Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

DynamicHinting

Priority aware Resource Management for Real-Time Operation in Wireless Sensor / Actor Networks

13.-14. August 2009
8. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"

M. Baunach
Chair for Computer Science V

Overview

I. Introduction and Motivation

why real-time matters …

II. Resource Management

… and Dynamic Hinting

III. Applications and Test Beds

real-world performance results

IV. Conclusion and Outlook

current and future work

DynamicHinting

I. Introduction and Motivation

Why Real-Time Matters...

Introduction

Today's Sensor Networks:
increasing size, pervasiveness, demands and complexity

Modular HW / SW concepts
(service oriented programming)

➔ Compositional Design ←

Intense interactions
(with environment / nodes)

➔ Sporadic & Periodic Events ←

Highly dynamic environments
(variable system load)

➔ Performance / Reactivity ←

Scheduling / Resource Management

&

Real-Time Operation

Resource sharing is a hard problem in time critical task systems!
Affects tasks, nodes & finally the overall system!

DynamicHinting

Related Work and Motivation

WSN research is still too limited to static design concepts!

Current (operating) systems for WSN/WSAN applications:

- Non-preemptive/run-to-completion tasks (e.g. TinyOS, Contiki)
  - Very common
  - Bad reactivity to sporadic events
- Preemptive tasks (e.g. Mantis, RETOS, SmartOS, threading extensions/libs)
  - Better reactivity might be possible
  - Rarely used
  - Most OS do not cover resource management issues

➔ Manual coordination and fine tuning of all tasks still required for proper operation. ←

Approaches for complex and compositional systems:

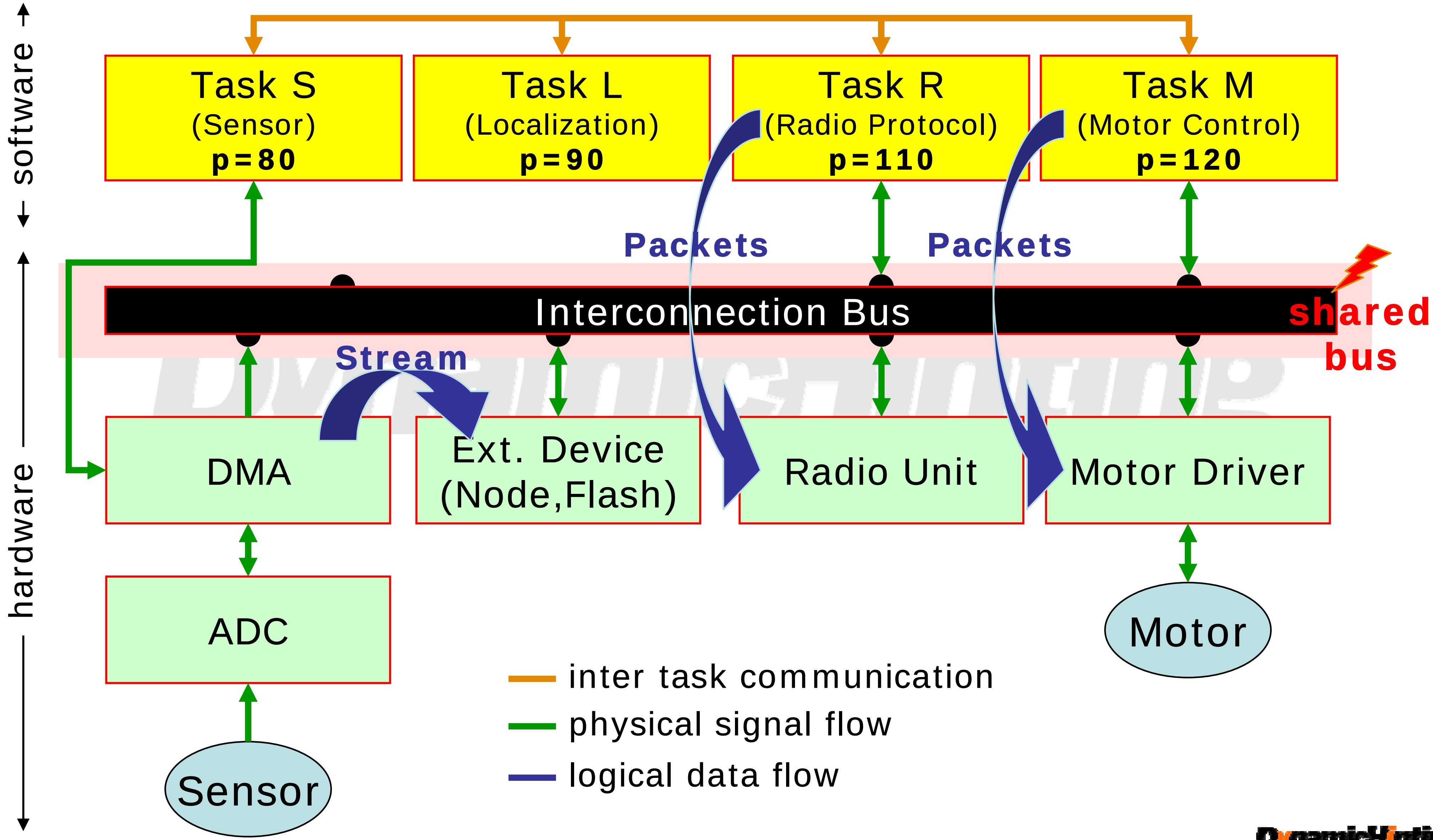
1. Decomposition into more but smaller (hardware) subsystems
2. **Concurrent task systems with cooperative resource sharing**
   (preemptive & prioritized for fast response on various events)

