TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

SymTA/S
Symbolic Timing Analysis for Systems

ARTIST2 PhD Course, June 12, DTU Copenhagen, Denmark

Razvan Racu

Arne Hamann

IDA
INSTITUTE OF
COMPUTER AND
COMMUNICATION
NETWORK ENGINEERING

Day schedule

0900 – 0945 Introduction to system performance verification

1000 – 1045 Compositional performance analysis

1100 – 1200 Hands-on tutorial 1: Basics SymTA/S

1330 – 1415 Sensitivity analysis

1430 – 1515 Design space exploration and robustness optimization

1530 – 1630 Hand-on tutorial 2: Advanced SymTA/S features

1630 – 1700 Discussion

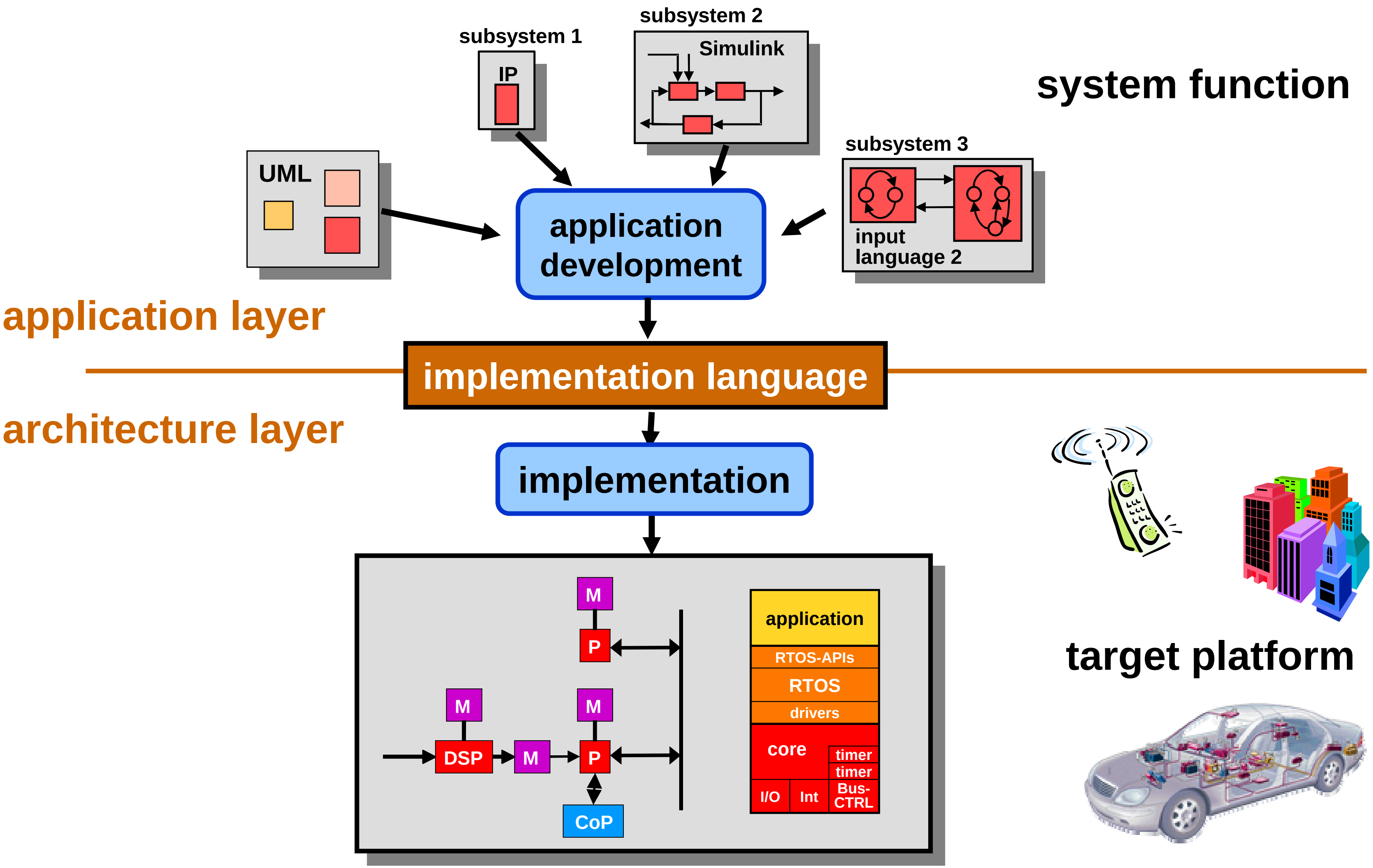
System design challenges

Functional vs. performance verification

- Separate function verification from performance verification
  - functional verification/test determines functional correctness independent of the target architecture
  - performance verification/test determines platform adherence to
    - load conditions and response times (deadlines)
    - jitter bounds
    - buffer sizes

This presentation is about performance verification !!

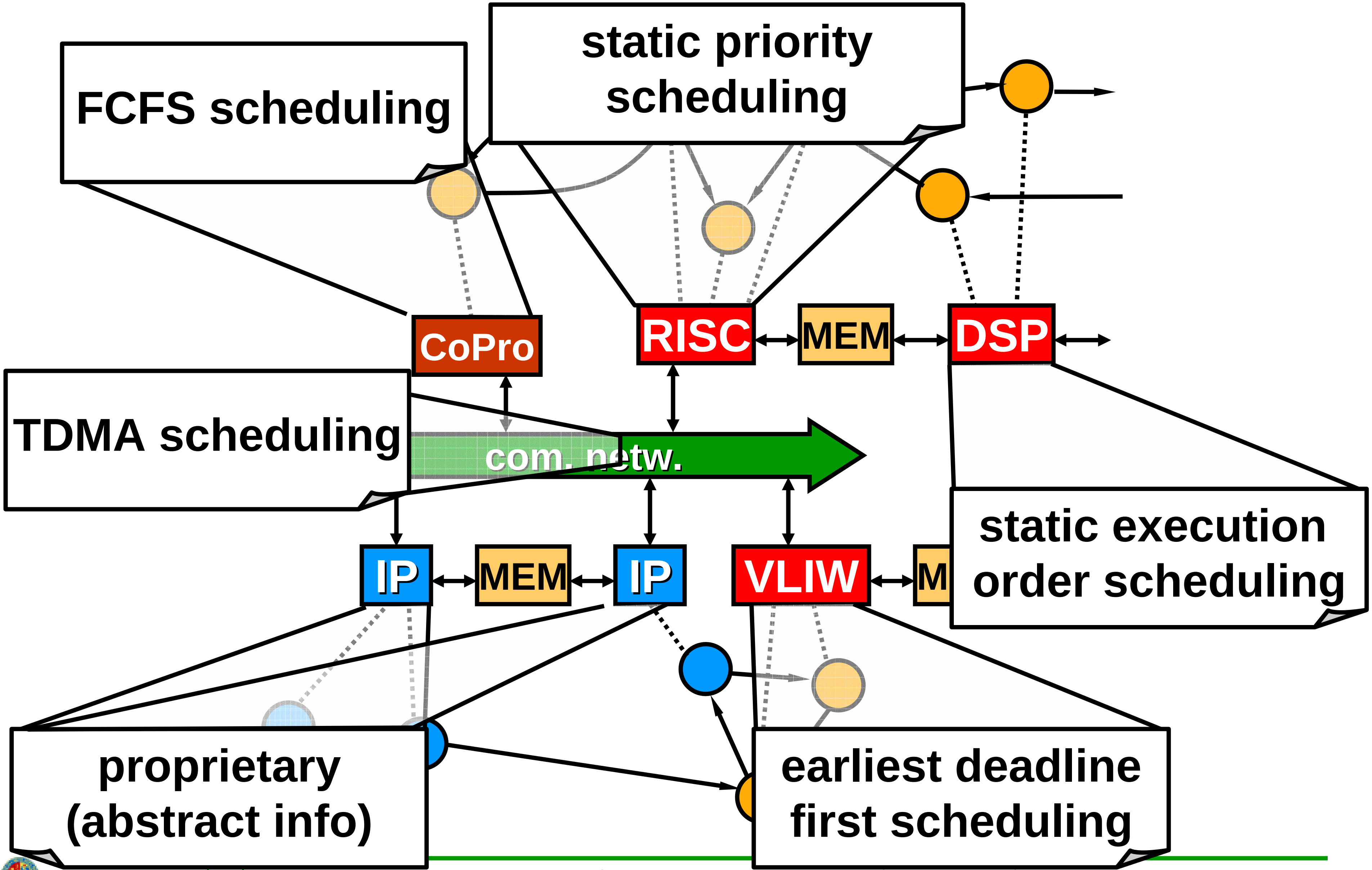
Introduction



Embedded system platform properties

- ES platforms are heterogeneous
  - components
  - networks
  - communication
  - scheduling (static, dynamic, event-, time-driven, ...)
  - ...
- Heterogeneity results from
  - hardware and software component specialization (cost, power, dependability)
  - HW/SW reuse

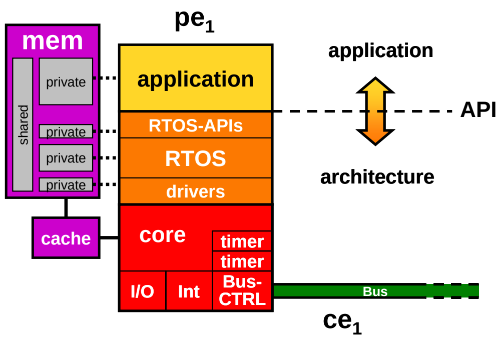
Heterogeneous resource sharing



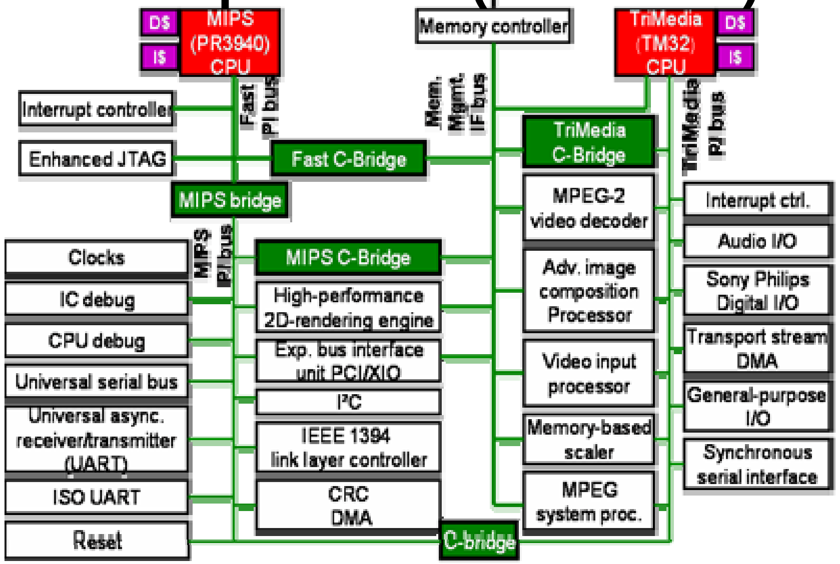
Exemple 1 : MPSOC

- **Heterogeneity resulting from**
  - **hardware and software component specialization**
  - **reuse**

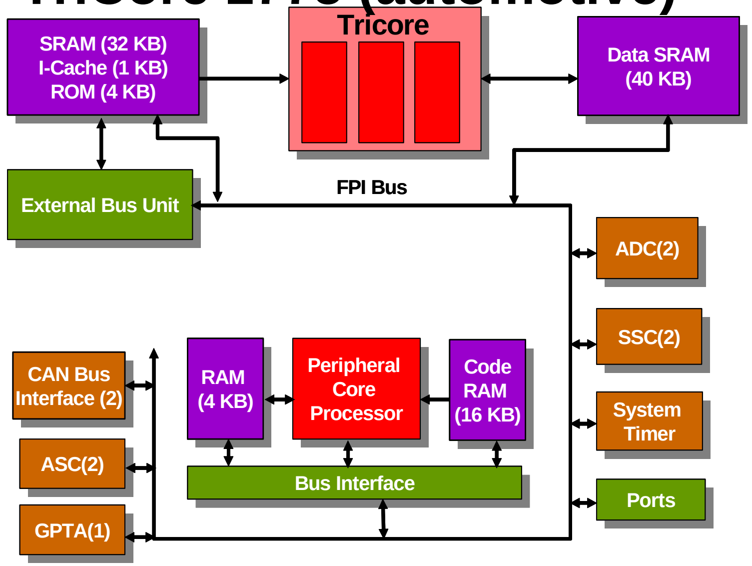
multilayered SW



Philips VIPER (consumer)

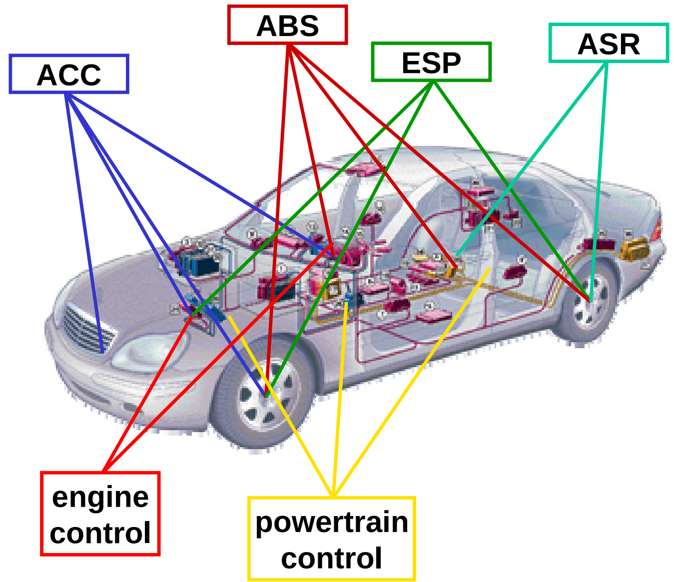


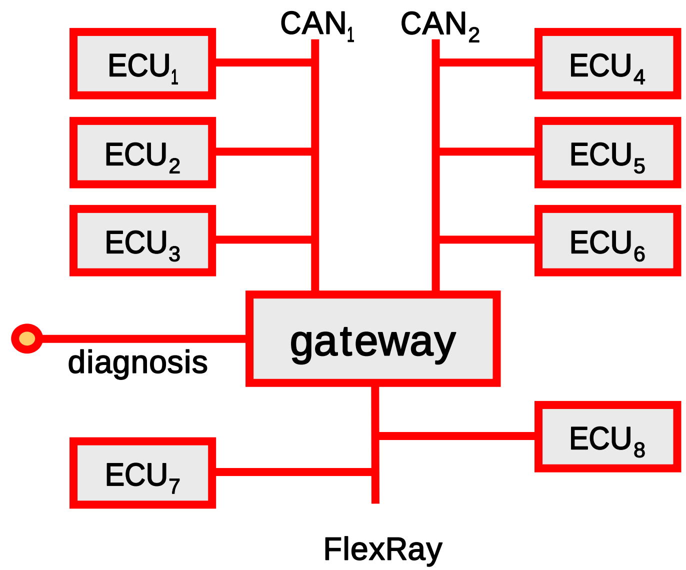
TriCore 1775 (automotive)



Example 2: Automotive Platform

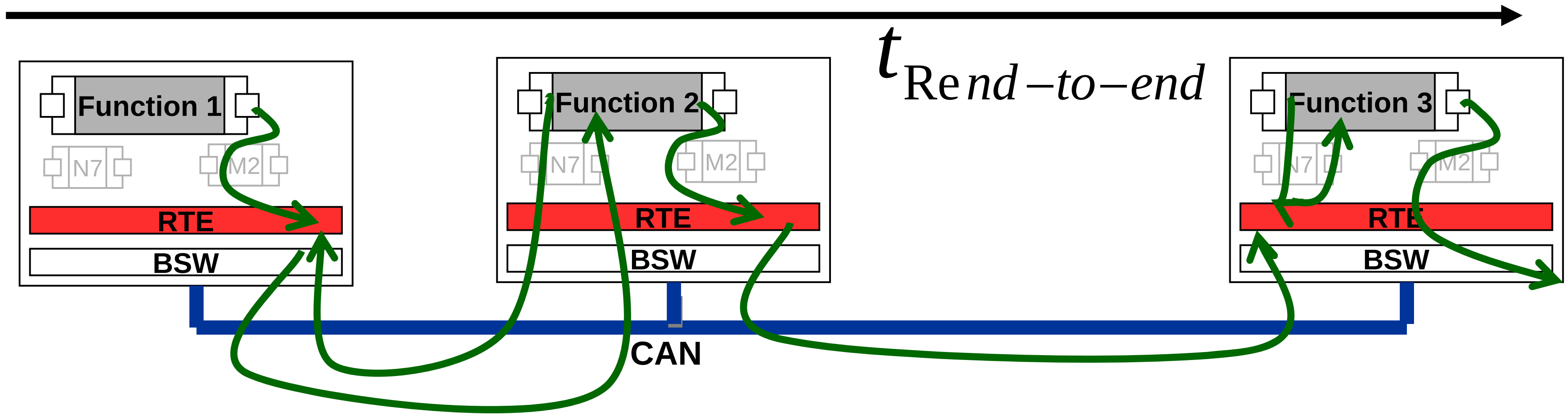
- Heterogeneous
  - 50+ ECUs
  - many suppliers
  - several RTOSes and protocols
  - strongly networked
- Complex
  - end-to-end deadlines
  - hidden dependencies
  - global memories





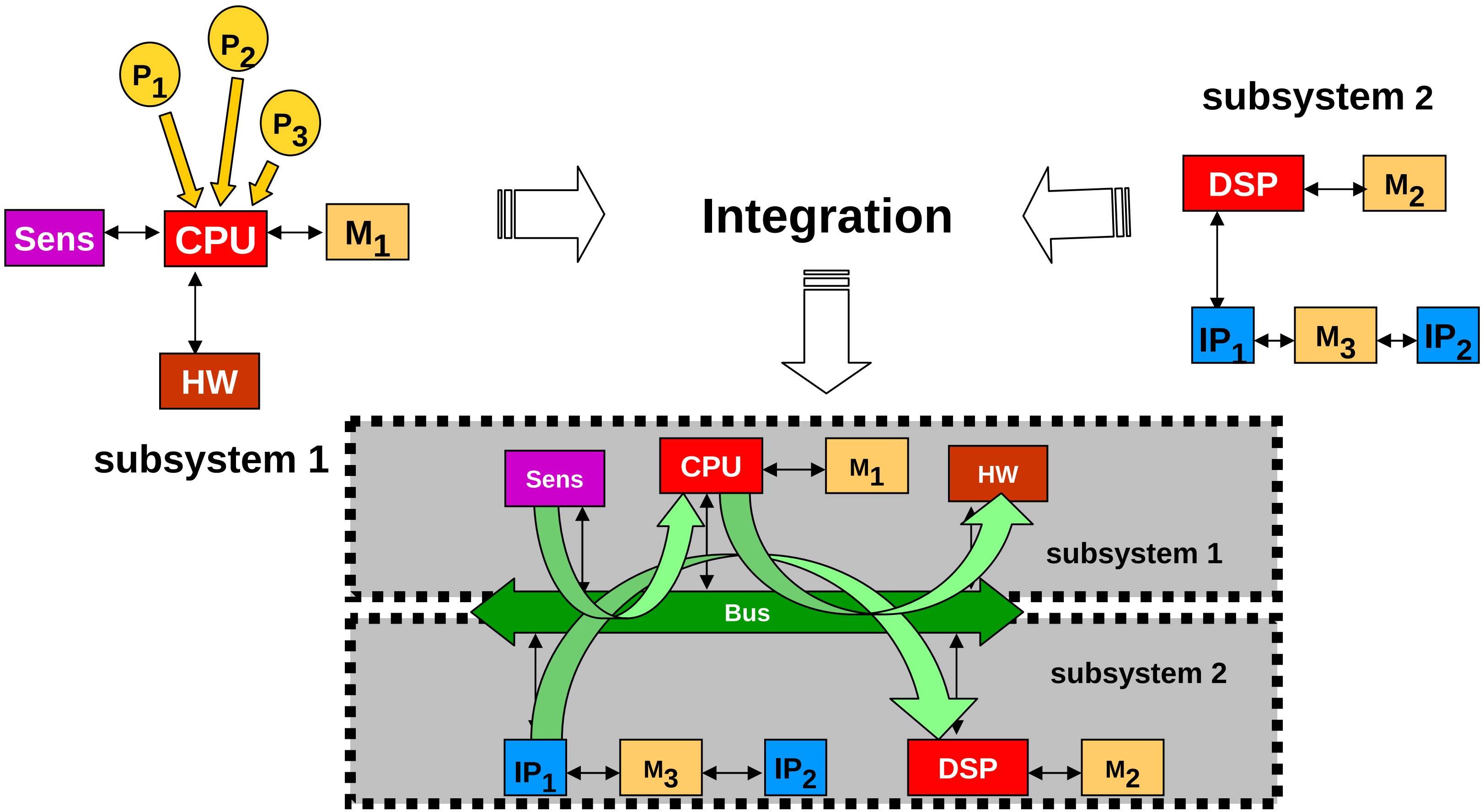
End-to-end times do not easily compose

$$t_{RFunction1} + t_{RFunction2} + t_{RFunction3} \neq t_{R\,end-to-end}$$



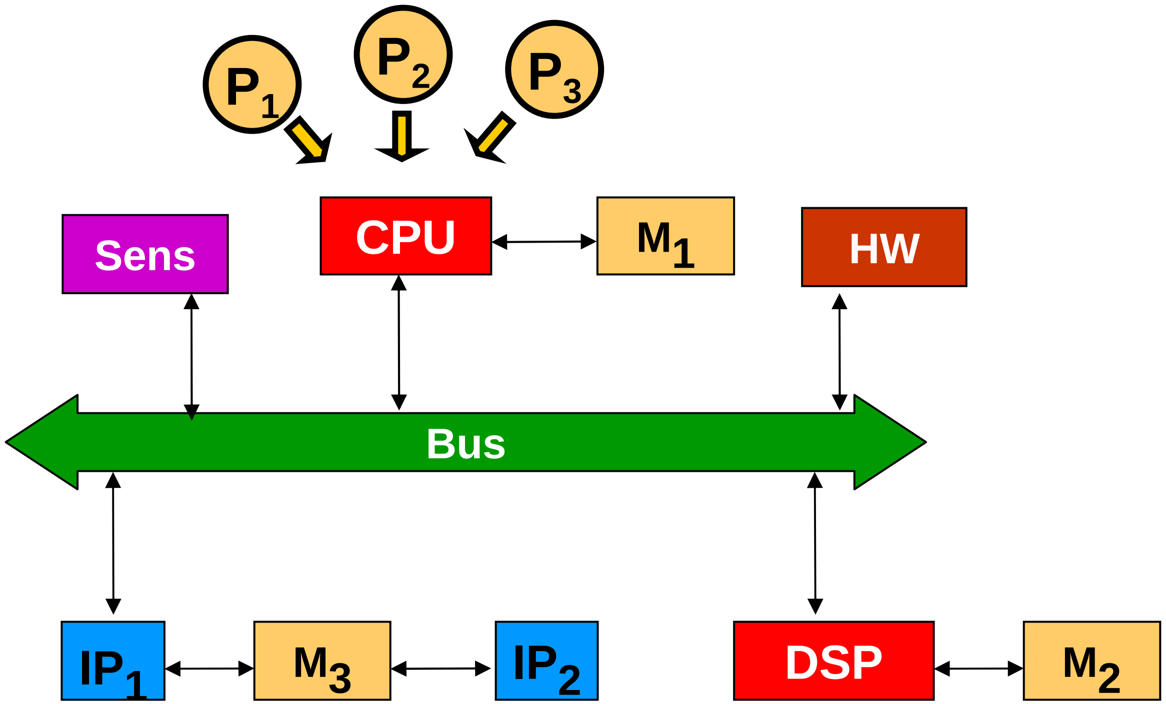
Design as integration problem

- **System design is to a large extend an integration problem**

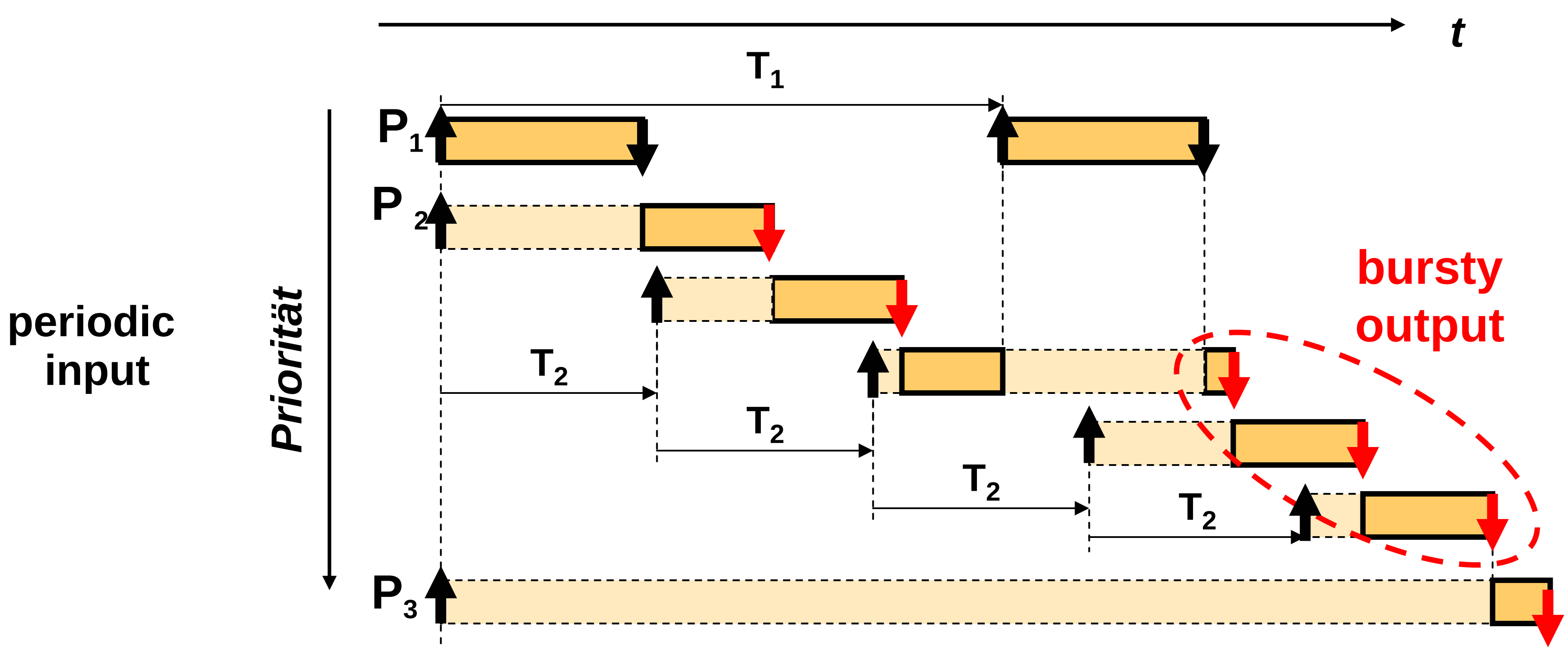


Coupling effects – a closer look

- **Example: 3 periodic tasks on CPU send data over the bus**
- **Static priority scheduling on CPU: P1 > P2 > P3**



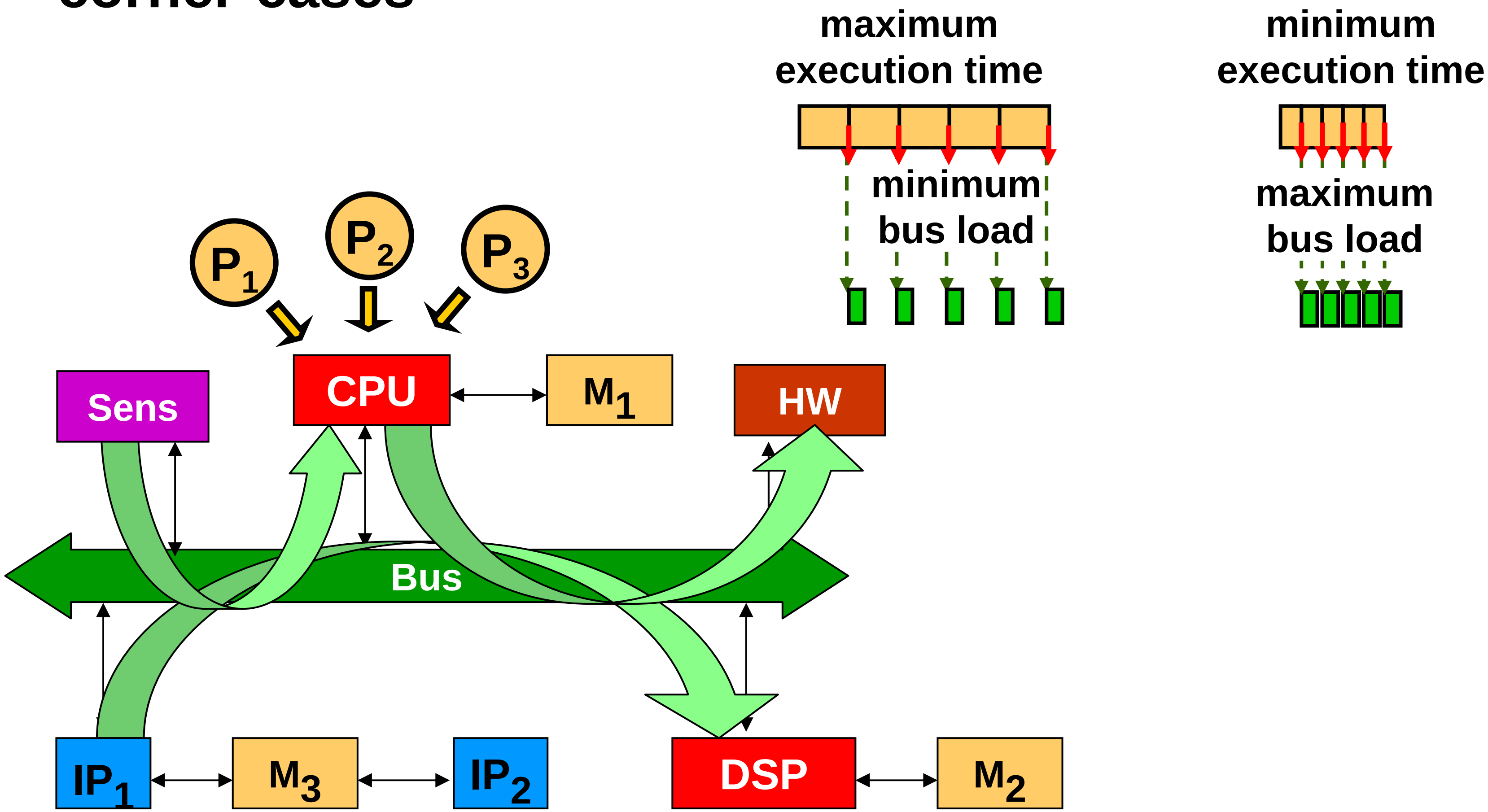
Coupling effects – creation of bursts



- **Complex execution traces with dynamic behavior**
- **Burst events at the output**
- **Consequences: transient overload, missed deadlines, data loss, ...**

