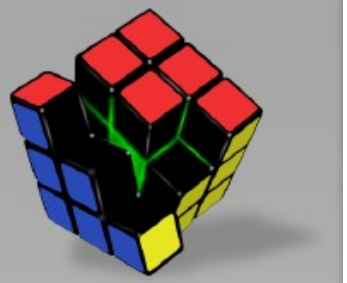


technische universität
dortmund

Power Management

Energiesparen in eingebetteten Betriebssystemen

Michael Müller
vorname.nachname@cs.tu-dortmund.de

AG Eingebettete Systemsoftware
Informatik 12, TU Dortmund

Inhalt

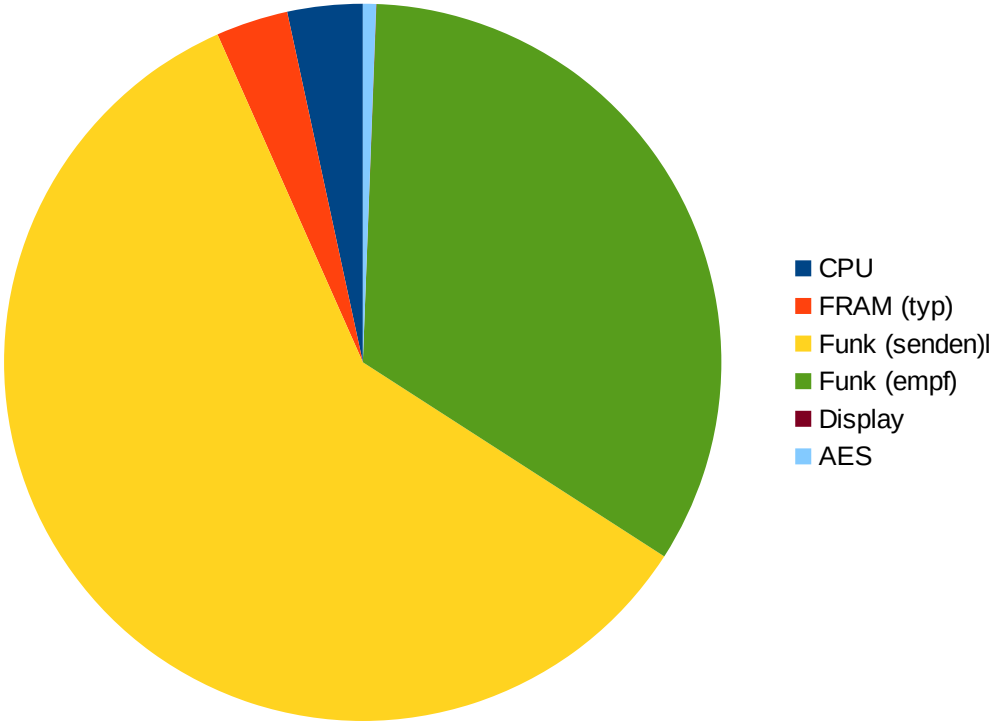
- Einleitung – Warum Energie sparen?
- Elektrotechnische Grundlagen
- Energiesparen per DVS
- Peripherie und Power-Management
- Energiemodelle in Betriebssystem
- Zusammenfassung

Inhalt

- Einleitung – Warum Energie sparen?
- Elektrotechnische Grundlagen
- Energiesparen per DVS
- Peripherie und Power-Management
- Energiemodelle in Betriebssystem
- Zusammenfassung

Energieverbrauch

Beispiel MSP430FR59xx mit TI CC1200 und Low Power Display



Modul	Stromaufnahme
CPU	1920 µA
FRAM	1845 µA[1]
Funken (Senden)	33600 µA
Funken (Empfangen)	19000 µA
Display	6 µA
AES	336 µA

Quelle: www.ti.com, Datasheet MSP430FR59xx, [2]
[1] typischer Verbrauch bei Ablauf eines Programms

Energieverbrauch

Beispiel MSP430FR59xx mit TI CC1200 und Low Power Display

Modul	Stromaufnahme
CPU	1920 µA
	[1]
(Empfangen)	
Display	6 µA
AES	336 µA

Funken sehr teuer! Aber CPU bereits auf Platz 2
→ Fazit: Nur **selten** Funken und CPU **sparsam** verwenden → Aber wie?

Quelle: www.ti.com, Datasheet MSP430FR59xx, [2]
[1] typischer Verbrauch bei Ablauf eines Programms

Inhalt

- Einleitung – Warum Energie sparen?
- Elektrotechnische Grundlagen
- Energiesparen per DVS
- Peripherie und Power-Management
- Energiemodelle in Betriebssystem
- Zusammenfassung

Wodurch enstehen die Verluste?

- Umwandlung von Energie in Wärme
- Verursacht durch Stromfluss in der Schaltung
- Physikalischer Zusammenhang zwischen Stromfluss und Energie
  - $E = P \circ \Delta t$ (Energie = Leistung ∘ Zeit)
  - $P = U \circ I$ (Leistung = Spannung ∘ Strom)

CMOS-Technologie

- CMOS heute dominierende Technologie für CPUs und Speicher
- Basiert auf komplementär verschalteten MOSFETs
- Stromfluss nur beim Schalten

Leistung einer CMOS-Schaltung: $P = C \circ U^2 \circ f + I_L \circ U$

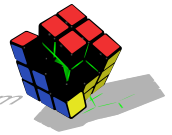
C	Schaltkapazität
U	Versorgungsspannung
f	Schaltfrequenz
I_L	Leckstrom (in der Regel vernachlässigbar)

CMOS-Technologie

- CMOS heute dominierende Technologie für CPUs und Speicher
- Basiert auf komplementär verschalteten MOSFETs
-

Fazit: Größtes Energiesparpontenzial durch Reduktion der Betriebsspannug
Aber Reduktion auch der Taktrate (Frequenz) nötig

C	Schaltkapazität
U	Versorgungsspannung
f	Schaltfrequenz
I_L	Leckstrom (in der Regel vernachlässigbar)



Techniken – Überblick

- Dynamic Voltage and Frequency Scaling (DVFS)
  - Dynamische Anpassung von Taktfrequenz und Spannung um Energie zu sparen
  - Setzt geschicktes Scheduling vorraus
- Clock-Gating
  - Abschalten des Taktes für CPU, Peripherie durch eine zusätzliche Hardwarekomponente (Gate)
  - Softwareumsetzung meist durch Sleep Modes
- Sleep Modes
  - Modi zum Abschalten bestimmter Komponenten
  - Werden durch Software-Instruktionen ausgewählt

