# Shared Resources in Multiprocessor Systems
# -
# Modeling Concepts and Observations

Andreas Schranzhofer

Mircea Negrean

Simon Schliecker

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

artist

Technische Universität
Braunschweig

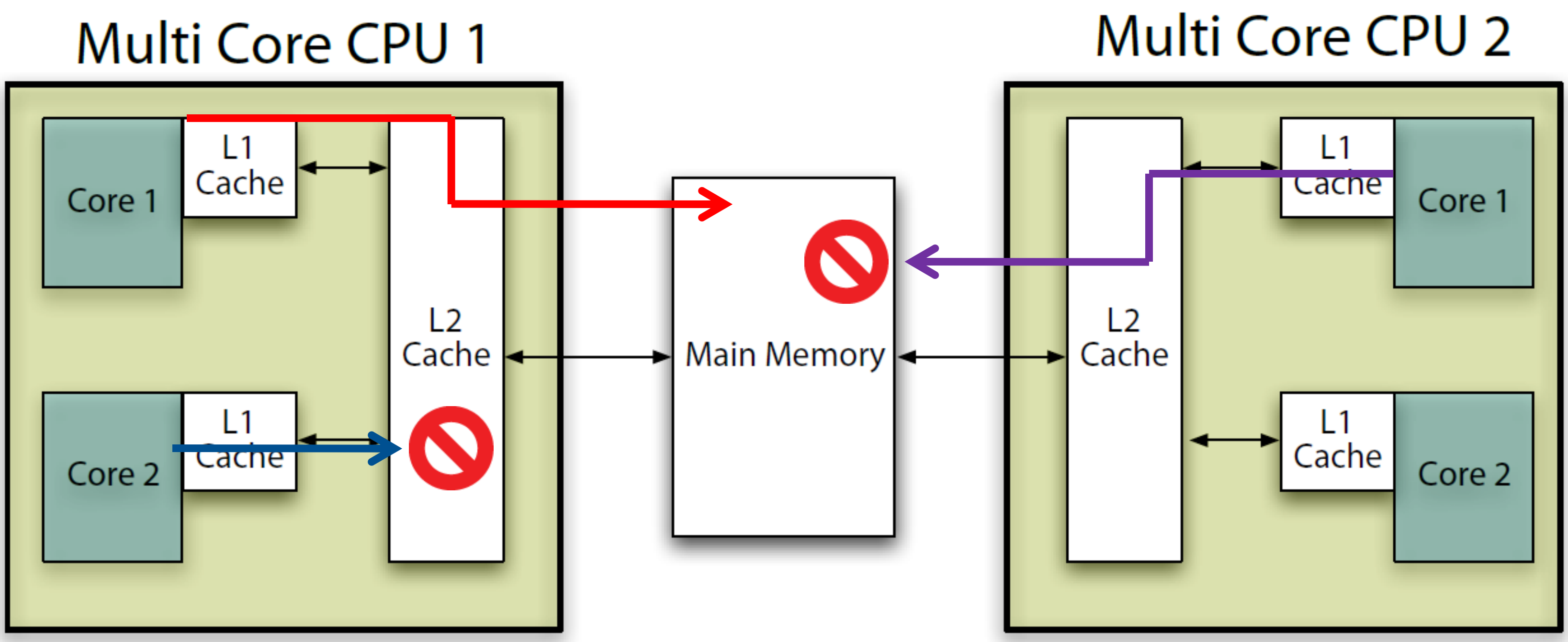
# Outline

- Multiprocessor Systems with Shared Resources
  (Investigated Scope ETHZ / TUBS)
- Remaining Sources of Overestimation
- Improvements with the aid of today‘s model
- Suggestions of improved model
- Conclusion

Interferences:
**CPU1/Core2** blocked by **CPU1/Core1** on L2 Cache
**CPU2/Core1** blocked by **CPU1/Core1** on Main Memory
**CPU1/Core2** blocked by **CPU2/Core1** on Main Memory

# Motivation (1)

- COTS Systems use shared resources (Memory, Bus)
- Multiple entities competing for shared resources
  - waiting for other entities to release the resource
  - accessing the resources



# Motivation (2)

Multi-Core Architecture with shared resource

- shared memory, communication peripherals, I/O peripherals

Blocking due to Interference on shared resource

- Depends on structure of tasks on the cores
- Depends on blocking vs. non-blocking execution semantics
- Depends on arbitration policy on the shared resource