Scheduling anomalies

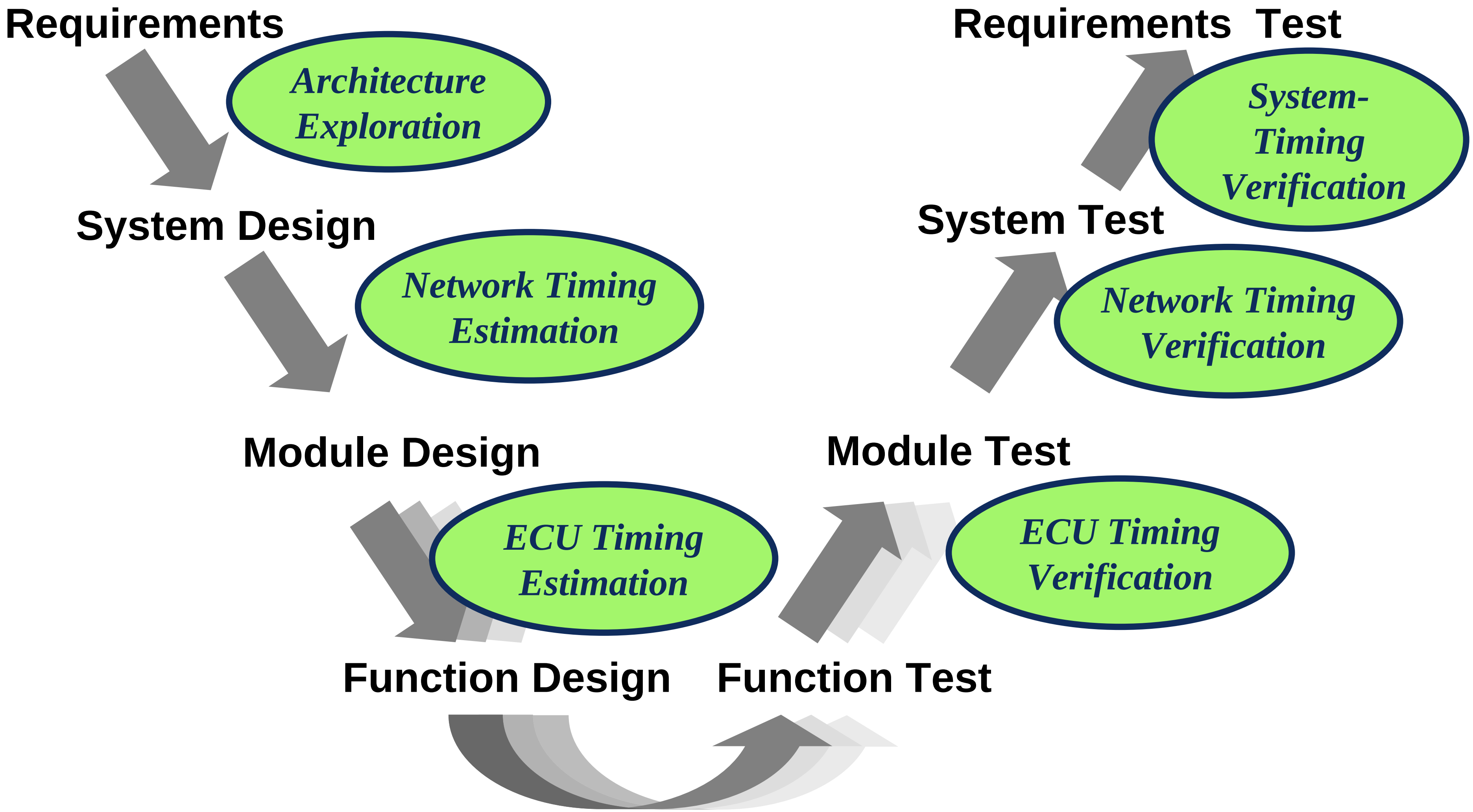
- **System corner-cases different of component corner-cases**



Key platform design challenges

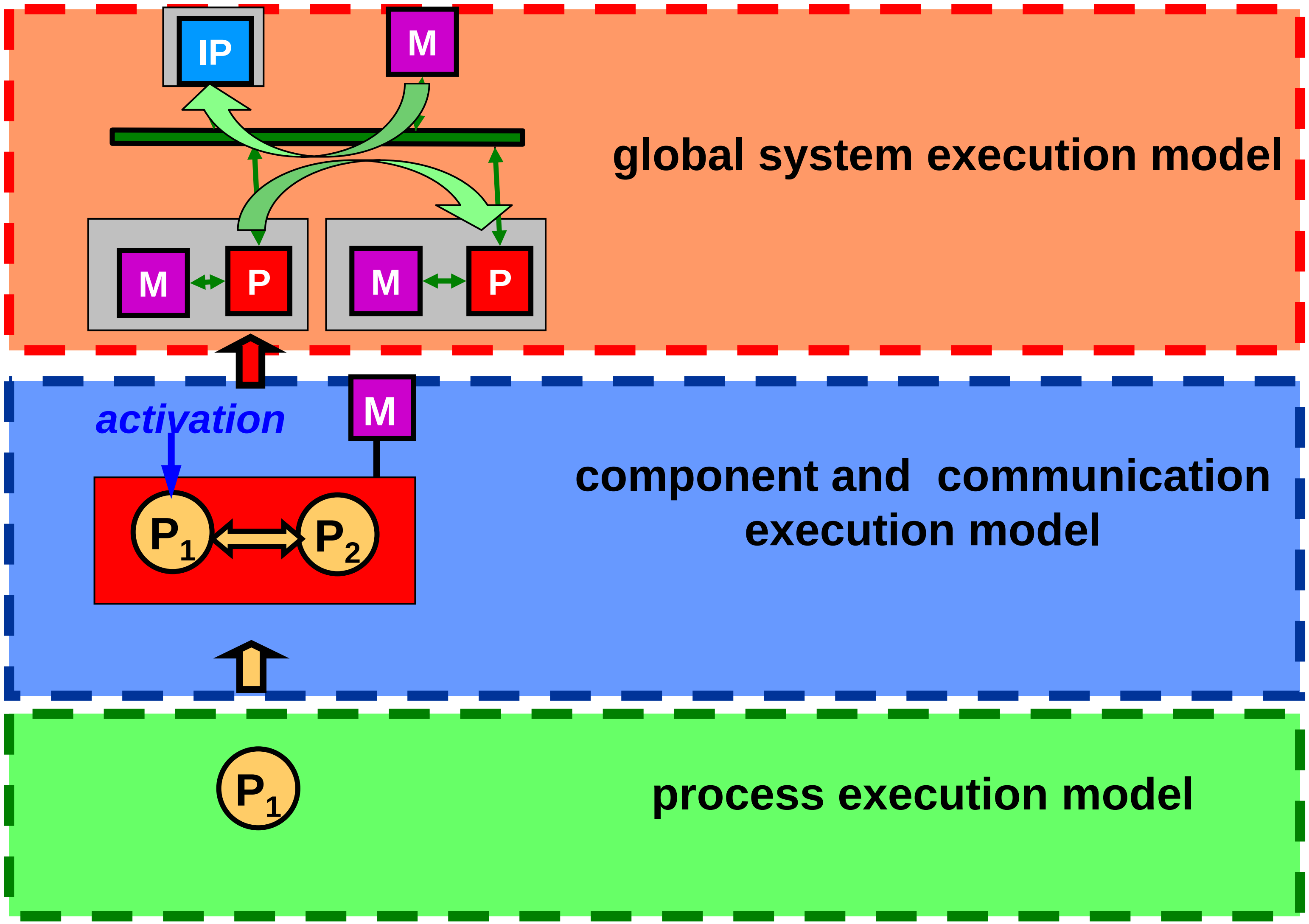
- **Increasing system complexity**
  - **from single processor to multi-processor (MpSoC)**
  - **from buses to networks (NoC)**
- **Complex dependencies and modifications threaten design robustness**
- **Global end-to-end constraints added for control applications**
- **Integration under optimization requirements**
  - **cost (memory, power, …)**
  - **robustness**
  - **extendibility – consider upcoming features, SW updates, platform updates in product lines**
- **Reliable system integration is key requirement**
  - **Performance verification required at every design stage**

Timing is everywhere

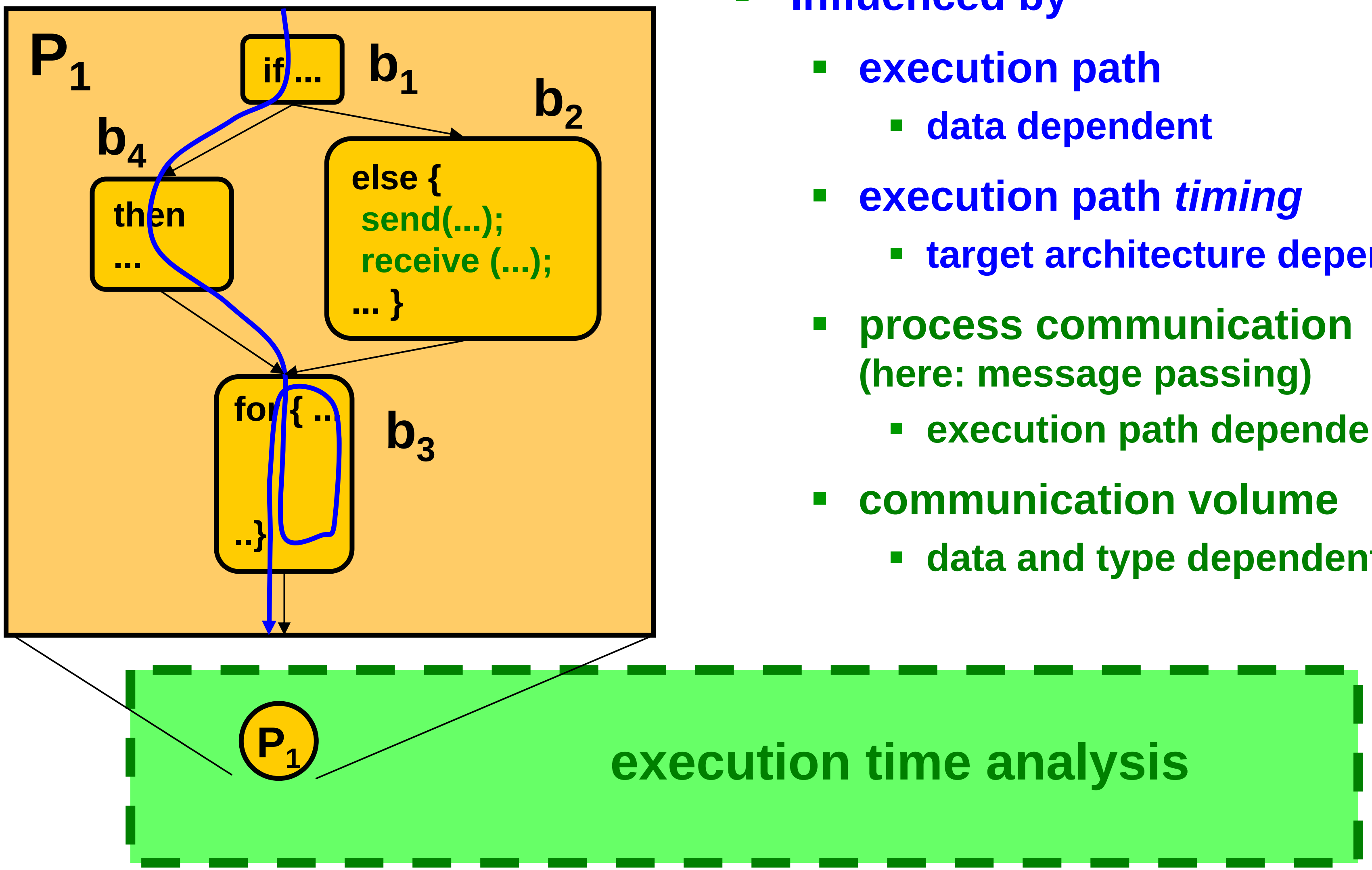


Performance verification flow

Target architecture performance – general view



Process execution model



- **Influenced by**
  - **execution path**
    - **data dependent**
  - **execution path *timing***
    - **target architecture dependent**
  - **process communication (here: message passing)**
    - **execution path dependent**
  - **communication volume**
    - **data and type dependent**

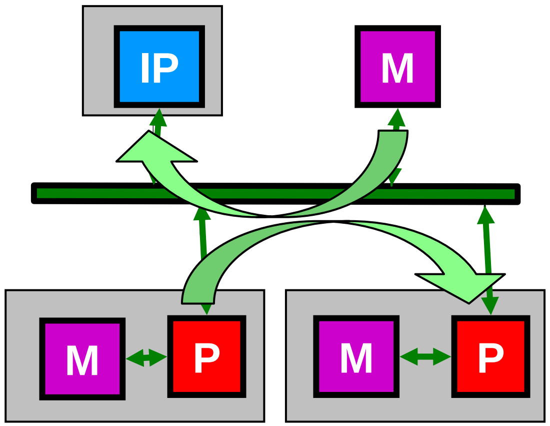
Process timing and communication

- **State of industrial practice - simulation/performance monitoring**
  - **trigger points at process beginning and end**
  - **data dependent execution → upper and lower timing bounds**
- **simulation challenges**
  - **coverage?**
  - **cache and context switch overhead due to run-time scheduling with process preemptions**
- **Alternative - formal analysis of individual process timing**
  - **provides conservative bounds**
  - **serious progress in recent years**

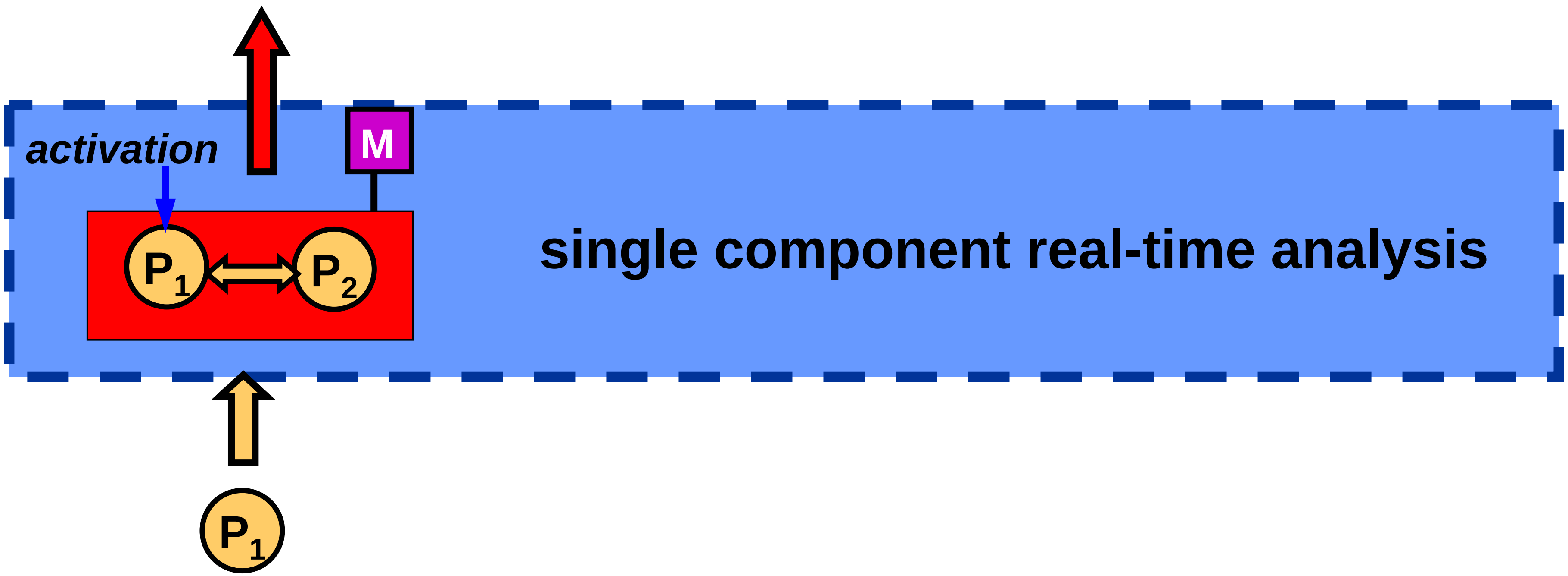
Formal process execution time analysis

- Active research area with dedicated events (e.g. Euromicro WS)
- Formal analysis using simple processor models
  - Li/Malik (Princeton) (95): Cinderella
- Detailed execution models with abstract interpretation
  - Wilhelm/Ferdinand (97 ff.): commercial tool AbsInt
- Combinations with simulation/measurement of program segments
  - Wolf/Ernst (99): SymTA/P
- All tools provide (conservative) upper execution time bounds (WCET) or time intervals (WCET/BCET)

Component and communication execution model



- Influenced by
  - resource sharing strategy
  - process activation



Component and communication execution model

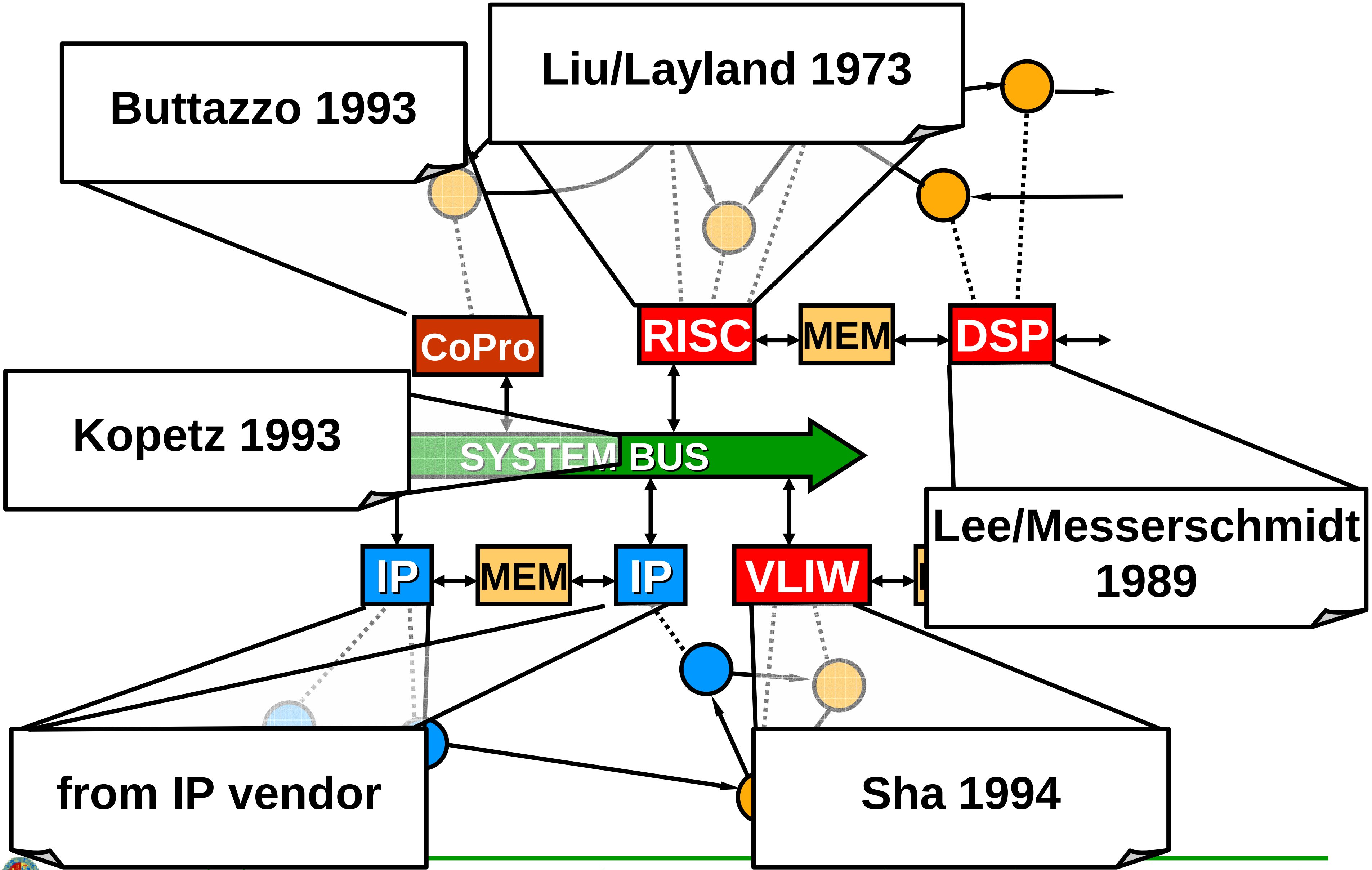
- Resource sharing strategy

  → process and communication scheduling

  - static execution order
  - time driven scheduling
    - fixed: TDMA
    - dynamic: Round-Robin
  - priority driven scheduling
    - static priority assignment: RMS, SPP
    - dynamic priority assignment: EDF

- Timing depends on environment model
  - determines frequency of process activations or communication

Scheduling Analysis Techniques

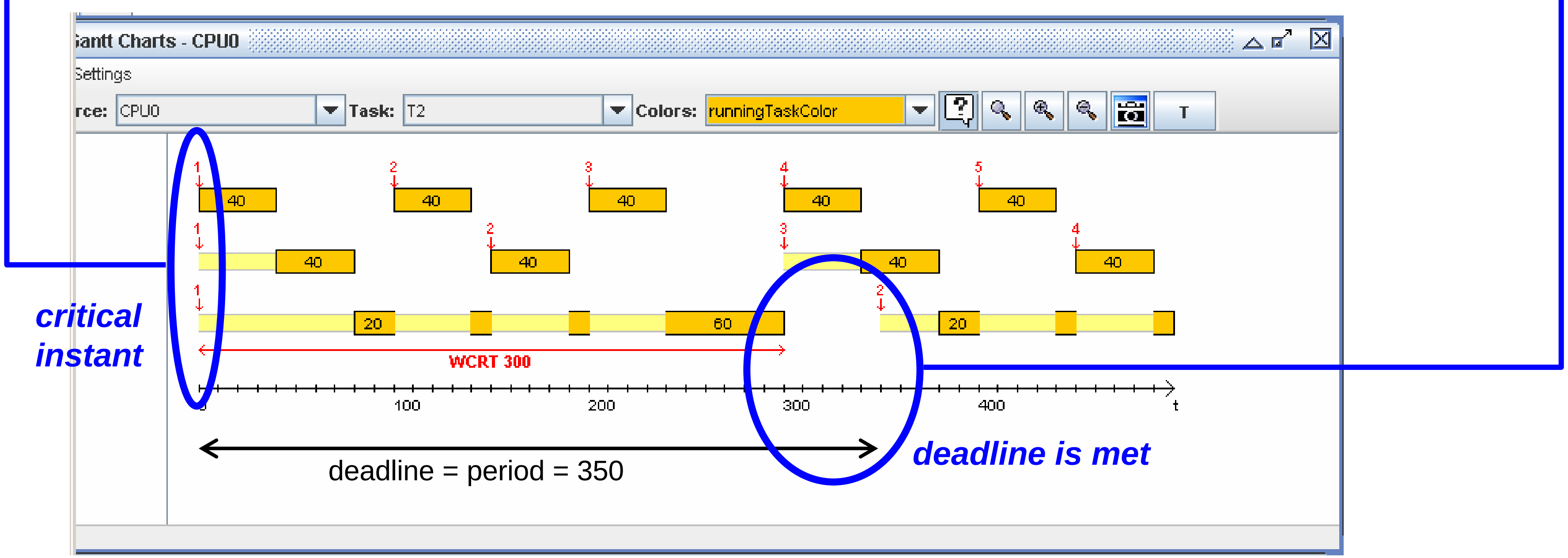


Example: Rate Monotonic Scheduling (RMS)

- Very simple system model
  - periodic tasks with deadlines equal to periods
  - fixed priorities according to task periods
  - no communication between tasks
  - (theoretically) optimal solution for single processors
  - several practical limitations but good starting point
- Schedulability tests for RMS guarantee correct timing behavior
  - processor utilization (load) approach
  - response time approach (basis for many extensions)

RMS Theory – The response time approach

- **Critical instant:**
  all tasks start at t=0 („synchronous assumption“ to ensure maximum interference in the beginning of task execution)
- **when each task meets its first deadline, it will meet all other future deadlines (proof exists!)**
- **test by „unrolling the schedule“ (symbolic simulation)**



RMS Theory – The response time formula

fix-point problem

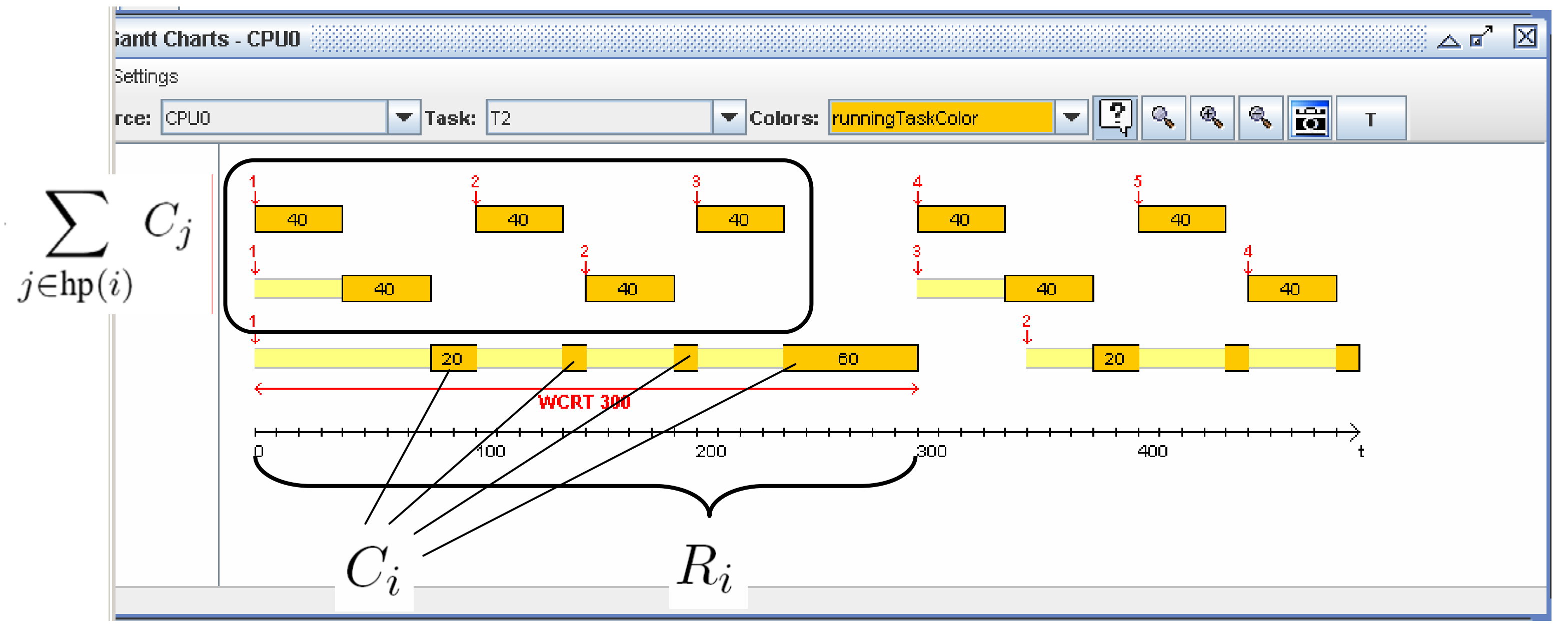
$$R_i = C_i + \sum_{j \in hp(i)} C_j \left\lceil \frac{R_i}{T_j} \right\rceil \leq D_i = T_i$$

response time

core execution time

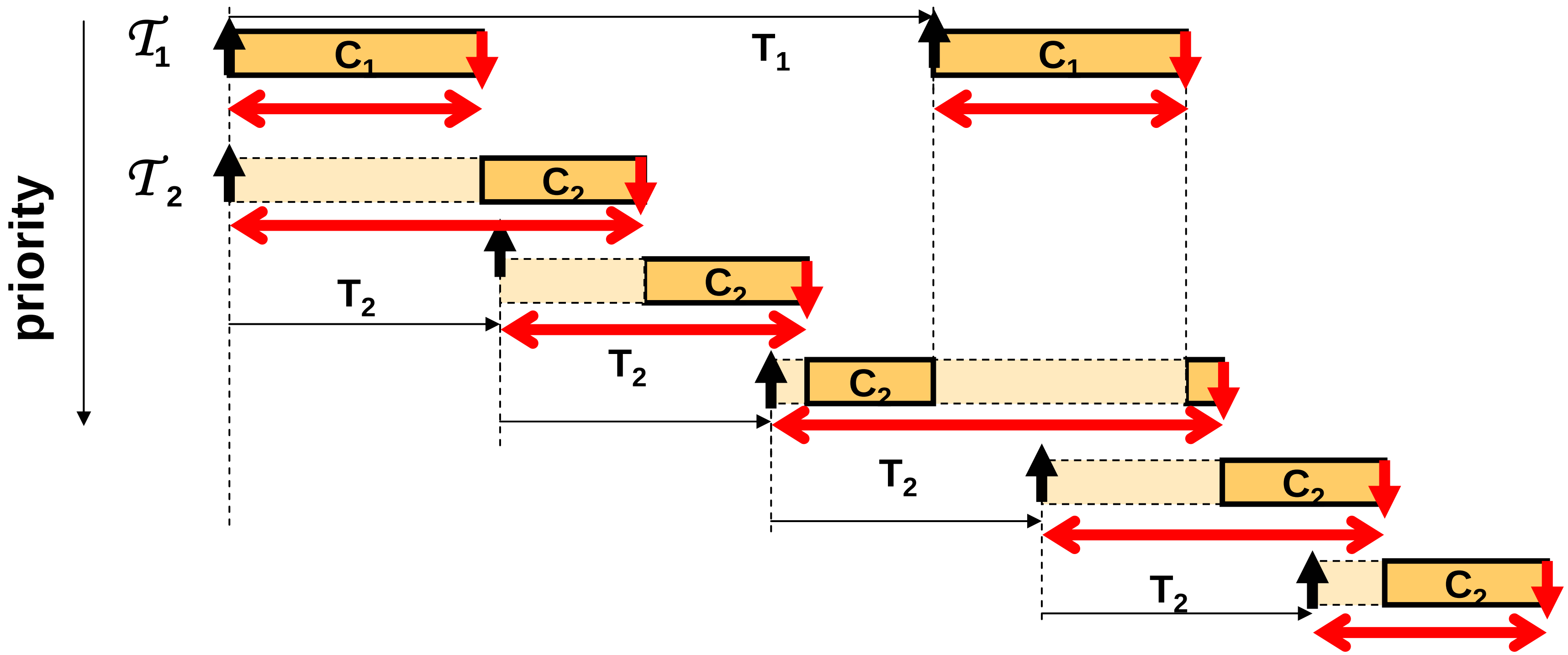
$\left\lceil \frac{R_i}{T_j} \right\rceil$: # of preemptions

