
Christian Dietrich, Peter Wägemann, Peter Ulbrich, Daniel Lohmann

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

April 17, 2017
Response Time of Control Systems

Event Detection

Computation

Propagation Delay

Physical Event

Physical Reaction

System Response Time

LUH, FAU SysWCET – Introduction and Motivation
Response Time of Control Systems

System Response Time

- Event Detection
- Computation
- Propagation Delay

Physical Event

Physical Reaction
Worst-Case Response Time (WCRT): 103 cycles
Worst-Case Response Time (WCRT): $103 + 200 + t(\text{RTOS})$ cycles?
Worst-Case Response Time of Systems

Worst-Case Response Time (WCRT): 241 cycles
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 \textit{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

```c
void thread_low() {
    timing_start();
    if (...) {
        activate(high);
    }
    ...
    timing_end();
}
```

Application -> System Analysis -> RTOS Semantic

System Analysis -> All Possible System Flows

All Possible System Flows -> RTOS Implementation

RTOS Implementation -> platin WCET Analysis

platin WCET Analysis -> WCRT
Outline

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

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

ISR

thread Med

thread Low

activates

L1

L3

activates

H

thread High

LUH, FAU SysWCET – System-Wide Control Flows
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
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)
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

- Introduction and Motivation
- Step 1: Operating-System State Transition Graph
- Step 2: System-wide Unified IPET Formulation
- Evaluation
- Conclusion and Future Work
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
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
Automatic SysWCET analysis ⇔ compositional WCRT analysis

<table>
<thead>
<tr>
<th></th>
<th>States</th>
<th>Solve Time</th>
<th>SysWCET</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Gathering</td>
<td>9506</td>
<td>14.72 s</td>
<td>5626 cyc.</td>
<td>6286 cyc.</td>
</tr>
</tbody>
</table>
Automatic SysWCET analysis ⇔ compositional WCRT analysis

<table>
<thead>
<tr>
<th>State</th>
<th>States</th>
<th>Solve Time</th>
<th>SysWCET</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Gathering</td>
<td>9506</td>
<td>14.72 s</td>
<td>5626 cyc.</td>
<td>6286 cyc.</td>
</tr>
<tr>
<td>Flight Control</td>
<td>7690</td>
<td>161.56 s</td>
<td>9279 cyc.</td>
<td>10057 cyc.</td>
</tr>
</tbody>
</table>
Automatic SysWCET analysis ⇔ compositional WCRT analysis

<table>
<thead>
<tr>
<th></th>
<th>States</th>
<th>Solve Time</th>
<th>SysWCET</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Gathering</td>
<td>9506</td>
<td>14.72 s</td>
<td>5626 cyc.</td>
<td>6286 cyc.</td>
</tr>
<tr>
<td>Flight Control</td>
<td>7690</td>
<td>161.56 s</td>
<td>9279 cyc.</td>
<td>10057 cyc.</td>
</tr>
<tr>
<td>Remote Control</td>
<td>4608</td>
<td>92.57 s</td>
<td>9768 cyc.</td>
<td>10541 cyc.</td>
</tr>
</tbody>
</table>
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