Sleep Modes

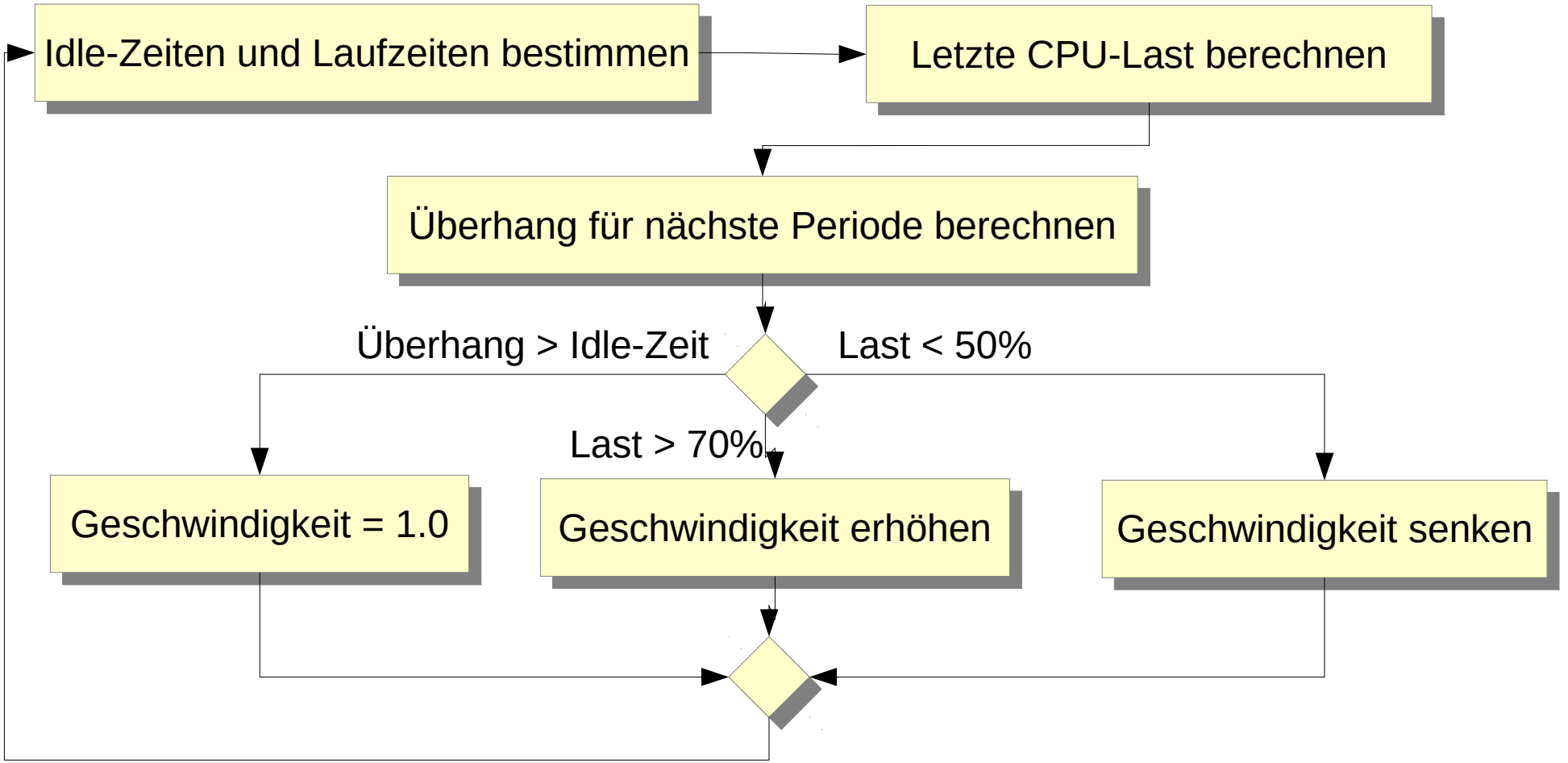
- Dienen dazu bestimmte Peripheriegeräte abzuschalten
- Modus bestimmt welche Geräte abgeschaltet werden
- Deutliches Energiesparpotenzial, z.B. MSP430

Modus	Eigenschaften	Stromaufnahme	Aufwachzeit
AM	Aktivmodus	~1420 µA	–
LPM0	CPU und MCLK angehalten	~225 µA	0 µs
LPM1	FRAM aus, DCO wenn nicht nötig	~180 µA	6 µs
LPM2	SMCLK, DCO aus	~0,9 µA	7 µs
LPM3	VCC für DCO aus	~0.6 µA	7 µs
LPM4	ACLK aus	~0.5 µA	7 µs
LPM3.5	Nur RTC arbeitet	~0.45 µA	250 µs
LPM4.5	Alle CLKs und Komponenten aus	~0.2 µA	250 µs