$\sum_{j \in hp(i)} C_j \left\lceil \frac{R_i}{T_j} \right\rceil$: interference term I_i

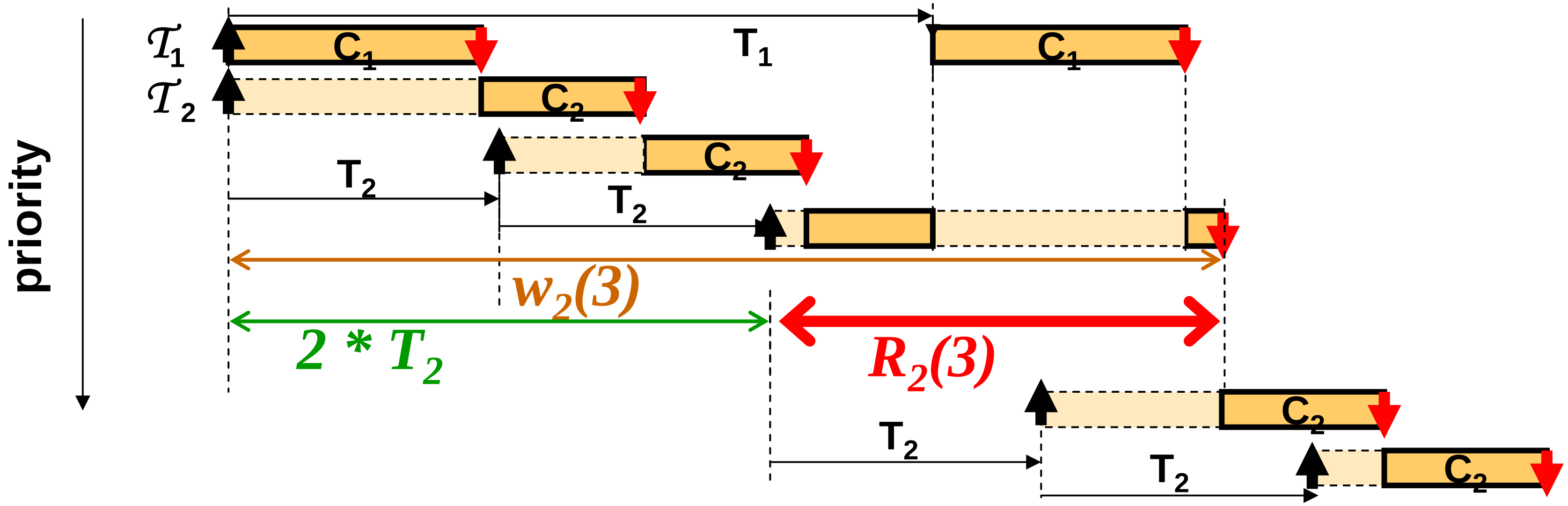


Example: Static priority w/ arbitrary deadlines

- **Assume:**
  - **tasks with periods T, worst-case execution times C**
  - **static priorities**
  - **deadlines (arbitrary) larger than the period**



Analysis uses "Busy Window" approach (Lehoczky)



$$w_i(q) = \quad q\, C_i \quad + \sum_{j \in \mathrm{hp}(i)} C_j \left\lceil \frac{w_i(q)}{T_j} \right\rceil$$

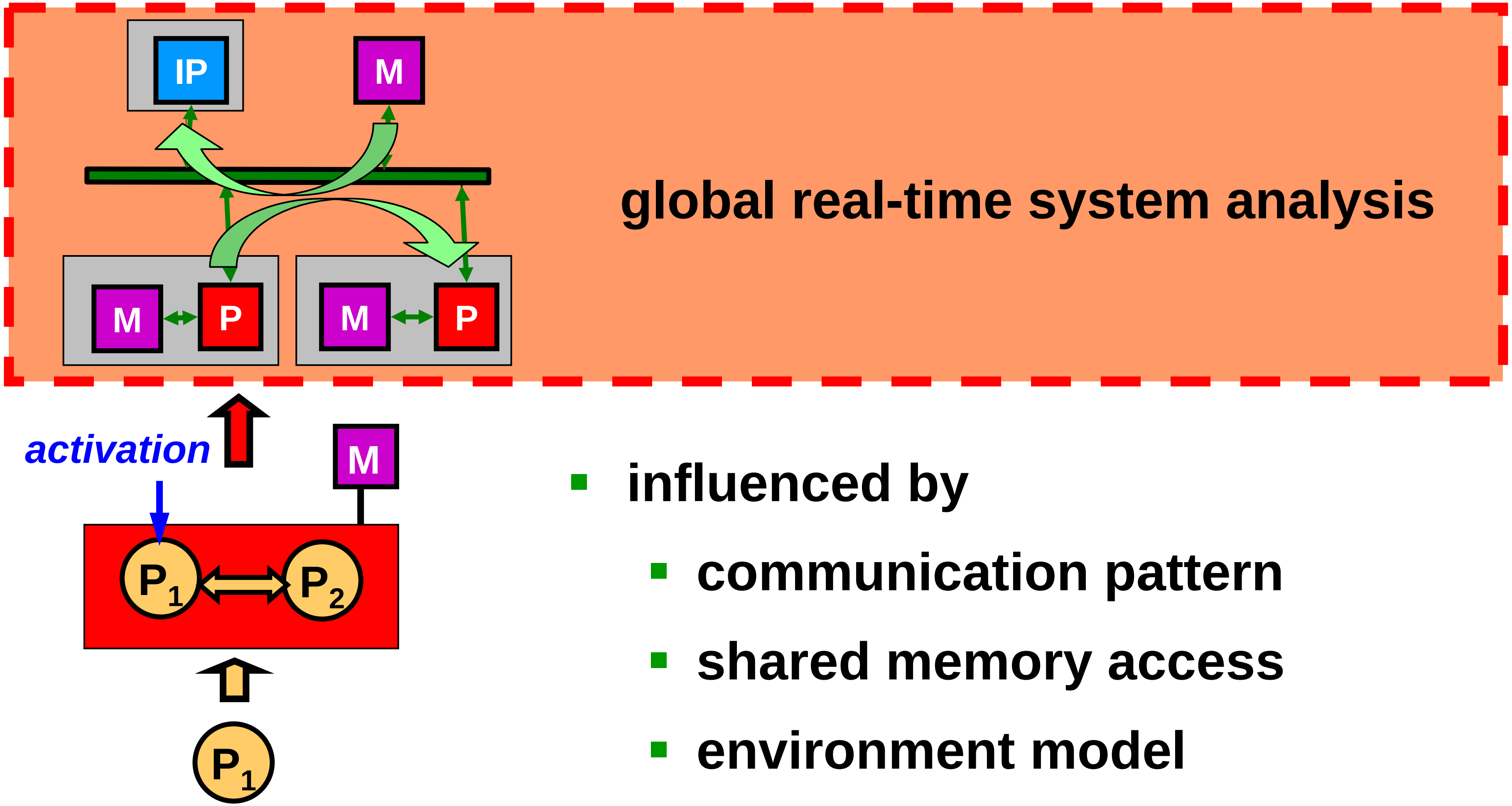
$$R_i(q) = \quad w_i(q) \quad - \quad (q-1)\, T_i$$

find fix point where equations hold!

Other Extensions in Literature

- **Jitter and burst activation**
- **Static and dynamic offsets between task activations**
- **Different task modes**
- **Execution scenarios**
- **Blocking and non-preemptiveness**
- **Scheduling overhead → context switch time**
- **etc...**

Global system execution model



- influenced by
  - communication pattern
  - shared memory access
  - environment model

System performance analysis

System performance analysis - state of the art 1/2

- Current approach: target architecture co-simulation, performance simulation
- Simulation challenges
  - identification of system performance corner cases
    - different from component performance corner cases
    - complex phase and data dependent “transient” run-time effects w. scheduling anomalies
    - target architecture behavior unknown to the application function developer
    - test case definition and selection?
  - simulation of incomplete application specifications ?
    - how to do design space exploration before code implementation is available?

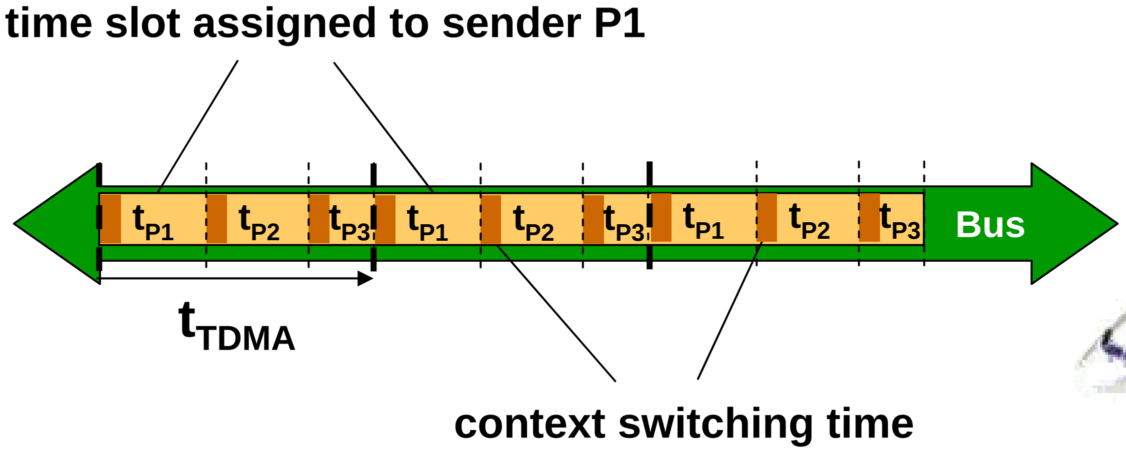
System performance analysis - state of the art 2/2

- **Load analysis**
  - **Example: "all deadlines are met if the resource load is below 69%"**
  - **Consider only average scenarios (no transient load)**
  - **No performance metrics → no constraint validation**

Conservative design

- Popular as a system level technique for safety critical systems design
- Strict separation of subsystems
  - fixed allocation of memory
  - fixed allocation of communication resources
  - fixed allocation of computation resources
- Spatial and temporal decoupling of resources
  - not-in-use allocated parts are locked
  - no coupling effects
- Requires system synchronization …
- … paid by timing overhead

TDMA 1/2



Time Triggered System (TDMA)

- **periodic assignment of fixed time slots for communication and processing**
- **unused slots remain empty**
- **requires system synchronization**
- **no coupling effects**

TDMA 2/2

- **Predictable, independent system capacity**

$$R_i = C_i + (t_{TDMA} - t_{Pi}) \times \left\lceil \frac{C_i}{t_{Pi}} \right\rceil$$

R_i response time P_i, C_i core execution time P_i

- **Used in avionics and automotive (TTP, FlexRay)**
- **Can be used at system level (Giotto - Berkeley)**

Conservative design - Summary

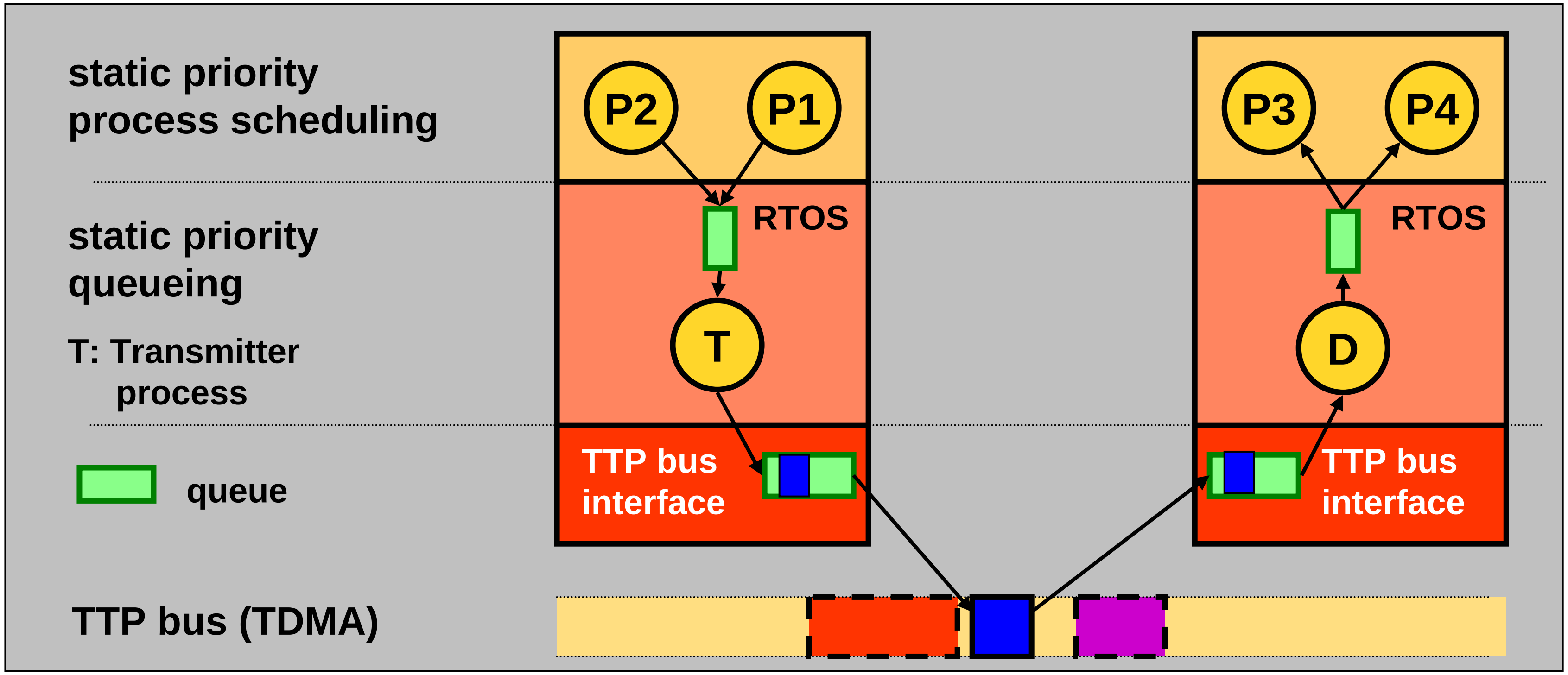
- Limitations
  - low resource utilization
  - extended response times (problem for adaptive control engineering)
  - requires general time base (scalability?)
  - little flexibility (fixed time slots)
  - not a general solution
  - inefficiency (performance, bandwidth, costs, power) increases with system size
- Time-triggered systems are a good example for systematic integration, but…
- … reliable integration does not necessarily require conservative design style

System level performance analysis

- Global approach („Holistic“)
  - local analysis scope extension to several subsystems
- Compositional approach
  - global flow analysis combined with local scheduling analysis

Analysis scope extension – „Holistic“

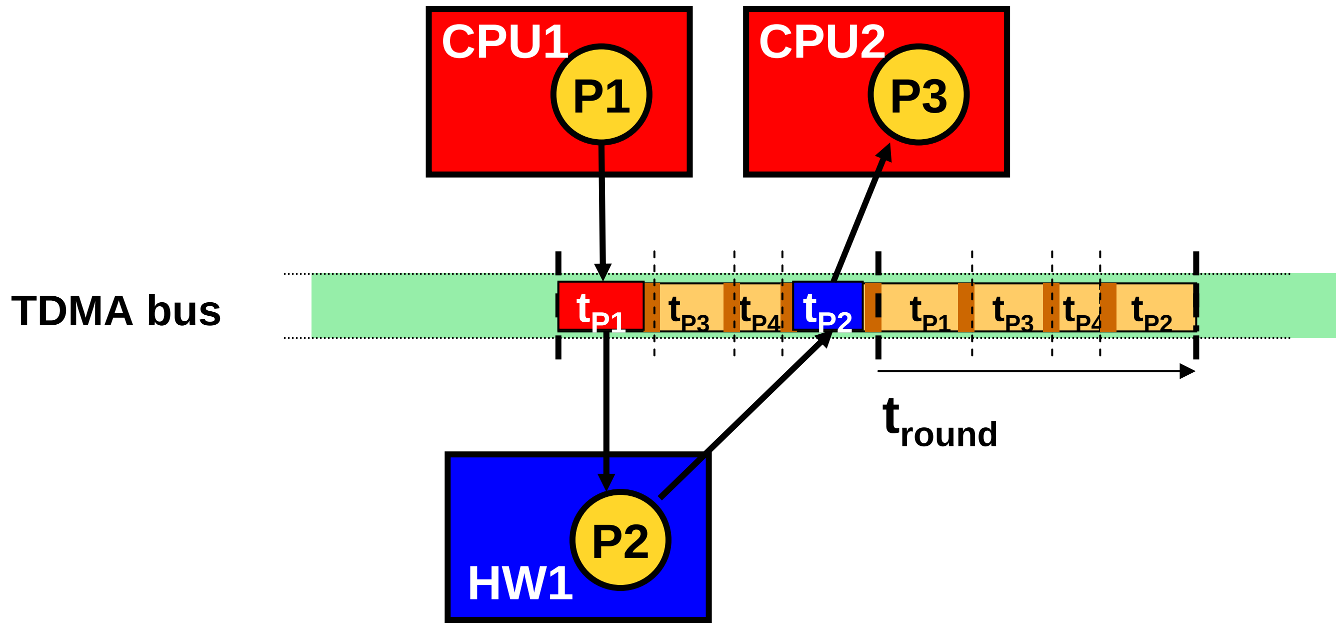
- Coherent analysis („holistic“ approach)
- Example: Tindell 94, Palencia/Harbour 98, Pop/Eles (DATE 2000, DAC 2002): TDMA + static priority – automotive applications



- **Problem: scalability**

Analysis scope extension (cont'd)

- **Benefit: scope extension can take global system knowledge into account**
- **Example: using dependency information to detect that P2 can send in the same TDMA round as P1, if $R_{P2} < t_{P3} + t_{P4}$, where R_{P2} is the worst-case response time of P2**



Compositional performance analysis

After the break!

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

SymTA/S

Compositional performance analysis

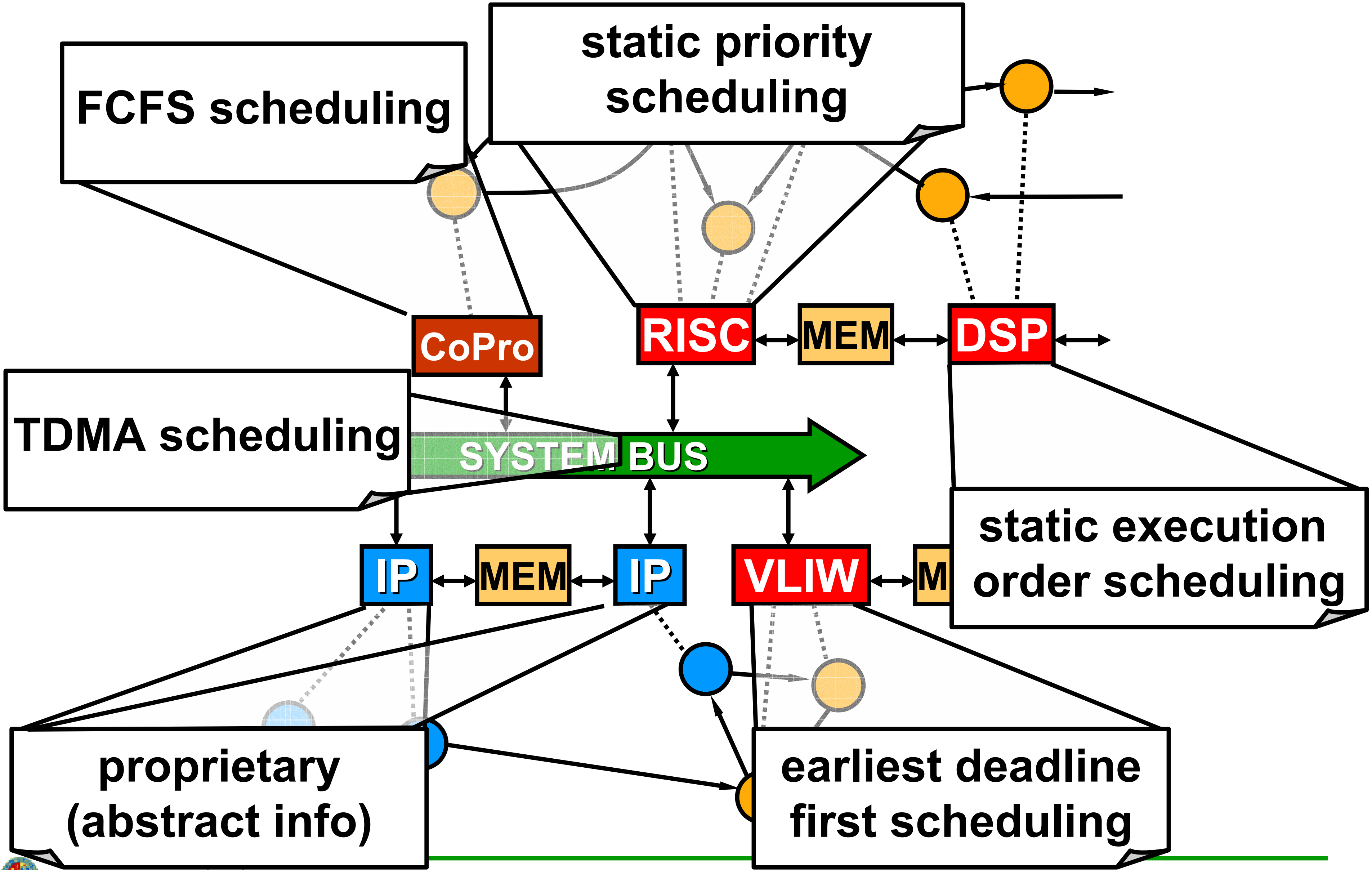
ARTIST2 PhD Course, June 12, DTU Copenhagen, Denmark

Razvan Racu

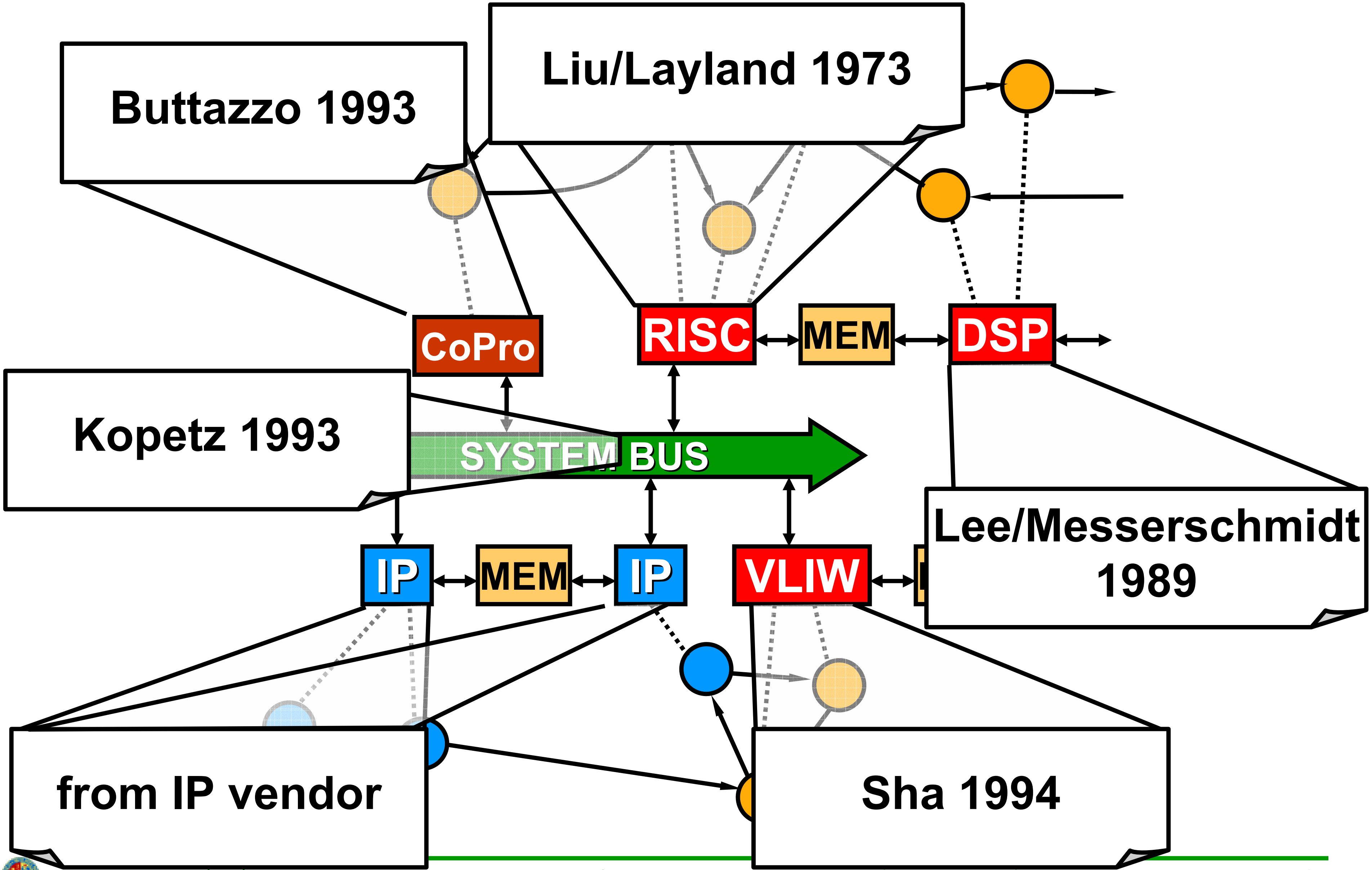
Arne Hamann

INSTITUTE OF
COMPUTER AND
COMMUNICATION
NETWORK ENGINEERING

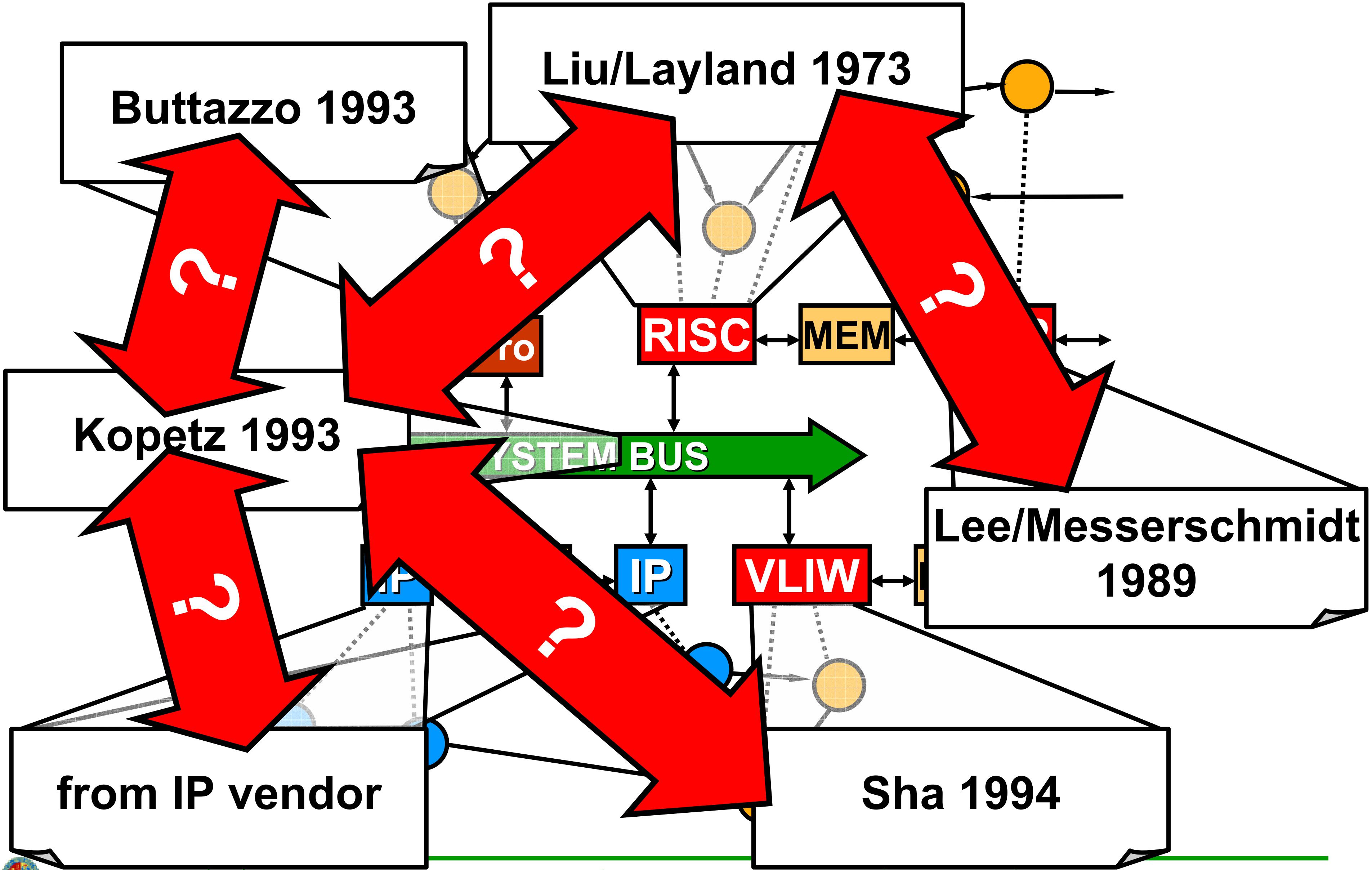
Multiple Scheduling Strategies



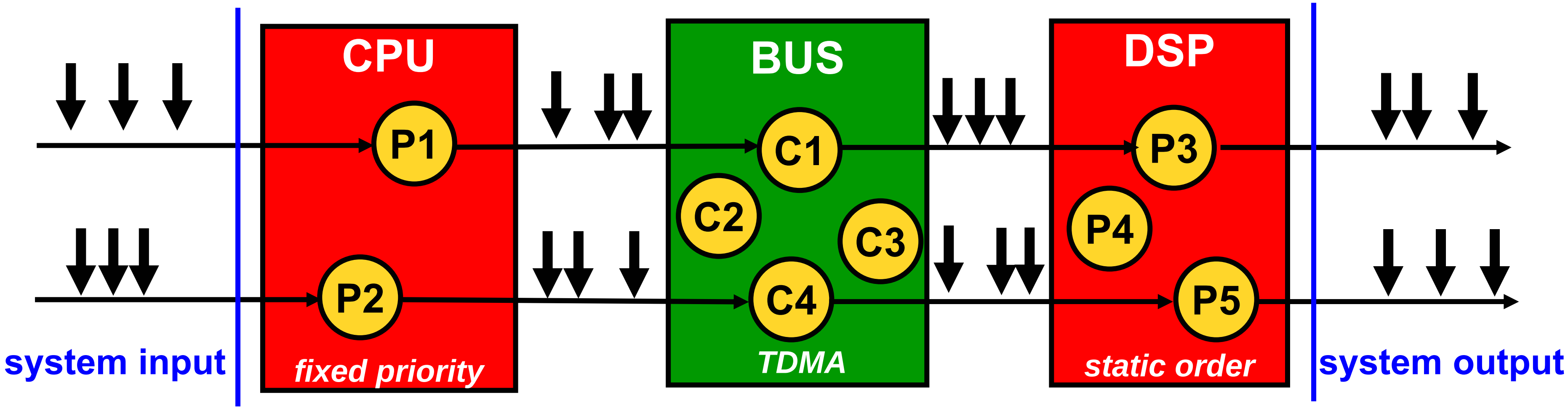
Corresponding Analysis Techniques



Integration ???