DynamicHinting

II. Resource Management

... and Dynamic Hinting

WSAN based Localization and Steering System



Resource Sharing in Preemptive Systems

Preemptive operation yields no instant advantage if a high priority task requires a shared resource which is currently held by any less important task!

➔ Priority Inversion and even Deadlocks might occur! ←

Task priorities are not obeyed as desired!
Unexpected behaviour, reduced reactivity & real-time capability!

Solution approaches:

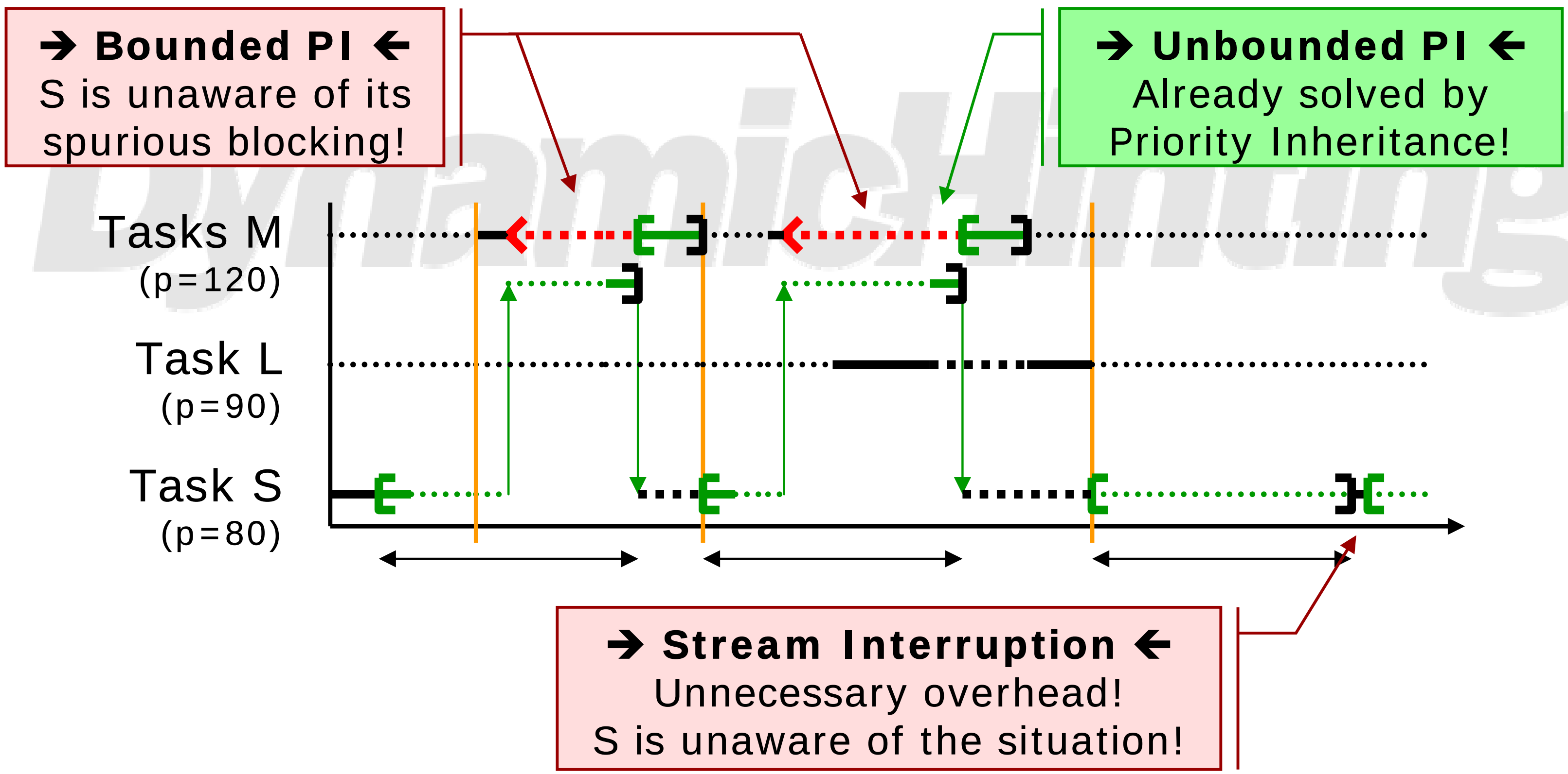
1. Terminate spurious tasks or withdraw resources.
2. Individual task priorities indicate the desired relevance.
   ➔ Adjust task priorities dynamically at runtime according to the current resource assignment situation.
   - Priority Ceiling / Highest Locker Protocol (PCP / HLP)
   - Priority Inheritance Protocol (PIP)

Resource Allocation – Example

Sensor task requires long-term allocation of the bus resource.
➔ Blocks other (sporadic but more important) tasks. ←

Idea: Regular/periodic release allows interleaved bus access.

Resource Allocation via Priority Inheritance Protocol (PIP):



Resource Allocation – Example

Sensor task requires long-term allocation of the bus resource.
➔ Blocks other (sporadic but more important) tasks. ➔

Idea: Regular/periodic release allows interleaved bus access.

Resource Allocation via Priority Inheritance Protocol (PIP):

Central Problem:

A task has no information about its current influence on the system.