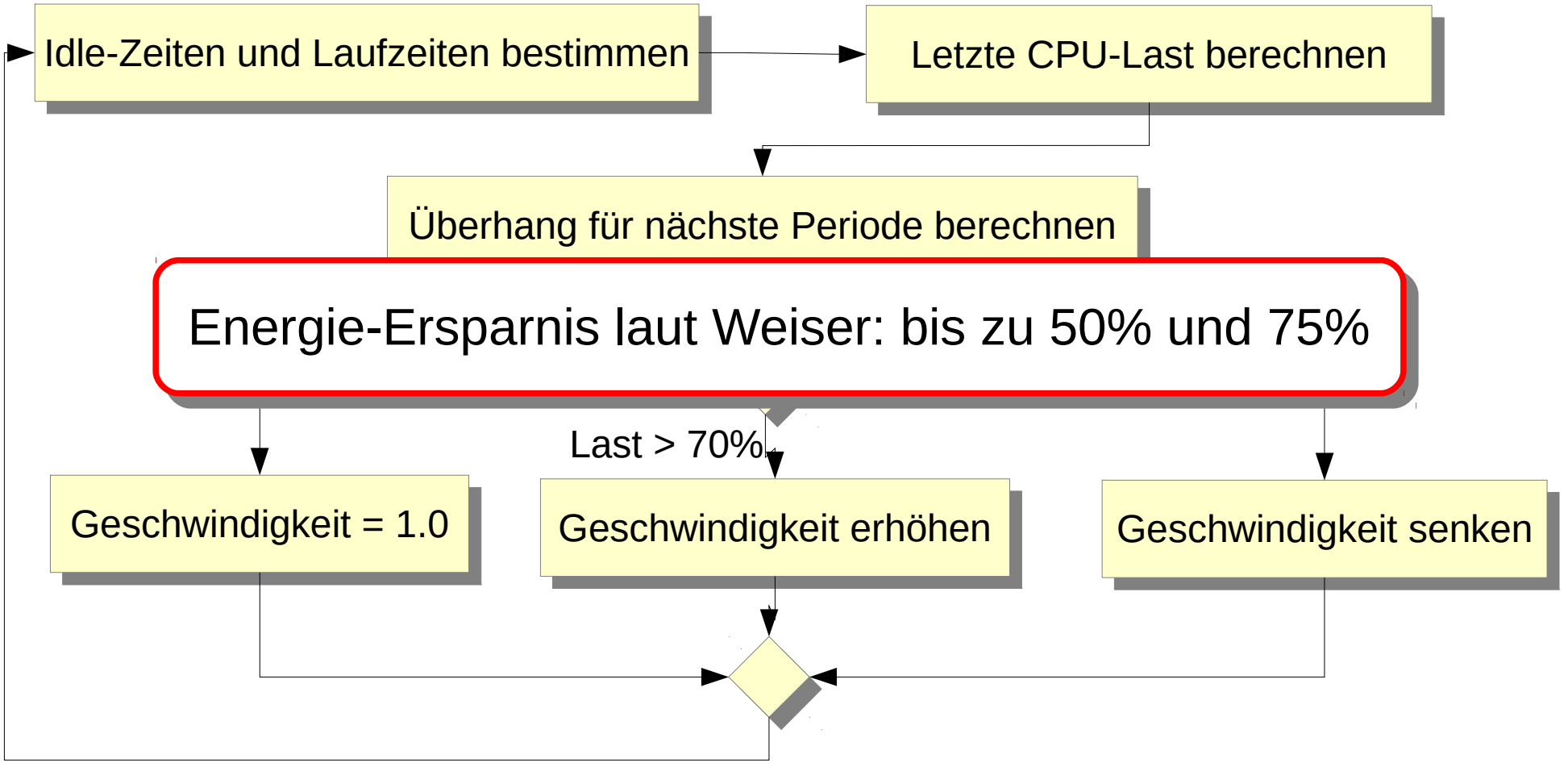
Inhalt

- Einleitung – Warum Energie sparen?
- Elektrotechnische Grundlagen
- Energiesparen per DVS
- Peripherie und Power-Management
- Energiemodelle in Betriebssystemen
- Zusammenfassung

Einfache Heuristik: PAST [7]



Einfache Heuristik: PAST [7]



DVS mit Echtzeit

- In Echtzeitsystem max. Laufzeit meist bekannt (WCET)
- Verwendung des Schlupfs um Laufzeit bei Reduktion der Frequenz auszudehnen
- Bestimmung der Durchführbarkeit eines Schedules mit Formeln (z.B. bei EDF) möglich

Inhalt

- Einleitung – Warum Energie sparen?
- Elektrotechnische Grundlagen
- Energiesparen per DVS
- Peripherie und Power-Management
- Energiemodelle in Betriebssystemen
- Zusammenfassung

Peripherie: Anwendung bisher

```
Every 5 minutes:
  Turn on SPI bus
  Turn on flash chip
  Turn on voltage reference
  Turn on I2C bus
  Log prior readings
  Start humidity sample
  Wait 5ms for log
  Turn off flash chip
  Turn off SPI bus
  Wait 12ms for vref
  Turn on ADC
  Start total solar sample
  Wait 2ms for total solar
  Start photo active sample
  Wait 2ms for photo active
  Turn off ADC
  Turn off voltage reference
  Wait 34ms for humidity
  Start temperature sample
  Wait 220ms for temperature
  Turn off I2C bus
```

- Abschalten aller nicht benötigten Peripherie-einheiten
- Im tiefstmöglichen Schlafmodus auf Messergebnis warten
- Die eigentliche Funktion des Programms ohne *Power Management*

Quelle: SuS Folien 3.8 PowerManagement

Peripherie: Beispiel TinyOS

- Verlagerung der Energieverwaltung in Treiber
- Kennt drei Arten von Geräten
  - Dedicated Devices (Low-Level etwa GPIO-Pin, ADC)
  - Shared Devices (z.B. SPI, I^2C)
  - Virtualized Devices (komplexe Geräte, wie Funkmodul)

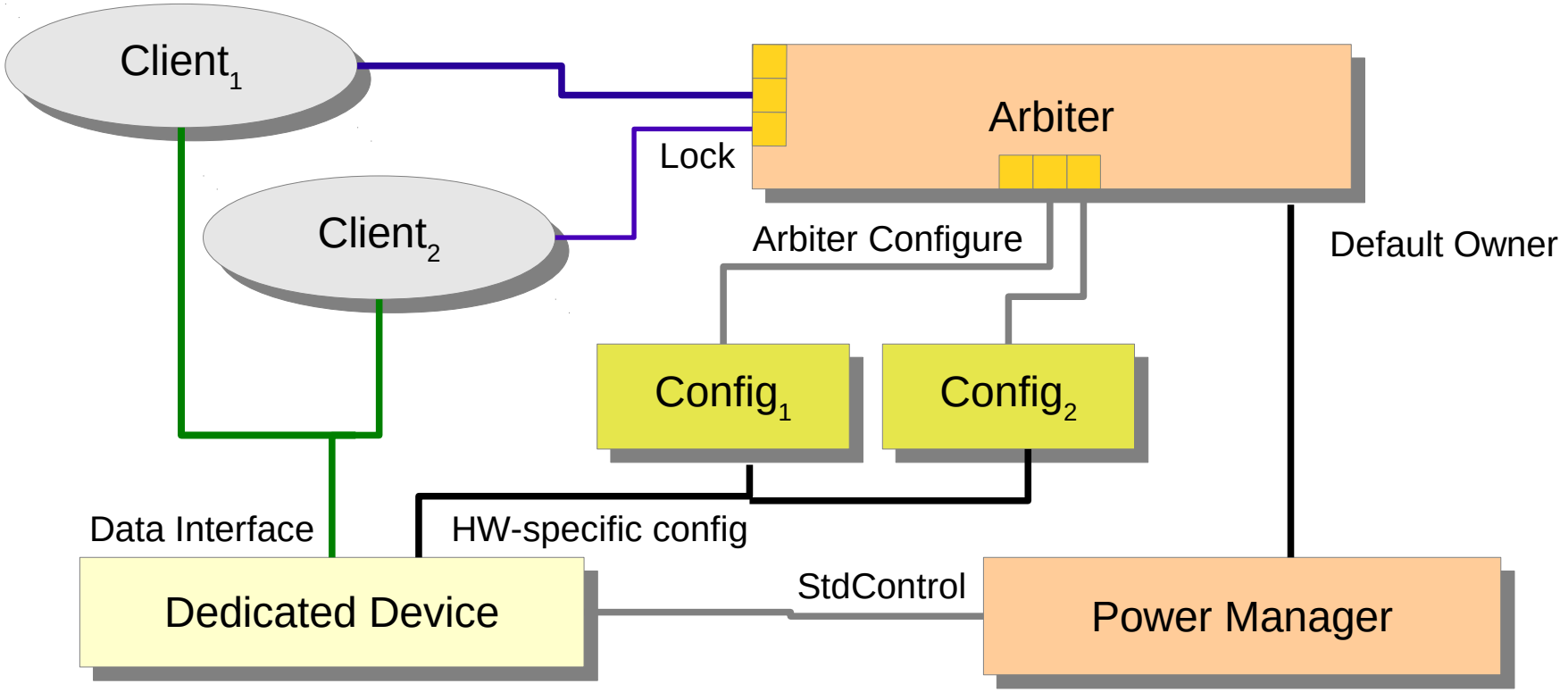
TinyOS – Dedicated Devices

- Repräsentieren Lowlevel-Devices (GPIO-Pins, Counter, Timer, ...)
- Erlauben nur einen Klienten den Zugriff
- Explizite Steuerung des Energieverbrauchs
- Standardschnittstellen zum Ein- und Ausschalten
  - StdControl, AsyncStdControl, SplitControl

```
interface SplitControl {
    command error_t start();
    event void startDone(error_t error);
    command error_t stop();
    event void stopDone(error_t error);
  }
```

