Research Summary

6. Februar 2025

Lehrstuhl Informatik 4 - System Software

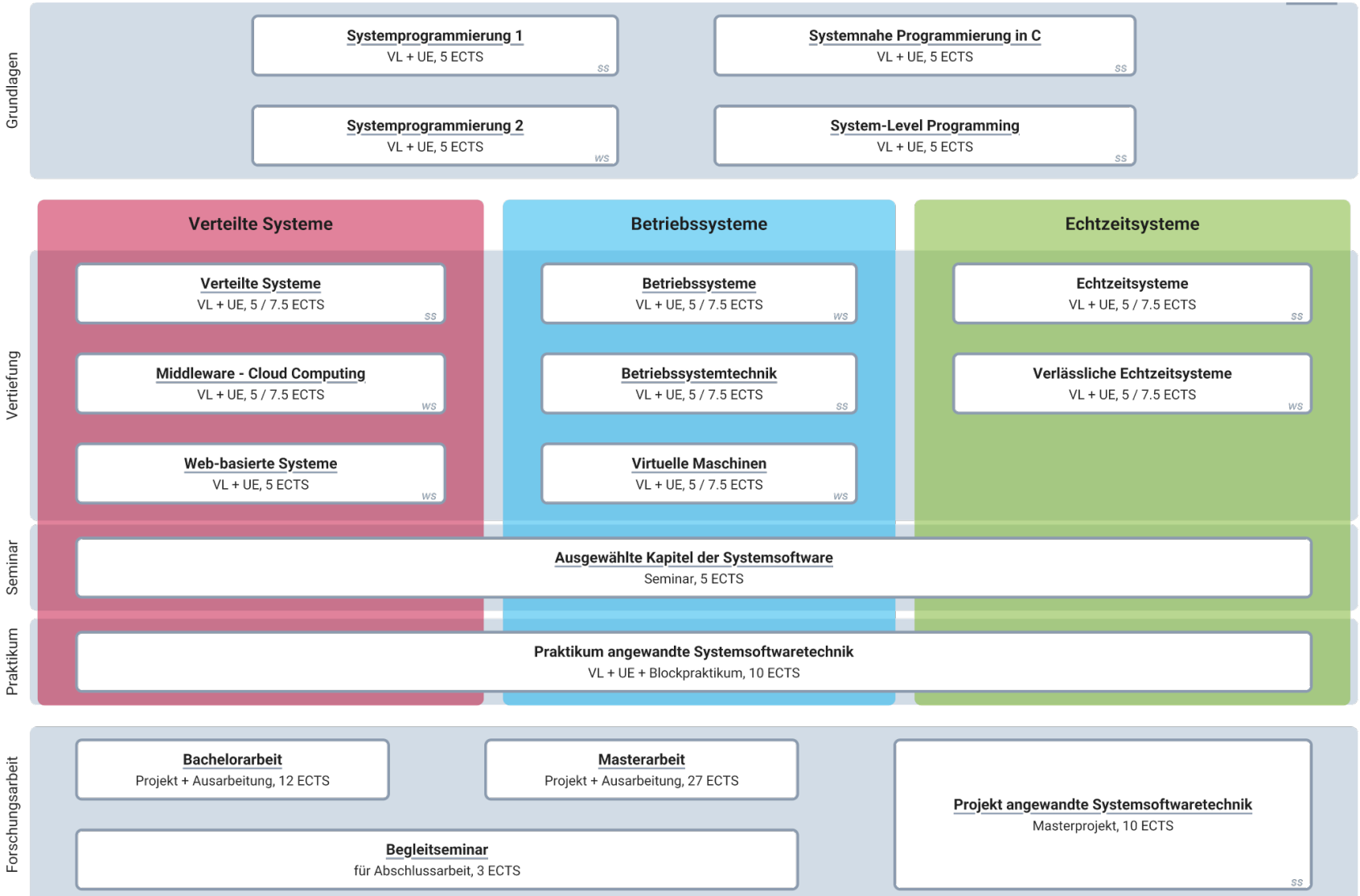
Friedrich-Alexander-Universität Erlangen-Nürnberg