Compositional approach



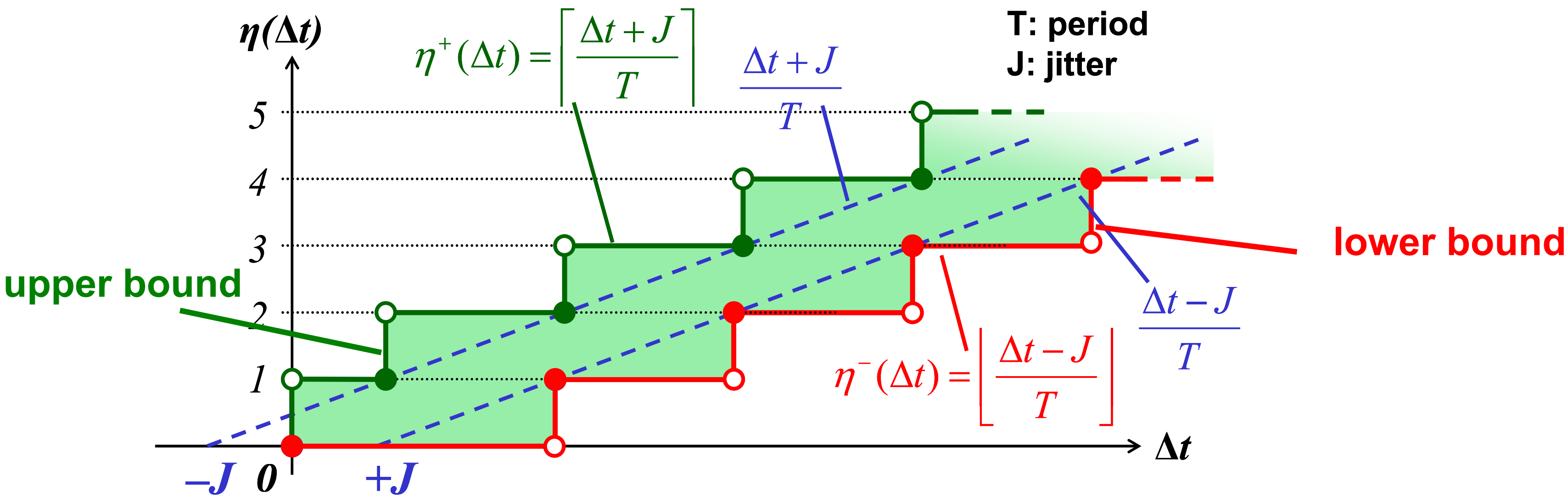
- Tasks are coupled by event sequences
- Composition by means of event stream propagation
  - apply local scheduling techniques at resource level
  - determine the behavior of the output stream
  - propagate to the next component

Idea

- Use network calculus + additional information as intermediate mathematical formalism
- Arrival curve functions of network calculus
  - $\eta^{+}(\Delta t)$ maximum number of activating events occuring in time window Δt
  - $\eta^{-}(\Delta t)$ minimum number of activating events occuring in time window Δt
  - d^{-} minimum event distance - limits burst density

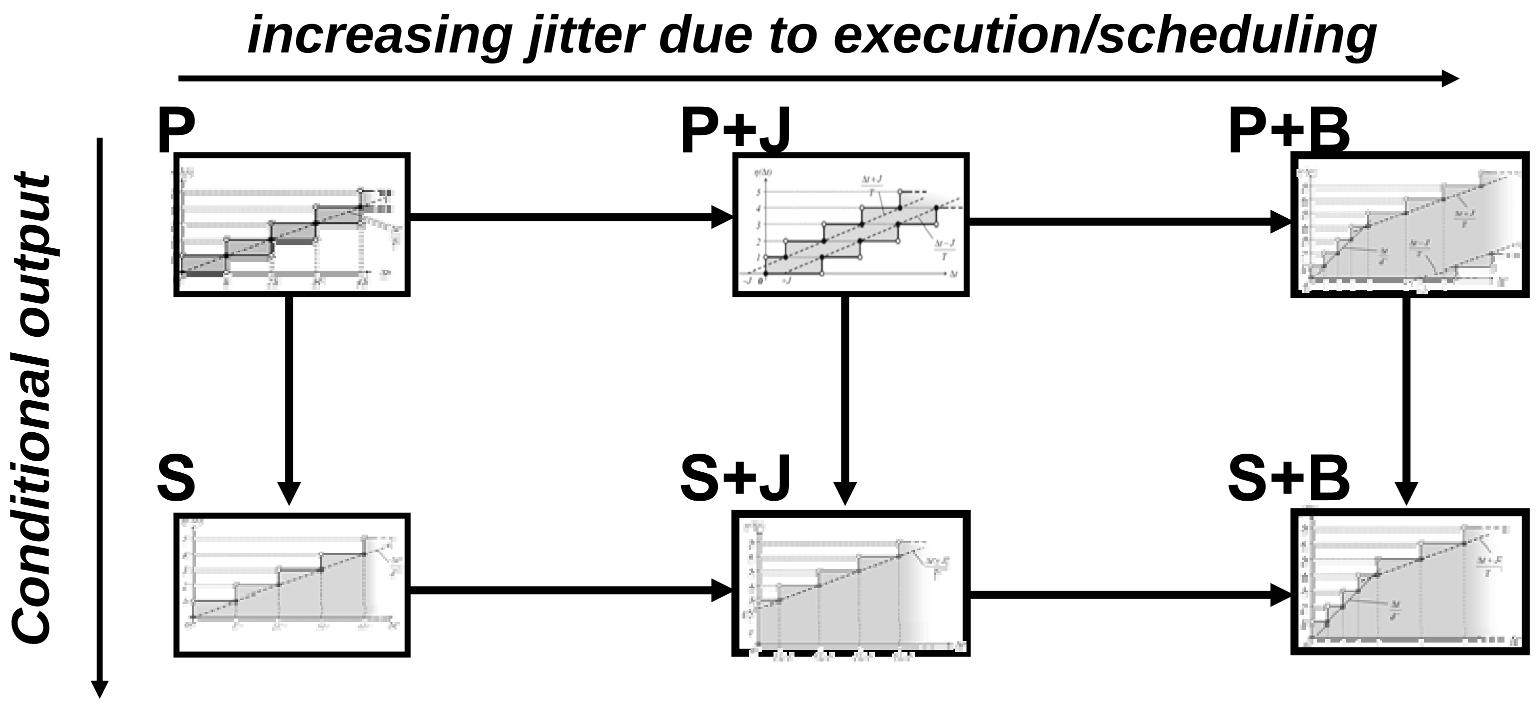
Event specification

- Derive event stream models with parameters
  - individual events replaced by stream variables (vectors) with stream parameters period, jitter, min. distance, …
  - derive arrival curve functions from model parameters



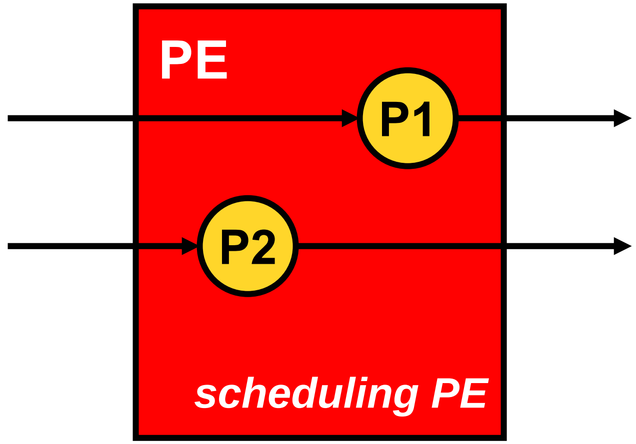
SymTA/S standard event models

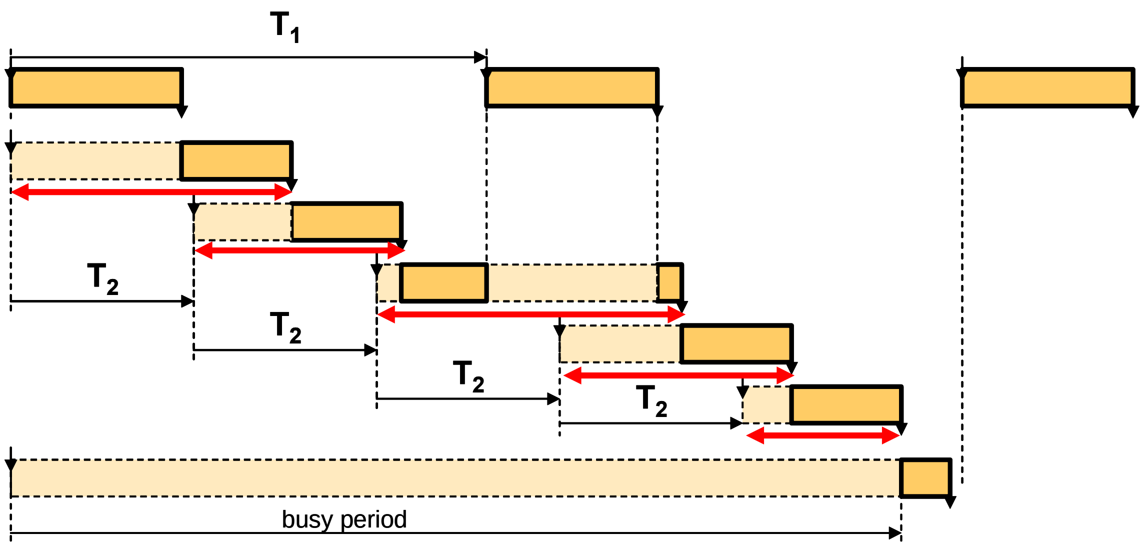
- Required by RTA
  - Periodic/sporadic
  - Periodic/sporadic with jitter
  - Periodic/sporadic with burst



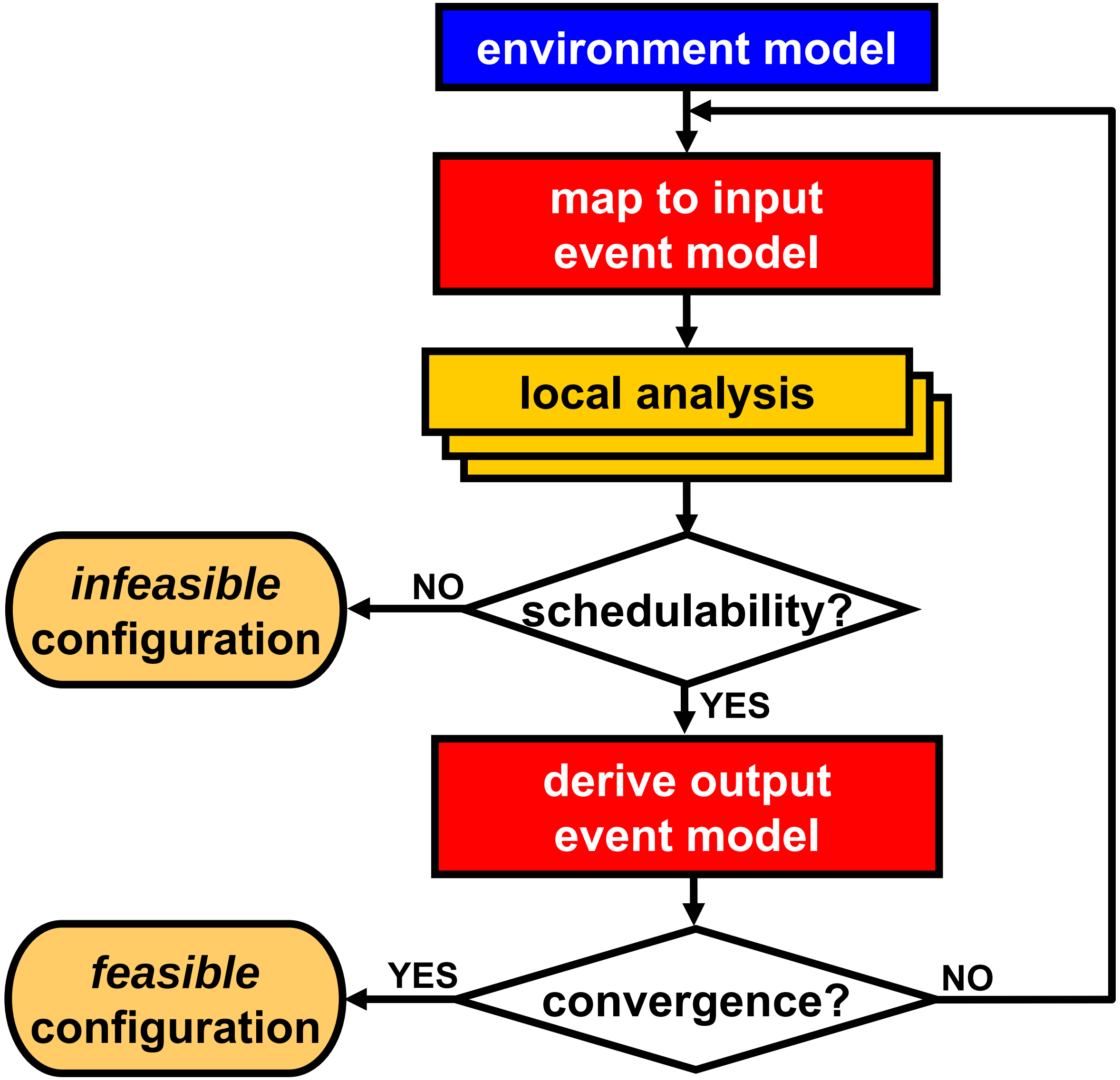
Input – output event model relation

- **Any scheduling increases jitter**
- **Jitter grows along functional path**
- **Increasing jitter leads to**
  - **burst and transient overloads**
  - **higher memory requirements**
  - **power peaks**



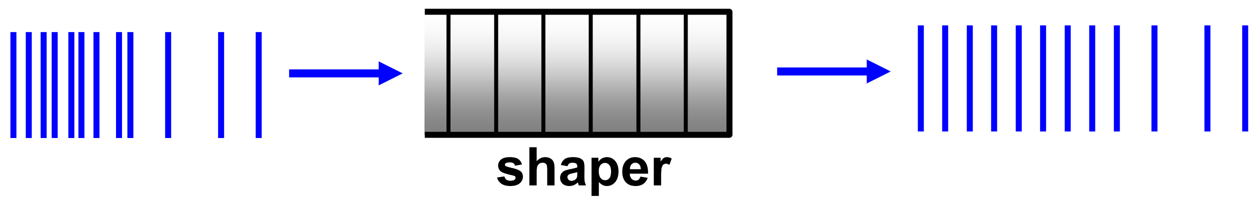


System analysis loop

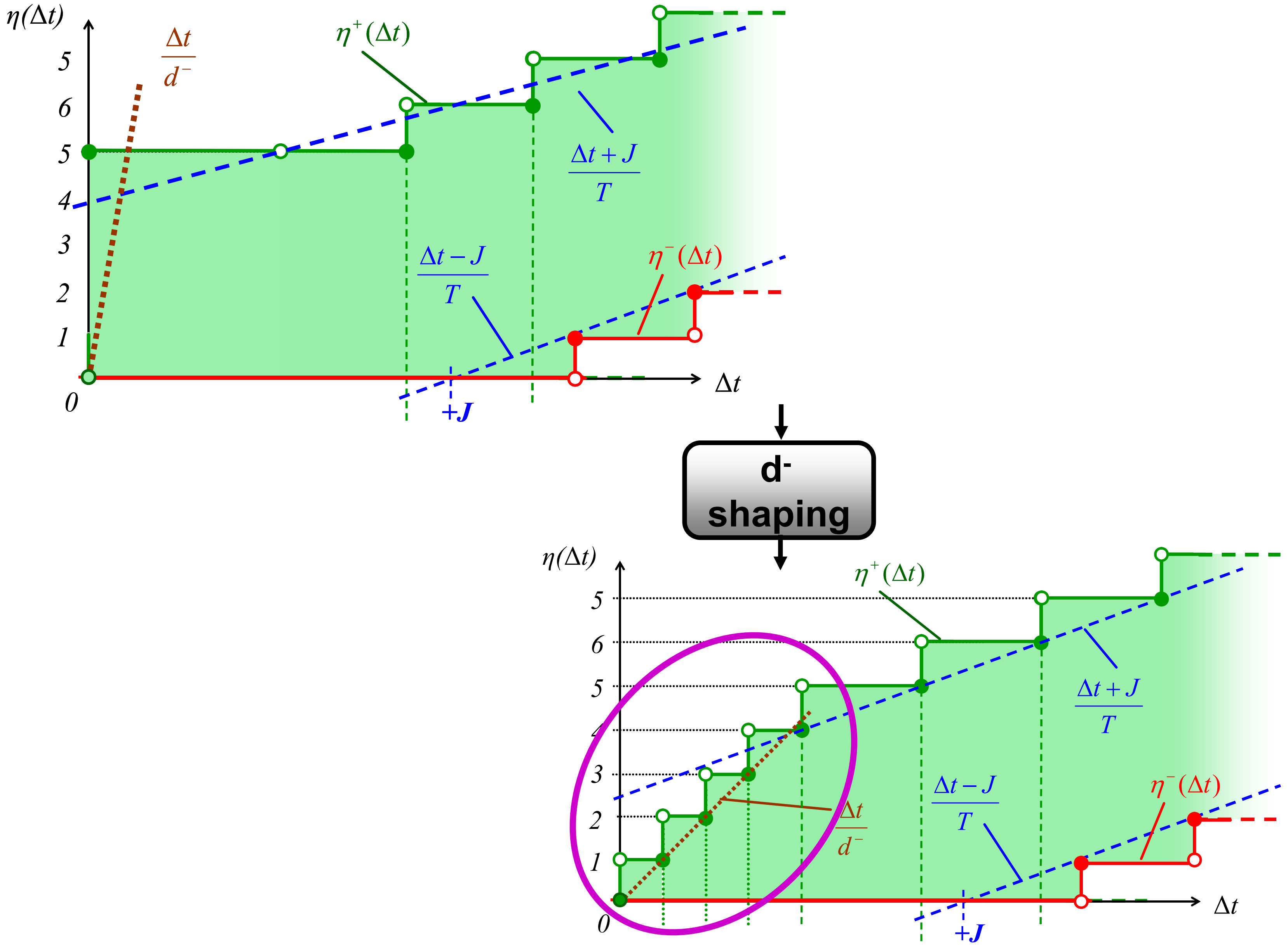


Reducing transient load in design

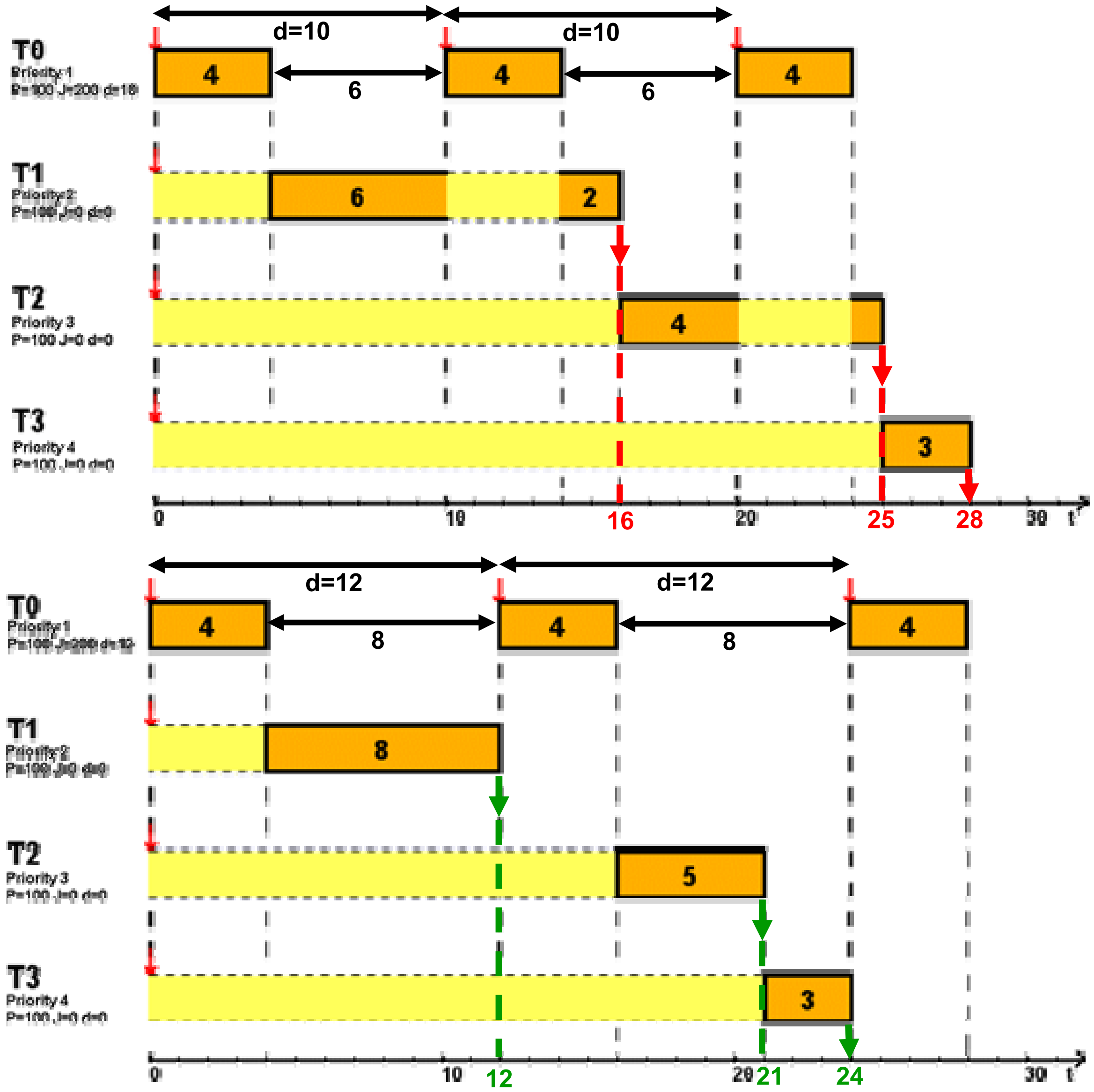
- **Re-synchronization**
- **Minimum event separation using „traffic shaping“**
- **Requires memory and possibly increases latency**



Traffic shaping - example

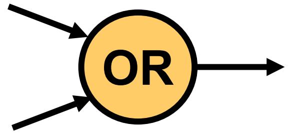


Optimization potential of Traffic Shaping

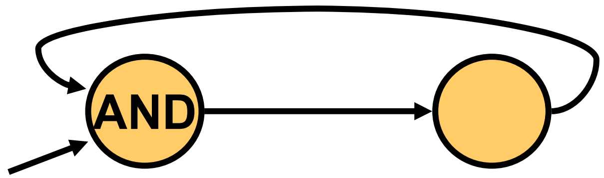


RTA event models are not sufficient

- **Event model transitions needed to couple different subsystems and scheduling domains**
- **More complex activation models needed**
  - **OR activation**
    - **typical in event driven systems**