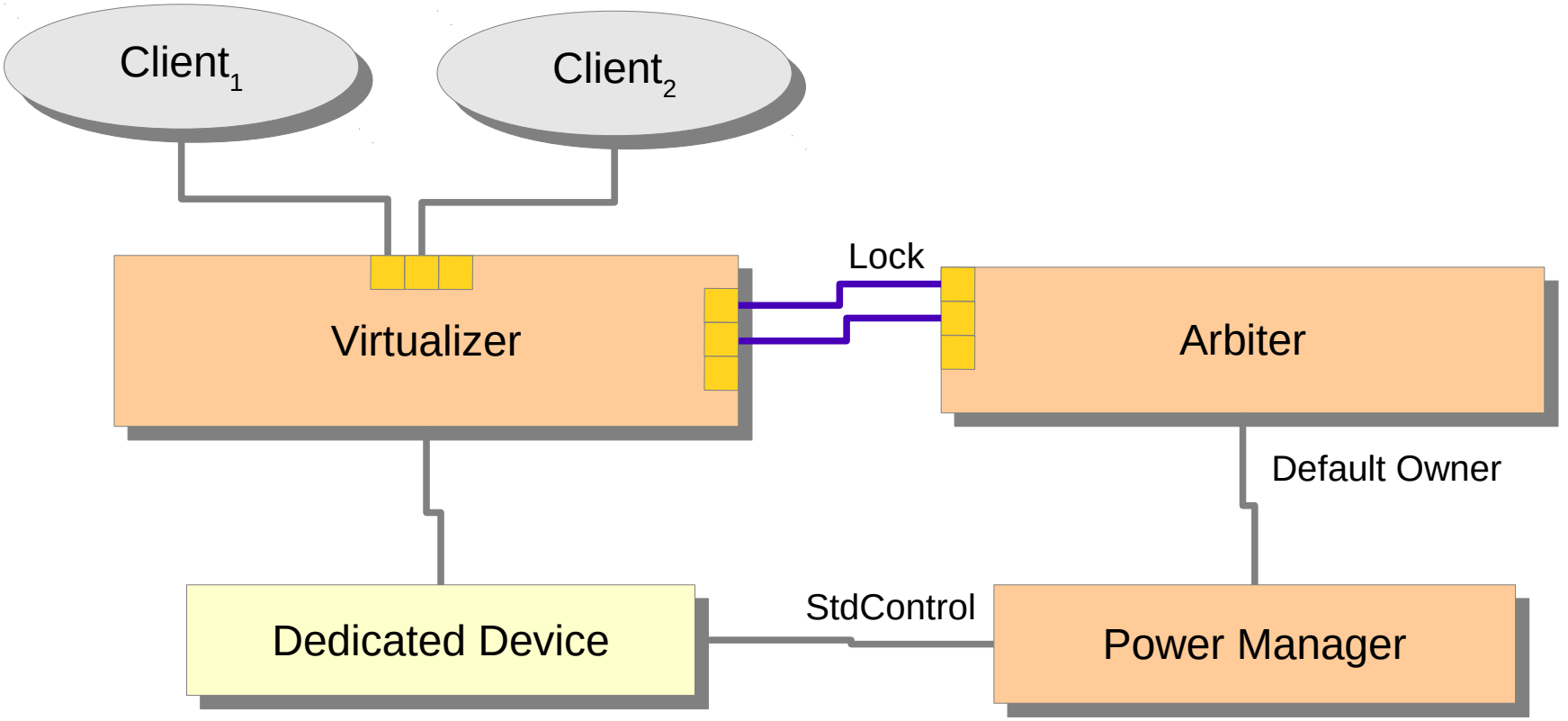
TinyOS – Shared Devices

- Implizites Power Management
- Gestattet mehreren Komponenten Zugriff auf ein Gerät
  - Üblicherweise geteilte Geräte (z.B. Busse, wie I^2C, SPI, UART)



TinyOS – Virtualized Devices

- Stellen einfache funktionale Schnittstelle bereit (z.B. Paket senden/empfangen)
- Erlauben (implizit) mehrere Nutzer