# Possible Scheduling Setups

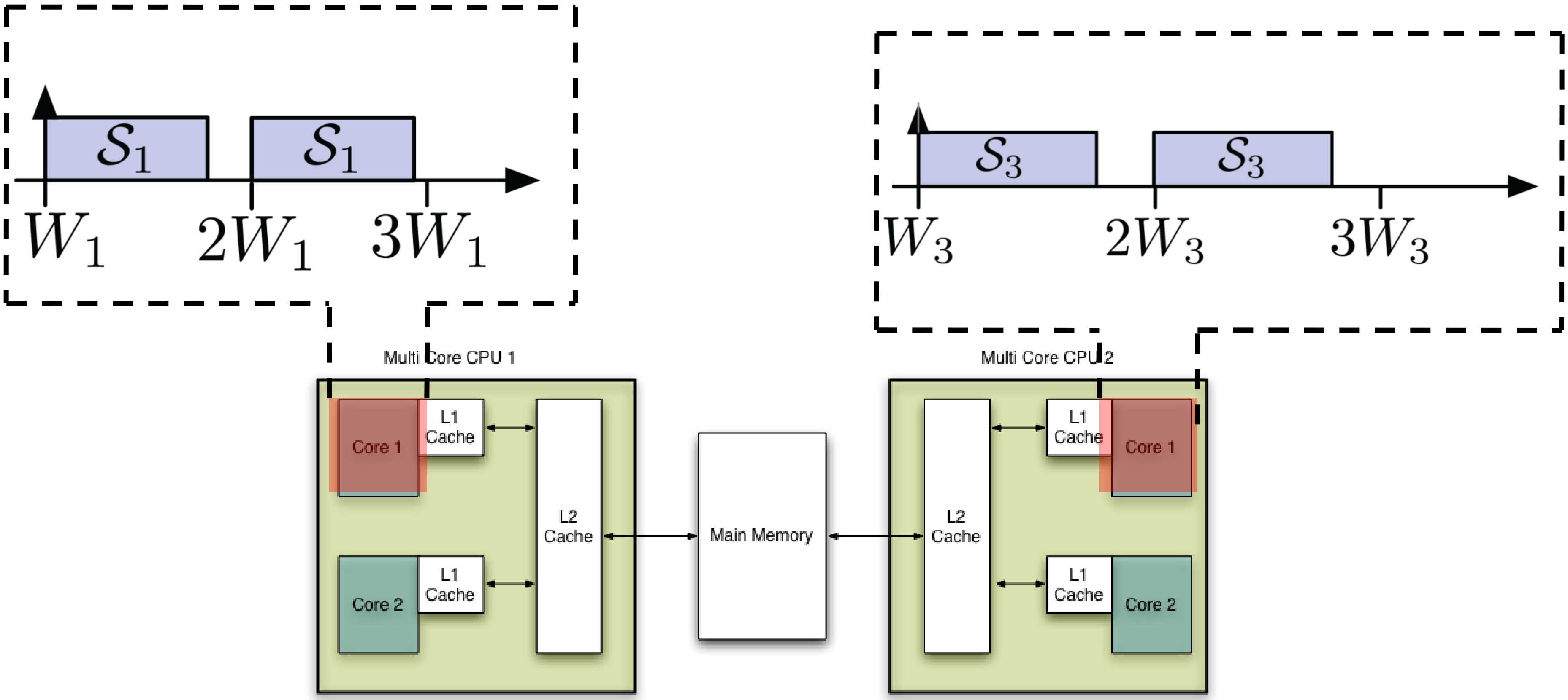
Local scheduling

- static execution order
- time-driven, event-driven
- priority based
- preemptive vs. non-preemptive
- stalling vs. suspending
- …

Shared resource arbitration

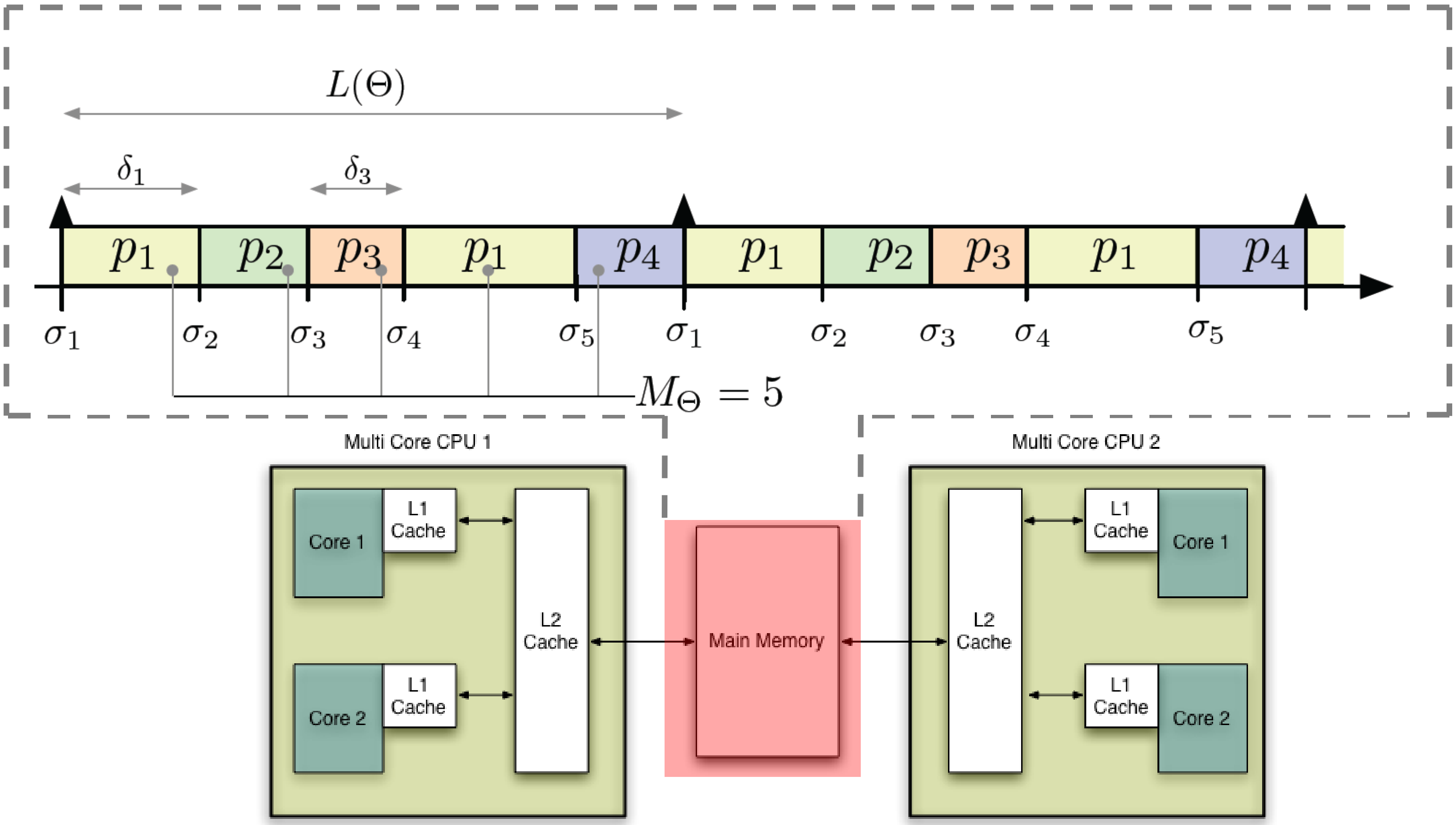
- static time-driven (TDMA) – eliminating interference
- dynamic: priority based, FCFS
- …