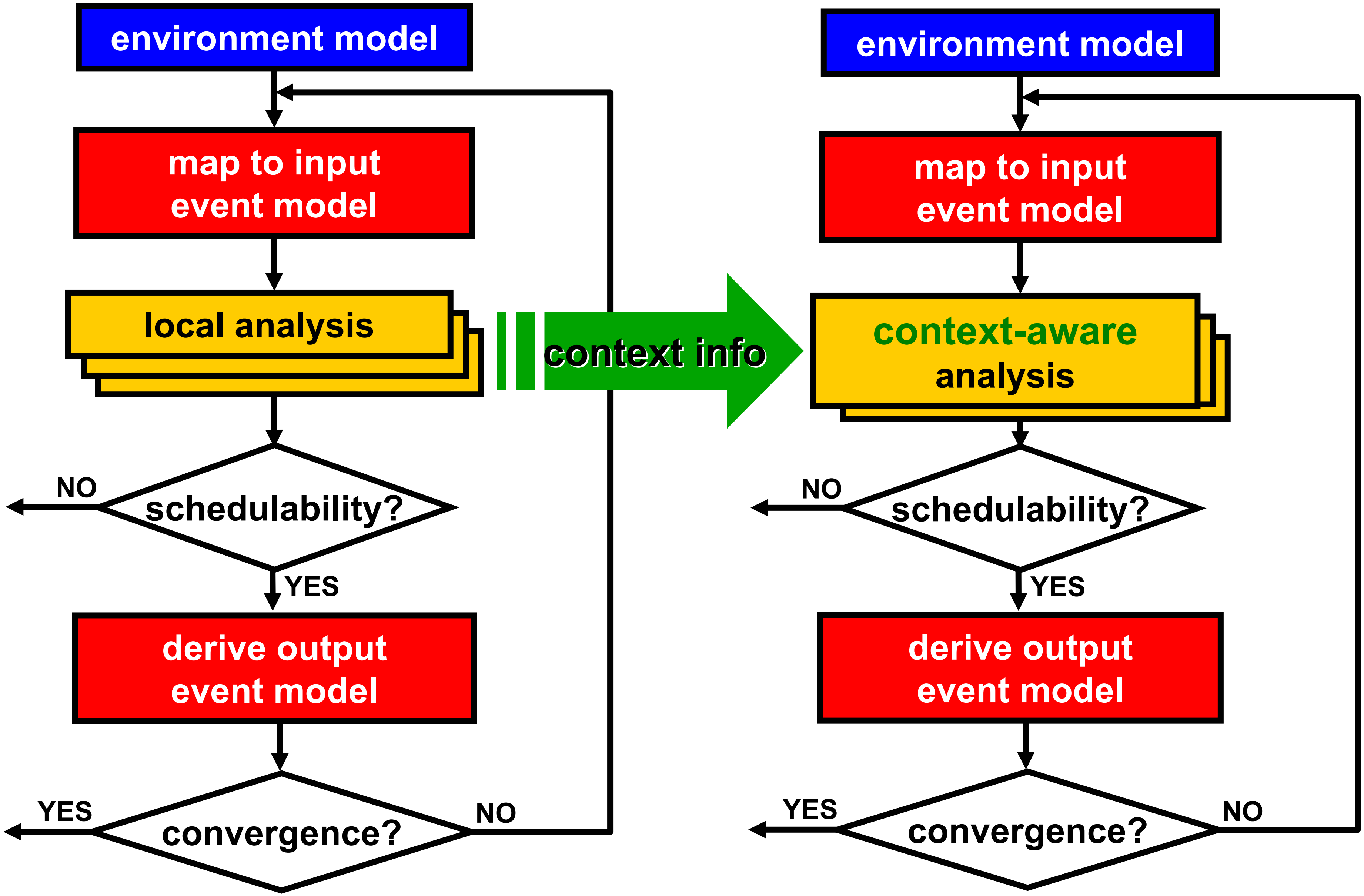


  - **AND activation and loops**
    - **typical for signal processing**



Analysis extensions

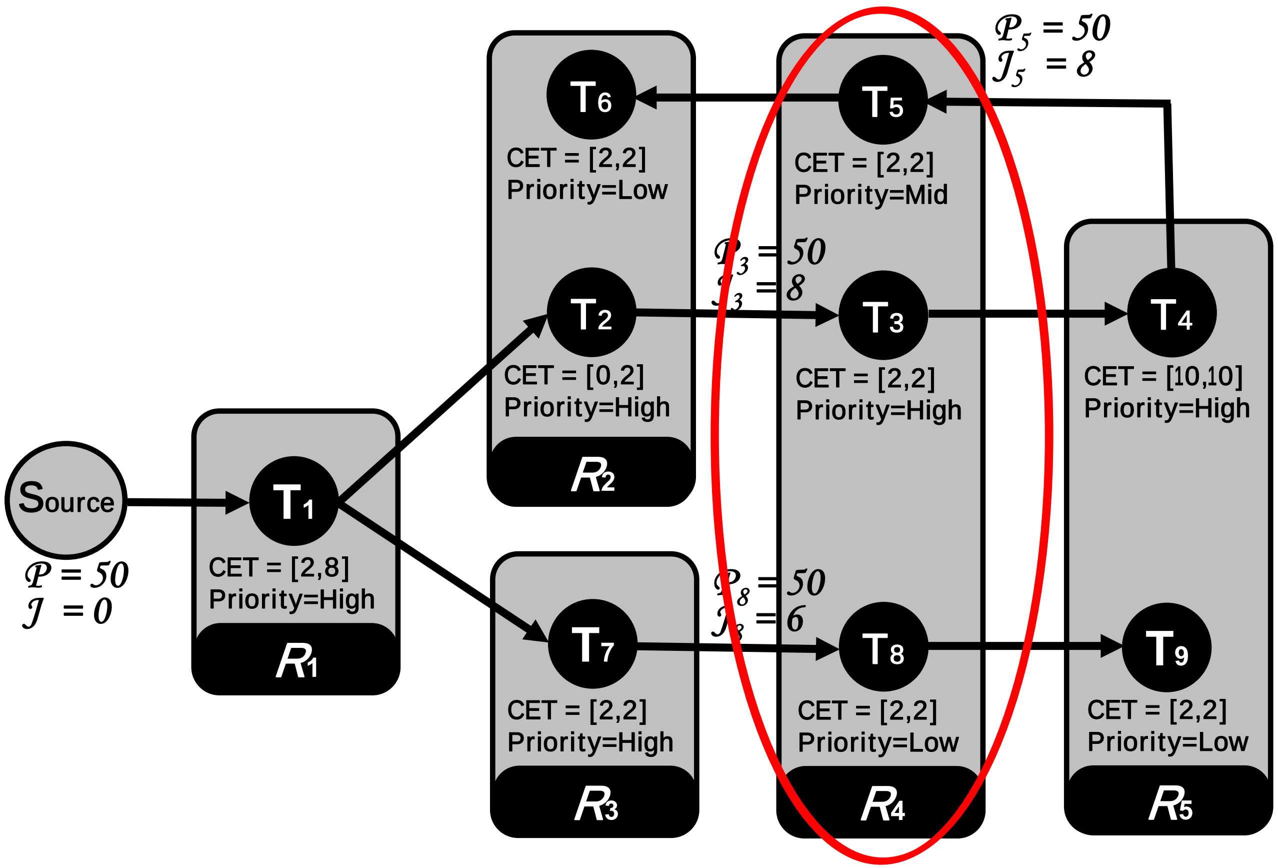
System analysis loop



Taking global dependencies into account

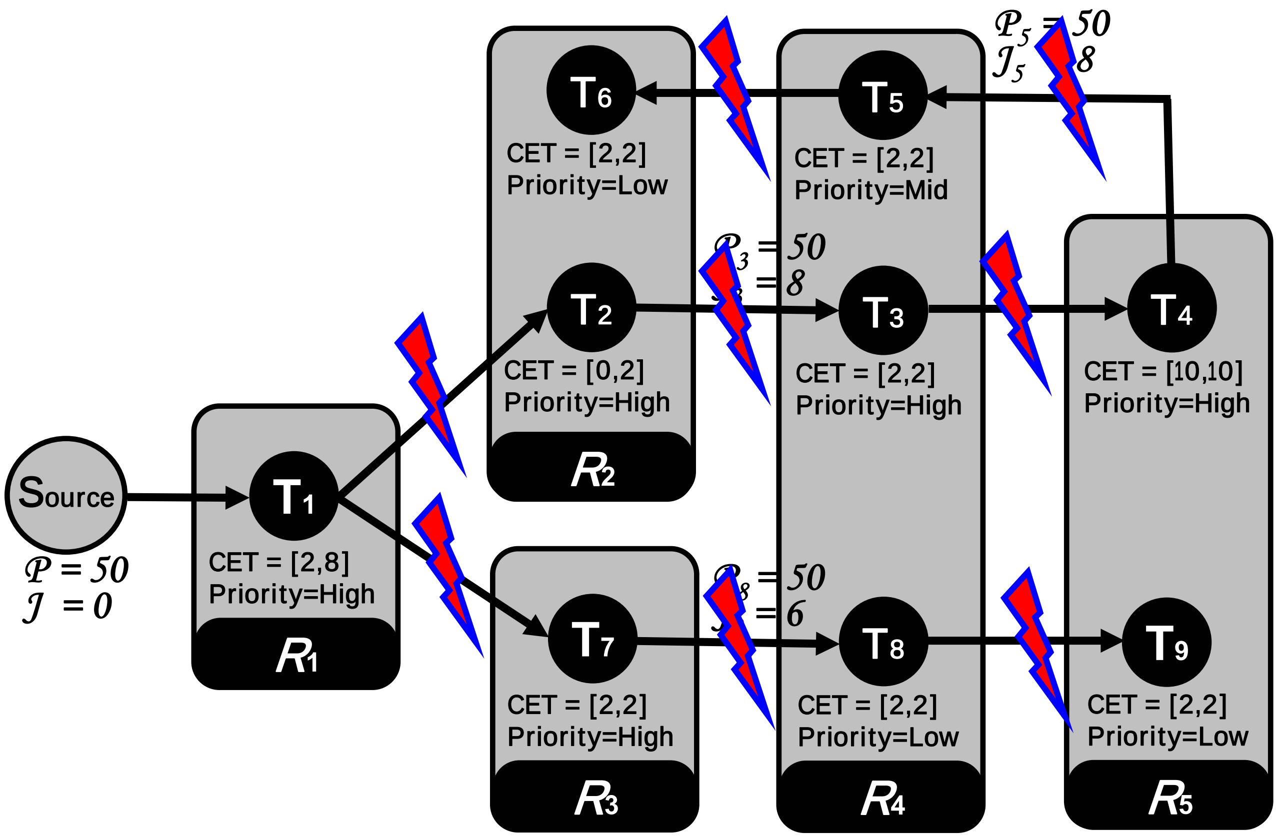
- „intra-context“ dependencies
  - different events in a single event stream often activate different task behaviors with different execution times or communication loads
- „inter-context“ dependencies
  - activating events in different event streams are often time-correlated which rules out the simultaneous activation of all tasks
- can be combined leading overall to less conservative analysis results

Motivating Example



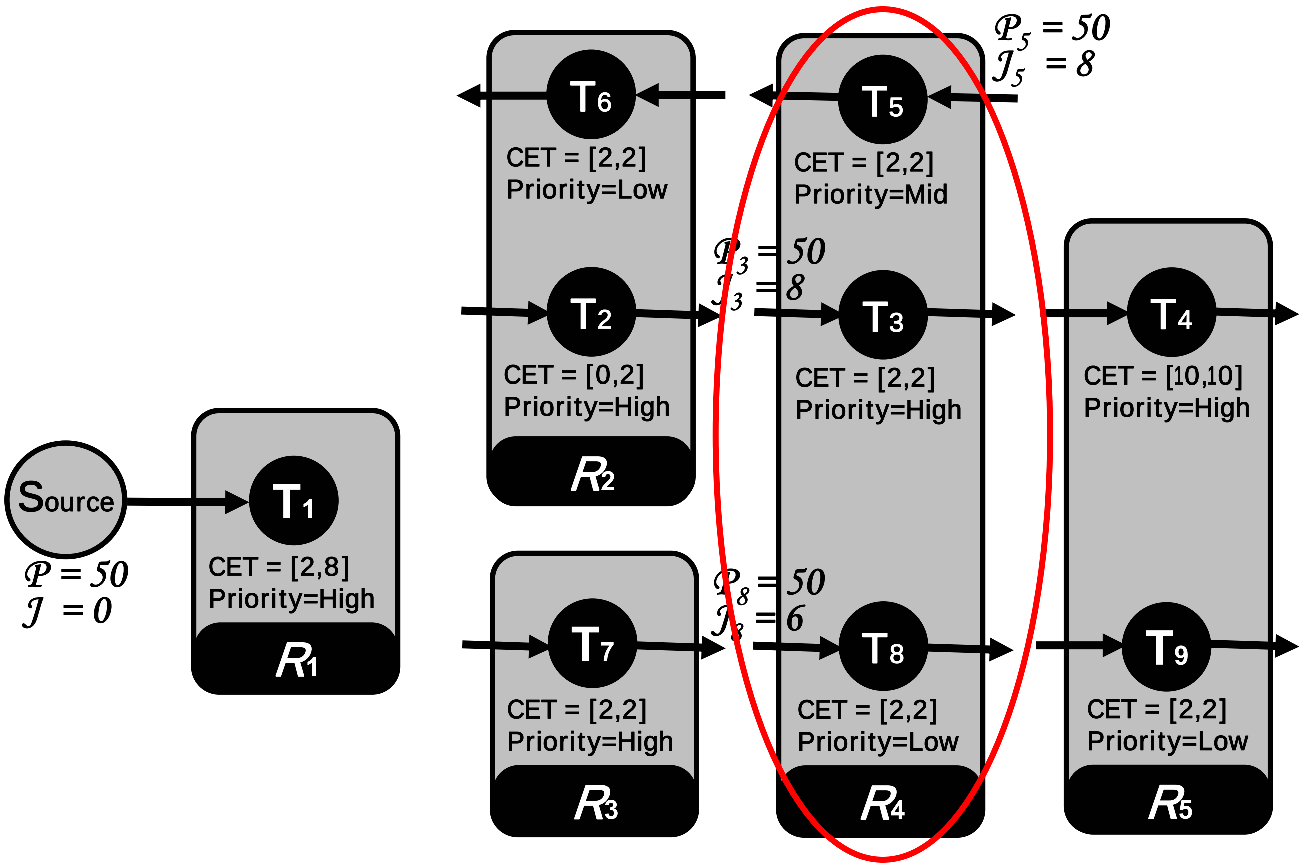
- **Static priority preemptive scheduling on all resources**
- **Compositional performance analysis approach (Richter)**

Lehoczky (1990)



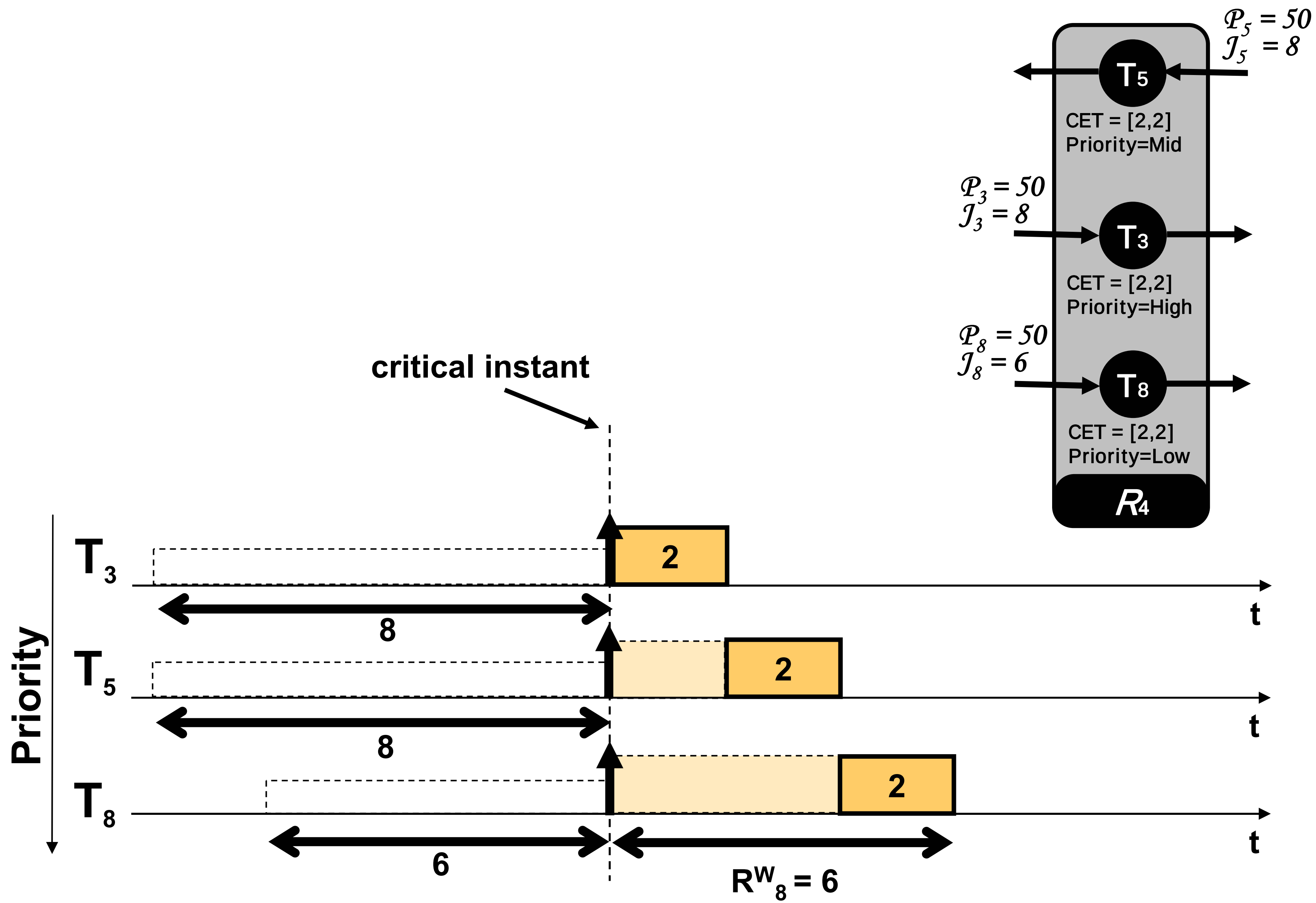
- Ignore correlation between tasks!

Lehoczky (1990)

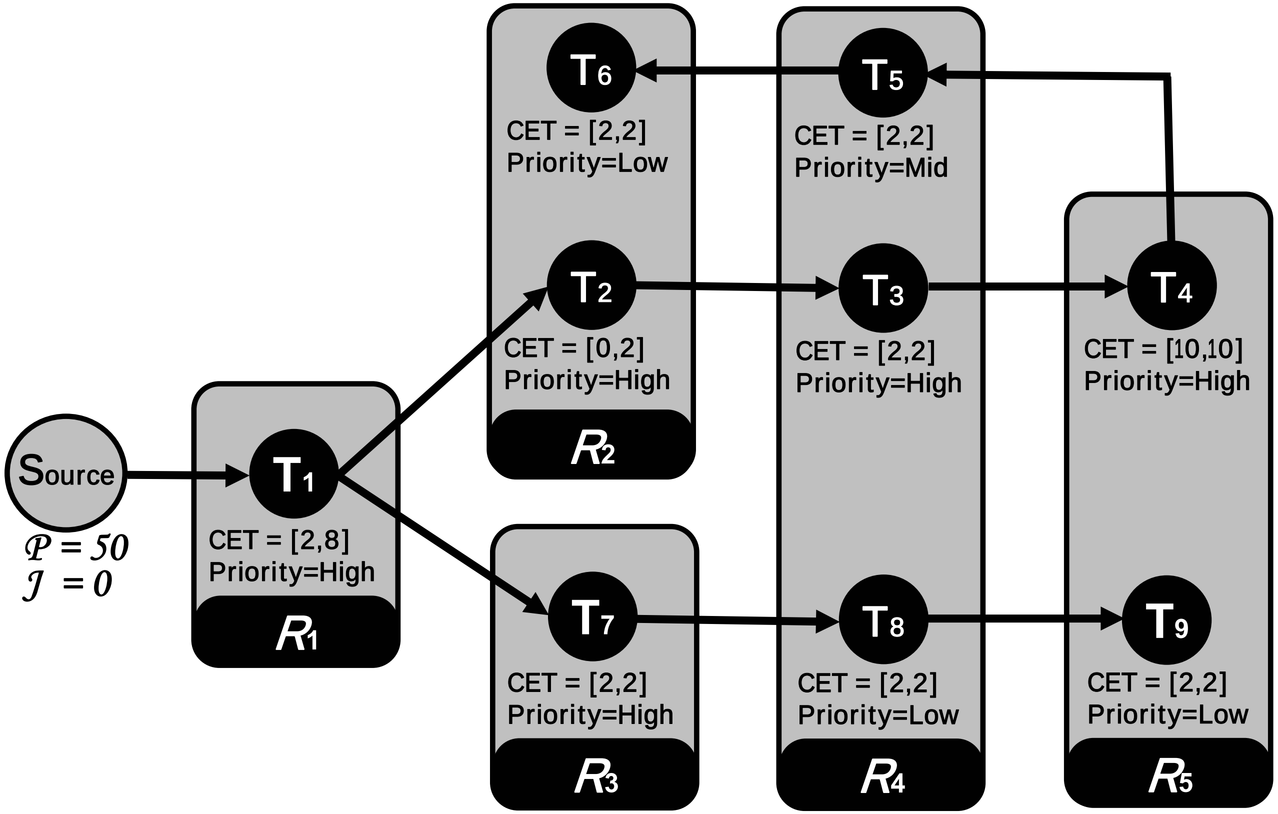


- **Ignore correlation between tasks!**

Lehoczky (1990)

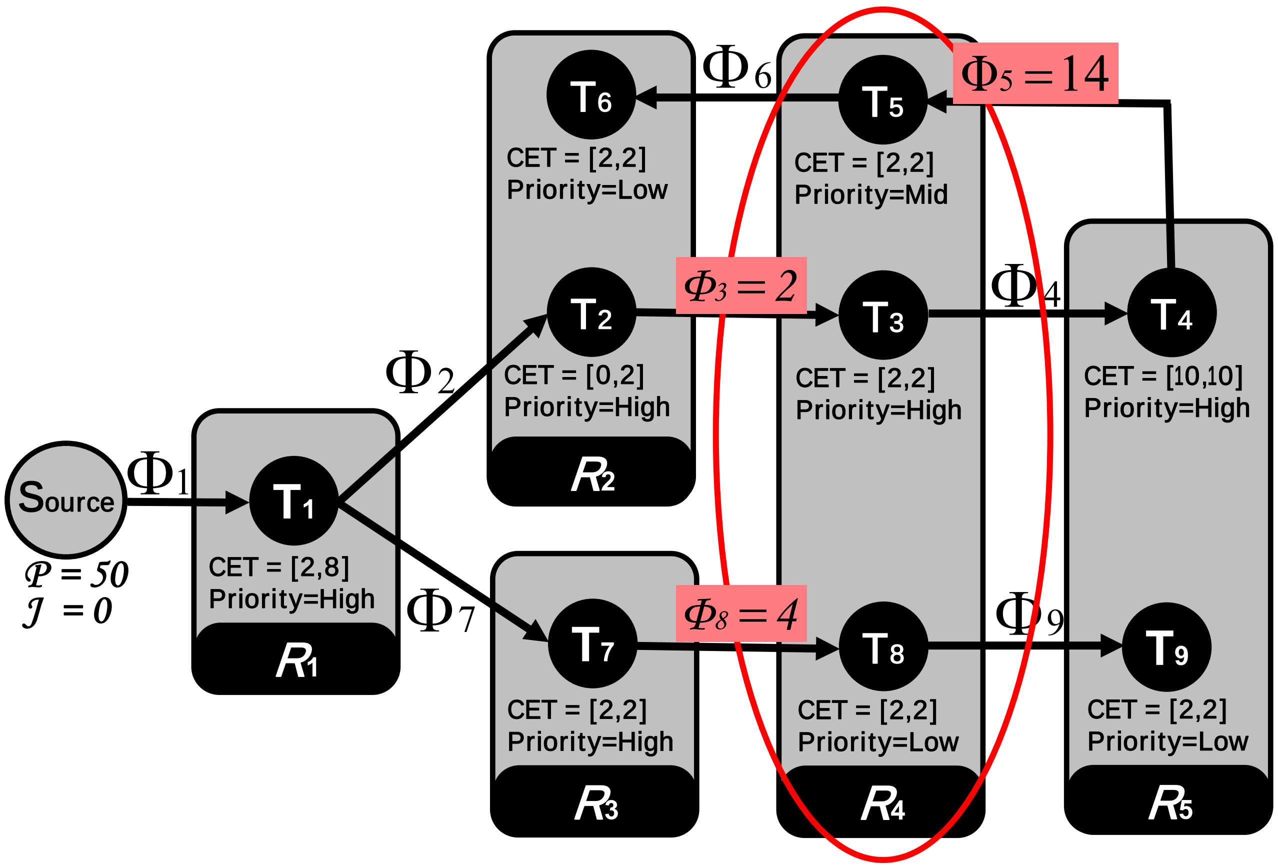


Tindell (1994)



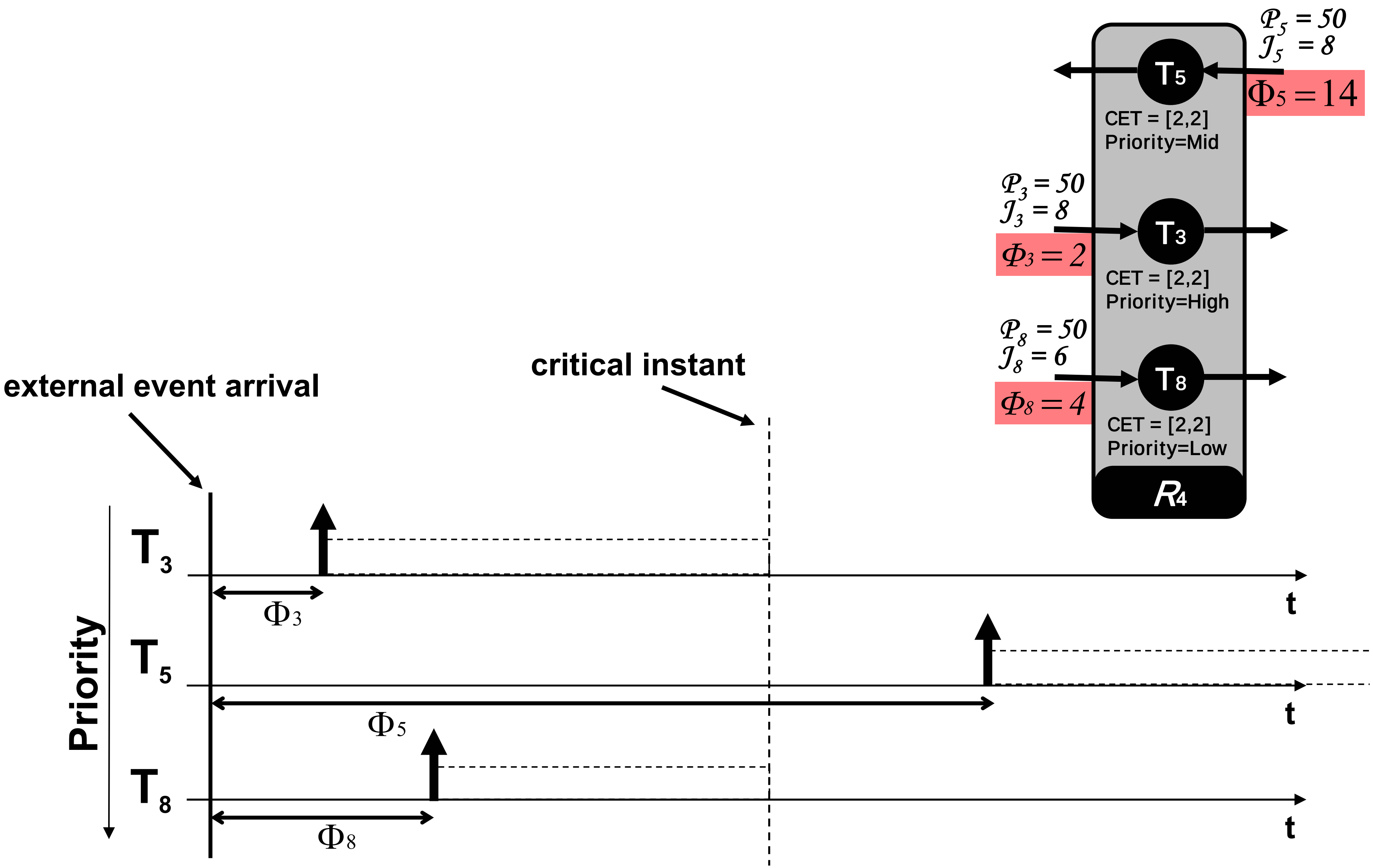
- **Periodic arrival of events at system inputs as timing-reference**

Tindell (1994)

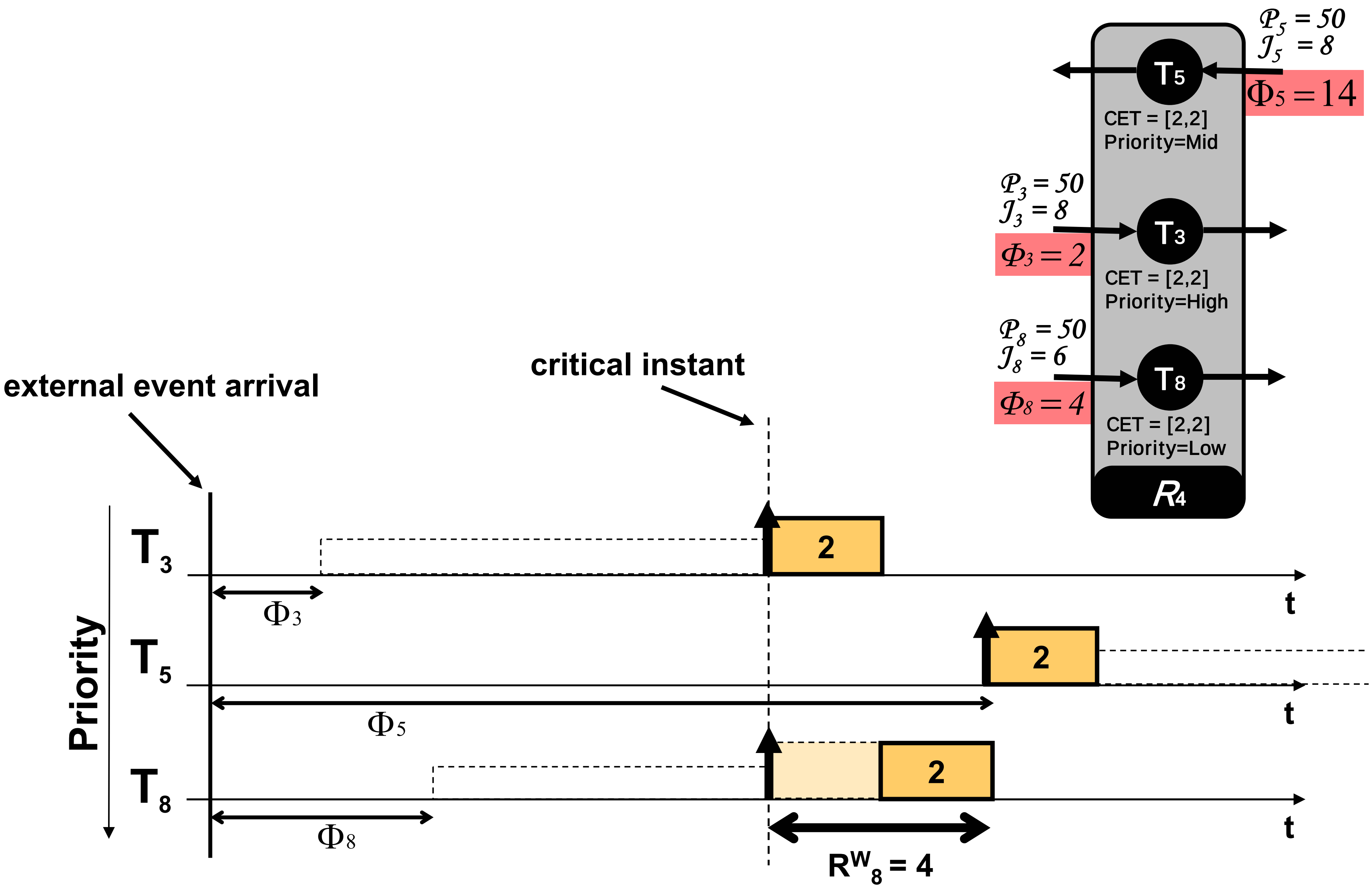


Global Offset Φ_i = earliest activation time of T_i relative to the periodical arrival of an external event at the system input

Tindell (1994)