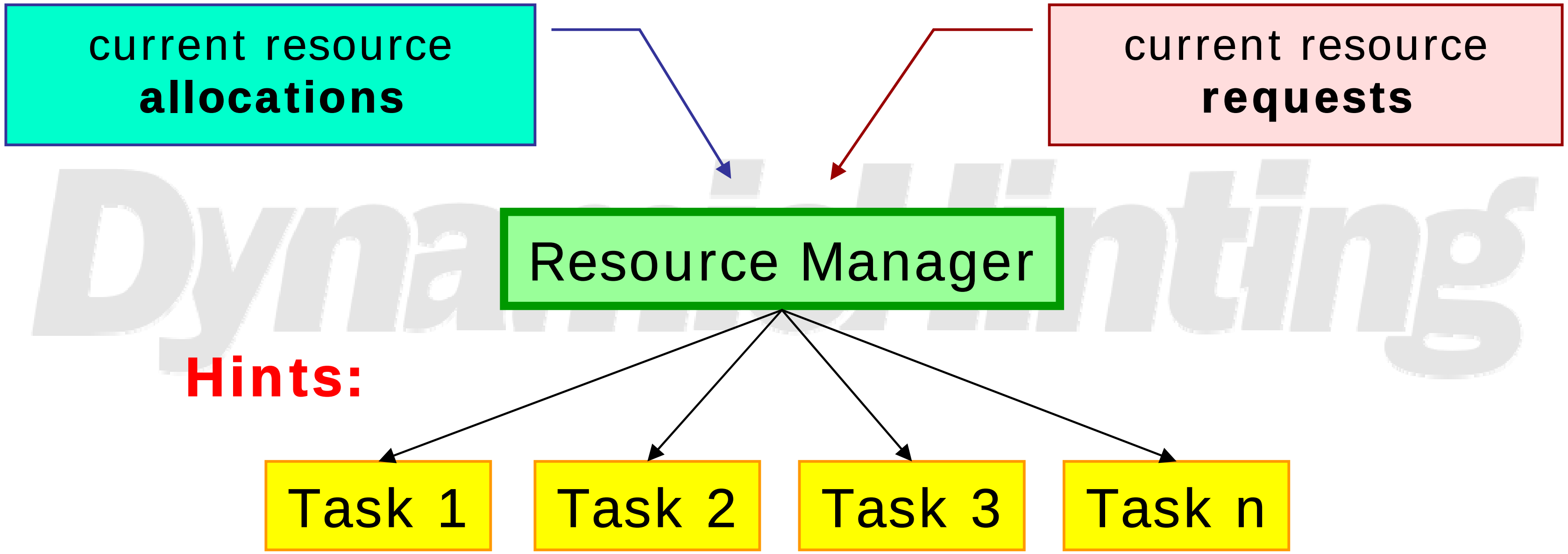
➔ Stream Interruption ➔
Unnecessary overhead!
S is unaware of the situation!

DynamicHinting

Dynamic Hinting – The Concept

Idea:

- **Take advantage of the resource manager's runtime knowledge** about current resource allocations & requirements
- Filter this information
  and forward it to tasks which currently block more relevant tasks.



- **Hints allow blocking and deadlocked tasks to adopt to the situation** and finally to contribute to the system's overall reactivity and stability.
- Still, the decision between following and ignoring a hint is made by each task autonomously and dynamically at runtime (e.g. by using TUFs).

How does a task receive its hints?

- **EQ: Explicit Querying**

  A task simply queries (e.g. regularly) if it currently blocks another more important task.

```
Resource* getHint(currentPrio*, isDeadlock*, remainingTime*);
```

- **EW: Early Wakeup**

  For idle periods, the task instructs the resource manager to wake it early in case of a hint.