# Static execution on the processing element



# TDMA on the shared resource

**Independence between tasks single source of interference**

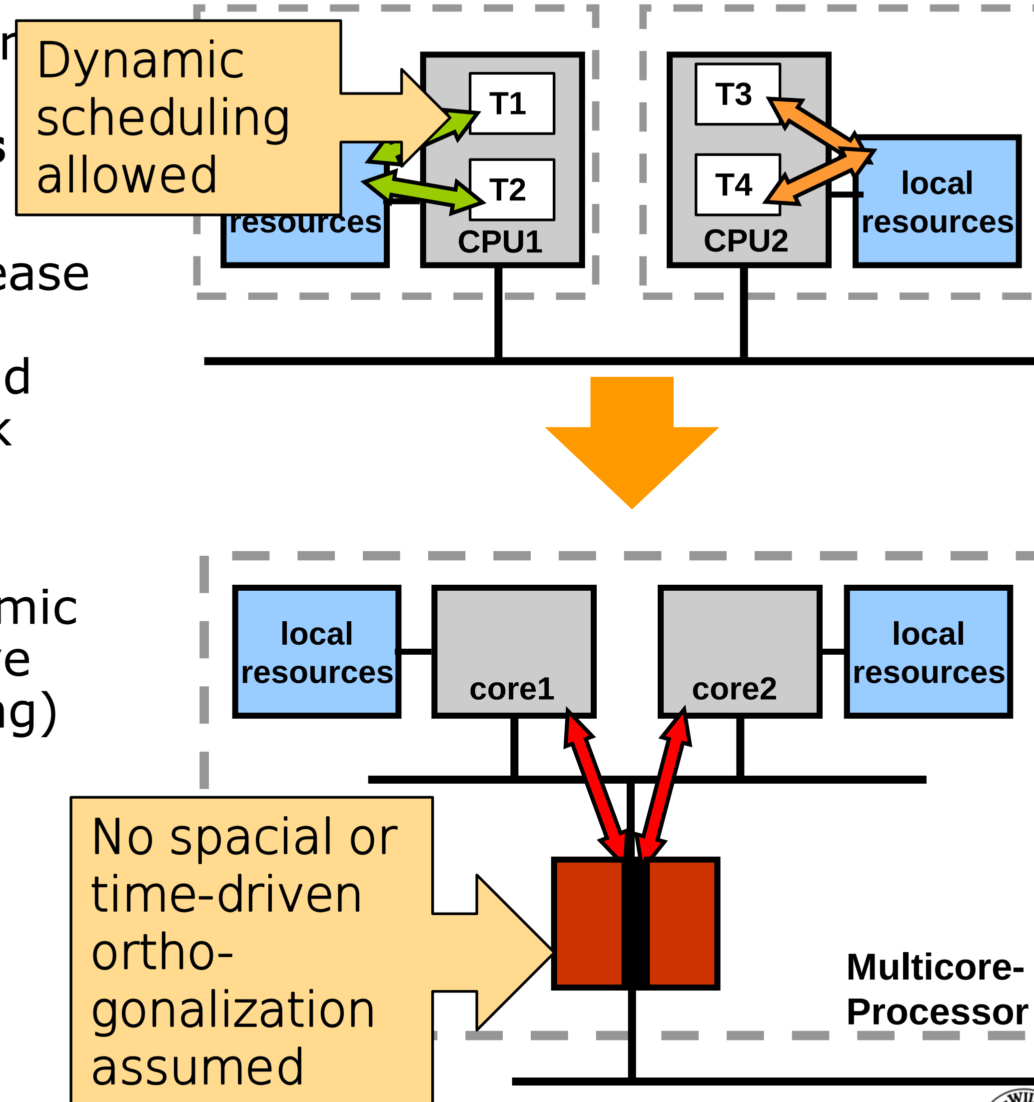


# Worst-Case Response Time

- Depends on the arbitration policy on the shared resource
- Depends on the resource access model [DAC 2010]
  - Uncertainty, when resource accesses happen
- Static arbitration (TDMA) [RTAS 2010]
  - Cores only interfere with the TDMA arbiter
  - Problem is already hard
- Dynamic arbitration (FCFS, RR) [DATE 2010]
  - What is the worst-case ? (non-compositionality)
  - Approximation of interference

# Motivational Setup b:

- Multicore system componen introduce a new degree of **inter-task dependencies**
- Common approach to increase predictability: static scheduling on the cores and interconnect, or single task per processor
- Common in practice: dynamic scheduling (e.g. automotive control systems, networking)



# Multicore with dynamically shared resources

To formally analyse multicore systems with dynamic shared resource arbiters, we need:

1. Traffic Models to the Shared Resource

2. Analysis of Latency of Shared Resource Accesses

3. Analysis of Impact of Shared Resource Access Latency on Task Response Time

(Not our scope today: When embedded into larger setup, find unknown input parameters with fixed-point iteration)

# Performance Analysis in the Presence of Shared Res.

- Shared Resources Delays
  - Introduction of Aggregate Resource Delays (WCET Workshop 2006)
  - Dedicated Shared Resource Analysis for M-PCP (DATE 2009)
- Dynamic Shared Resource Load Estimation
  - Improved Shared Resource Load Estimation (NewCAS 2006), also for Caches (DATE 2010)
- Analysis Dependencies
  - TII 2009: Solve Analysis dependencies in dynamic scheduling and arbitration setups
- Applications:
  - Application of Analysis to Multimedia quad-core (CODES 2008, mixed  with simulation)
  - Overview (TCAD 2008 with automotive software and networking aspects, TECS 2010 (coming soon)

# Problems

- Trade-Offs:

Dynamic schedules: efficient +, analyzability ?

Static schedules: efficient ?, analyzability +

- Wanted:

Efficient +, analyzability +

- Questions:

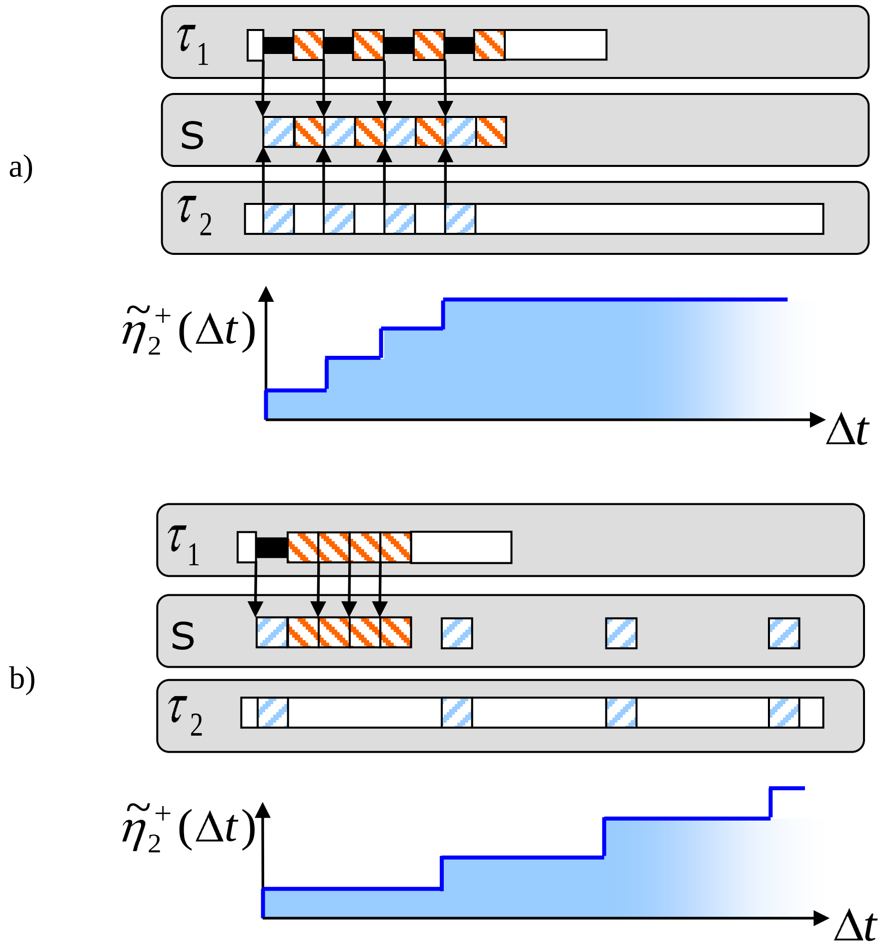
Are load models accurate enough ?