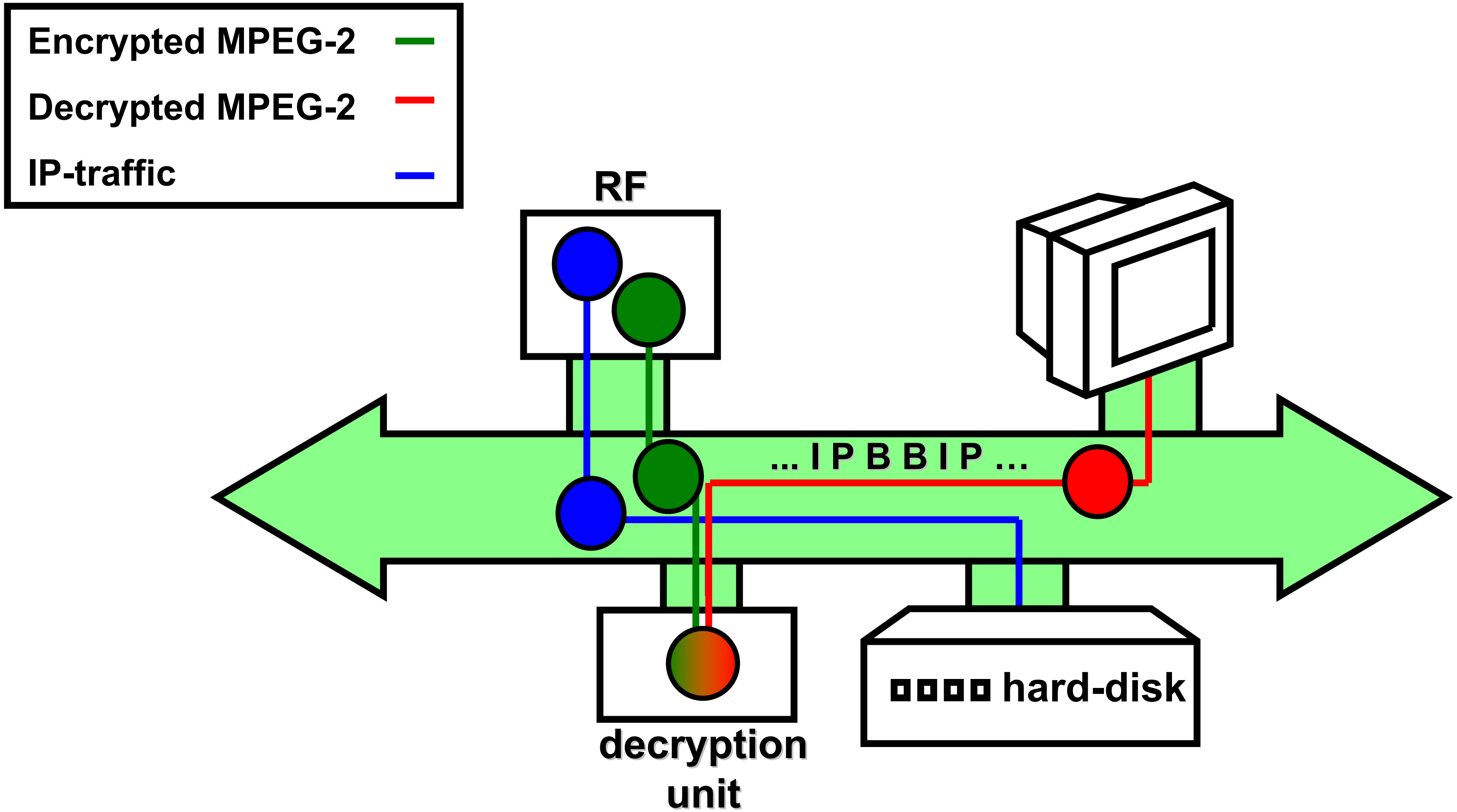
Tindell (1994)



Further Techniques

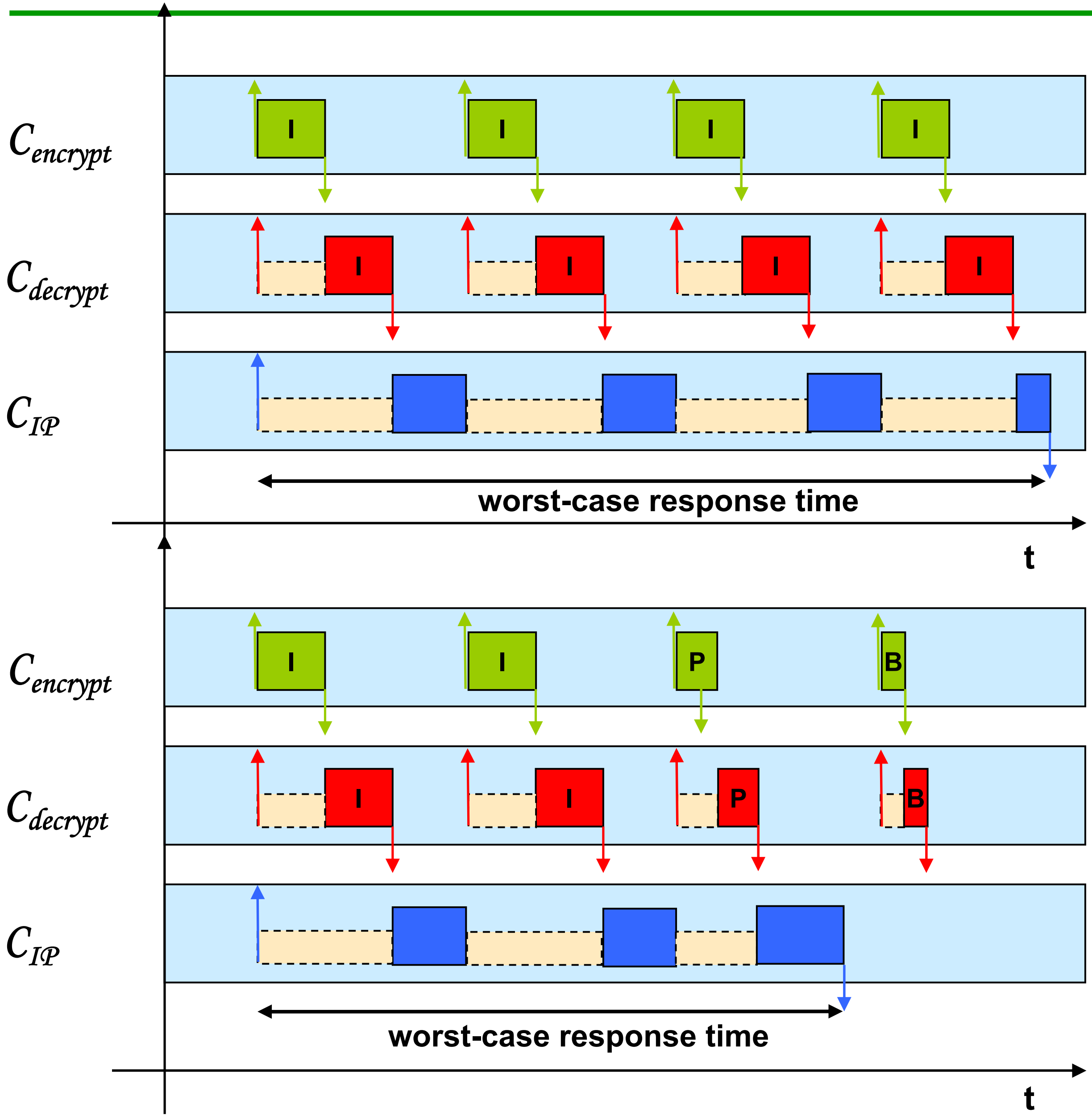
- **Relative offsets and relative jitter**
  - **Extends idea of global offsets**
  - **Describes the earliest activation time of a task relative to a timing-reference *ref***
  - **Reference is not necessarily a periodic external event**
  - **Enables tighter response time calculation**
- **Precedence relations**
  - **Explicitly considers precedence relations between tasks (i.e. task i cannot start until task j has finished execution)**
  - **Orthogonal to offset based techniques**

Set-top box



- set top box: decript video + download file via IP

Intra context dependencies



classical analysis:

each frame is assumed to be an I-Frame

intra context analysis:

information about frame types allows accurate load calculation

That‘s all !

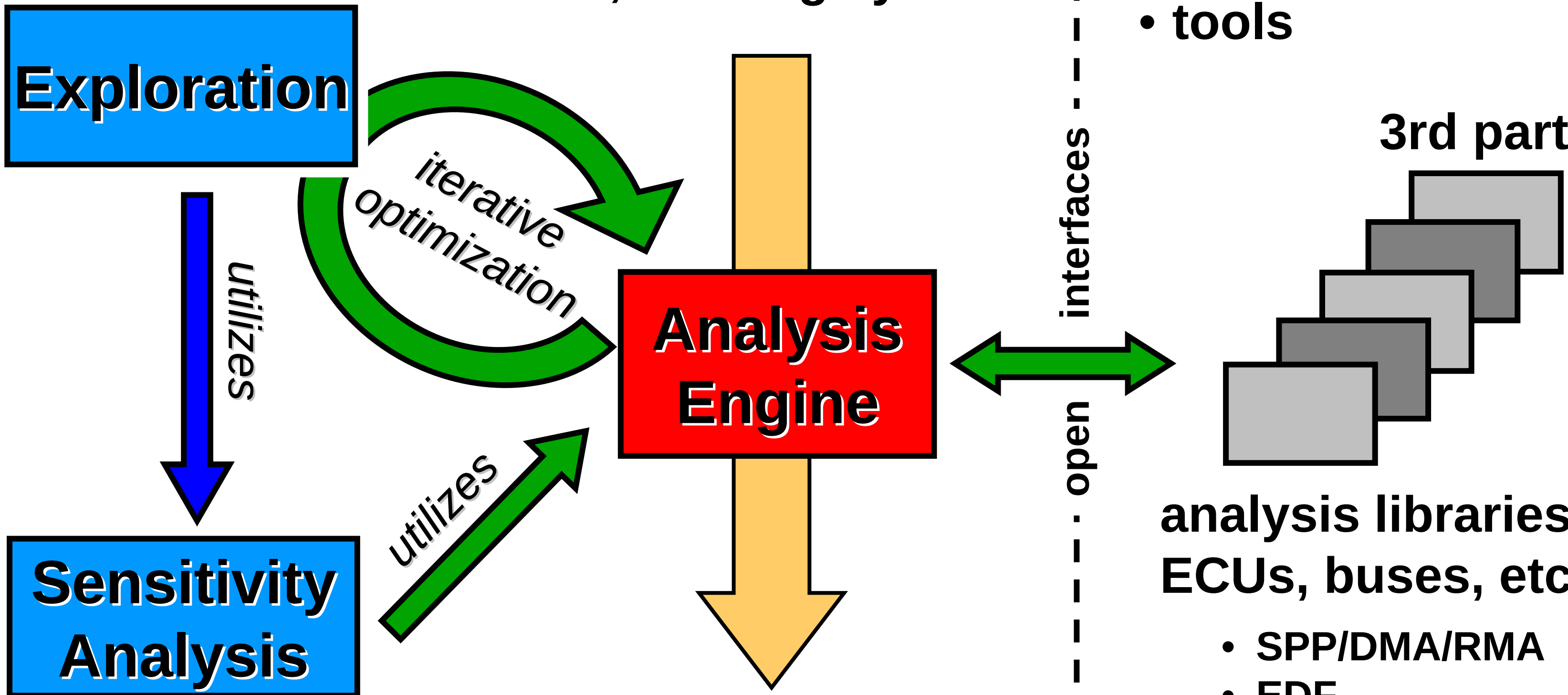
Hands-on Session

SymTA/S Tool

SymTA/S Tool Suite

idea, specification,
sketch, existing system

flow integration:
- data bases
- tools



verified system

analysis libraries for ECUs, buses, etc ...

- SPP/DMA/RMA
- EDF
- TDMA
- RR

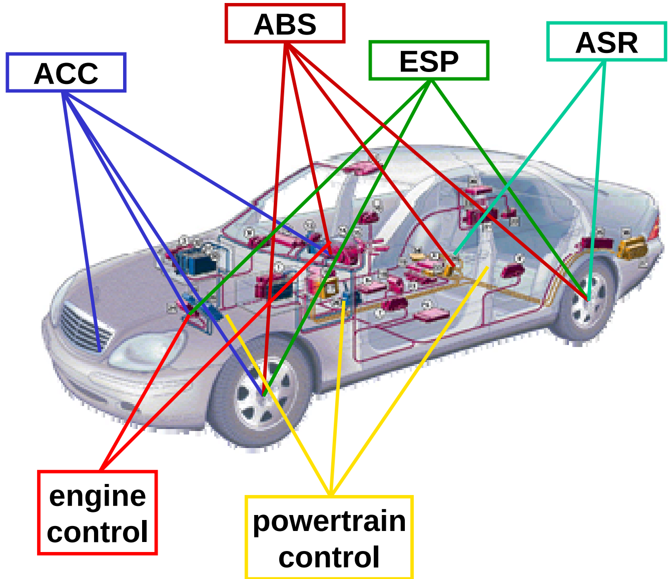
Industry related:

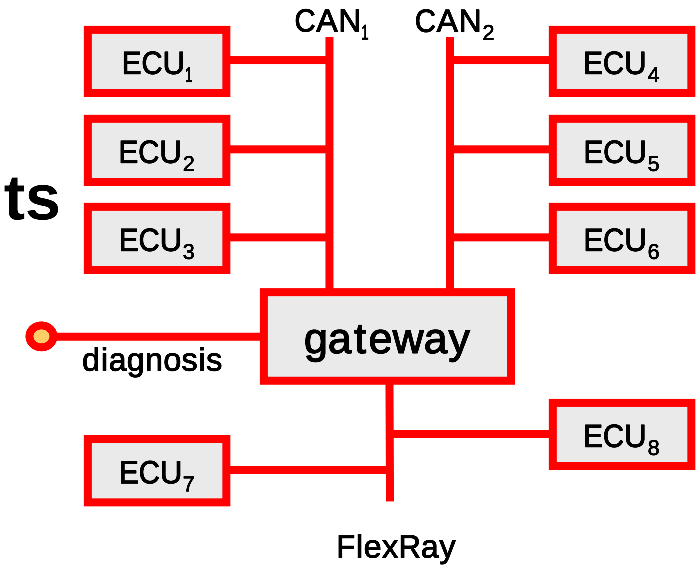
- OSEK flavours
- CAN

Sensitivity Analysis

Challenges

- Heterogeneous
  - Hundreds of functions
  - 50+ ECUs
  - Several RTOSes and protocols
  - Strongly networked
  - Many suppliers
- Complex performance requirements
  - End-to-end deadlines
  - Hidden timing dependencies

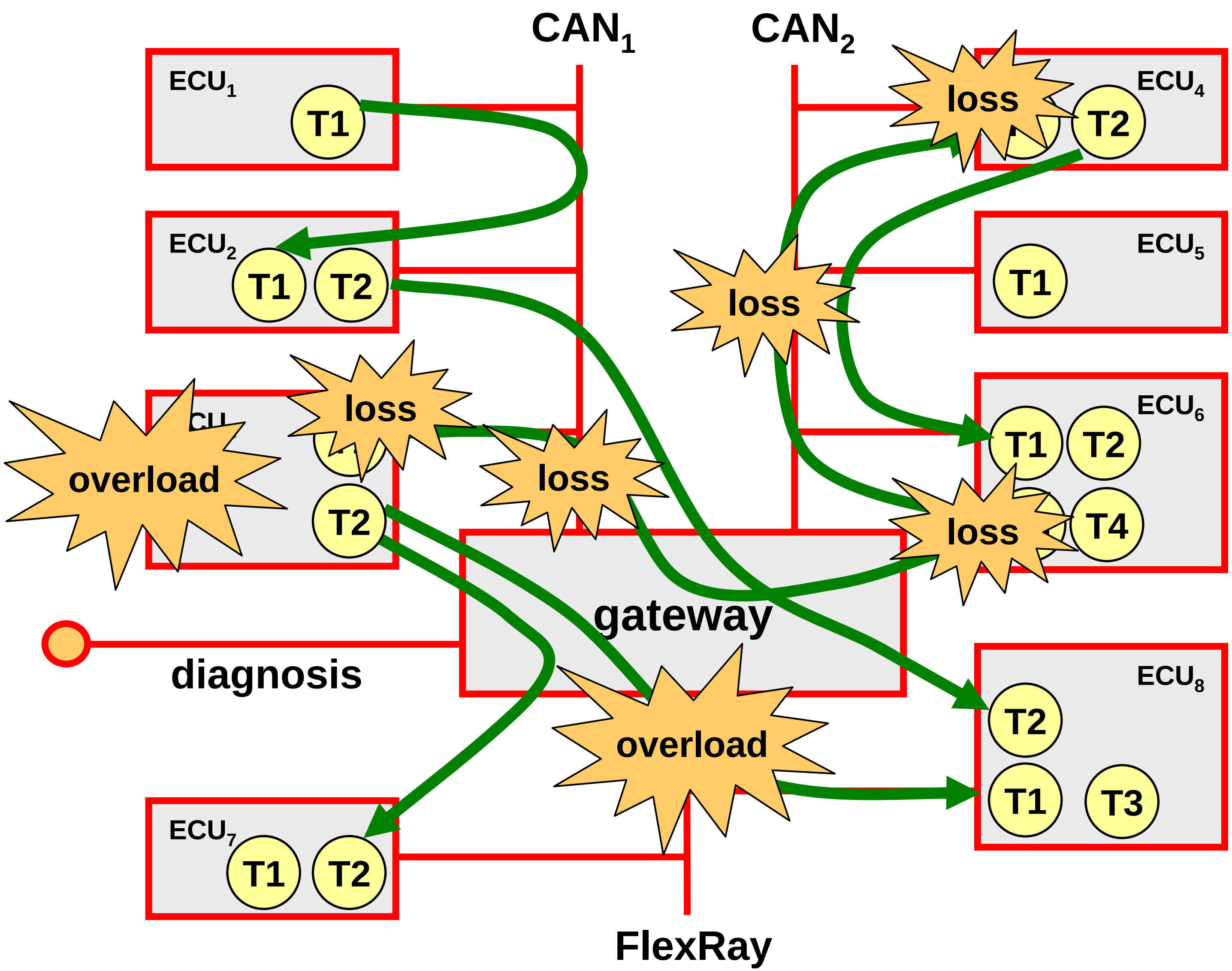




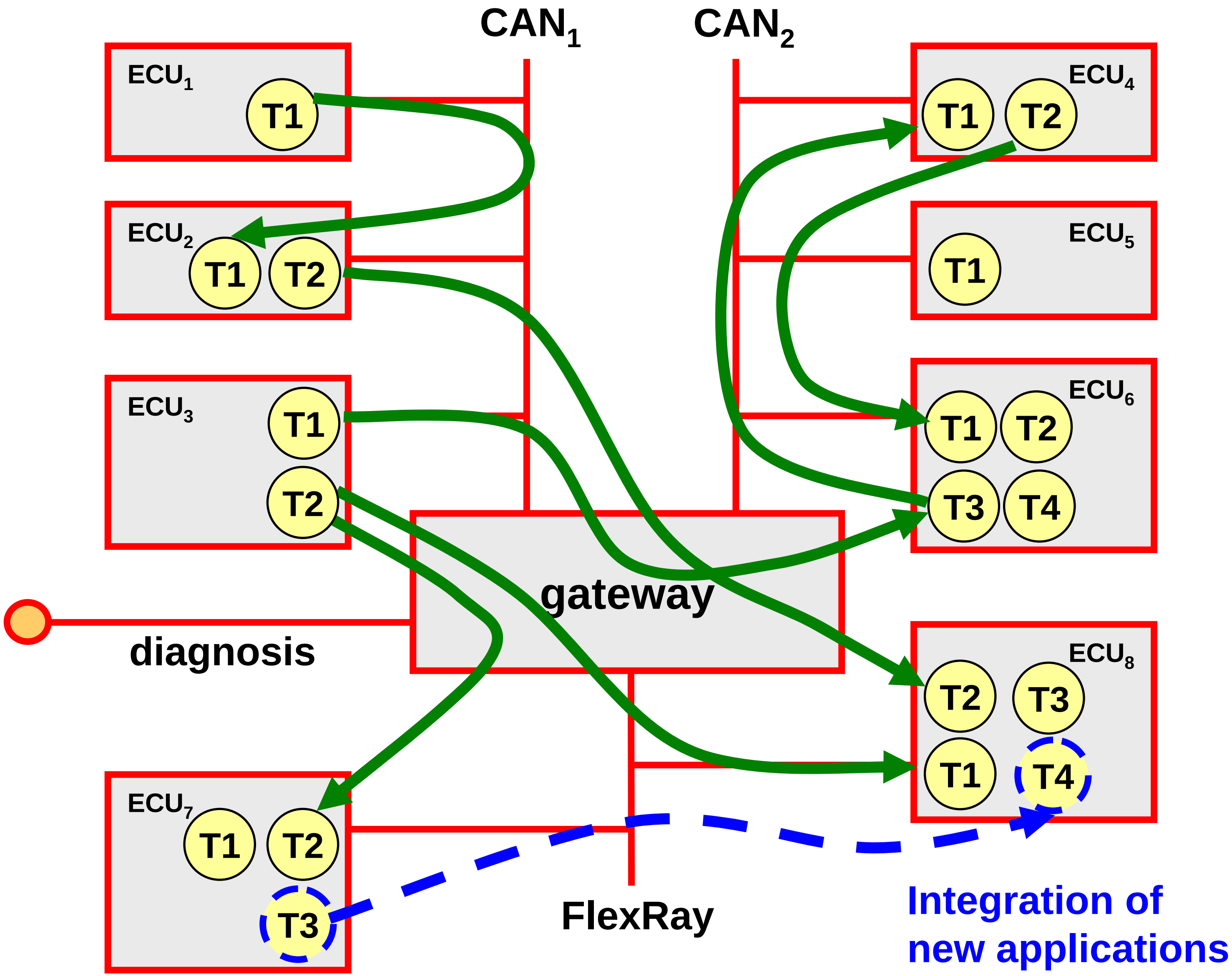
Motivation

- **Modifications of design properties**
  - **During the design process**
    - Refinement of early design data estimations
    - Refinement and changes of specification
    - Exchange of platform components: replace CPU or memory type
  - **In the product lifecycle**
    - Product updates (HW, firmware and SW)
    - Integration of new components or subsystems
    - Change in the environment: applications (smart phone), technical system (motor speed)
  - **In the field**
    - Dynamic systems
    - Unplanned environment situations (resilience)
- **Such changes introduce uncertainties and increase design risk**

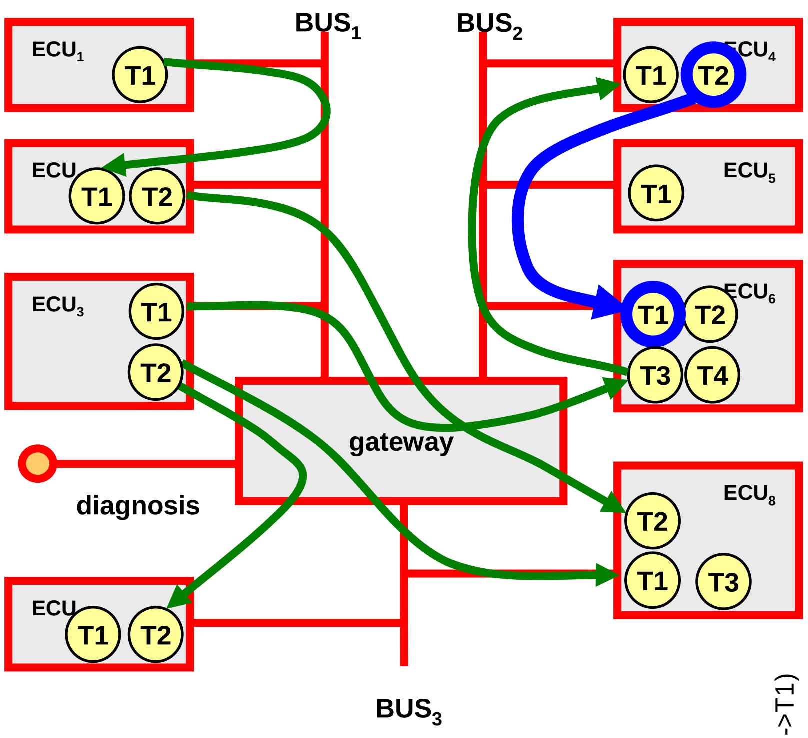
Domino effects due to parameter changes

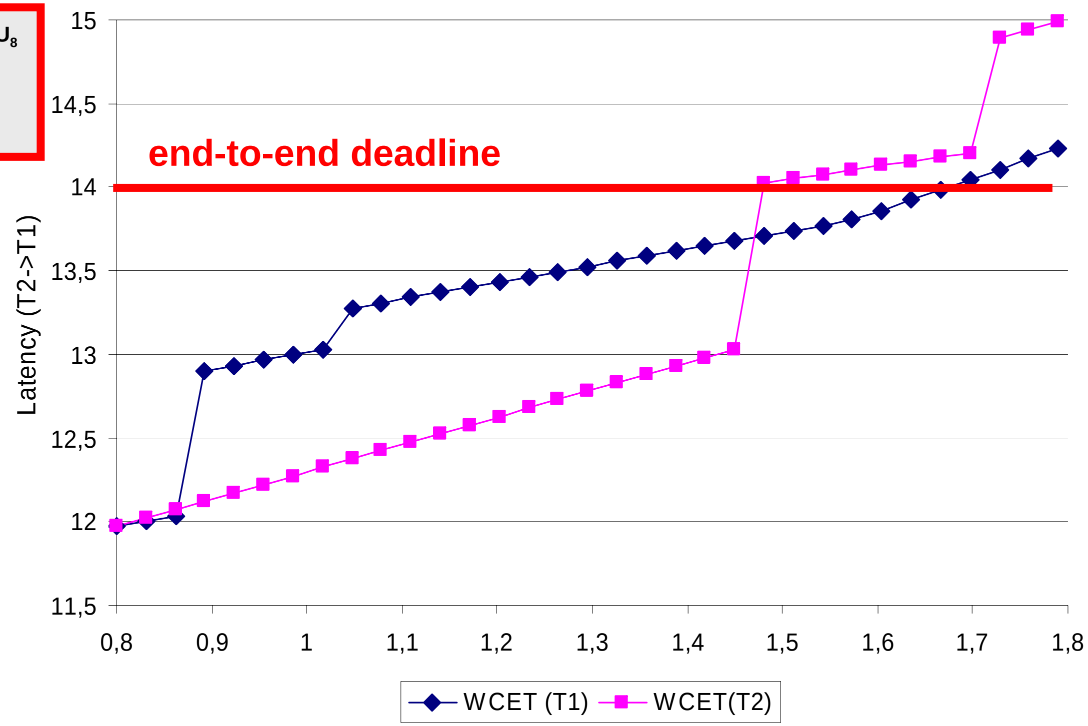


Multi-dimensional sensitivity analysis



Example: WCET variation





Sensitivity analysis

- Sensitivity analysis identifies limits of feasible design
  - How far can system properties be changed before the system fails → slack ?
  - What is the impact of property changes on the performance metrics?