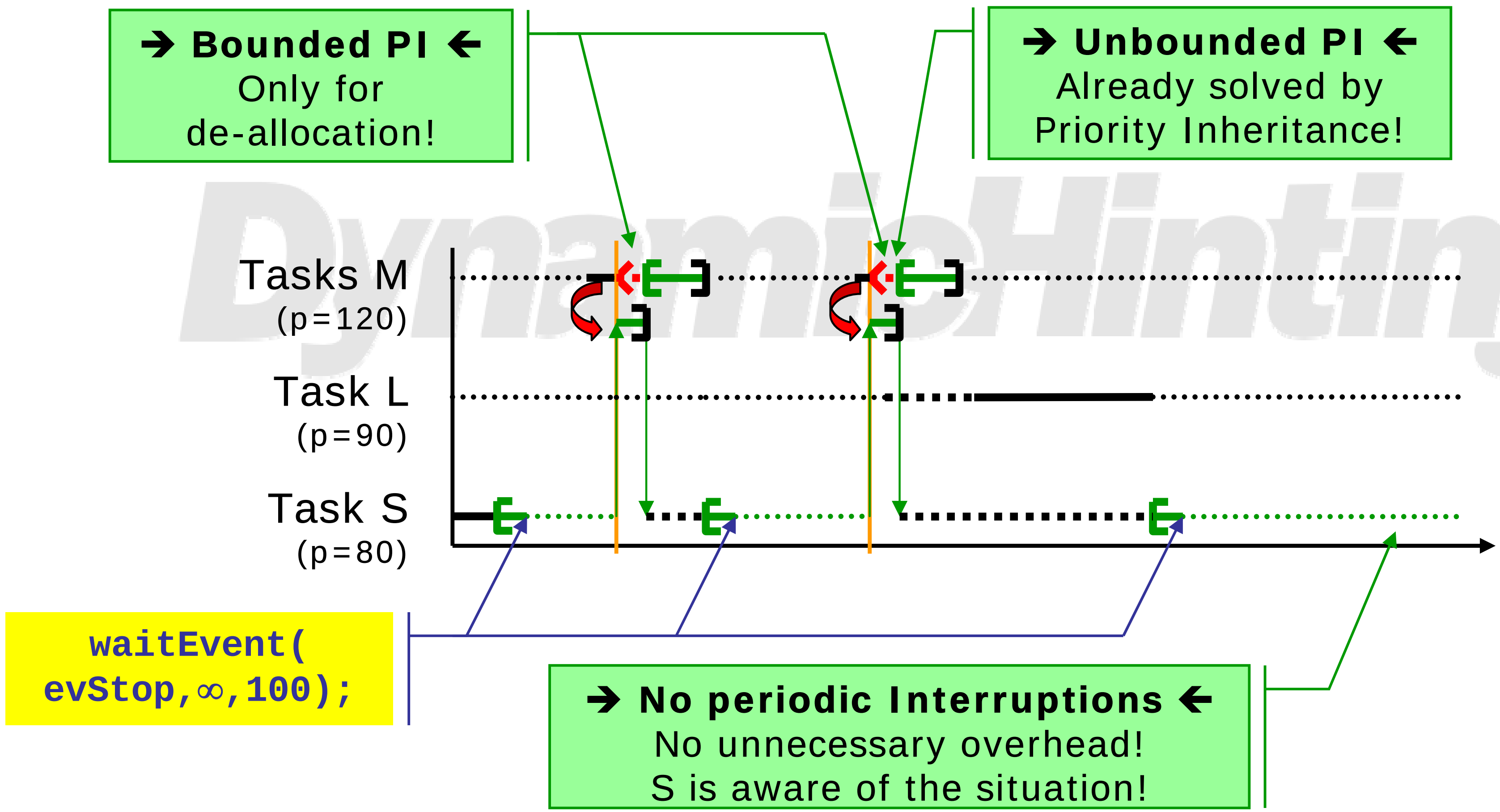
```
result_t sleep(deadline | timeout, prioThreshold);
result_t waitEvent(event, deadline | timeout, prioThreshold);
result_t getResource(resource, deadline | timeout, prioThreshold);
```

- **...**

Resource Allocation – Improved Example

Resource Allocation via Dynamic Hinting and Early Wakeup:

Policy upon a hint: Always perform an immediate stream interruption!



DynamicHinting

III. Applications and Test Beds

Real World Performance Results

Integration of Dynamic Hinting into the operating system

SmartOS

- **Preemptive tasks** with variable base priorities
- Integrated **timing concept** (1µs resolution)
- **Resource protection** mechanism
- Inter-Task communication
- Event handling system (includes IRQ timestamping)
- Available for TI MSP430 (Renesas SH2A, AVR under construction):
  ROM size: ~4 KB
  RAM size: ~100 B