Are load models expressive enough ?

Does the analysis exploit the models expressivness ?

Do we need more structure ?

# Load Models Matter




☐ Task executing

↓ Request for S

$\tau_1$ accessing S

$\tau_2$ accessing S

$\tau_1$ stalled

- Shared Resource Delay is the sum of
  - operation time
  - **interfering operations**
  - arbitration overhead
  - …

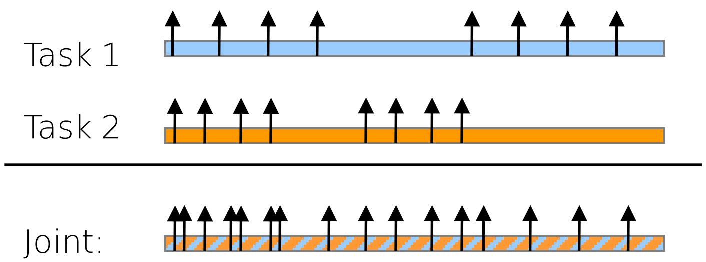
***More accurate shared resource load model***

***» » » More accurate shared resource delay computation***

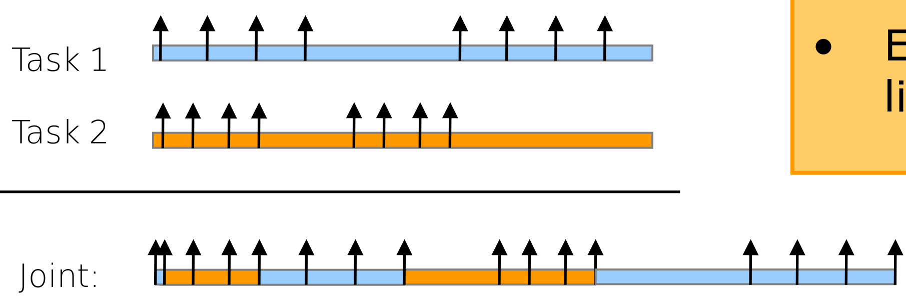
***» » » More accurate worst-case response time analysis***

# **Multiple tasks** executing on the same processor

- Obvious Bound: Sum over all requests




- Better Bound: only one task c
in time (exclusive task execut

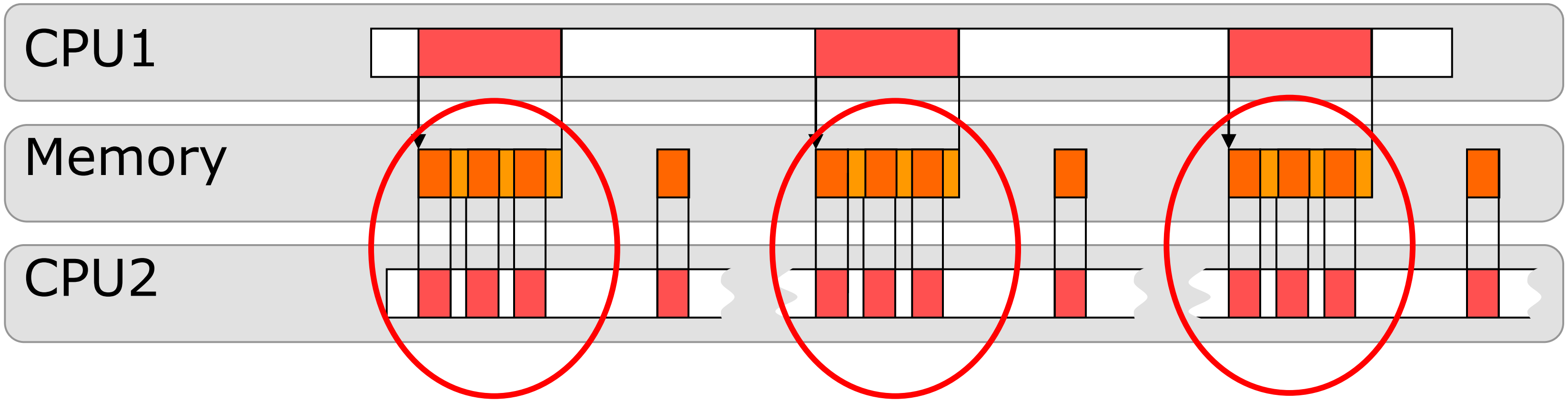



- instance of “bounded nonlinear Knapsack problem”
  → NP-hard
- Can not be constructed iteratively for increasing *n*, because a different combination may be worst-case for e.g. *n=5* than for *n=10*.
- Efficient solutions exist in literature

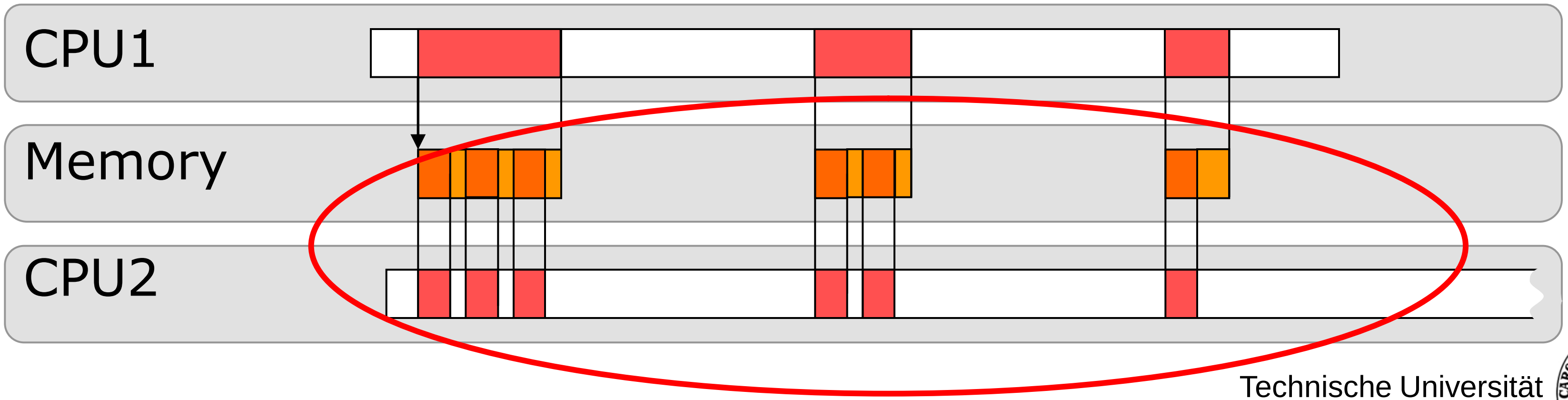
$$\tilde{\delta}^e_{T\to S}(n) = \min\{\sum_{j\in T} \tilde{\delta}^e_{j\to S}(n_j) \mid \sum_{j\in T} n_j = n\}$$

