SysWCET: Integrated Response-Time Analysis for Fixed-Priority Real-Time Systems

Christian Dietrich, Peter Wägemann,
Peter Ulbrich, Daniel Lohmann

{dietrich,lohmann}@sra.uni-hannover.de
{waegemann,ulbrich}@cs.fau.de

April 17, 2017

Computer Science 4
Operating Systems and Distributed Systems
FAU

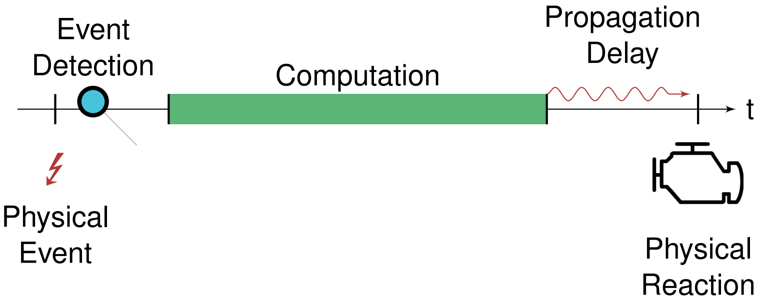
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Institute of Systems Engineering
System and Computer Architecture
Leibniz Universität Hannover

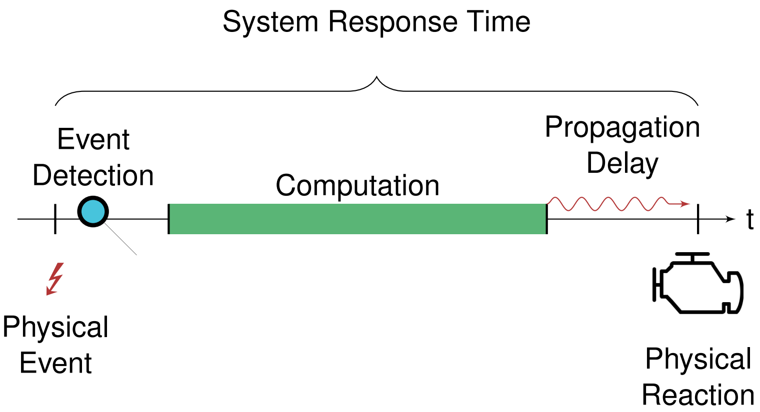
DFG

Leibniz Universität Hannover

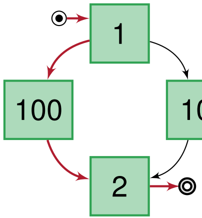
Response Time of Control Systems



Response Time of Control Systems

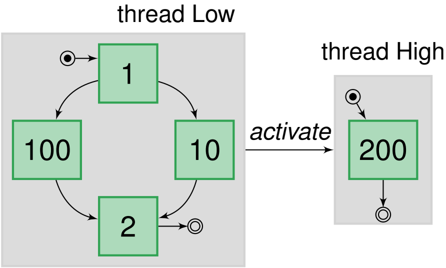


Worst-Case Response Time of Systems



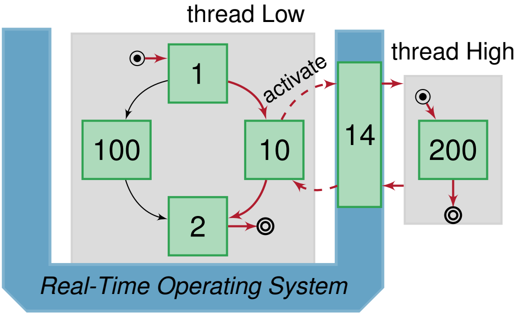
Worst-Case Response Time (WCRT): 103 cycles

Worst-Case Response Time of Systems



Worst-Case Response Time (WCRT): 103 + 200 + t(RTOS) cycles?

Worst-Case Response Time of Systems



Worst-Case Response Time (WCRT): 241 cycles

Worst-Case Response Time of Systems

thread Low

Local WCET analysis stops at the syscall boundary.

Threads and RTOS interact with each other.