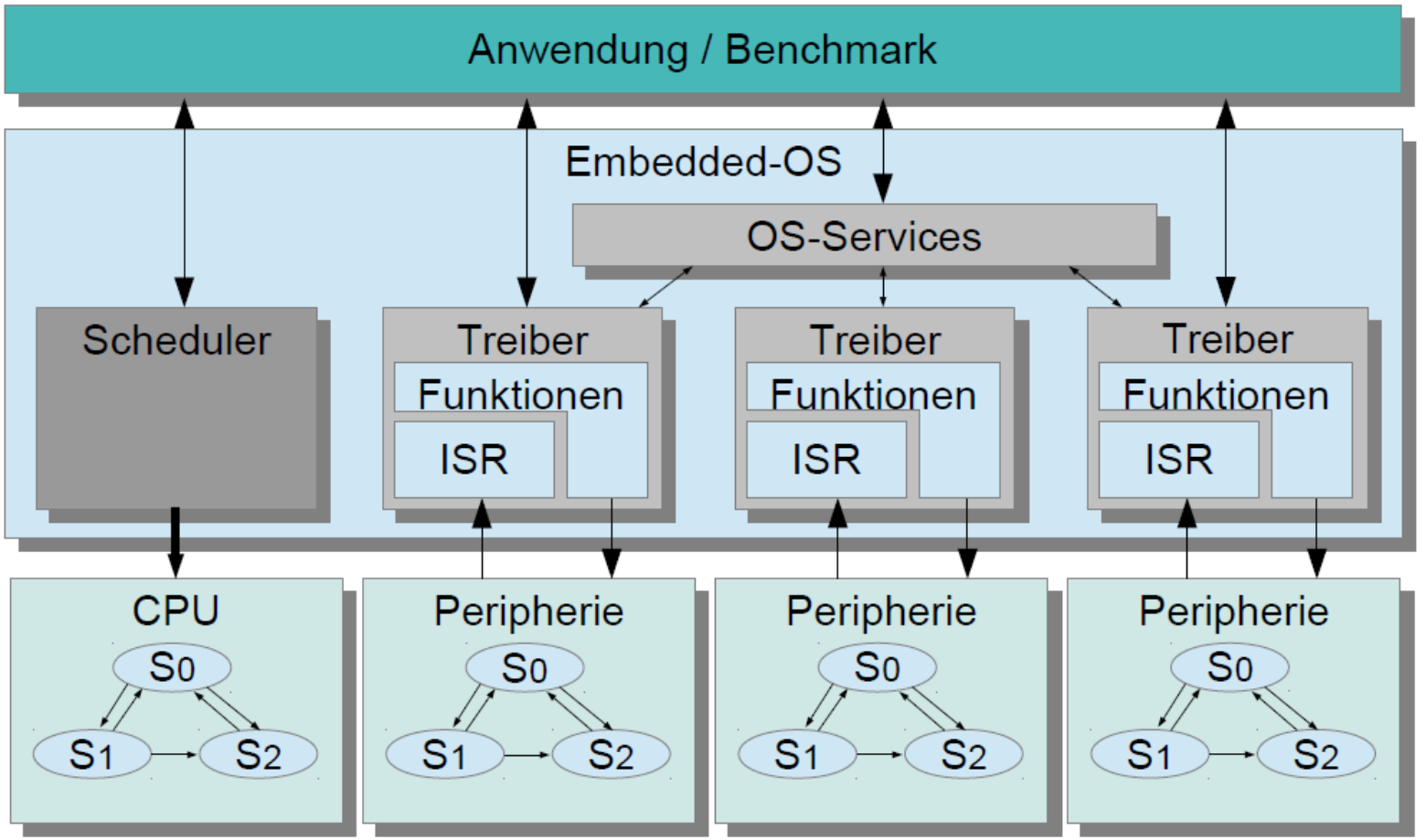
TinyOS

- Verwendung von asynchroner E/A-Schnittstelle
  - Erlaubt parallele Ausführung von E/A-Aktionen
- Protokollierung von E/A-Aufträgen erlaubt “Blick in die Zukunft”
  - Damit effiziente Ansteuerung von Geräten möglich
- 98% des Optimums erreichbar

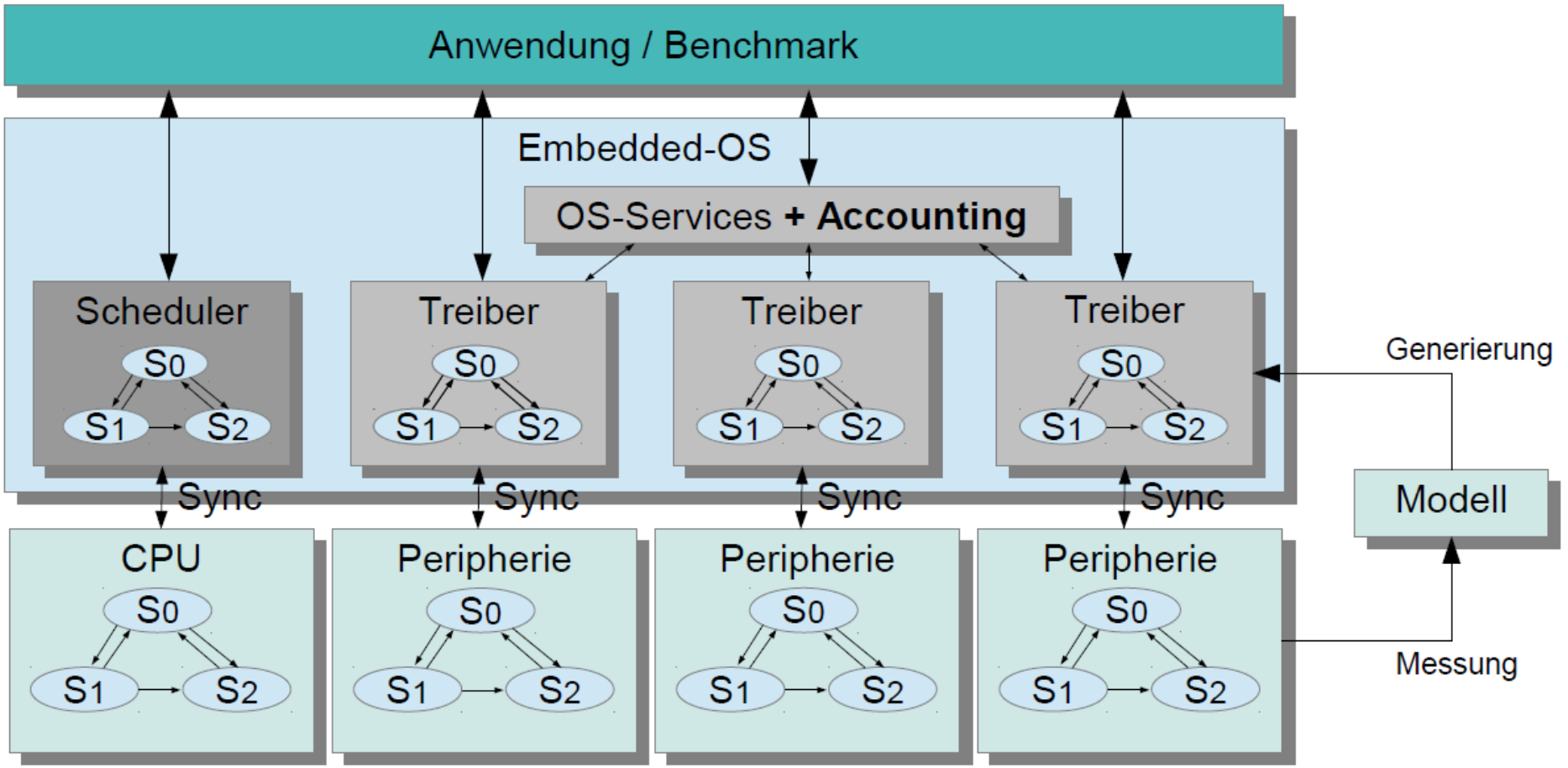
Inhalt

- Einleitung – Warum Energie sparen?
- Elektrotechnische Grundlagen
- Energiesparen per DVS
- Peripherie und Power-Management
- Energiemodelle in Betriebssystemen
- Zusammenfassung