# Aggregate Busy Time Model Avoids Multiple Worst-Cases

a) Classic assumption: Each request a worst-case

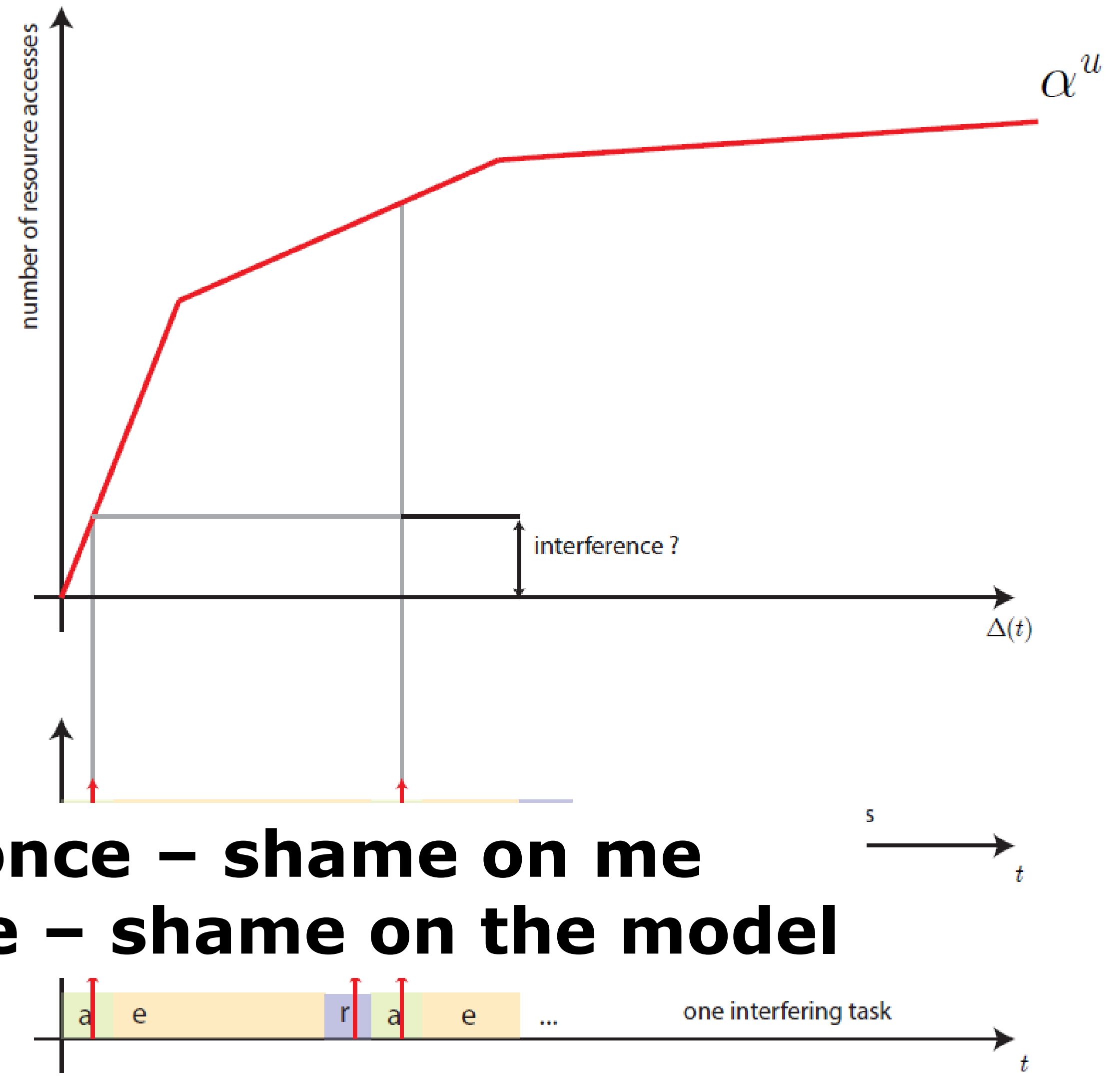



b) Improvement: aggregate request delay



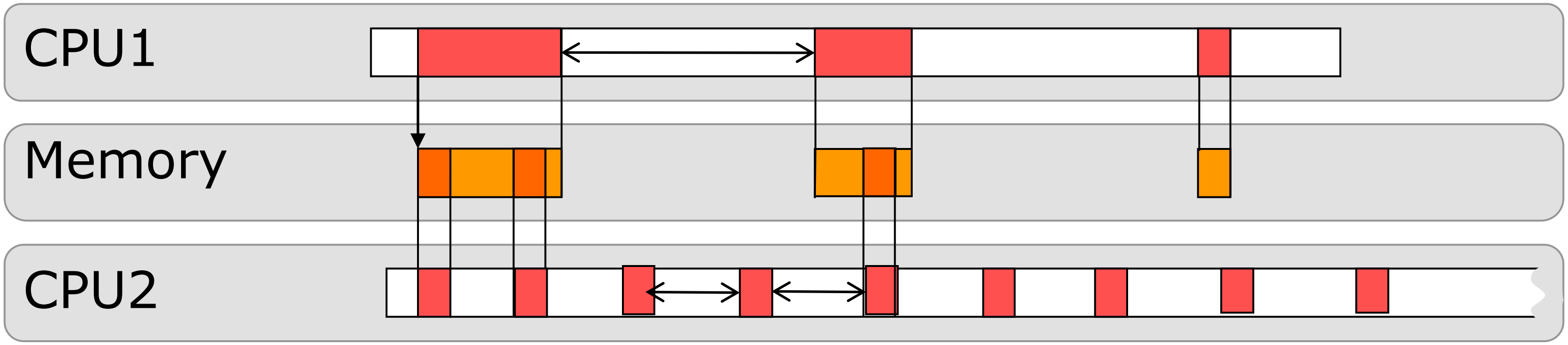
# Infeasible interference

- some interferences cannot happen
  - but are considered in the WCET analysis




**Interfere me once – shame on me**
**Interfere me twice – shame on the model**

# Observation: some interference scenarios are infeasible




Consider best case behavior to identify infeasible interference

**Scenario I: Maximum distance between interference is smaller than minimum distance of requests**

➔ Not every request can cause interference (depending on periods and phasing)