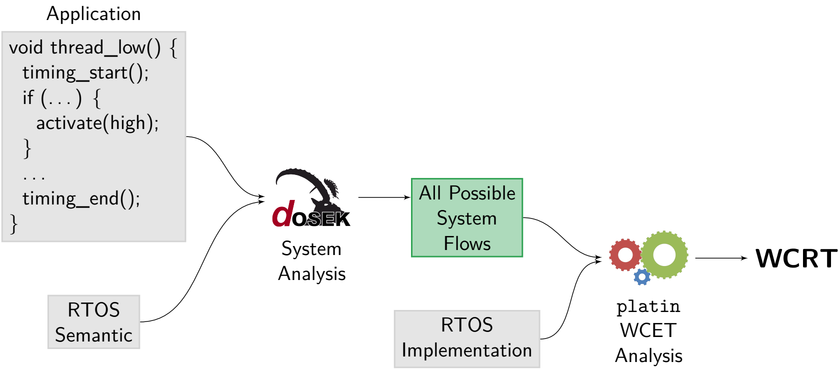
Threads are often **not** independent.

Worst-Case Response Time (WCRT): 241 cycles

The Problem with Compositional WCRT Analysis

- Commonly used approach is compositional
  - Calculate WCET of each component pessimistically in isolation
  - Aggregate WCETs bottom-up according to their interference
- Individual WCET have to be pessimistic to be safe
  - Always assume longest path in one thread
  - Worst-case execution time of the kernel for each syscall
- ⇒ System-wide unified formulation for WCRT problem
  - Unified formulation for machine-level and scheduling analysis
  - RTOS semantic and environment model must be considered
  - Possibility for cross-thread flow facts

Operation and Toolchain Overview



Outline

- Introduction and Motivation
- **Step 1: Operating-System State Transition Graph**
- Step 2: System-wide Unified IPET Formulation
- Evaluation
- Conclusion and Future Work

ISE SRA

Event-Triggered Real-Time Control Systems

- System Model
  - Event-triggered real-time systems: execution threads, ISRs, etc.
  - Fixed-priority scheduling semantics
  - Ahead of time knowledge
    - System objects (thread, resources, periodic signals) and their configuration
    - Application structure including syscall locations and arguments

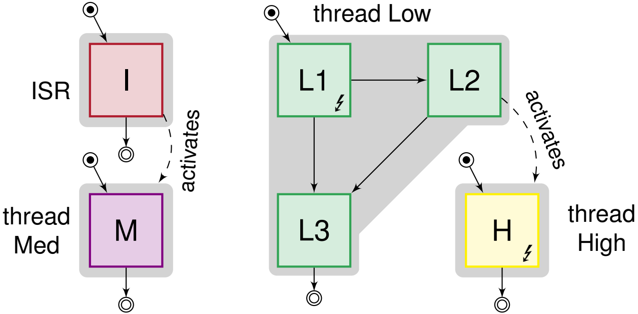
Event-Triggered Real-Time Control Systems

- System Model
  - Event-triggered real-time systems: execution threads, ISRs, etc.
  - Fixed-priority scheduling semantics
  - Ahead of time knowledge
    - System objects (thread, resources, periodic signals) and their configuration
    - Application structure including syscall locations and arguments

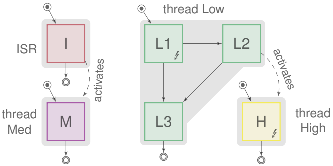
AUTOSAR

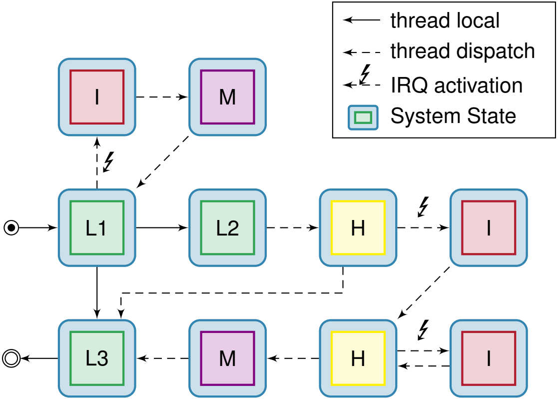
- Assumption apply to a wide range of systems: OSEK, AUTOSAR
  - Industry standard widely employed in the automotive industry
  - Static configuration at compile-time
  - Fixed-priority scheduling with threads and ISRs
  - Stack-based priority ceiling protocol (PCP) for resources