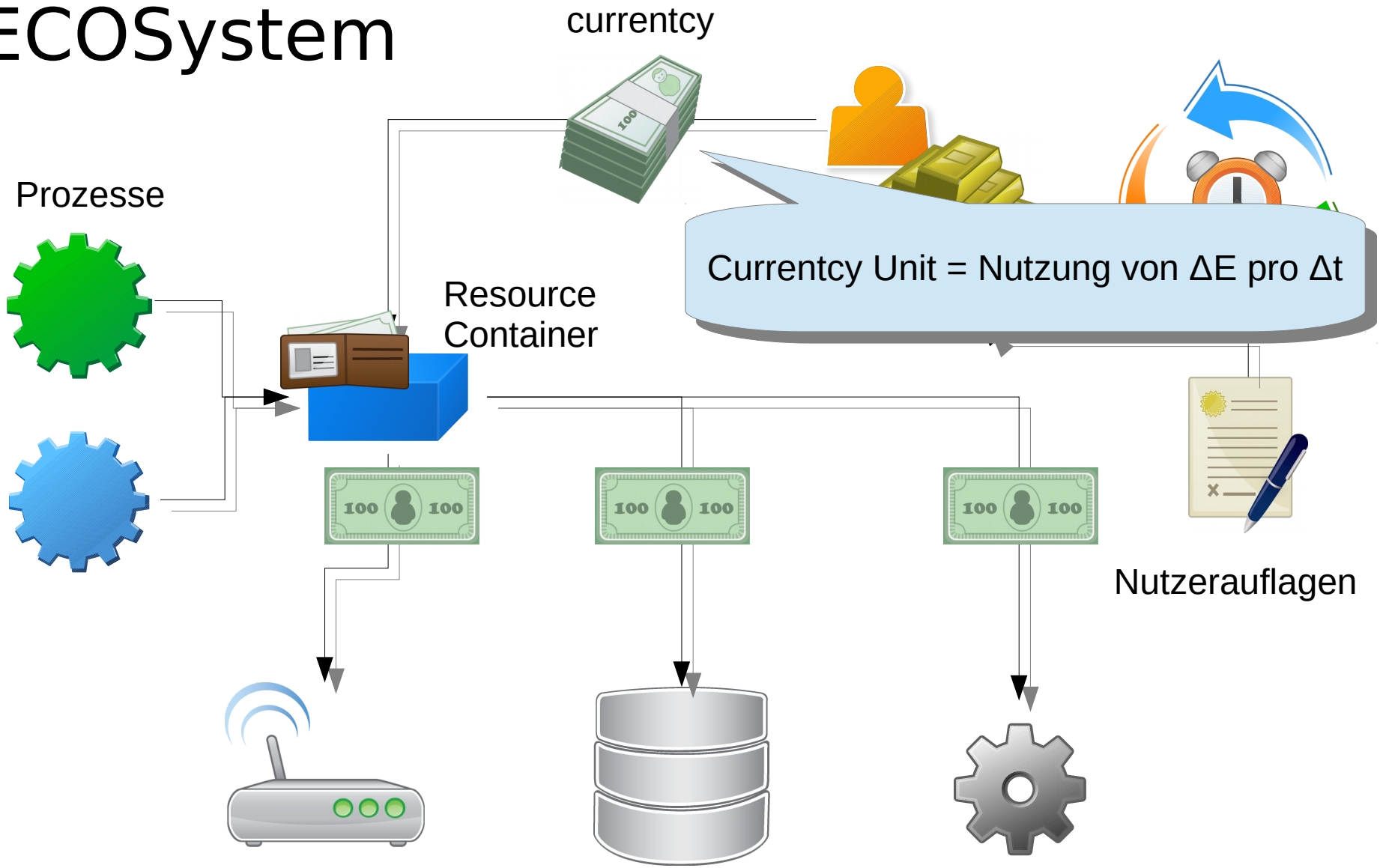
Herkömmliches Betriebssystem



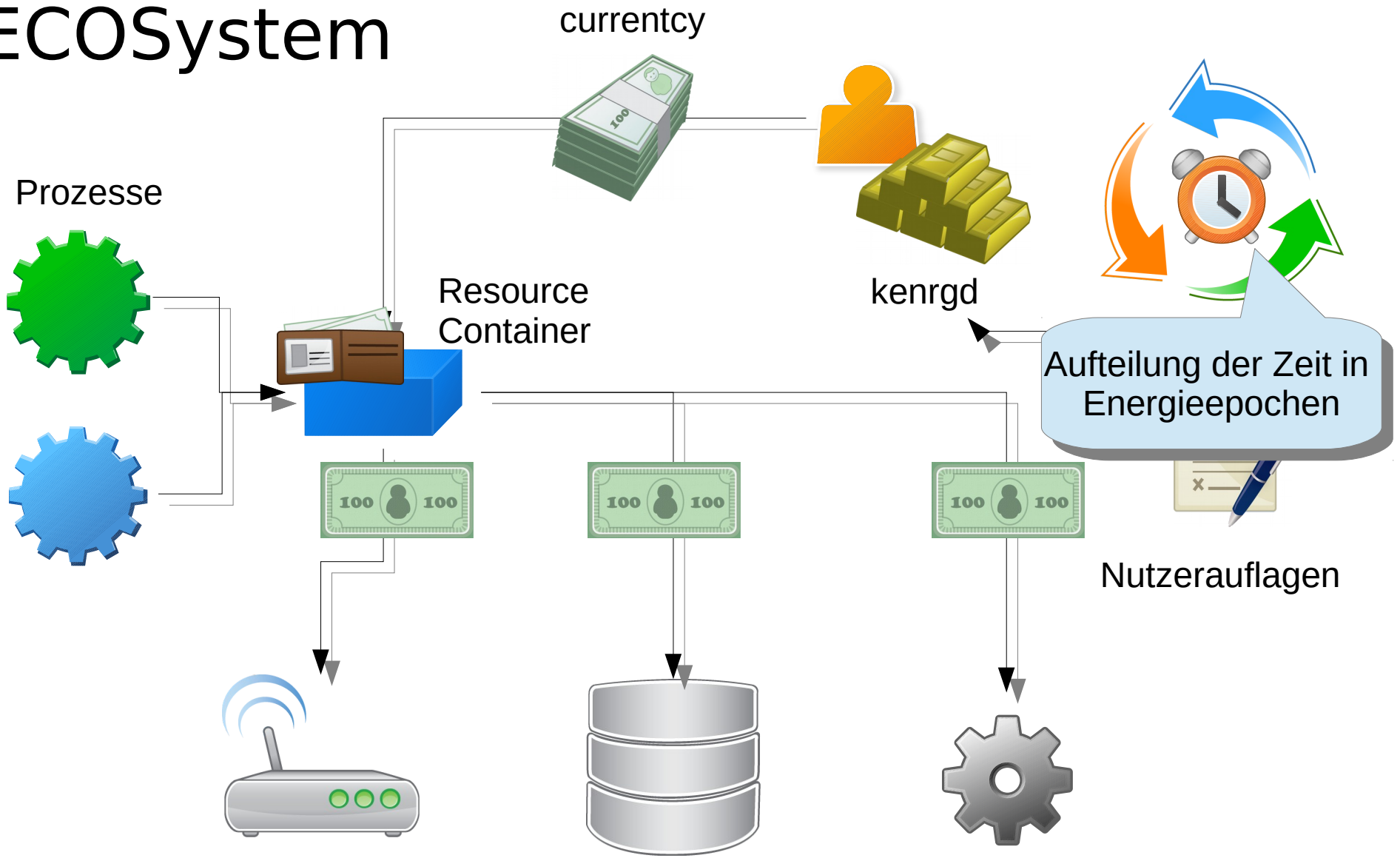
Energiegewahres Betriebssystem