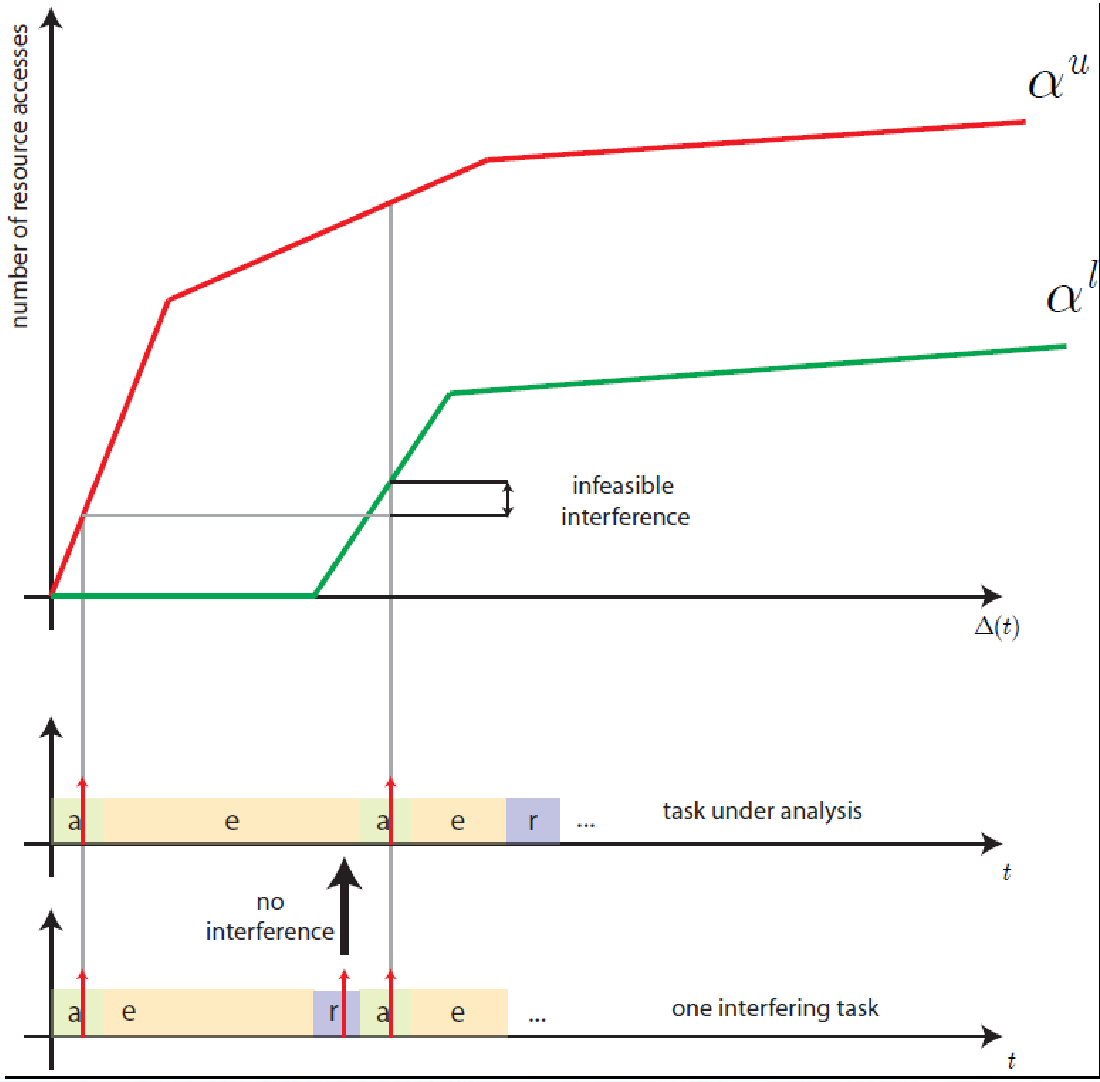
**Scenario II: Minimum distance between interference is larger than maximum distance between requests**

➔ Not every request can cause interference (depending on periods and phasing)

- Refined models should consider distance between **n** requests

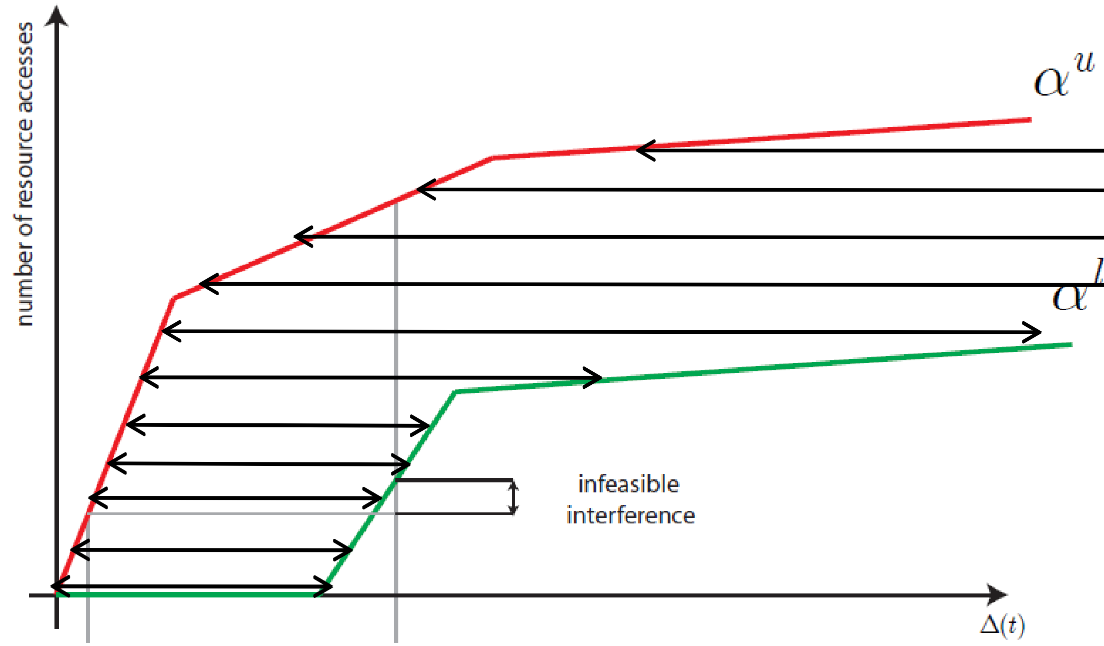
# Infeasible Interference

- Consider only feasible interference
- dyn. Programming
  - exclude accesses that cannot cause interference
- gain depends on lower curve