Friedrich-Alexander-Universität
Technische Fakultät

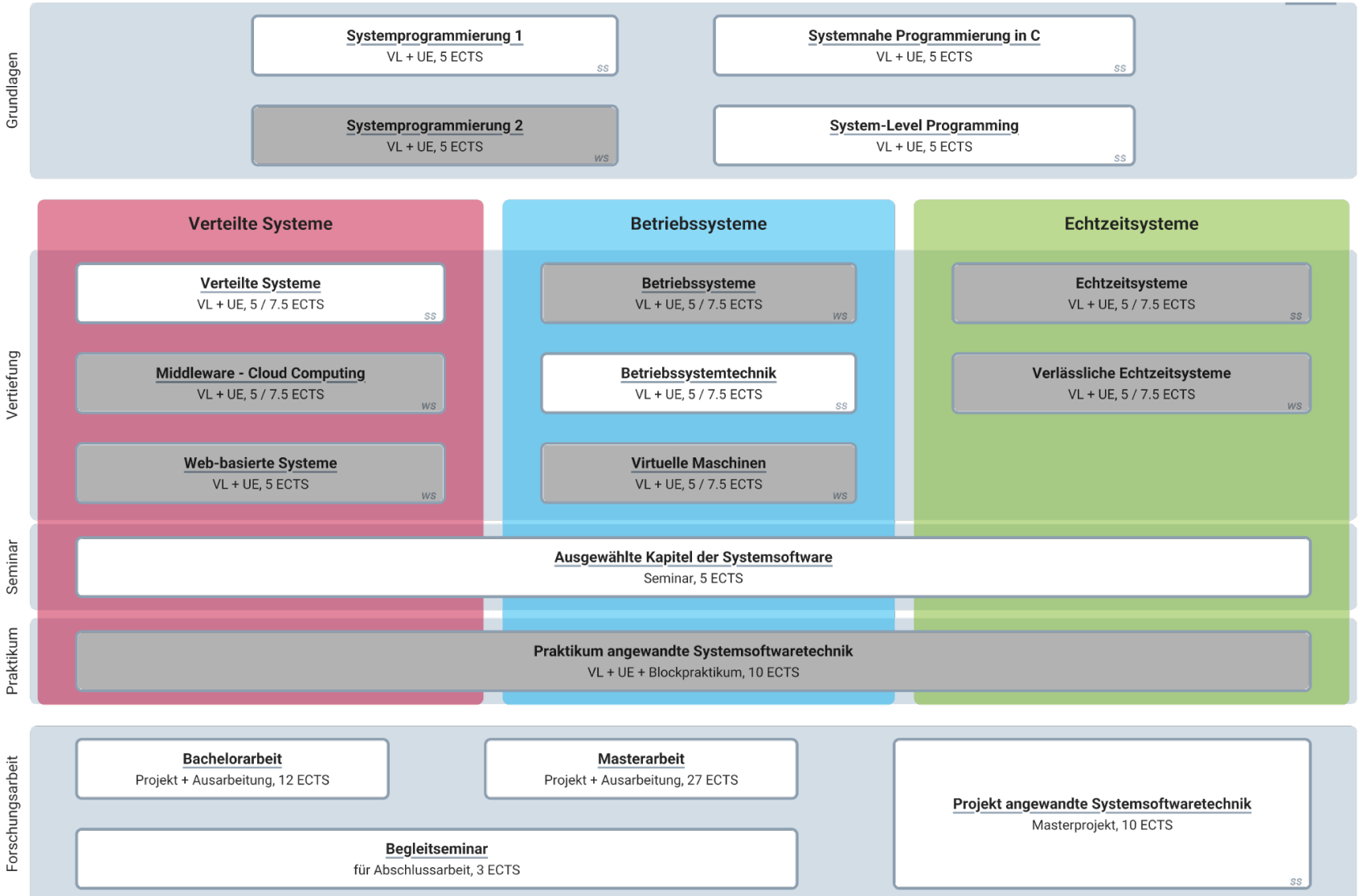
Lehrstuhl für Informatik 4
Systemsoftware

Lehre

Lehrangebot



Lehrangebot *im kommenden Sommersemester*



Replizierte Systeme und Blockchain-Systeme

- Grundlagen von replizierten Systemen (PBFT & Bitcoin)
- Byzantinische Fehlertoleranz: neue Forschungsergebnisse
- Moderne Blockchain-Systeme (Avalanche, Sui, Aptos, ..)