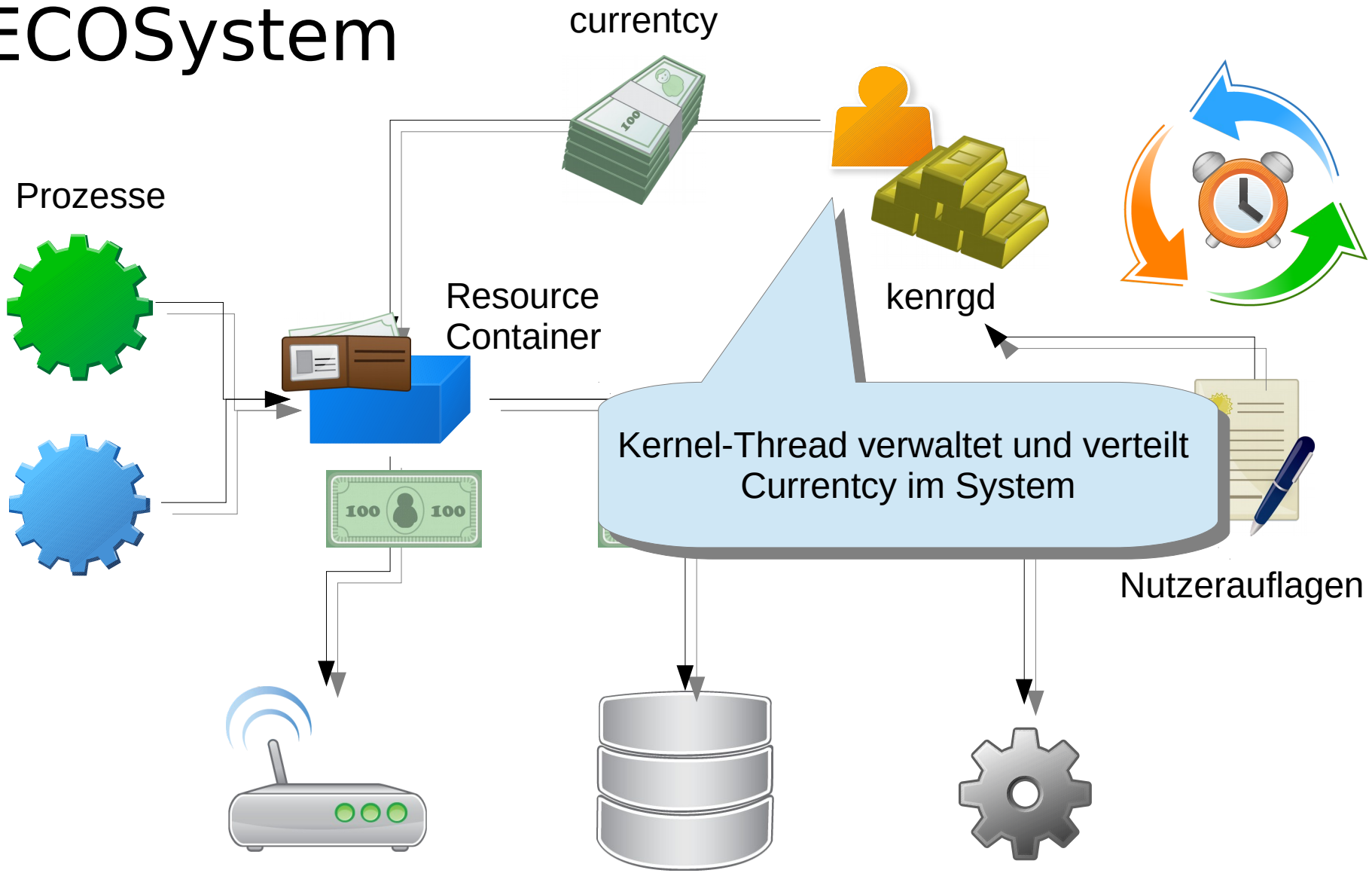
ECOSystem


currentcy
Prozesse
Resource Container
Currentcy Unit = Nutzung von ΔE pro Δt
Nutzerauflagen