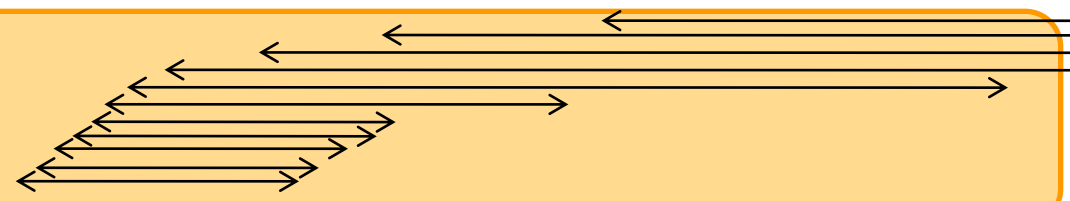


# New metric: task requests' live intervals

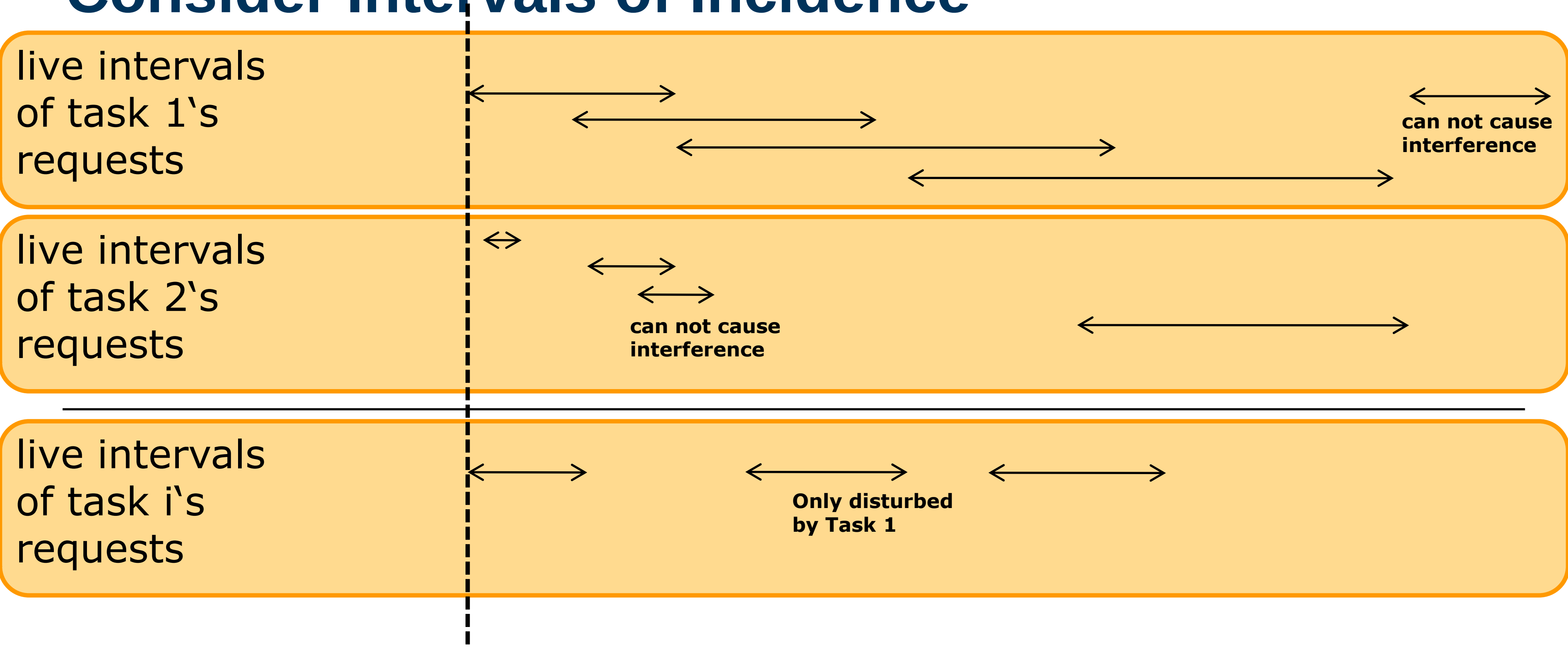
- For each of the tasks' requests, deterimine the interval within which the request may take place




live intervals
of task i's
requests



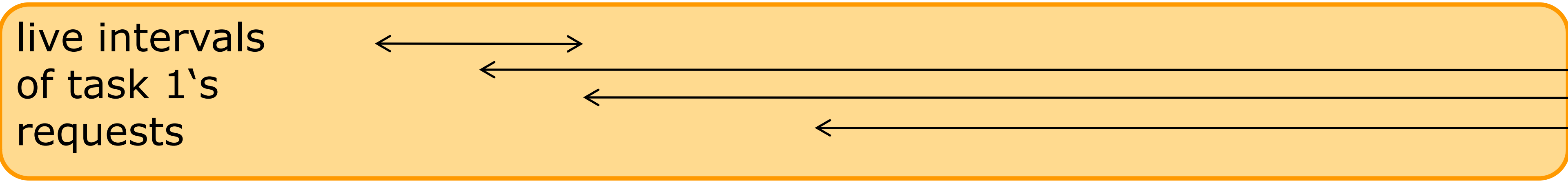
# Consider Intervals of Incidence



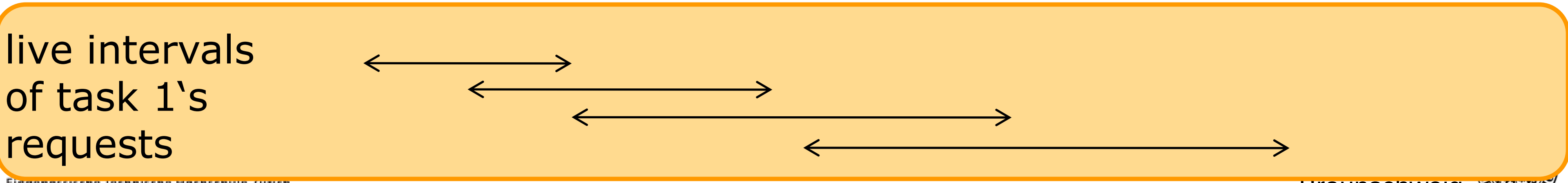

- By identifying the requests' "live-intervals", possible co-incidences and "an-cidences" may be identified
- May increase accuracy (because is spans a larger investigated time window), at the cost of complexity (when investigating different trace scenarios)

# WACI: Possibly need new metrics

- Intuitive metric: *Maximum distance between n requests*
  - If request are conditional, the minimum number of requests can be zero
  - does not deliver required information!




