft	Bitrate	Länge	Sendeleistung	Abweichung	Median	Minimum
CC1200 TX power	✓	✓	✓	0.8 %	12 %	0.2 %
CC1200 TX time	✓	✓	–	0.1 %	87 %	0.1 %
CC1200 RX power	✓	–	–	<0.1 %	0.3 %	<0.1 %
nRF24 TX power	✓	–	✓	1.2 %	34 %	<0.1 %
nRF24 TX time	✓	–	–	<0.1 %	16 %	<0.1 %
nRF24 RX power	✓	–	–	<0.1 %	2.2 %	<0.1 %

- Nach Monte Carlo-Kreuzvalidierung ($\frac{2}{3}$ Training, $\frac{1}{3}$ Validierung)
- Test mit synthetischem Verbraucher:
  - Disjunkte Parameter in Trainings- und Validierungsmenge
  - 100 % Erkennung der Sollfunktion (Abweichung $\leq 0.7\,\%$) bei ≥ 7 Parameterwerten
  - $\geq 90\,\%$ (Abweichung $\leq 1.4\,\%$) sonst

Zusammenfassung

- Automatisiertes Verfahren: Treiber + Automat $\rightarrow$ Energiemodell
  - Benchmark-Erstellung
  - Accounting-Code im Treiber
  - Auswertung der Messdaten

Zusammenfassung

- Automatisiertes Verfahren: Treiber + Automat $\rightarrow$ Energiemodell
  - Benchmark-Erstellung
  - Accounting-Code im Treiber
  - Auswertung der Messdaten
- Hohe Modellgüte, kaum manuelle Intervention notwendig

Zusammenfassung

- Automatisiertes Verfahren: Treiber + Automat $\rightarrow$ Energiemodell
  - Benchmark-Erstellung
  - Accounting-Code im Treiber
  - Auswertung der Messdaten
- Hohe Modellgüte, kaum manuelle Intervention notwendig
- Ausblick: Berücksichtigung des Energiebedarfs der Treiber-Implementierung
  - Prüfsummenberechnung, Font-Rendering, ...
- Berücksichtigung des Modell-Overheads
  - Z.B. abhängig von Fließkomma-Unterstützung der Hardware

Literatur I

[McC+11] John C McCullough u. a. “Evaluating the effectiveness of model-based power characterization”. In: USENIX Annual Technical Conf. Bd. 20. 2011.