Interesse? Übersicht der Themen:

- `https://sys.cs.fau.de/lehre/ss25/akss`
- Reservierung: Email an berger@cs.fau.de

Externe Vorlesung: Applied Software Architecture (ASA)

Modulbeschreibung

- A software system is far more than just a collection of algorithms and data structures: building it means carefully designing and crafting an architecture that ensures that the system is capable of fulfilling all relevant requirements and needs.
- The lecture teaches students how to design, document, implement and evaluate the architecture of a complex software system.
- A focus is put on the practical applicability of the lecture's contents. The lecturers have a background in academia and diverse roles in industry, and are thus capable of bridging the gap between theory and practice.
- The contents of this module are aligned with the syllabus of the iSAQB Certified Professional for Software Architecture - Foundation Level programme.

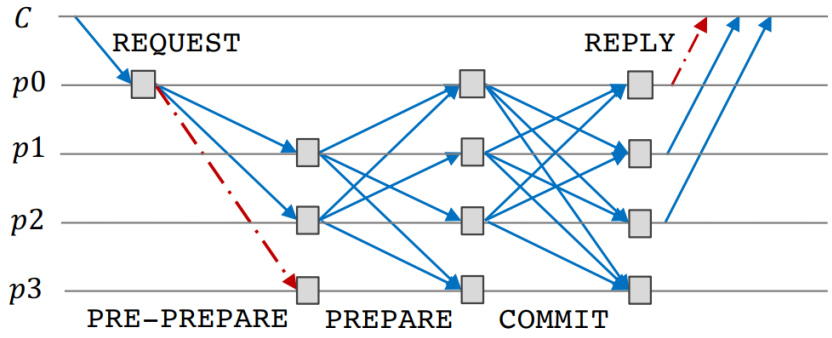
Abschlussarbeiten

Ein kurzer Ausflug zu den *verteilten Systemen*

Byzantinische Fehlertoleranz

Fuzzing *Byzantine Fault-Tolerant* (BFT) Protocols

BFT Protokollausführung

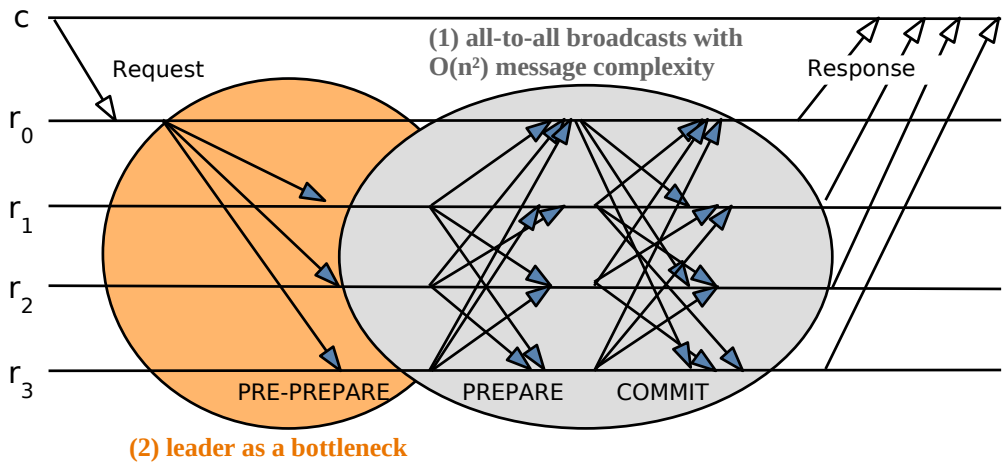


- *BFT Protokolle* dienen der Ausfallsicherheit von Systemen
  - .. sie sind aber sehr komplex!
- *Fuzzing* ist eine randomisierte Technik zur Fehlersuche
- **Thema:** Implementierung einer Fuzzing Methodik in einem modernen BFT Framework mit anschließender Evaluation