- More relevant: *Maximum distance between n requests,* ***if n requests are issued*****.**
  - to be explored

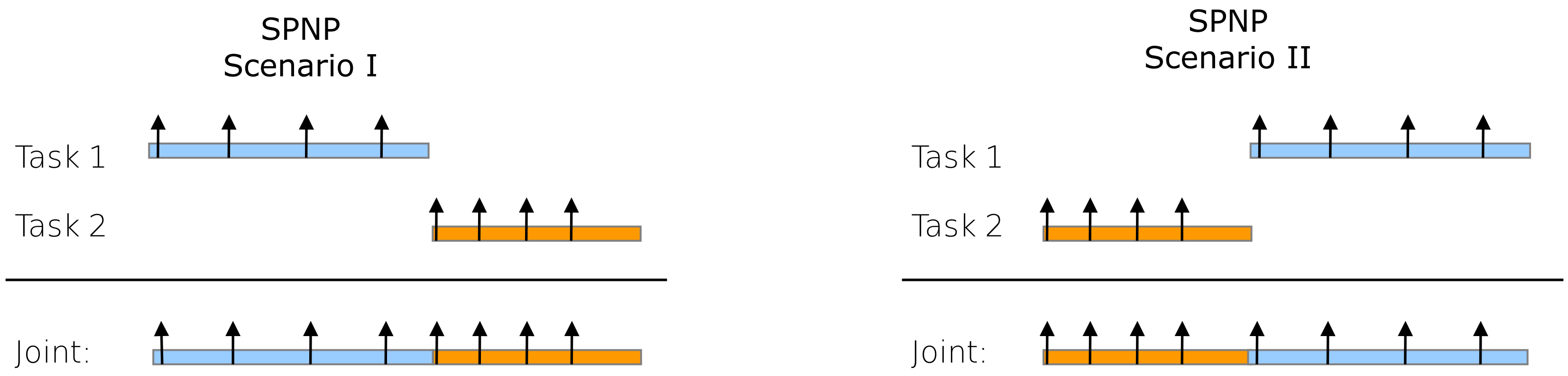



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Braunschweig

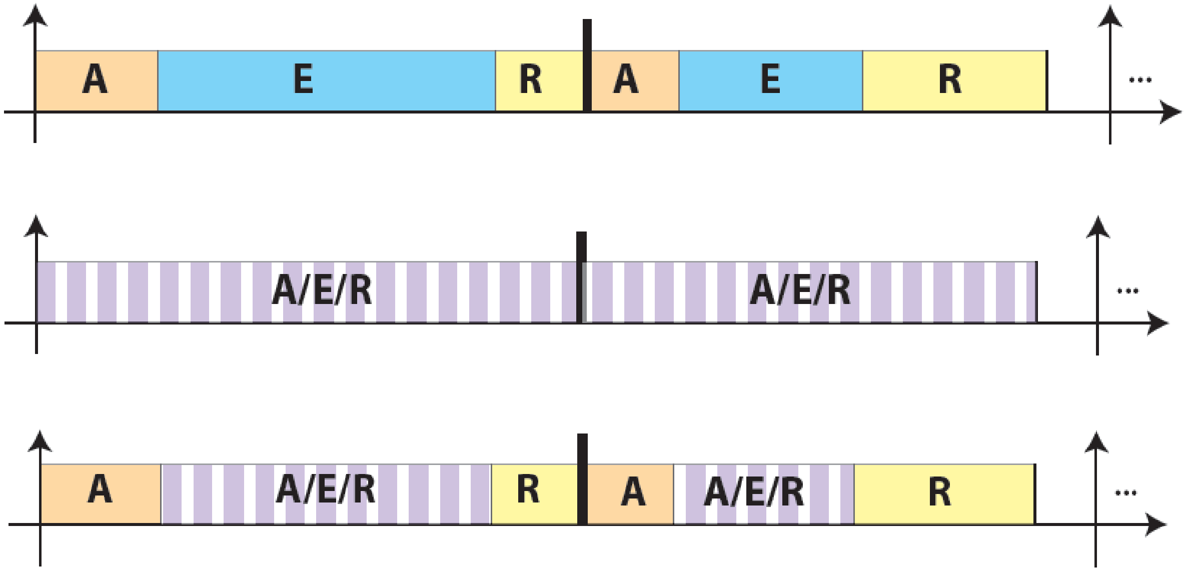
# Exploit details from more structure

- Consider execution imposed by schedulers
  - e.g. SPNP (static priority non-preemptive) scheduling



# Structured Resource Access Models

- 3 Models to specify resource accesses:
  - Dedicated Model
  - General Model
  - Hybrid Model




- 2 Models to execute superblocks:
  - Sequential
  - Time-triggered

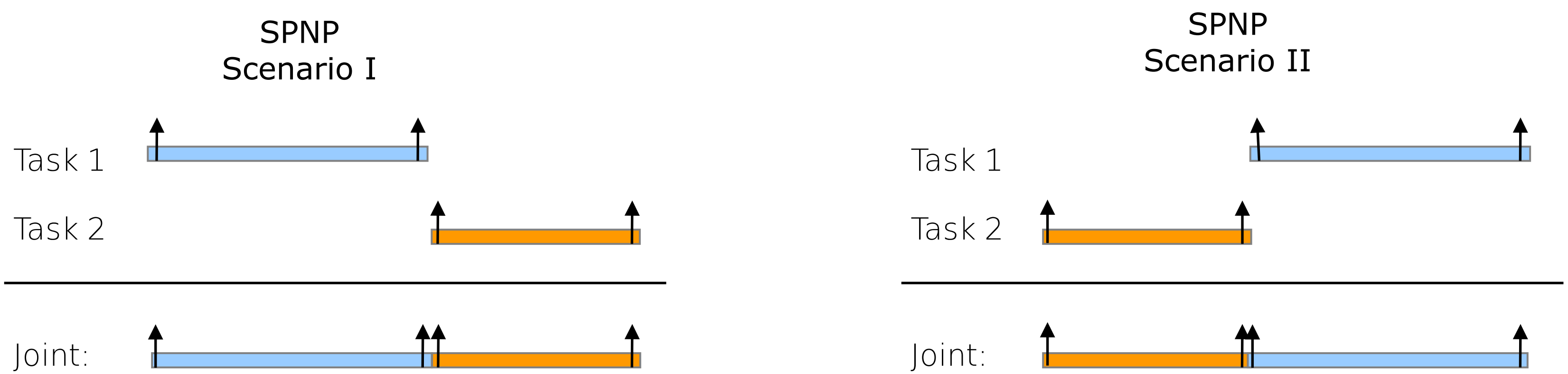
Wouldn't it be nice to support also priority-based preemptive or non-preemptive?

# Exploit details from more structure



- Consider execution imposed by schedulers
  - e.g. SPNP (static priority non-preemptive) scheduling

- Consider task execution model
  - e.g. tasks communicate at the beginning and at the end of their execution




→ Correlate local scheduling policy and task execution model to derive improved bounds of the shared resource load

# Conclusion

- Resource Sharing in Multi-Core Systems is an important issue in terms of
  - Analyzability, Predictability, Accuracy
  - Efficiency

- Increase analyzability by increasing structuring
  - Separation of computation and communication
  - Suitable arbitration on the shared resource

- Improve accuracy by including model aspects
  - Better exploit existing model ("minimum behavior")
  - Possibly new models needed to better capture behavior