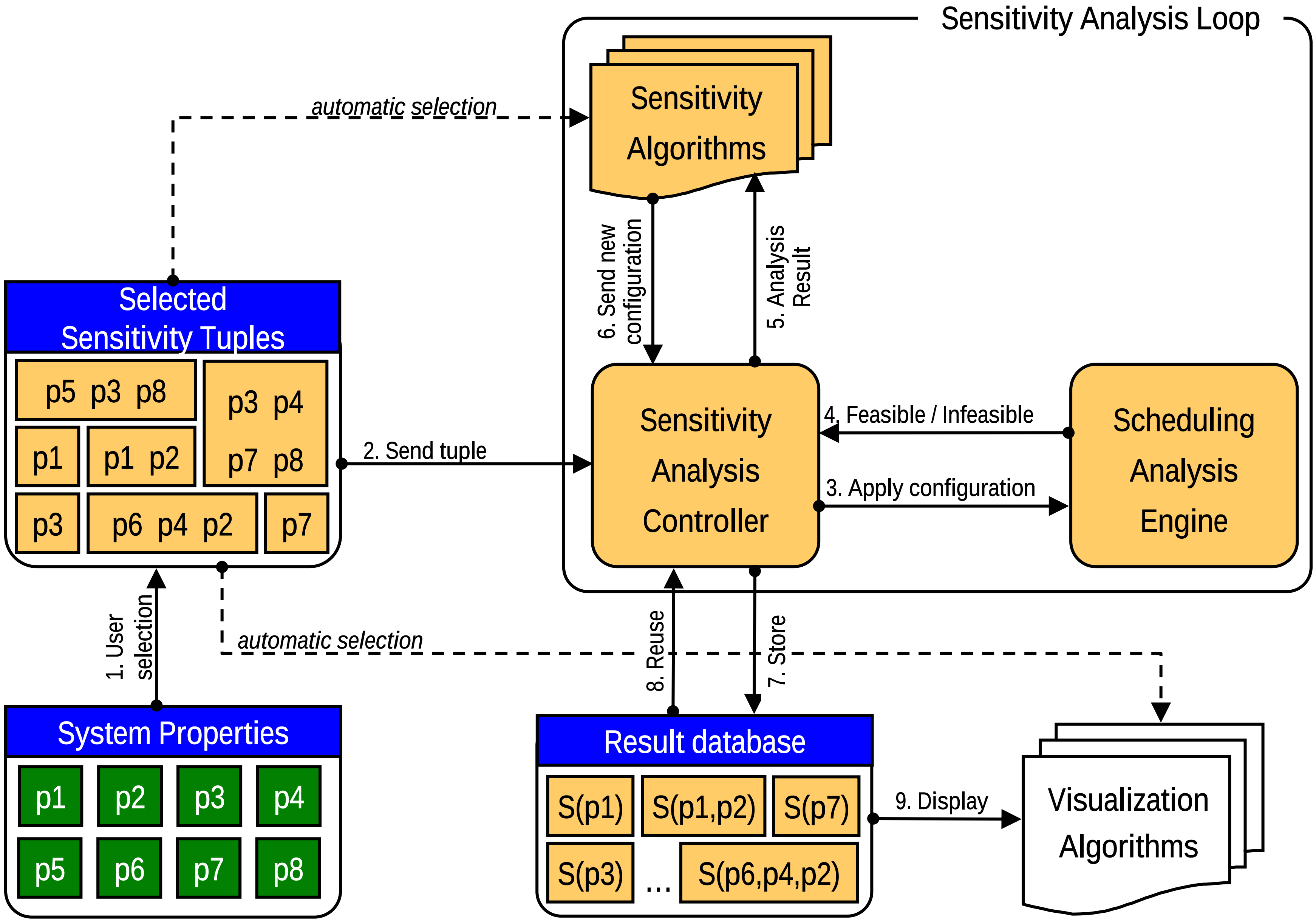
Sensitivity analysis key features

- **Evaluates design risk linked with a specific component**
  - **helps to controls parameter changes**
  - **captures „domino"- effects**
  - **metric for design robustness**
- **Assistance for system dimensioning/configuration**
  - **choose optimal bus bandwidth, CPU clock speed**

Design properties considered

- All design data can be subject to changes → complex issue
- Here we assume
  - Fixed architecture
  - Fixed mapping of functions to components
- Modification of performance related SW and HW component properties
  - Platform component performance (processor and communication links)
  - Execution times of individual processes
  - Process communication volumes
- Considered performance metrics
  - Predictable design → worst case data
  - Response times
  - End-to-end latencies

Sensitivity analysis framework in SymTA/S



Sensitivity Analysis Framework

- Based on SymTA/S analysis engine
- Formally derived search space boundaries
  - based on load conditions
  - finds discontinuity points (scheduling anomalies)
- Binary search technique
  - optimal → minimum number of search steps
  - bidirectional search space
    - feasible → infeasible
    - Infeasbile → feasible
  - transparent with respect to scheduling algorithms
  - applicable only on monotonic search spaces
    - if non-monotonic behavior, then split search space in monotonic sub-spaces

Sensitivity Analysis - Algorithms

- **One dimensional analysis**
  - **Formally derived search space boundaries**
  - **Binary search like search**
- **Two dimensional analysis**
  - **Divide-and-conquer like search algorithm**
  - **Parameter specific heuristics for search space reduction**

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

SymTA/S
Design space exploration
and
System Robustness Optimization

ARTIST2 PhD Course, June 12, DTU Copenhagen, Denmark

Razvan Racu

Arne Hamann

INSTITUTE OF
COMPUTER AND
COMMUNICATION
NETWORK ENGINEERING

Design Space Exploration Framework

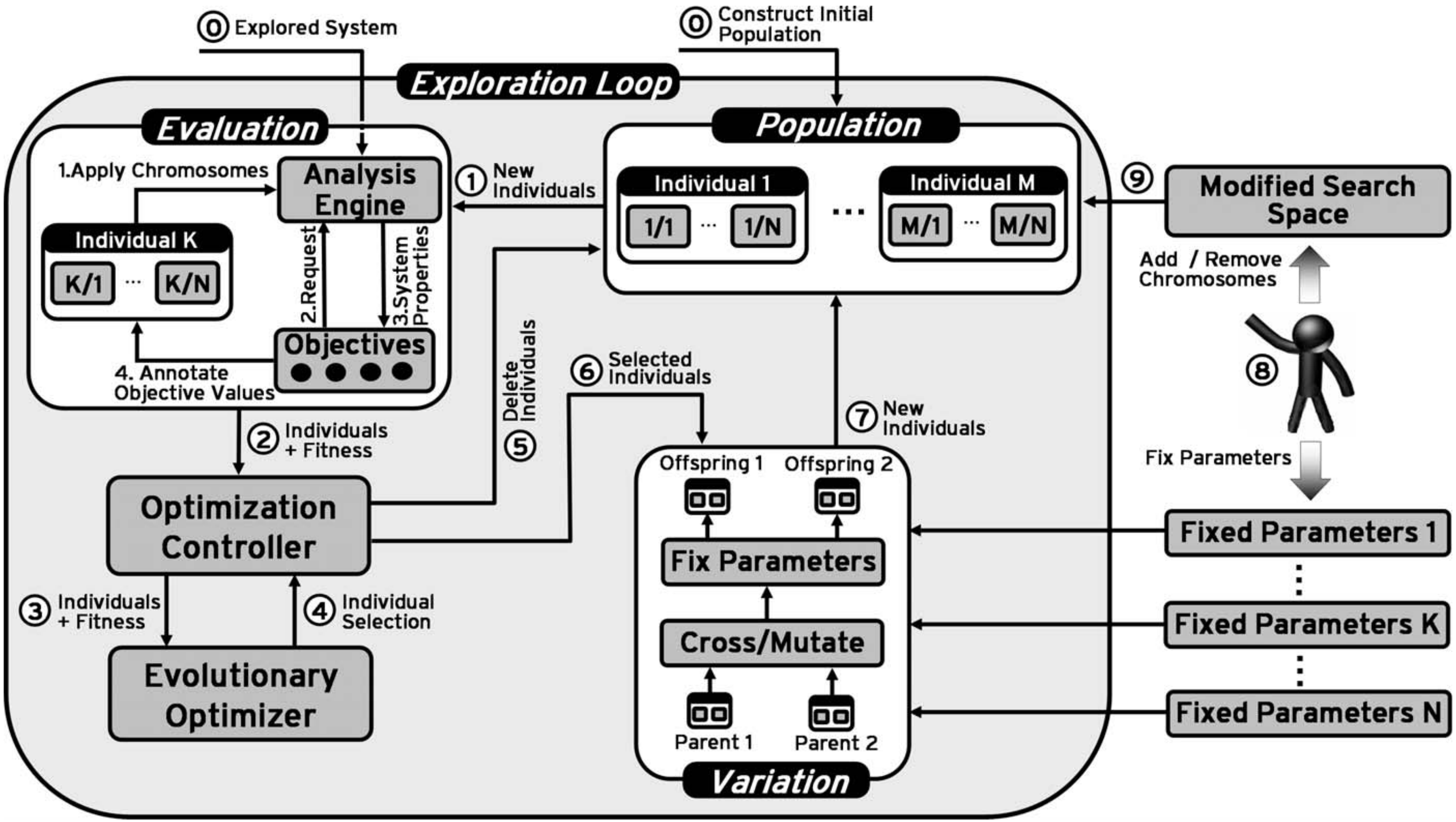
Outline

- SymTA/S design space exploration framework
- Problem independent selector algorithms
- Example application: Timing optimization in SymTA/S

Design Space Exploration Framework

- Compositional search space encoding scheme
- Dynamic search space modification
  - user-controlled exploration
  - automatic search space adaptation
- High flexibility and extensibility
- Pareto-optimization of arbitrary optimization objectives
  - Evolutionary search techniques, PISA, ETH Zurich
- Exploration speed-up through meta-heuristics
  - problem independent
  - problem dependent

Exploration loop



Selector example: FEMO

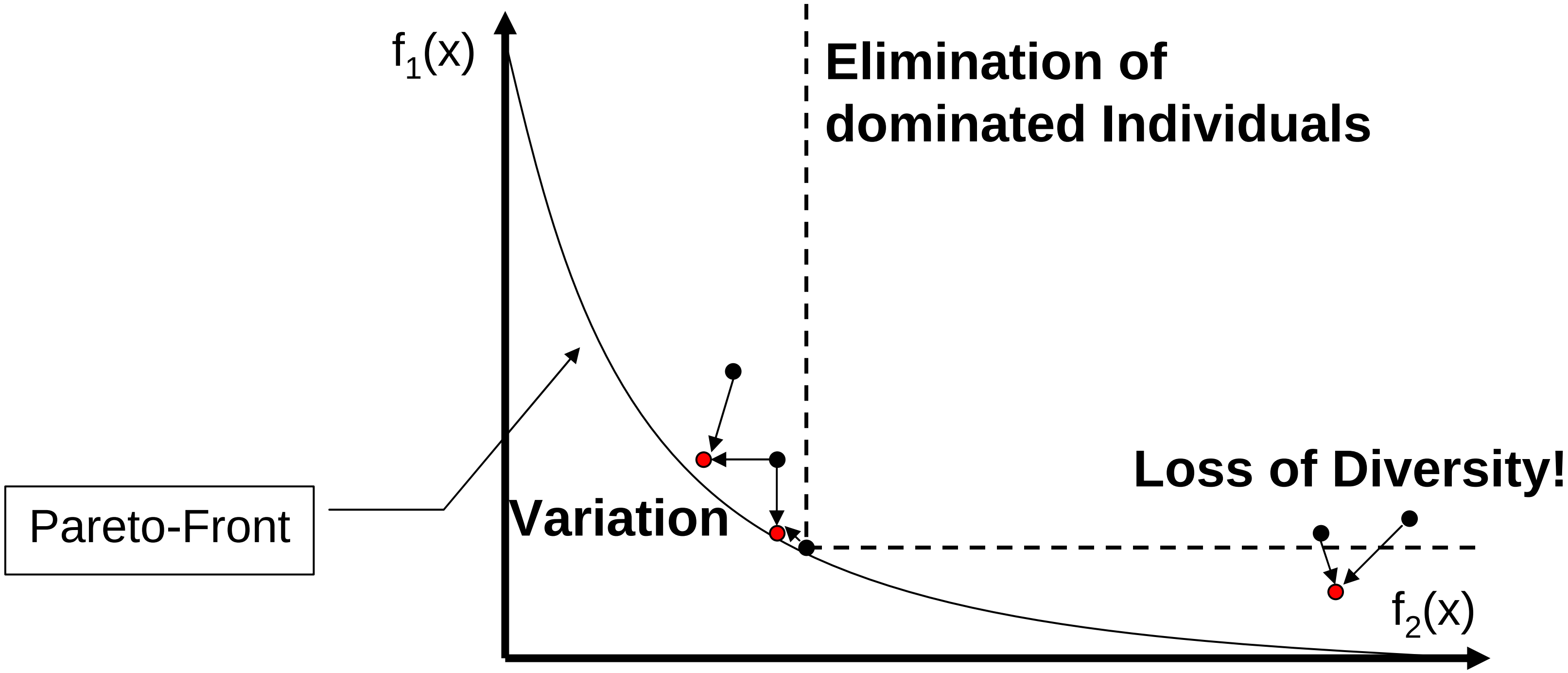
- **Fair Evolutionary Multi-objective Optimizer (FEMO)**
- **Developed by Zitzler and Thiele (~2002), ETH Zürich**
- **Idea: Offspring based selection**
  - **Count for each individual the number of his offsprings**
  - **Select individuals with equal rate for procreation → Fairness**
- **Remove all dominated (i.e. not Pareto-optimal individuals) after each generation → variable population size**
  - **All individuals in population are Pareto-optimal, none is “better” than another**
  - **Possible problem: search space coverage**

FEMO Algorithm

- **Add random initial individuals to the population**
- **Repeat until stop condition:**
  - **Select individual i with the least offsprings**
  - **Create offspring i' through crossover and mutation**
  - **Remove all individuals from population that are Pareto-dominated by i'**
  - **Add i' to the population if it is not Pareto-dominated by any other individual**

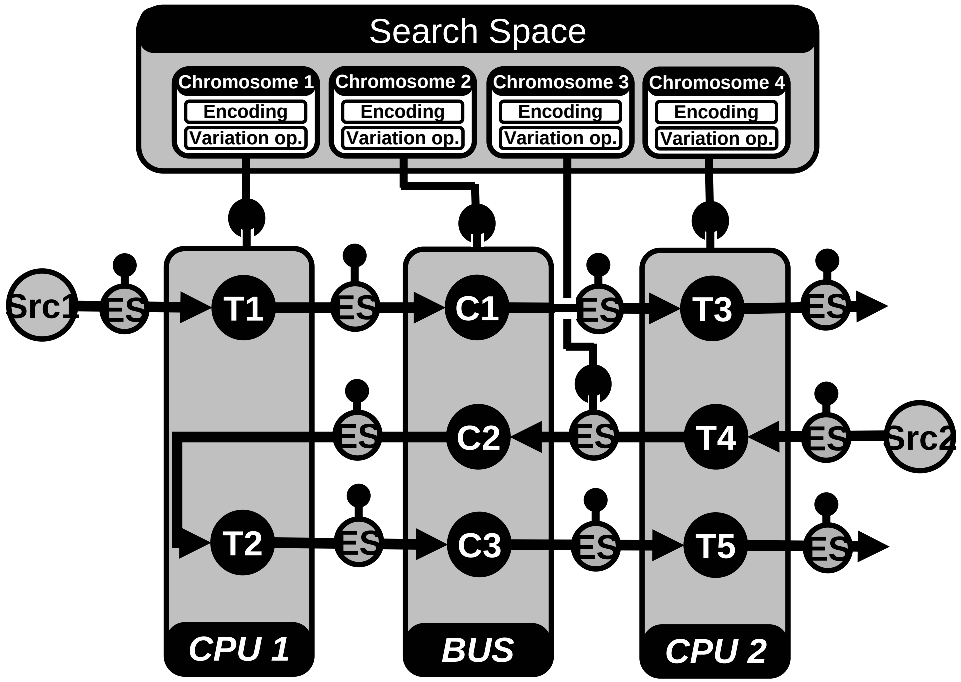
FEMO: Evolutionary Search Strategy

- **Diversity vs. Convergence speed**

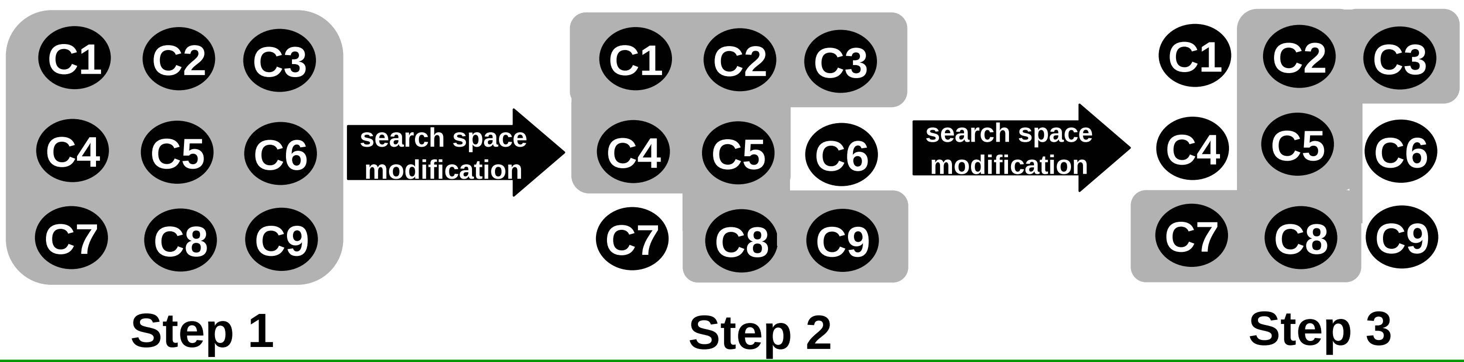


Exploration control

- **Compositional encoding**



- **Search space adaptation**



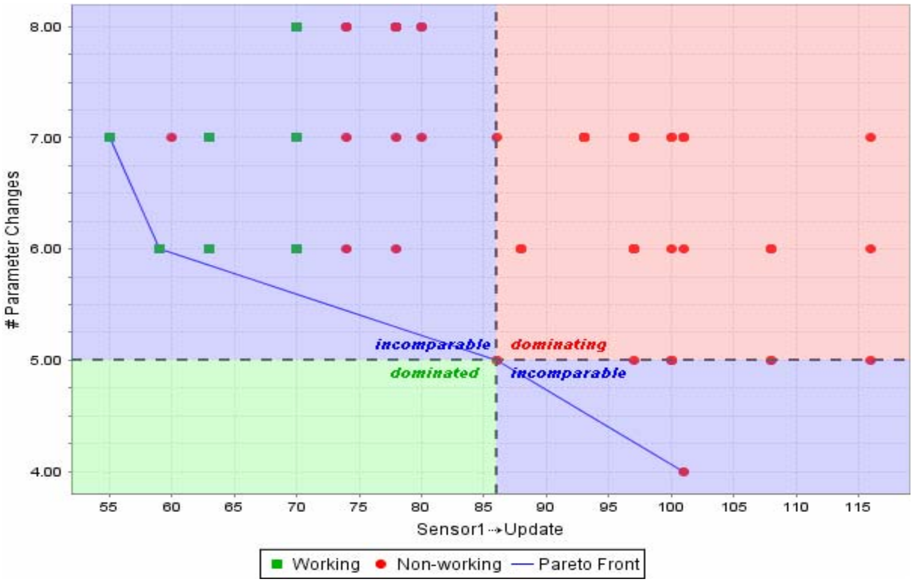
Application domains

- **System optimization**
  - **timing (jitter, end-to-end deadlines)**
  - **buffer sizes**
  - **power dissipation**
  - **mapping**
- **Robustness optimization**
- **Multi-dimensional sensitivity analysis**
- **System generation**
- **…**

Example application: Timing optimization 1/2

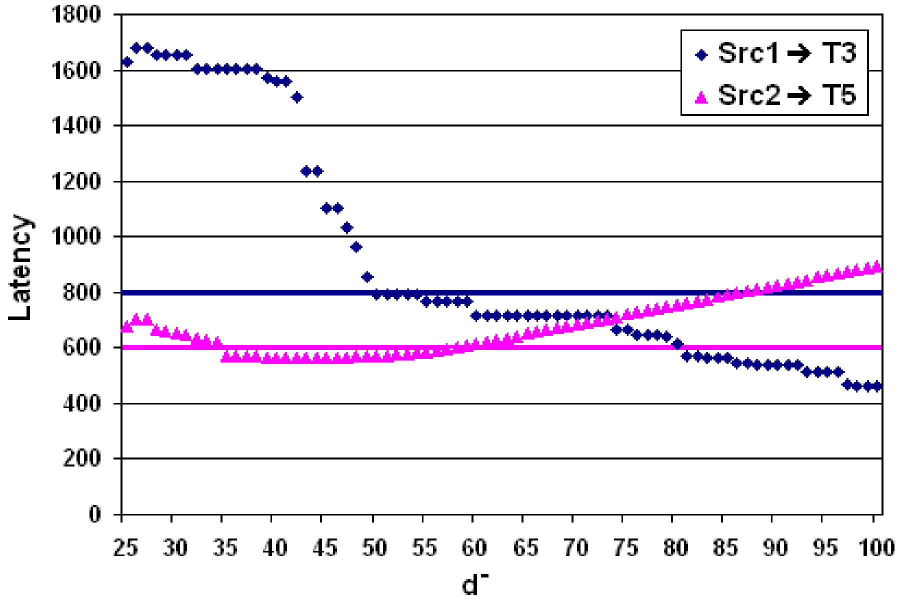
- **Search space**
  - **scheduling parameter for various policies: SPP, TDMA ,RR, EDF**
  - **optimization of parameters for real world RTOSes and bus protocols: ERCOSEK, CAN**
  - **optimization through traffic shaping**
  - **mapping optimization**
  - **…**
- **Optimization Objectives**
  - **end-to-end latencies, worst-case response times**
  - **buffer sizes**
  - **power consumption**
  - **system cost**
  - **# parameter changes**
  - **…**

Example application: Timing optimization 2/2



Pareto-front: end-to-end deadline vs. # parameter changes

Influence of Traffic Shaping on System Performance



Robustness Optimization

Outline

- **System property variations**
- **Sensitivity Analysis**
- **Stochastic Multi-dimensional Sensitivity Analysis**
- **Robustness Metrics**
  - **Hypervolume calculation**
  - **Minimum Guaranteed Robustness (MGR)**
  - **Maximum Possible Robustness (MPR)**
- **Experiments**

System Property Variations (1)

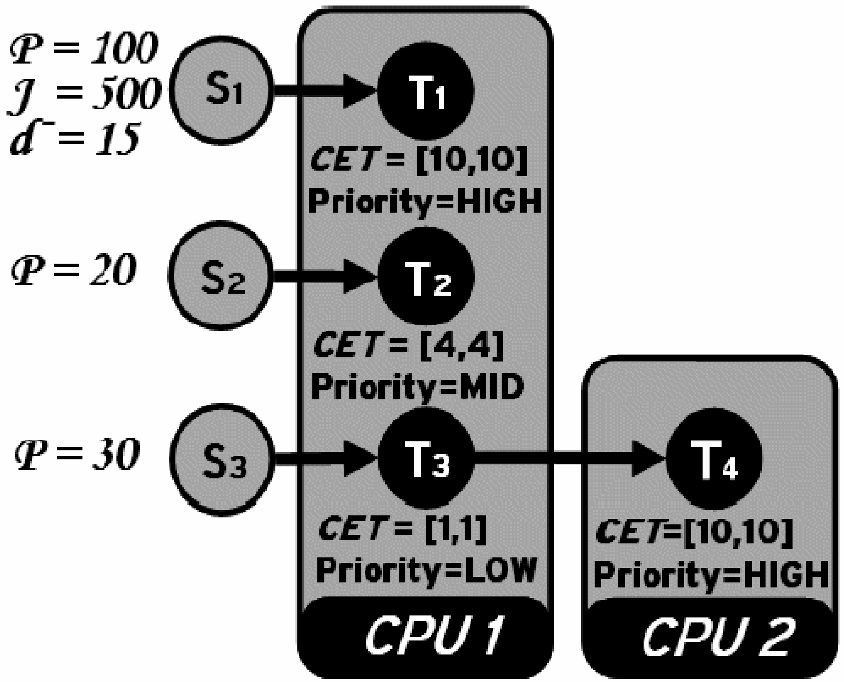
- Two types of system property variations
  - Variations influencing the system load
    - Software execution path length
    - Communication volumes
    - Input data rates
  - Variations influencing the system service capacity
    - Processor clock-rate
    - Communication link performance

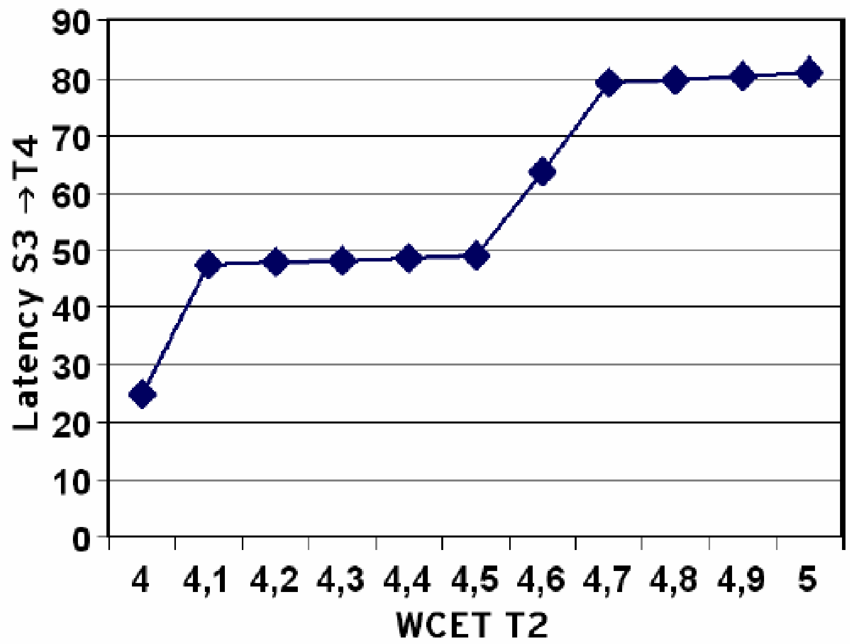
System Property Variations (2)

- **Why do system property variations occur?**
  - **Specification changes, late feature requests, product variants, software updates, bug-fixes**
- **Robustness to property variations**
  - **decreases design risk, and**
  - **increases system maintainability and extensibility**
- **Property variations can have severe unintuitive effects on system performance**

Example: WCET Variation

- **End-to-End latency S3→T4 as a function of execution demand of T2**





System Property Variations (3)

- Property variations invalidate the assumption under which the system was dimensioned and configured
- →Correct function and performance of the system is put at risk
- How can we increase the robustness of the system to property variations ?
  - Adaptivity: feedback-based scheduling, self-organizing systems,…
  - Sensitivity analysis: achieve robustness without on-line parameter adaptation

Problem Formulation

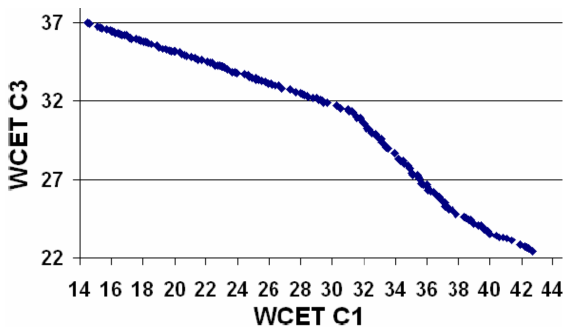
- **Find parameter configuration that …**
- **… maximizes the robustness of the given system w.r.t. changes of several properties**
- **Robustness = the system can sustain a certain degree of property variations without severe performance degradation**
- **→Multi-dimensional optimization problem**
- **Not included: dynamic parameter adaptations as a reaction to property variations**

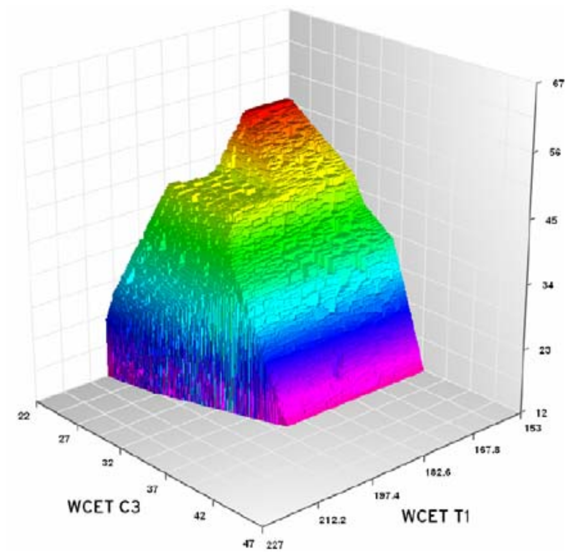
Sensitivity Analysis (1)

- Calculates maximum/minimum admissible values for given system properties
- Supported system properties
  - WCETs / BCETs
  - Communication volume
  - CPU clock rate
  - Bus throughput, …

Sensitivity Analysis (2)

- **One-dimensional case**
  - **maximum/minimum feasible property value**
- **Multi-dimensional case**
  - **front separating feasible and non-feasible system property combinations: sensitivity front**





Sensitivity Analysis (3)

- **Recent results:**
  - **One-dimensional sensitivity analysis**
    - **Calculates slack for a single system property**
    - **Vestal: Trans. on Software Engineering 1994**
    - **Racu: RTAS 2005**
  - **Multi-dimensional sensitivity analysis**
    - **Considers interdependencies between multiple system properties**
    - **Racu / Hamann: ECRTS 2006**
- **Problem: computational effort grows exponentially with problem dimension**

Stochastic Sensitivity Analysis (1)

- **Solution: scalable stochastic analysis to bound system sensitivity**
- **Sensitivity analysis formulated as multi-objective optimization problem**
- **Search space: System properties including WCETs, Periods, Jitters, …**
- **Optimization objectives: maximization / minimization of considered system properties**
  - **Pareto-optimization**

→Pareto-front of optimization task corresponds to sought-after sensitivity front

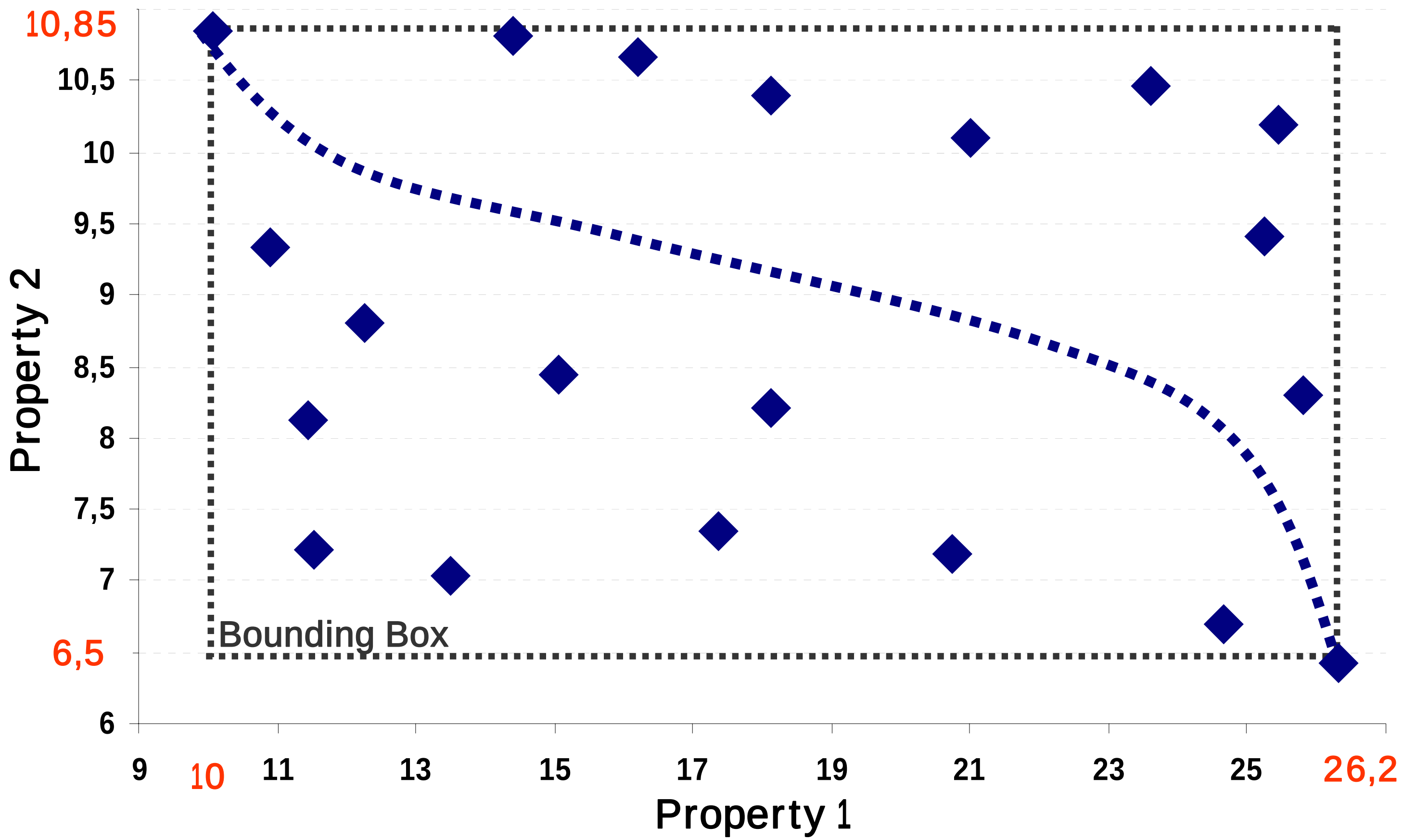
Stochastic Sensitivity Analysis (2)

