Temporal isolation in an HSF-enabled real-time kernel in the presence of shared resources
– International Workshop OSPERT 2011 –

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

5 July 2011
Outline

1. Introduction
2. Inter-subsystem resource sharing
3. Temporal Isolation
4. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
5. Conclusions
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 shared resources.
A Solution: Hierarchical Scheduling

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

Global Scheduler

Local scheduler

Subsystem 0

Local scheduler

Subsystem 1

Local scheduler

Subsystem n

R1

R2
A Solution: Hierarchical Scheduling

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

Tasks, located in arbitrary subsystems, may share logical resources
i.e. deal with local and global priority inversion!
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!**
1. Introduction

2. Inter-subsystem resource sharing

3. Temporal Isolation

4. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection

5. Conclusions
Global resource sharing problem

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

- SRP locally
- SRP globally

![Diagram showing resource sharing and blocking times]

Legend:
- active
- holding mutex
Global resource sharing problem

Two SRP-based solutions for fixed-priority scheduling:

- **HSRP:** React upon budget depletion while a resource is locked; i.e. allow to use an overrun budget
  1. **with payback:** the consumed overrun budget is subtracted from the next budget provisioning;
  2. **no payback:** no penalty for overrun consumption.
Global resource sharing problem

Two SRP-based solutions for fixed-priority scheduling:

- **HSRP:** React upon budget depletion while a resource is locked; i.e. allow to use an overrun budget
  1. **with payback:** the consumed overrun budget is subtracted from the next budget provisioning;
  2. **no payback:** no penalty for overrun consumption.

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

1. Introduction
2. Inter-subsystem resource sharing
3. Temporal Isolation
4. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
5. Conclusions
Temporal isolation stunned by overrun

When a task unwinds:

Legend:
- critical section
- normal execution
- budget arrival

SubSys$_1$
SubSys$_2$
SubSys$_3$

$t_0$
$t_1$
$te$
time

No longer a guarantee for:
- overrun durations
- blocking durations
Global effects of local SRP-based arbitration:

- What if Task 1 exceeds its WCET?
Sources of unpredictable interferences

Global effects of local SRP-based arbitration:

Legend:
- active
- holding mutex
- preempted

- What if Task 1 exceeds its WCET?
- What if a Task 3 exceeds its critical-section length?
What if an interfering task exceeds its WCET?
What if an interfering task exceeds its WCET?

- A solution to prevent an increase of resource-holding times:

  Disable local preemptions.
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?
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.
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 critical-section lengths.
Outline

1. Introduction
2. Inter-subsystem resource sharing
3. Temporal Isolation
4. B-HSTP: Basic Hierarchical Synchronization with Temporal Protection
5. Conclusions
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**;
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 subsystem may access 1 resource at each time;

4. A subsystem can keep at most 1 resource **busy**;

5. When a subsystem blocks on a **busy** resource:
   - Suspend until next period.
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 subsystem may access 1 resource at each time;
4. a subsystem can keep at most 1 resource **busy**;
5. when a subsystem blocks on a **busy** resource:
   - suspend until next period.

**A basic scheme to monitor and enforce critical-section lengths!**
Grasping B-HSTP: an example
On the performance of B-HSTP

<table>
<thead>
<tr>
<th>Event</th>
<th>1-level SRP</th>
<th>HSRP</th>
<th>SIRAP</th>
<th>B-HSTP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BC</td>
<td>WC</td>
<td>BC</td>
<td>WC</td>
</tr>
<tr>
<td>Resource lock</td>
<td>124</td>
<td>124</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td>Resource unlock</td>
<td>106</td>
<td>106</td>
<td>196</td>
<td>725</td>
</tr>
<tr>
<td>Budget depletion</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>383</td>
</tr>
<tr>
<td>Budget replenishment</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>

Is B-HSTP expensive?

- **No:**
  1. not more expensive compared to servers in HSFs;
  2. cheaper than existing solutions in literature.

- **Yes:** overheads are not periodic and difficult to analyse.
1 Introduction

2 Inter-subsystem resource sharing

3 Temporal Isolation

4 B-HSTP: Basic Hierarchical Synchronization with Temporal Protection

5 Conclusions
Conclusions

We presented:

1. **B-HSTP**: a basic scheme to **monitor** and **enforce** critical-section lengths;

2. an implementation in a COTS micro-kernel, $\mu$C/OS-II;

3. a comparison with straightforward SRP-based resource arbitration,
   - see OSPERT 2010: SIRAP and HSRP implementation;

Lessons learned:

1. Timers are expensive and, thus, impractical for short critical sections!

2. Timers are necessary to guarantee temporal isolation!

Can we do better?
Conclusions

We presented:

1. **B-HSTP**: a basic scheme to monitor and enforce critical-section lengths;
2. an implementation in a COTS micro-kernel, \( \mu \)C/OS-II;
3. a comparison with straightforward SRP-based resource arbitration, see OSPERT 2010: SIRAP and HSRP implementation;

Lessons learned:

1. Timers are expensive and, thus, impractical for short critical sections!
2. Timers are necessary to guarantee temporal isolation!
Conclusions

We presented:

1. **B-HSTP**: a basic scheme to **monitor** and **enforce** critical-section lengths;
2. an implementation in a COTS micro-kernel, $\mu$C/OS-II;
3. a comparison with straightforward SRP-based resource arbitration, see OSPERT 2010: SIRAP and HSRP implementation;

Lessons learned:

1. Timers are expensive and, thus, impractical for short critical sections!
2. Timers are necessary to guarantee temporal isolation!

Can we do better?


What is next?


Questions . . . ?