Operating-System State Transition Graph

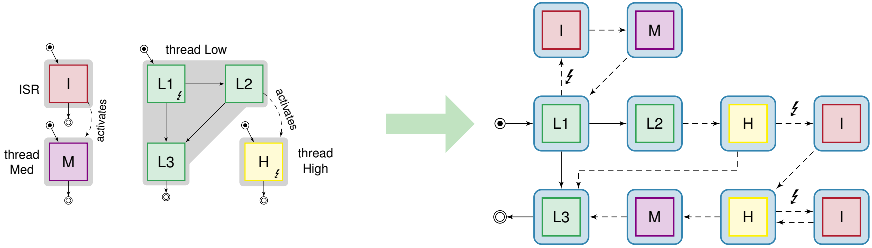


Operating-System State Transition Graph





Operating-System State Transition Graph



- Explicit enumeration of all system states
- Operating system ↔ Application ↔ Environment
- Includes interrupt activations, synchronization protocol, preemption

Outline

- Introduction and Motivation
- Step 1: Operating-System State Transition Graph
- **Step 2: System-wide Unified IPET Formulation**
- Evaluation
- Conclusion and Future Work

SysWCET IPET Formulation

- Implicit Path Enumeration Technique
  - Calculate upper bound on WCET of programs
  - Utilizes Integer Linear Programming (ILP)
  - *Execution frequency* on longest path
  - Allows integration of flow facts (e.g., mutual exclusive paths)

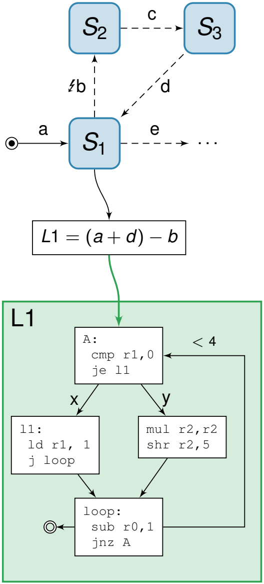
SysWCET IPET Formulation

- Implicit Path Enumeration Technique
  - Calculate upper bound on WCET of programs
  - Utilizes Integer Linear Programming (ILP)
  - *Execution frequency* on longest path
  - Allows integration of flow facts (e.g., mutual exclusive paths)

- SysWCET Idea in a Nutshell
  1. IPET on State Transition Graph: *state frequency*
  2. One IPET fragments for each program block
  3. Derive block frequency from state frequency

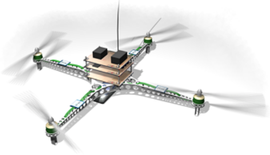
Layered IPET Construction

- Operating-System State Layer
  - State and state-transition frequency ILP variables
  - How often visits the system S_1 for the WCRT?
  - Restrict IRQ count globally
    $(b + \ldots) \cdot 1000 < T_{WCRT}$
- Glue Layer
  - Derive block activations from state frequency
  - Subtract *completed* IRQ activations
- Machine Layer
  - Construct IPET fragment for each ABB
  - RTOS' machine code is included
  - Block frequencies drive machine-level IPETs
  - Flow Facts inside/across IPET fragments