Interesse?
Für nähere Infos, Mail an: berger@cs.fau.de

Integration von Skalierungstechniken in ein BFT Framework

Existierende Flaschen-hälse im PBFT Protokoll



- *BFT Protokolle* werden oft in Blockchains eingesetzt
  - viele Replikate, $\rightarrow$ *Skalierbarkeit ein Kriterium*
- Ideen: Trennung von zwei Aspekten (Verbreitung von Daten $\leftrightarrow$ Agreement), Multi-Leader, Kryptographie
- **Thema:** Implementierung einer Auswahl von Ideen in ein modernes Rust-basiertes BFT Framework und Evaluation

Interesse?
Für nähere Infos, Mail an: berger@cs.fau.de

Kompartimentierung von Rust-Anwendungen

- Ihr kennt jetzt MMU-basierte Isolation
- $\exists$ weitere Isolationsmechanismen
- Kann die Sicherheit einer Anwendung davon profitieren?

[1] https://www.eetimes.com/wp-content/uploads/media-1314103-armchericompartmentalization.png

Kompartimentierung von Rust-Anwendungen

- Ihr kennt jetzt MMU-basierte Isolation
- $\exists$ weitere Isolationsmechanismen
- Kann die Sicherheit einer Anwendung davon profitieren?

$\rightarrow$ Ja, durch Kompartimentierung

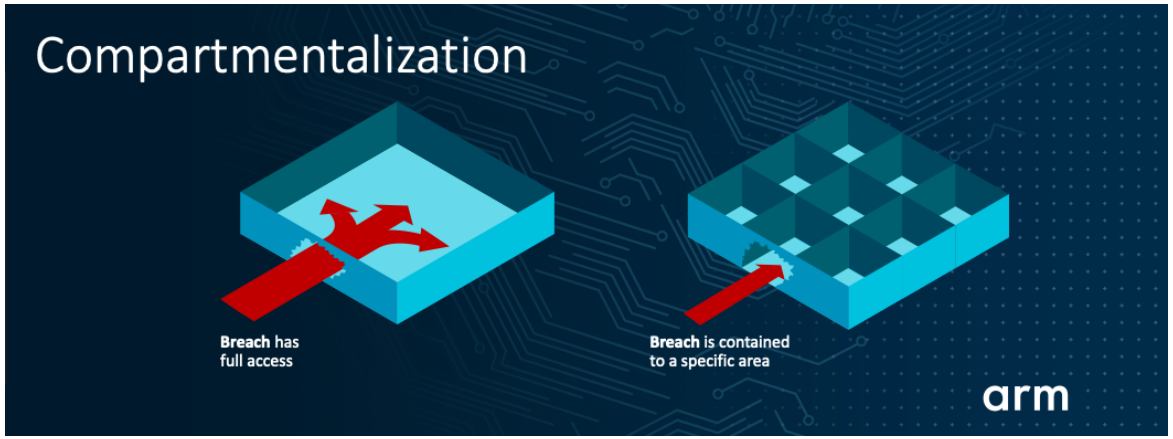


Abbildung 1: Kompartimentierung: Konzept[1]

[1] https://www.eetimes.com/wp-content/uploads/media-1314103-armchericompartmentalization.png

Mögliche Themenbereiche: Kompartimentierung in Rust

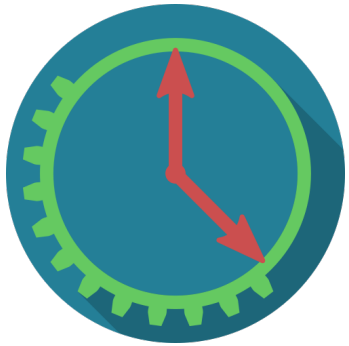
- Wie erhalten wir überhaupt die Semantik?
- = Wie fügen wir korrekt Fernaufrufe ein?

