Dependable resource sharing for compositional real-time systems
– Embedded Real-time Computing Systems and Applications (RTCSA) –

M.M.H.P. van den Heuvel - R.J. Bril - J.J. Lukkien

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

30 August 2011
1. Introduction
2. Inter-component resource sharing
3. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
4. HSTP: Self-donations and third-party donations
5. HSTP: Budget overruns versus self-blocking
6. Conclusions
Why hierarchical scheduling: an example for automotive

- Reduce the number of nodes
- Trend: Fewer and more powerful nodes
Why hierarchical scheduling: an example for automotive

- Reduce the number of nodes
- Trend: Fewer and more powerful nodes

**Integration problem:**
- temporal isolation between legacy applications on a single processor;
- applications may share more resources.
A Solution: Hierarchical Scheduling

- component: server, set of tasks and local (task) scheduler
- server: a budget allocated each period
A Solution: Hierarchical Scheduling

- Component: server, set of tasks and local (task) scheduler
- Server: a budget allocated each period

Tasks, located in arbitrary components, may share logical resources i.e. deal with local and global priority inversion!
Outline

1 Introduction

2 Inter-component resource sharing

3 B-HSTP: Basic Hierarchical Synchronization with Temporal Protection

4 HSTP: Self-donations and third-party donations

5 HSTP: Budget overruns versus self-blocking

6 Conclusions
Resource sharing: a closer look

1. Access shared memory:
   - shared buffers;
   - memory-mapped devices.

2. Operating-system services:
   - in-kernel: short non-preemptive critical sections;
   - other services: mutually exclusive between resource users.

3. Global non-preemptive access to the processor (pseudo-resources):
   - reduce cache misses;
   - less pipeline flushes.

Critical sections may be long w.r.t. WCETs!
Global resource sharing problem

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

---

Three SRP-based solutions for HSFs:

**HSRP:** React upon budget depletion while a resource is locked; i.e. allow to use an overrun budget;

**BROE** and **SIRAP:** Prevent budget depletion during resource access; i.e. before granting access, first check the remaining budget.

---
Global resource sharing problem

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

Three SRP-based solutions for HSFs:

- **HSRP**: React upon budget depletion while a resource is locked; i.e. allow to use an overrun budget;
Global resource sharing problem

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

Three SRP-based solutions for HSFs:

- **HSRP**: React upon budget depletion while a resource is locked; i.e. allow to use an overrun budget;

- **BROE** and **SIRAP**: Prevent budget depletion during resource access; i.e. before granting access, first check the remaining budget.
Temporal isolation stunned by overrun

When a task unwinds:

Legend:
- critical section
- normal execution
- budget arrival

No longer a guarantee for:
- overrun durations
- blocking durations
Sources of unpredictable interferences

Global effects of local SRP-based arbitration:

What if Task 1 exceeds its WCET?
Global effects of local SRP-based arbitration:

- What if Task 1 exceeds its WCET?
- What if a Task 3 exceeds its critical-section length?
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 prevent excessive overruns:

- Modified SRP with enforced blocking durations.

It is now similar to the Immediate-PCP in:


Martijn van den Heuvel (TU/e, SAN)
Confine temporal faults

What if an interfering task exceeds its WCET?

- A solution to prevent an increase of resource-holding times:
  
  **Disable local preemptions.**

---

Martijn van den Heuvel (TU/e, SAN)  Dependable resource sharing in HSFs  30 August 2011  10 / 22
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?
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.
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 prevent excessive overruns:
  Modified SRP with enforced blocking durations.
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 prevent excessive overruns: Modified SRP with enforced blocking durations.

It is now similar to the Immediate-PCP in:

Outline

1 Introduction

2 Inter-component resource sharing

3 B-HSTP: Basic Hierarchical Synchronization with Temporal Protection

4 HSTP: Self-donations and third-party donations

5 HSTP: Budget overruns versus self-blocking

6 Conclusions
Global resource arbitration rules of B-HSTP

1. Use **resource-access budgets** (RAB) guarded by timers:
   1. Lock: replenish RAB with maximum critical-section length;
   2. Unlock: cancel timer;
   3. RAB depletion: mimic resource unlock and mark resource **busy**;

2. follow SRP as long as timing contracts are respected;
3. at most 1 task per component may access 1 resource at each time;
4. a component can keep at most 1 resource **busy**;
5. when a component blocks on a **busy** resource: suspend until next period.

A basic scheme to monitor and enforce blocking durations!
Global resource arbitration rules of B-HSTP

1. Use **resource-access budgets** (RAB) guarded by timers:
   - **Lock**: replenish RAB with maximum critical-section length;
   - **Unlock**: cancel timer;
   - **RAB depletion**: mimic resource unlock and mark resource **busy**;

2. follow SRP as long as timing contracts are respected;
3. at most 1 task per component may access 1 resource at each time;
4. a component can keep at most 1 resource **busy**;
5. when a component blocks on a **busy** resource:
   - suspend until next period.
Global resource arbitration rules of B-HSTP

1. Use **resource-access budgets** (RAB) guarded by timers:
   1. Lock: replenish RAB with maximum critical-section length;
   2. Unlock: cancel timer;
   3. RAB depletion: mimic resource unlock and mark resource **busy**;

2. follow SRP as long as timing contracts are respected;
3. at most 1 task per component may access 1 resource at each time;
4. a component can keep at most 1 resource **busy**;
5. when a component blocks on a **busy** resource:
   - suspend until next period.