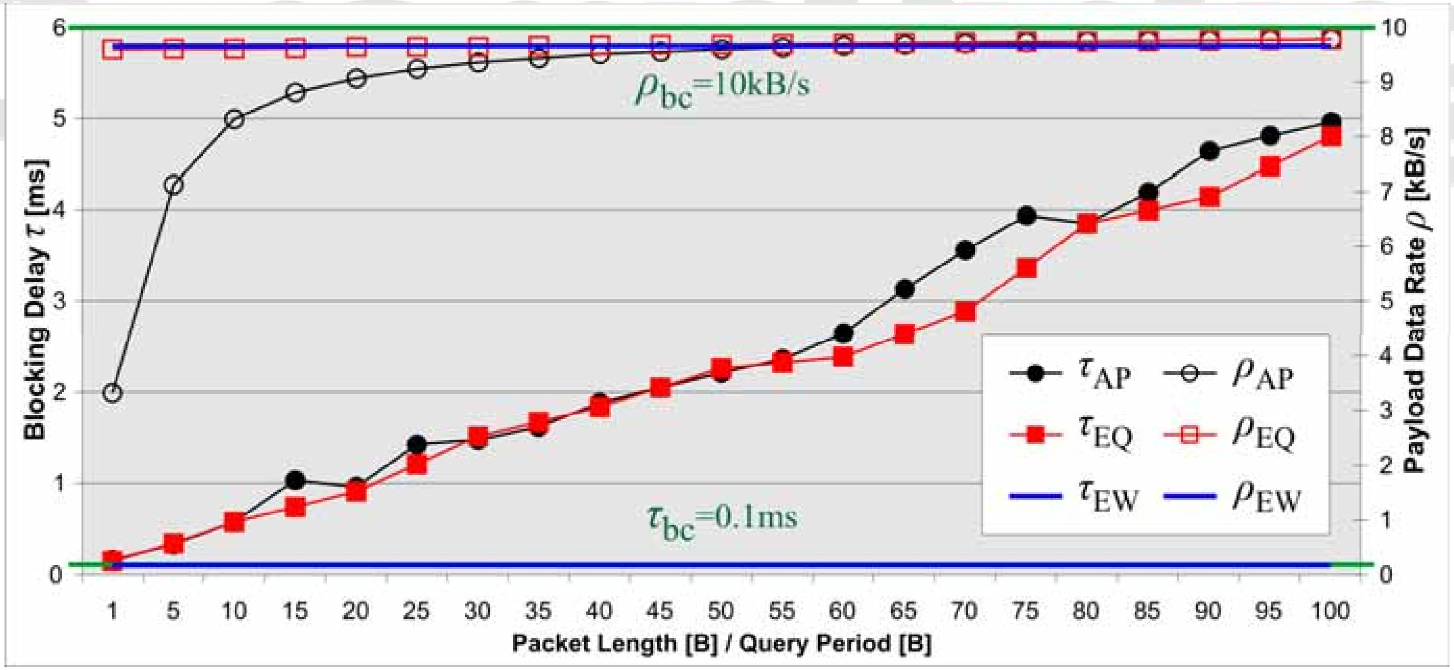
Application Example – Streamtest

Task S shares a common data bus with two time critical tasks M, R.

- S requires long term allocation of the bus.
- M, R require short but sporadic access to the bus.

Test Modes:

- **AP**: Atomic Packets (regular stream interruption, 2B for header/trailer)
- **EQ**: Explicit Querying (regular check, release only if necessary)
- **EW**: Early Wakeup (S is only resumed in case of a hint)



DynamicHinting

DynamicHinting

IV. Conclusion & Outlook

Current and Future Work...

Dynamic Hinting – Conclusion & Outlook

Dynamic Hinting:

Analyzes the current resource situation to provide tasks with information about their spurious blocking of more important tasks.

- On demand resource de-allocations become possible!
- Blocking delays (even BPI) can be reduced significantly.
- Better accounting for the intended task priorities.
- Deadlock-Recovery

➔ Implementation of cooperative tasks facilitates compositional software-design & real-time operation! ➔

Current / future work:

- Adjust acceptance of hints to the current system situation (TUFs)
- Remote resource management in distributed systems (WSAN)
- Application of model checking in systems with Dynamic Hinting

DynamicHinting

- End -

Thank you for your attention.

Questions?