- Wo darf getrennt werden, wo nicht?
- = Welche Stellen sind unntrennbar? Synchronisation?

- Wie verhärten wir die Schnittstellen?
- = Und isolieren einen Einbruch vom Rest?

Interesse?
Für nähere Infos, Mail an: onciul@cs.fau.de

JITTY OS
Binärkompatibeles Unix-/Linux-ähnliches Forschungsbetriebssystem unseres Lehrstuhls

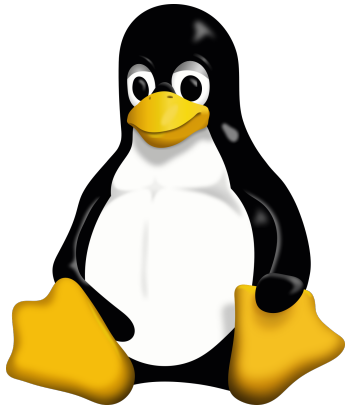


- Virtio: *Straightforward*, *Efficient*, *Standard* & *Extensible*
- Implementierung von paravirtualisierten Treiber für Netzwerk, Blockgeräte & Konsole
- Vergleich mit existierenden (emulierten) Gerätetreibern

Interesse?
Für nähere Infos, Mail an: ott@cs.fau.de

IOMMU-gewahrer Allokator

IOMMU $\approx$ MMU für DMA-fähige Geräte
Übersetzung von Speicherzugriffen auf dem PCI Bus mit Hilfe von **Tabellen**, **Caches**, **TLBs**



- Bisher: Fokus auf *Sicherheit* $\oplus$ *Performanz*
- Implementierung von *hybriden Ansatz* mit spez. Allokator
- Vergleich mit gegebenen Ansätzen in Linux und Literatur

Interesse?
Für nähere Infos, Mail an: ott@cs.fau.de

Lock ≠ Lock
Verschiedene Locks für verschiedene Anwendungszwecke:
Spinlock, Ticketlock, RWLock, SeqLock, Queued Spinlock, …

- Idee: Verwendung von Allzwecklocks (+ minimaler Attributierung)
- Protokollierung von Zugriffsmuster durch *JIT-Übersetzer*
- Austauschen der Implementierung zur Laufzeit

Interesse?
Für nähere Infos, Mail an: ott@cs.fau.de

NEON – Energy-Aware Operating Systems using Non-Volatile RAM

Aktueller Stand: Wir haben ein modifiziertes Linux, welches zur Laufzeit statt DRAM nichtflüchtigen Arbeitsspeicher verwendet
$\rightarrow$ Ständiges Sichern des Speicherinhalts ist nicht mehr nötig

Fragen:

- Können wir dadurch die Codebasis verkleinern (ohne Garantien zu verlieren)?
- Lässt sich dadurch die Performance verbessern?
- Sparen wir dadurch Energie?

Interesse?
Für nähere Infos, Mail an: preisner@cs.fau.de

Härtung von Rust-basierten Anwendungen

- Transiente Speicherfehler (Bitflips) können bspw. durch Strahlung ausgelöst werden
- Häufige Verwendung von Standardhardware in Satelliten(schwärmen) (bspw. Starlink).

$\Rightarrow$ Hohe Gefahr von transienten Speicherfehlern!

- **Lösungsansatz:** Härtung der Software durch Fehlertoleranzmechanismen (bspw. Prüfsummen)
- Software-Härtung ist für C/C++ gut verstanden aber neue Systemsoftware wird oft in Rust implementiert

- **Thema:** Wie können Rust-Anwendungen automatisch gegen transiente Speicherfehler gehärtet werden?

Interesse?

Fiona & Rüdiger, Mail an: ruediger.kapitza@fau.de

Sicherheitsgarantien für Nutzer von Webanwendungen

- HTTPS ermöglicht eine verschlüsselte Verbindung zu einem entfernten Webserver
- Wir können mit großer Sicherheit sagen, dass wir mit dem Eigentümer einer bestimmten Domain verbunden sind
- **Problem:** Ob der Webserver sicherer ist oder was mit unseren Daten passiert, bleibt offen
- **Confidential Computing (CC)** ermöglicht es, Webserver durch Hardware-Mechanismen zu schützen und aus der Ferne festzustellen, ob ein System als *sicher* gelten kann.