A basic scheme to monitor and enforce blocking durations!
Outline

1. Introduction
2. Inter-component resource sharing
3. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
4. HSTP: Self-donations and third-party donations
5. HSTP: Budget overruns versus self-blocking
6. Conclusions
Depleting a resource-access budget (RAB) ... and then?

Wait until a replenishment in the next component period:

- any remaining normal budget is lost;

---

Use remaining budget?

Legend:
- critical section
- normal execution
- task arrival

Use remaining budget:
replenish the RAB with a self-donation.

Unresolved in:

D. de Niz, L. Abeni, S. Saewong, and R. Rajkumar.
Resource sharing in reservation-based systems.
Replenish the RAB with a self-donation

- Allow for preemptions before RAB replenishment, i.e. **avoid double blocking**;

- Problem-causing component may resolve the problem itself, before another component notices:

  ![Diagram](image)

  **Legend:**
  - critical section
  - normal execution
  - budget arrival

  - Repeat this until the entire normal budget is depleted;

- SRP-based blocking for independent components.
Blocking on a busy resource... and then?

- Wait until the malicious component resolves the problem;
- Re-try in each component period;
- Waiting does not resolve anything & budget is lost:

```
Legend:
C1
C2
time
t1
t2
t
X2
Self-suspension is not allowed!
```

Use a limited form of donations compared to BWI-based solutions:

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

R. Santos, G. Lipari, and J. Santos.
Improving the schedulability of soft real-time open dynamic systems: The inheritor is actually a debtor.
Replenish the RAB with a third-party donation

- Allow for preemptions before RAB replenishment, i.e. **avoid double blocking** upon replenishment of the donatee:

\[
C_1 \quad \downarrow \\
C_2 \quad \downarrow \\
C_3 \\
t_1 \quad \quad t_e \quad \quad \text{time}
\]

Legend:
- critical section
- normal execution
- budget arrival
- donate budget

- No transitive donations like in BWI;
- No chained blocking;
- Repeat this until the entire normal budget is depleted.
Only resource-sharing components (red) have a shared problem!
Outline

1. Introduction
2. Inter-component resource sharing
3. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
4. HSTP: Self-donations and third-party donations
5. HSTP: Budget overruns versus self-blocking
6. Conclusions

Martijn van den Heuvel (TU/e, SAN) Dependable resource sharing in HSFs 30 August 2011 19 / 22
HSTP complements the existing protocols

- **Overrun:**
  - also donate the overrun budget, $X_{sl}$, 
  - i.e. it is meant to complete a critical section!

- **Self-blocking:**
  - **do not** donate when there is insufficient budget, 
  - **because** this may cause unnecessary deadline misses:

![Diagram showing self-blocking and normal execution]

- Optionally: SIRAP allows to give away the remaining budget, 
  - i.e. it is accounted for as idled time anyway!
1. Introduction
2. Inter-component resource sharing
3. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
4. HSTP: Self-donations and third-party donations
5. HSTP: Budget overruns versus self-blocking
6. Conclusions
Conclusions

We presented HSTP:

1. A basic scheme to monitor and enforce blocking durations;
2. SRP-compliant blocking for independent components;

4. Complements existing protocols: use an overrun budget for donations whenever it becomes available; adhere to the self-blocking conditions whenever applicable;

Open problem:

- how to allocate budgets in the presence of misbehaving components?
- how to account for donations?

Questions...
Conclusions

We presented **HSTP**: 
1. A basic scheme to *monitor* and *enforce* blocking durations; 
2. SRP-compliant blocking for independent components; 
3. In malicious situations: 
   - resource-sharing components have a *shared problem*; 
   - problem-causing component tries to resolve it using *self-donations*; 
   - blocking components may provide help, i.e. *third-party donations*. 

Open problem: 
- how to allocate budgets in the presence of misbehaving components? 
- how to account for donations?

Questions...
Conclusions

We presented **HSTP**:  
1. A basic scheme to **monitor** and **enforce** blocking durations;  
2. SRP-compliant blocking for independent components;  
3. In malicious situations:  
   - resource-sharing components have a **shared problem**;  
   - problem-causing component tries to resolve it using **self-donations**;  
   - blocking components may provide help, i.e. **third-party donations**.  
4. Complements existing protocols:  
   - use an overrun budget for donations whenever it becomes available;  
   - adhere to the self-blocking conditions whenever applicable;
We presented **HSTP**:

1. A basic scheme to **monitor** and **enforce** blocking durations;
2. SRP-compliant blocking for independent components;
3. In malicious situations:
   - resource-sharing components have a **shared problem**;
   - problem-causing component tries to resolve it using **self-donations**;
   - blocking components may provide help, i.e. **third-party donations**.
4. Complements existing protocols:
   - use an overrun budget for donations whenever it becomes available;
   - adhere to the self-blocking conditions whenever applicable;

Open problem:

- how to allocate budgets in the presence of misbehaving components?
- how to account for donations?
Conclusions

We presented **HSTP**:

1. A basic scheme to **monitor** and **enforce** blocking durations;
2. SRP-compliant blocking for independent components;
3. In malicious situations:
   - resource-sharing components have a **shared problem**;
   - problem-causing component tries to resolve it using **self-donations**;
   - blocking components may provide help, i.e. **third-party donations**.
4. Complements existing protocols:
   - use an overrun budget for donations whenever it becomes available;
   - adhere to the self-blocking conditions whenever applicable;

Open problem:

- how to allocate budgets in the presence of misbehaving components?
- how to account for donations?

Questions . . . ?