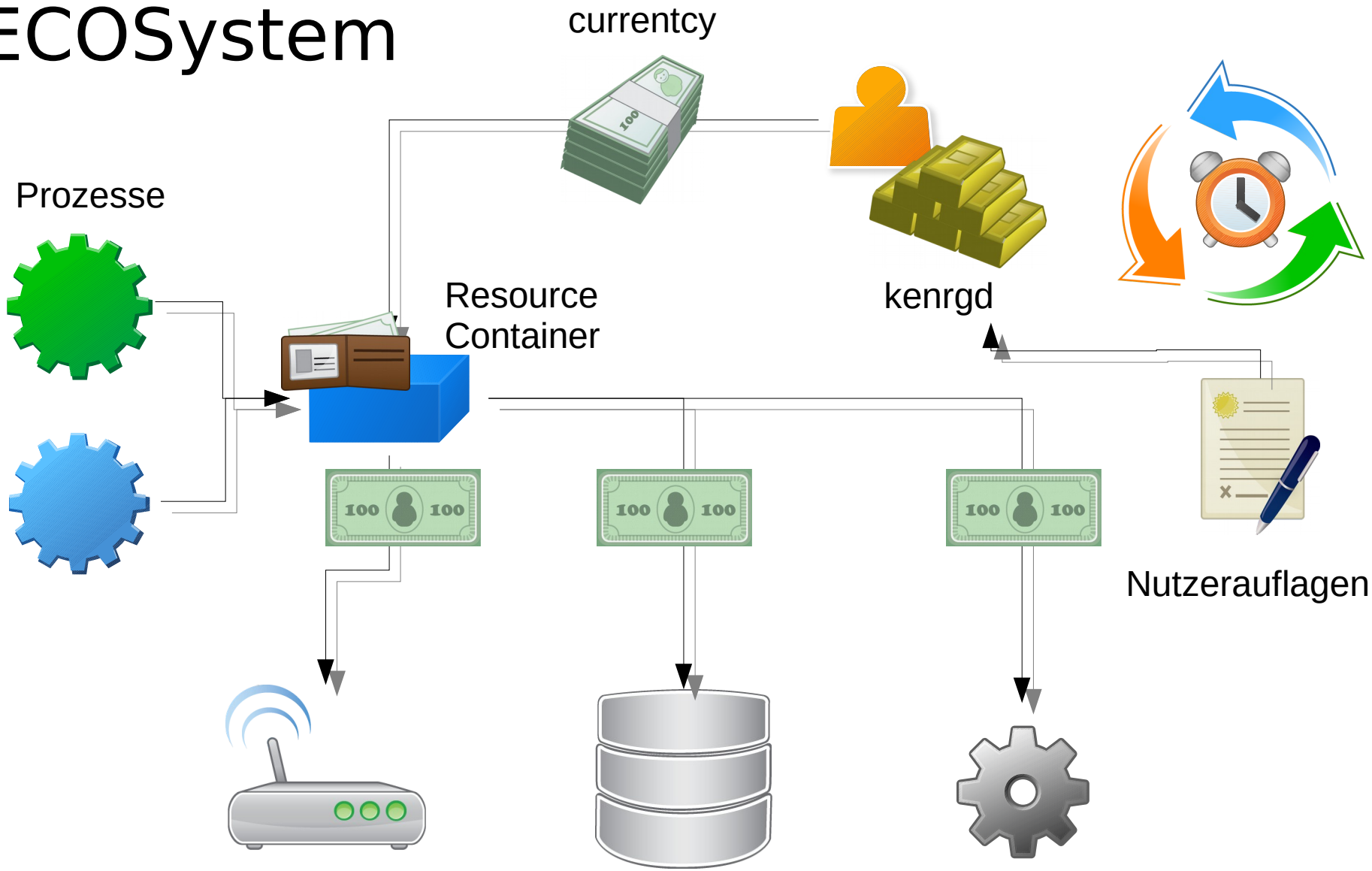
ECOSystem


currentcy
Prozesse
Resource Container
kenrgd
Aufteilung der Zeit in Energieepochen
Nutzerauflagen

ECOSystem


currentcy
Prozesse
Resource Container
kenrgd
Kernel-Thread verwaltet und verteilt Currentcy im System
Nutzerauflagen

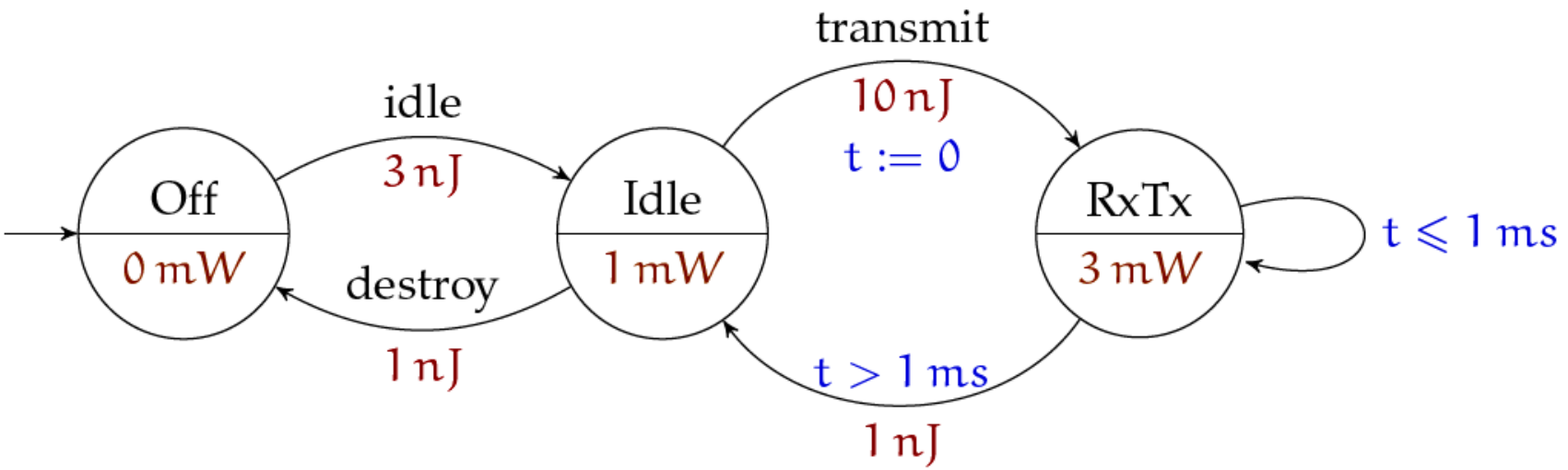
ECOSystem


currentcy
Prozesse
Resource Container
kenrgd
Nutzerauflagen

ECOSystem - Zusammenfassung

- Energie als First-Level-Resource im System
- Prozesse zahlen mit Energiewährung für Benutzung von Geräten
- Erreichen festgelegten Batterielaufzeit (mit max. Abweichung von 10%) möglich
  - Allerdings auf Kosten der Performance konkurriender Anwendungen

Energiemodell – PTAs



Quelle: [2]

- Modellierung des Geräts als Finite State Machine
- Transitionen zusätzlich gewichtet mit Zeitschranken und Energieverbrauch
- Zustände gewichtet mit Leistungsaufnahme

Zusammenfassung

- Power-Management wichtig insb. in eingebetteten Systemen
- Dynamic Voltage Scaling erlaubt Energie bei der Verwendung der CPU zu sparen
  - Setzt genaue Kenntnis über Verhalten der Prozesse vorraus
  - Meist nicht gegeben → Heuristiken und Hints nötig
- Erleichterung der Anwendungsentwicklung durch Verlagerung des Power-Managements in Gerätetreiber
- Energiemodelle erlauben das Accounting und Abschätzen von Energie durch das OS
  - Schwierig zu erstellen (messen), vereinfachen daher ungenau

Literatur

[1] Zeng, H., Ellis, C., Lebeck, A., Vahdat, A.: ECOSystem. ACM SIGPLAN Notices. 37, 123 (2002).

[2] Rapczynski, R., Entwurf eines energiegewahren Treibermodells für eingebettete Betriebssysteme, Masterarbeit, TU-Dortmund (2014).

[3] D. Grunwald, C.B. Morrey, P. Levis, M. Neufeld, and K.I. Farkas, *Policies for dynamic clock scheduling*. In Proceedings of the 4th Conference on Symposium on OSDI. USENIX Association, 2000.

[4] M. Weiser, B. Welch, A. Demers, and S. Shenker. *Scheduling for reduced CPU energy*. In Proceedings of the 1st USENIX Conference on OSDI (Monterey, California). Vol. 1. USENIX Association, 1994.

[5] K. Klues, V. Handziski, C. Lu, A. Wolisz, D. Culler, D. Gay, and P. Levis. *Integrating concurrency control and energy management in device drivers*. In Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles (SOSP '07). ACM, New York, pages 251-264, 2007.

Literatur

[6] D. Grunwald, C.B. Morrey, P. Levis, M. Neufeld, and K.I. Farkas, *Policies for dynamic clock scheduling*. In Proceedings of the 4th Conference on Symposium on OSDI. USENIX Association, 2000.

[7] M. Weiser, B. Welch, A. Demers, and S. Shenker. *Scheduling for reduced CPU energy*. In Proceedings of the 1st USENIX Conference on OSDI (Monterey, California). Vol. 1. USENIX Association, 1994.