- **Thema:** Welche Sicherheitsgarantien könnten den Nutzern über den Browser gegeben werden und wie könnten diese durch den Einsatz von CC umgesetzt werden?

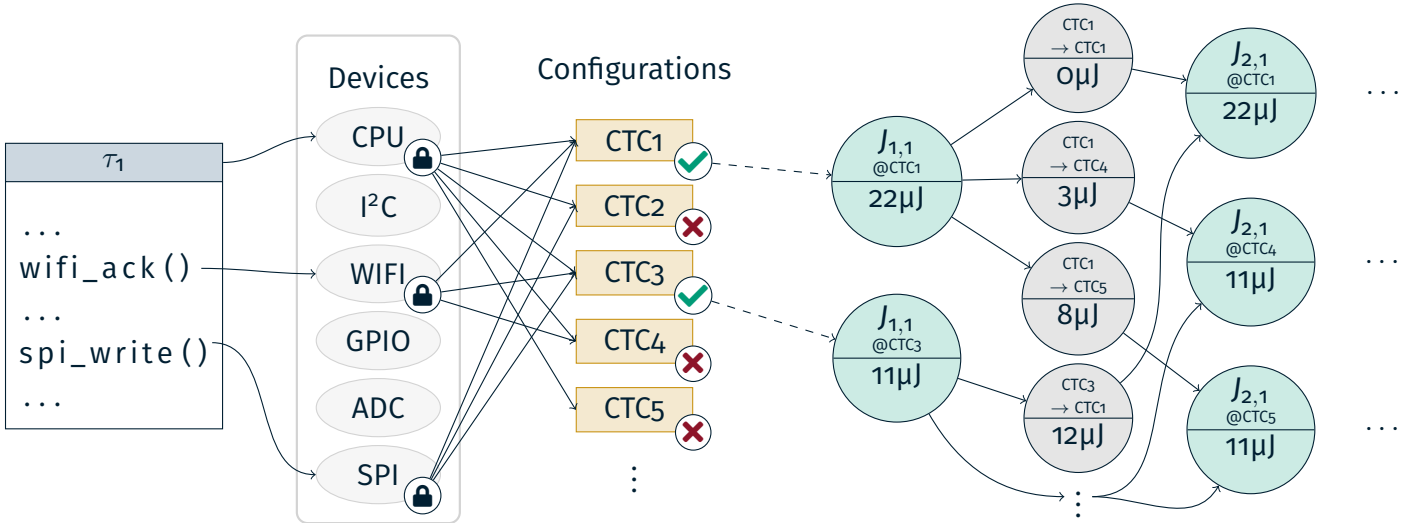
Interesse?

Mail an: ruediger.kapitza@fau.de

Task-Reordering for Embedded Systems

Energy-Constrained Real-Time Systems

- **Ziel**: mit Clock-Tree-Optimierungen Energiebedarf ↓
- **Bisher**: für feste Reihenfolge an Tasks
- **Frage**: Lassen sich Tasks umordnen, um Energie zu sparen?



Wissen zu mathematischer Optimierung/Algorithmik nötig!

Interesse an Echtzeitsystemen? Mail an dengler@cs.fau.de

Weitere Themen auf Anfrage

- **Thema:** Wie kann man JS im Browser **weitgehend** durch Rust ersetzen, indem man Webassembly verwendet?
- **Thema:** Wie kann man effizient Sicherungspunkte (Snapshots) für Confidential Virtual Machines (CVMs) (z.B. AMD SEV-SNP oder Intel TDX) ermöglichen?
- **Thema:** Entwicklung eines Frameworks für die flexible Erstellung von ganzheitlich validierbaren CVMs. Siehe: *"Trustworthy confidential virtual machines for the masses"*
- Noch nichts dabei? `https://sys.cs.fau.de/theses`

Interesse:
Mail an: ruediger.kapitza@fau.de