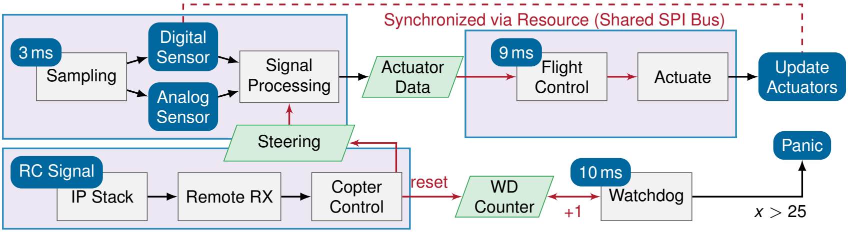
Outline

Hardware Model and Scenario: i4Copter



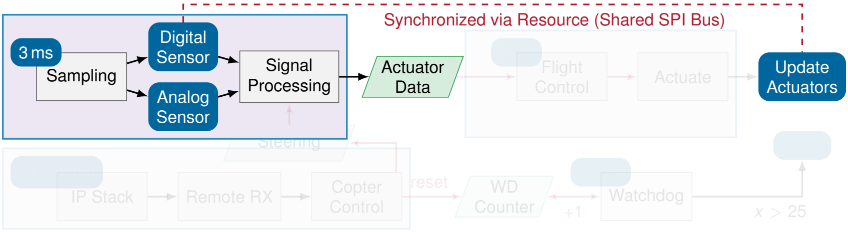
- Currently: basic processor model
  - No inter-instruction cost (no pipelines, no caches)
  - Count machine instructions on PATMOS ISA
  - SysWCET combinable with more complex models
- dOSEK as operating-system implementation
  - Generative approach with in-depth application analysis
  - Exports partial OS state transition graphs
- i4Copter: a safety-critical embedded control system
  - Developed in cooperation with Siemens Corporate Technology
  - 11 threads, 3 periodic signals, 1 interrupt, PCP resources, interrupt locks
  - Analyze only thread interactions without computations

Evaluation



- Automatic SysWCET WCRT analysis
  - Code annotations mark the start and endpoints of analysis
  - dOSEK calculates OS state-transition graph
  - `platin` WCET analyzer builds and solves IPET
- Manual compositional WCRT analysis
  - Calculate task WCETs in isolation with `platin`
  - Manual cumulation of individual results according to OS config

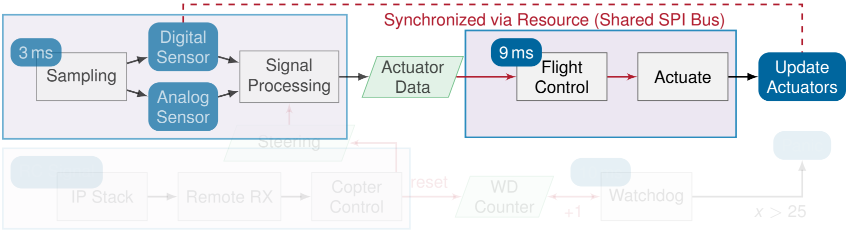
Evaluation



- Automatic SysWCET analysis ⇔ compositional WCRT analysis

	States	Solve Time	WCRT	
			SysWCET	Manual
Signal Gathering	9506	14.72 s	5626 cyc.	6286 cyc.

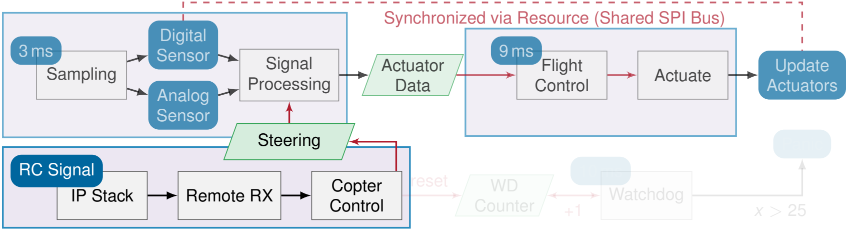
Evaluation



- Automatic SysWCET analysis ⇔ compositional WCRT analysis

			WCRT	
	States	Solve Time	SysWCET	Manual
Signal Gathering	9506	14.72 s	5626 cyc.	6286 cyc.
Flight Control	7690	161.56 s	9279 cyc.	10057 cyc.

Evaluation



- Automatic SysWCET analysis ⇔ compositional WCRT analysis

			WCRT	
	States	Solve Time	SysWCET	Manual
Signal Gathering	9506	14.72 s	5626 cyc.	6286 cyc.
Flight Control	7690	161.56 s	9279 cyc.	10057 cyc.
Remote Control	4608	92.57 s	9768 cyc.	10541 cyc.

Outline

- Introduction and Motivation
- Step 1: Operating-System State Transition Graph
- Step 2: System-wide Unified IPET Formulation
- Evaluation
- **Conclusion and Future Work**

Conclusion and Future Work

- WCRT is the WCET of the whole system while executing a job
  - RTOS, interrupts, and other threads interfere with execution
  - Compositional WCRT analysis accumulates pessimism
- SysWCET provides automatic system-wide WCRT analysis
  - Unified IPET formulation spanning multiple threads
  - Support for fixed-priority event-triggered RTOS
  - Based on RTOS-aware application analysis
- Directions of future research
  - Support more complex hardware models (pipelines, caches,. . .)
  - More dense OS state transition graph for more efficient operation
  - Extraction and formulation of cross-thread flow facts