Composing and synchronizing real-time components through virtual platforms in vehicular systems

Martijn van den Heuvel

System Architecture and Networking (SAN)
Department of Mathematics and Computer Science
Eindhoven University of Technology
The Netherlands

16th September 2014

Thanks to...

Promotor and co-promotor:
- Prof. Johan J. Lukkien
- Dr. Reinder J. Bril

Technical support:
- Dr. Mike Holenderski
- Dr. Erik J. Luit
- Dr. Richard Verhoeven

Research collaboration (MDH):
- Dr. Moris Behnam
- Prof. Thomas Nolte

Various M.Sc. students and postgraduates.

An automotive example

Why this growth of electronic parts?

- Increasing number of applications;
- Extensive networking between them.
An automotive example: electronic stability program (ESP)

- Increasing number of applications, for example: ABS, TCS, ESP;
- Extensive networking between them.

An automotive example: deployment of IVDC advances

- 4X Local controllers for steering, braking, suspension;
- Front and rear IVDC;
- 1X Global IVDC state estimation and supervisory control.

Demo: IVDC with active suspension

Experimental setup:
- 4X Local controllers for active suspension;
- 1X Global IVDC and supervisory control.

Our contribution:
- Virtualization techniques applied to a central node:
  - Synchronization between independently developed components.
Hierarchical composition of schedulers for vehicular systems

- component: server, set of tasks and local (task) scheduler
- server or virtual processor: a CPU budget allocated each period

Tasks, located in arbitrary components, may share resources
i.e. deal with local and global priority inversion!
Resource sharing across components: a closer look

- **Add a resource arbiter to each scheduler**, i.e., at the task level and at the component level.

- **Tasks may request:**
  1. Access to shared memory:
     - shared buffers;
     - memory-mapped devices.
  2. Operating-system services:
     - in-kernel: short non-preemptive critical sections;
     - device drivers and other services: mutually exclusive between users.
  3. Global limited-preemptive access to the processor:
     - reduce cache misses;
     - less pipeline flushes.

---

Challenges for resource sharing across components

1. **Independent development and integration of components;**
   - M. M. H. P. van den Heuvel, R. J. Bril, and J. J. Lukkien.
   - Transparent synchronization protocols for compositional real-time systems.

2. **Independent timing analysis of components and their integration;**
   - Opaque analysis for resource sharing in compositional real-time systems.
   - Optimal and fast composition of resource-sharing components in hierarchical real-time systems.
   - In RTCSA, Aug. 2014.

3. **Scheduling components and containment of temporal faults.**
   - M. M. H. P. van den Heuvel, R. J. Bril, and J. J. Lukkien.
   - Dependable resource sharing for compositional real-time systems.

---

A programmer’s perspective:

- resources are shared between tasks, not between components.
- tasks claim and protect their resources;
- Postpone binding of SRP primitives until component integration.

**Question from integrator to programmer:**

Is the resource local or global to a component?

---

Martijn van den Heuvel (TU/e, SAN) 16th September 2014 15 / 32
Challenges for resource sharing across components

- Independent development and integration of components;
- Independent timing analysis of components and their integration;
- Scheduling components and containment of temporal faults.

**Independent timing analysis of components and their integration:**

  Opaque analysis for resource sharing in compositional real-time systems.
  Optimal and fast composition of resource-sharing components in hierarchical real-time systems.
  In RTCSA, Aug. 2014.

Some analyses in literature do not support this independence!

The key idea – Specify partial interfaces and merge them

- Incremental analysis: separate concerns of local and global scheduling;
- Easier analysis: no timing details on global resource sharing;
- Avoid protocol-specific knowledge to compute $Q_s$ and $\{X_{sl}\}$.

**Analytical perspective:**

**The key idea – Specify partial interfaces and merge them**

**Gained independence:**

- Detect which resources are shared by others.
- Arbitrate shared resources by SRP.

**Merge them at composition!**
The key idea – Prune dependencies at composition

Gained independence:
- Detect which resources are shared by others.

Add an arbiter for shared resources only

Gained independence:
- Detect which resources are shared by others.
- Arbitrate shared resources by SRP;

Challenges for resource sharing across components

1. Independent development and integration of components;
2. Independent timing analysis of components and their integration;
3. Scheduling components and containment of temporal faults.

Scheduling and containment of temporal faults

Temporal isolation stunned by overrun when a task unwinds:

No longer a guarantee for:
- overrun durations
- blocking durations

Legend:
- critical section
- normal execution
- budget arrival

Component development

Component C

Component Interface ΩC

Admission control

Virtual platform of C

Other virtual platforms

Resource allocation

Global scheduling and global resource arbitration

M. M. H. P. van den Heuvel, R. J. Bril, and J. J. Lukkien.
Dependable resource sharing for compositional real-time systems.
Sources of unpredictable interferences

Global effects of local SRP-based arbitration:

- What if task $\tau_1$ exceeds its WCET?
- What if a task ($\tau_2$ or $\tau_3$) exceeds its critical-section length?

![Diagram showing time, $\tau_1$, $\tau_2$, $\tau_3$, resource ceiling ($rc_{sl}$), highest priority of any (local) task sharing resource $R_l$](image)

Confine temporal faults

- What if an interfering task exceeds its WCET?
  - A solution to prevent an increase of resource-holding times: Disable local preemptions.

- What if a task exceeds its critical-section length?
  - **NO solution** to prevent an increase of resource-holding times.
  - To protect independent components: 
    - Extend SRP with enforced blocking durations.

It is now similar to the protected Immediate-PCP in:

- D. de Niz, L. Abeni, S. Saewong, and R. Rajkumar.
  - Resource sharing in reservation-based systems.

Challenges for resource sharing across components

- Independent development and integration of components;
- Independent timing analysis of components and their integration;
- Scheduling components and containment of temporal faults.
  - Performance evaluation of analysis methods for arbitrating nonpreemptive resource access in compositional real-time systems.
  - In VHRE, Sep. 2014.

Scheduling of resource-sharing components

- **Budget depletion** during a critical section can lead to excessive blocking times:

![Diagram showing $P_s$, $Q_s$, $B_s$](image)

- **Solutions**:
  - React upon budget depletion by allowing budget overruns.
    - **Overrun with payback (OWP)**: the consumed overrun is subtracted from the next budget;
    - **Overrun without payback (ONP)**: no penalty for overrun.
  - Prevent budget depletion by checking the remaining budget.
    - **SIRAP**: insert idle time during task execution;
    - **BROE**: delay budget supply.
Timing analysis of resource-sharing components

- synthetic periodic tasks: 8 tasks/component;
- tasks generated by UUnifast with total utilization of 0.5;
- random task deadline constraints: \( WCET_i + \alpha(P_i - WCET_i); P_i \);
- short non-preemptive critical section.

\[ WCET_i + \alpha(P_i - WCET_i); P_i \]

\( N = 5 \) components.

Implicit deadlines \( (\alpha = 1) \).

**Challenges for resource sharing across components**

- Independent development and integration of components;
- Independent timing analysis of components and their integration;
- Scheduling components and containment of temporal faults.

**Current and future challenges:**

- **Mixed criticality:**
  - deal with uncertain timing specs of tasks;
  - graceful degrade functions by enabling/disabling optional ones.

- **Industrial standards:** timing augmented descriptions of components.

**Questions?**

Let’s pass the remote...