- Uses multi-criteria evolutionary algorithms to approximate sensitivity front
  - responsible for sensitivity front coverage
  - Currently used SPEA2 (ETH Zurich): diversified sensitivity front approximation through Pareto-dominance based selection and density approximation
- Can be used for system properties subject to maximization (e.g. WCETs) and minimization (e.g. Periods)
- In the following: properties are subject to maximization

Creation of the Initial Population

- **Creates a certain number of points representing a first approximation of sensitivity front**
- **Uses 1-dim sensitivity analysis**
  - **to bound the search space in each dimension (bounding hypercube)**
  - **to generate points representing the extrema of the sought-after sensitivity front**
- **Randomly place the rest of the initial points in bounding hypercube**

Initial Population - Example



Bounding the Search Space (1)

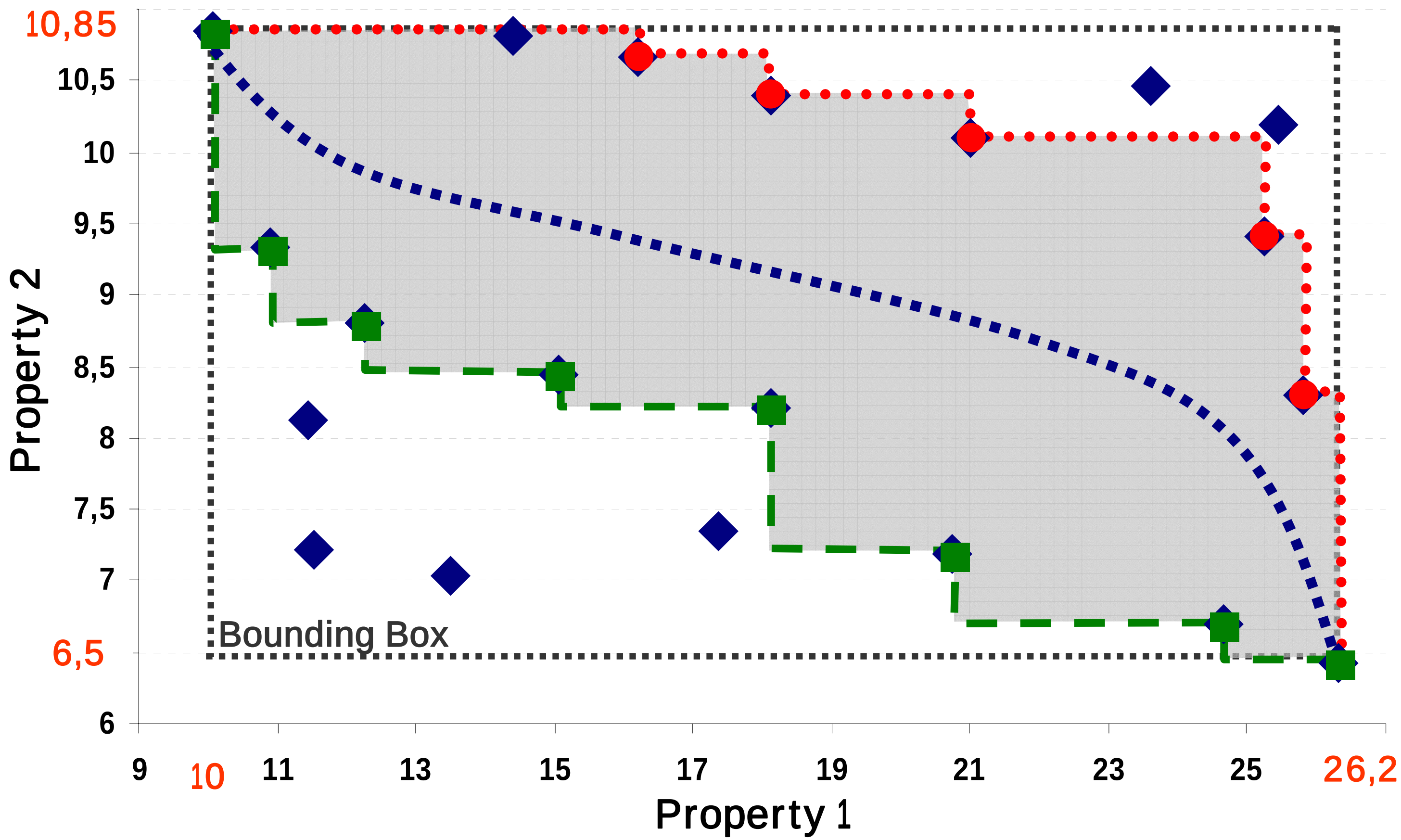
- **Extension for stochastic sensitivity analysis for robustness optimization**
- **Idea: bound search space containing the sought-after sensitivity front**
  - **Bounding working Pareto-front $\mathcal{F}^{n}$**
    - **evaluated Pareto-optimal working points**
  - **Bounding non-working Pareto-front $\mathcal{F}^{nw}$**
    - **evaluated Pareto-optimal non-working points**
- **Bounding Pareto-fronts can be used to derive multi-dim. robustness metrics (later)**

Bounding the Search Space (2)

- **Space between bounding Pareto-fronts is called interesting region**
- **Variation operators use algorithm ensuring that generated offsprings (points) are contained in interesting region**
  - **Below bounding non-working Pareto-front**
  - **Above bounding working Pareto-front**

→ Efficiently focuses exploration effort

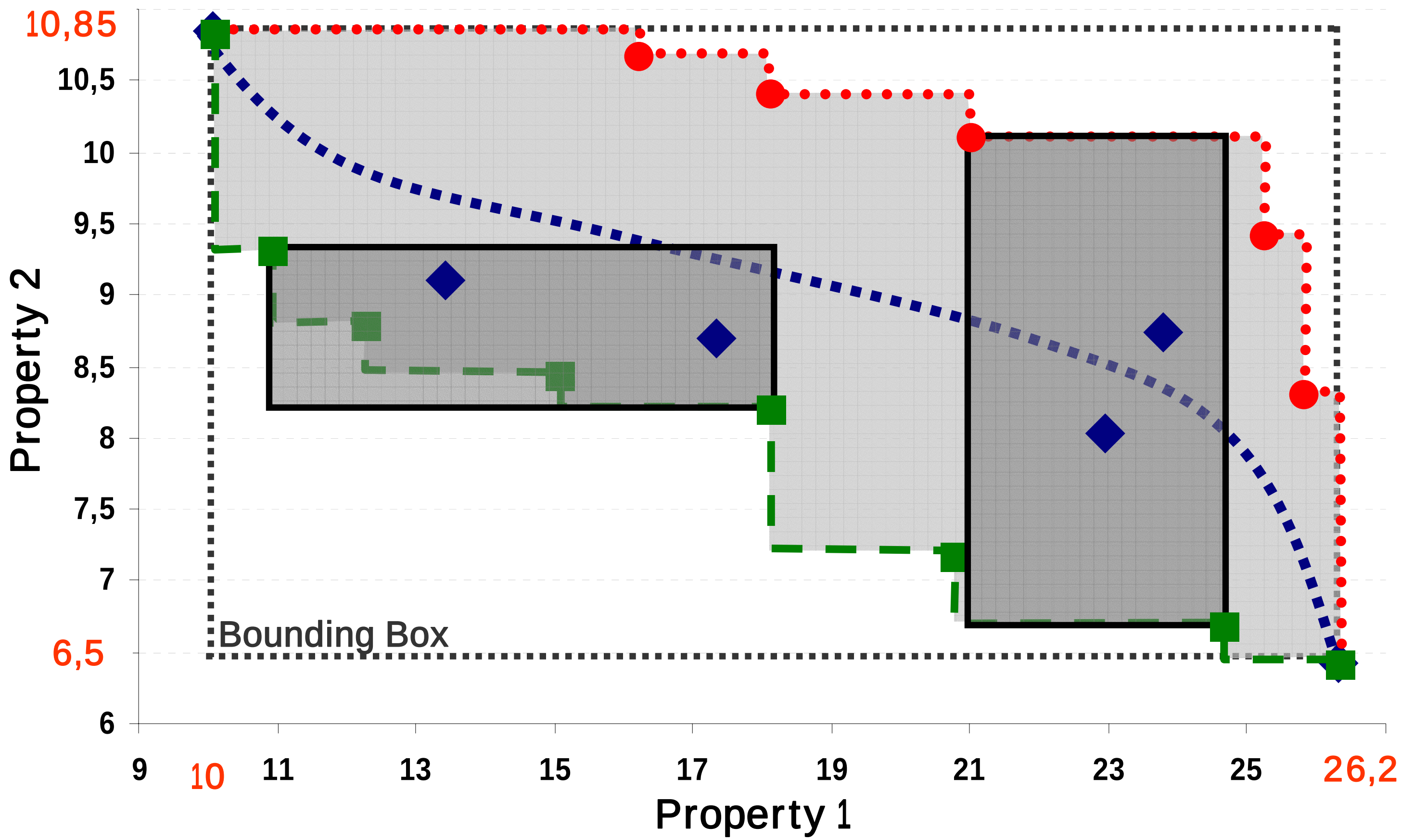
Bounding the Search Space (3)



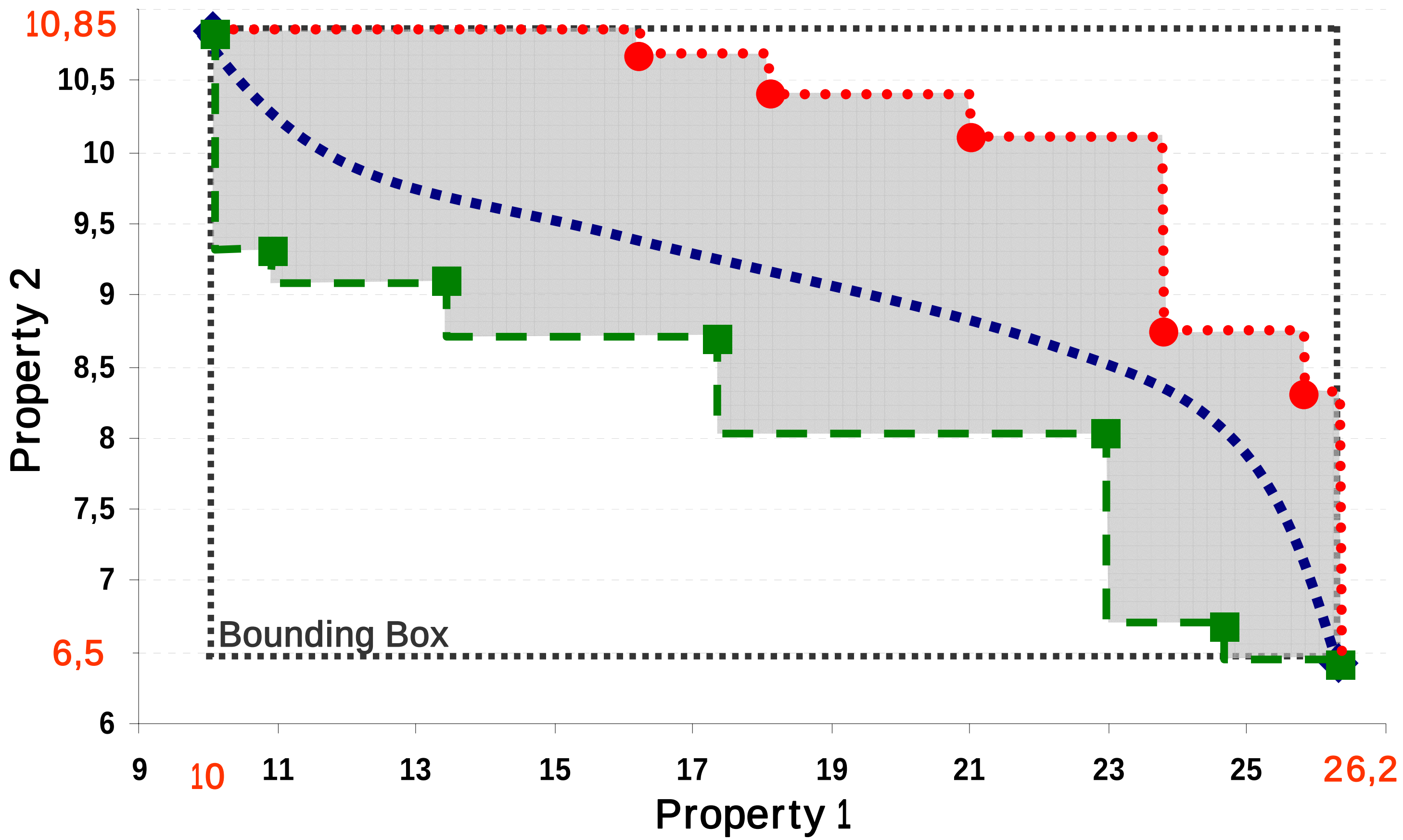
Random Crossover (1)

- **Takes as input two parent points to create two offspring points**
- **The two parent points define hypercube in which the created offspring points are randomly placed**
- **Simple standard operator that locally refines the approximation of the sought-after sensitivity front**

Random Crossover (2)



Random Crossover (3)



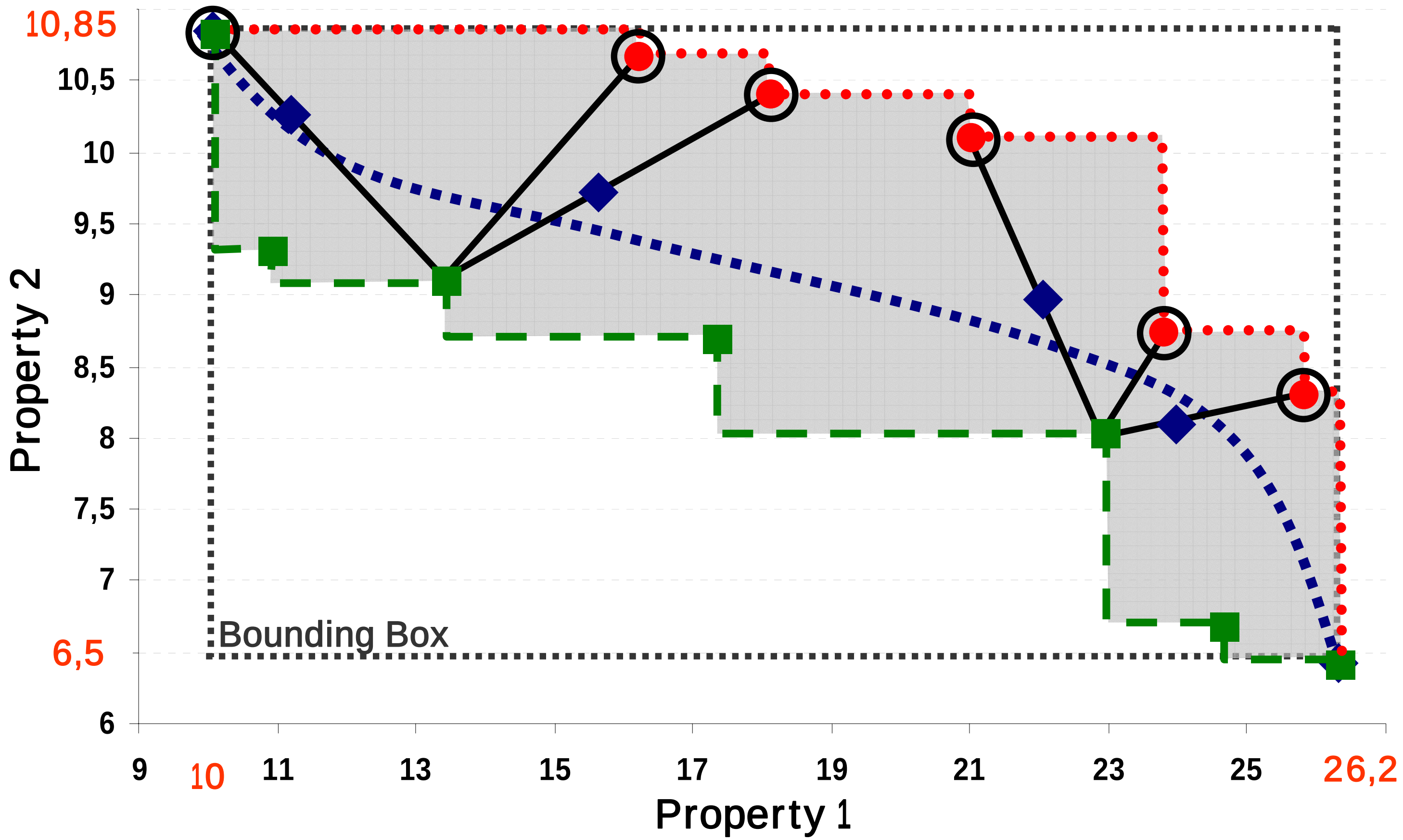
Front Convergence Mutate (1)

- **Takes as input one parent point to produce one offspring point**
- **Heuristic operator adapted to optimization problem**
- **Increases convergence speed**
- **Directly supports the convergence of the bounding Pareto-fronts**

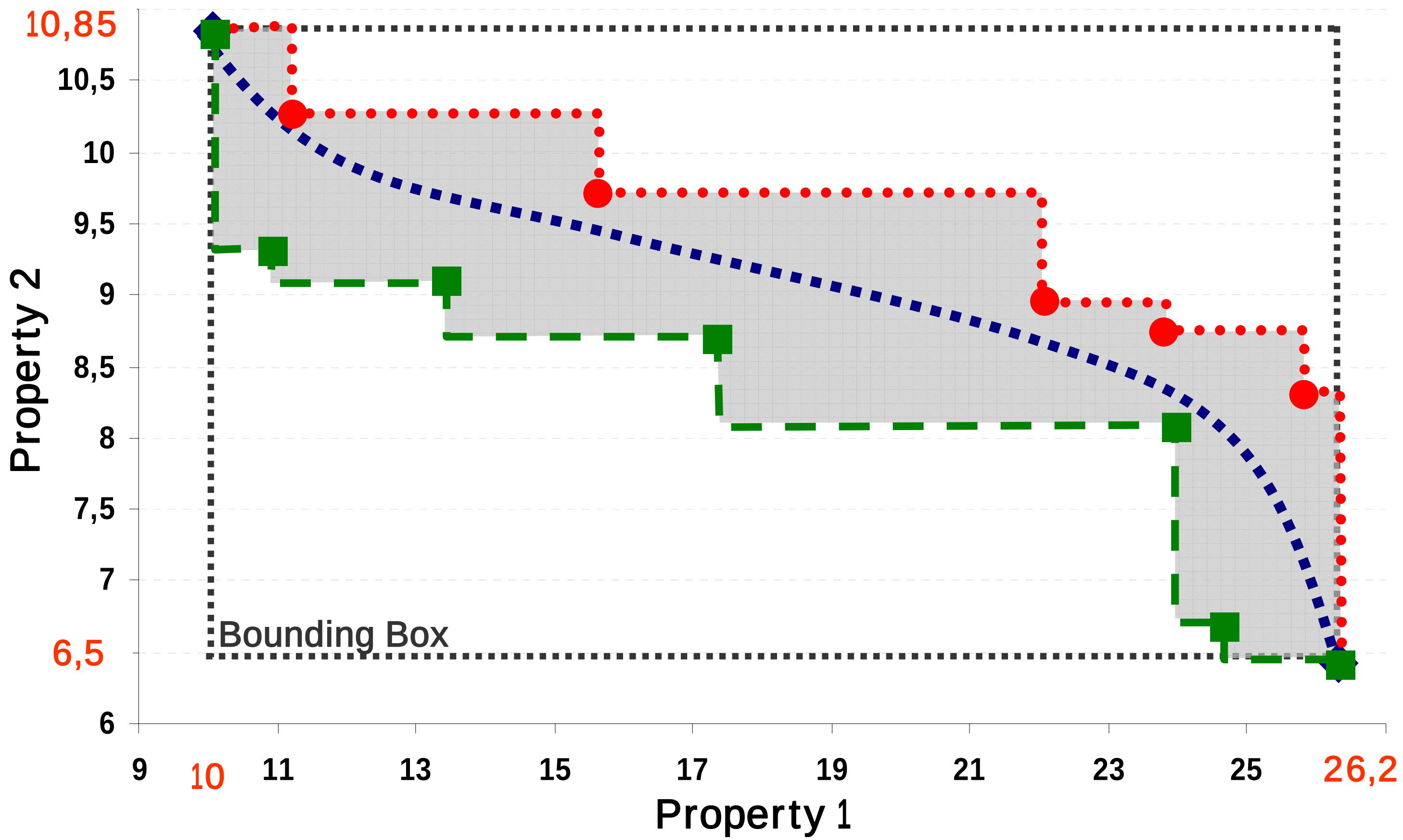
Front Convergence Mutate (2)

- **Strategy:**
  - **Determine X closest points on opposite Pareto-front**
  - **Choose randomly one of these points**
  - **Place offspring point randomly on straight line connecting the parent point and the chosen random point**

Front Convergence Mutate (3)



Front Convergence Mutate (4)

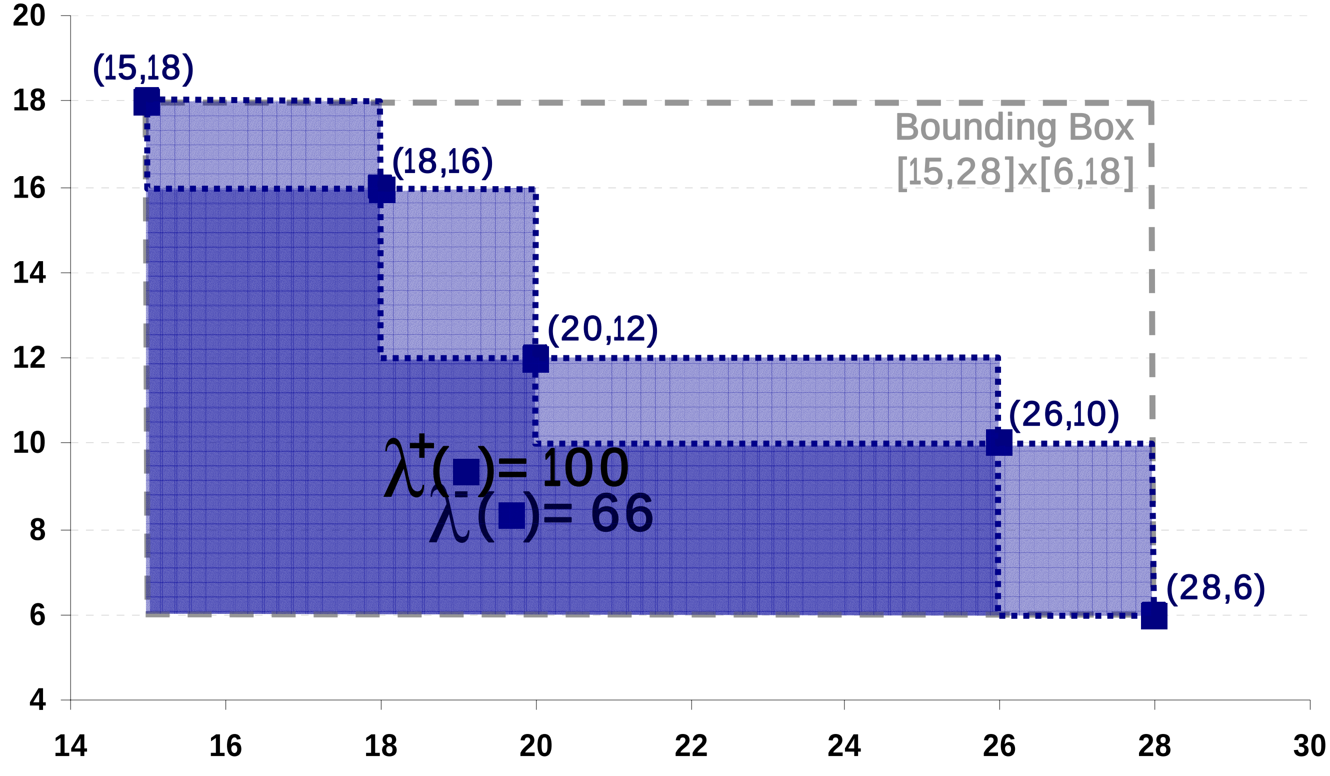


Hypervolume Calculation

- **Hypervolume as basis of the proposed robustness metrics**
- **Hypervolume is defined in a given hypercube and associated to a point set**
- **Two different notions of hypervolume**
  - **inner hypervolume λ^{-}: Volume of space Pareto-dominated by the given points inside the given hypercube**
  - **outer hypervolume λ^{+}: Volume of space Pareto-dominated by all points not Pareto-dominating any of the given points**

Hypervolume Calculation (2)

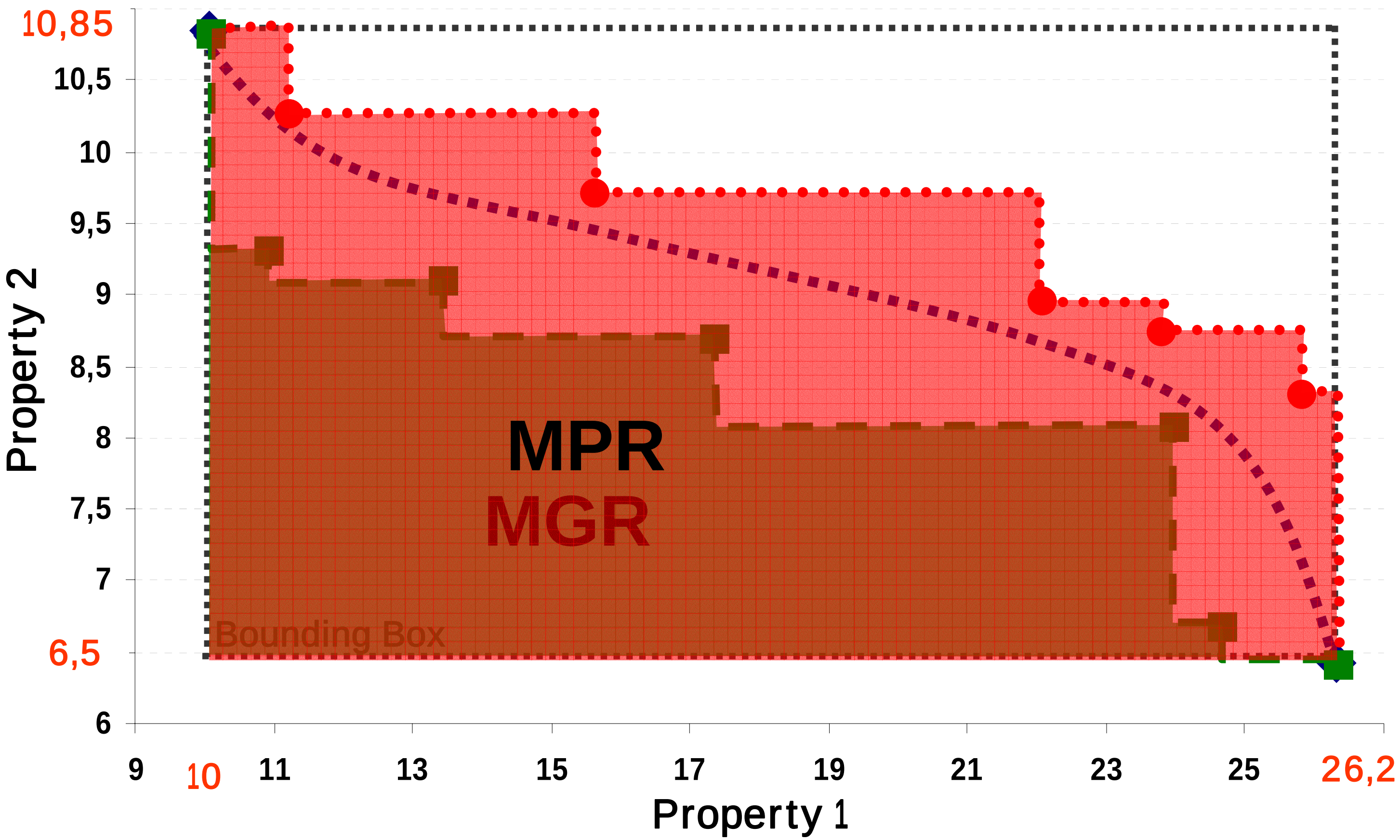
- **2D-case**
  - **inner hypervolume: lower step function**
  - **outer hypervolume: upper step function**



Robustness Metrics

- **Given a set of properties we want to achieve robustness for …**
- **… use stochastic sensitivity analysis to derive upper and lower robustness bounds**
  - **Minimum Guaranteed Robustness (MGR)**
    - **Defined as inner hypervolume of the bounding working Pareto-front $\mathcal{F}^{w}$**
  - **Maximum Possible Robustness (MPR)**
    - **Defined as outer hypervolume of the bounding non-working Pareto-front $\mathcal{F}^{nw}$**

Robustness Metrics (2)



Obviously: MGR <= Real Robustness <= MPR

Robustness Exploration

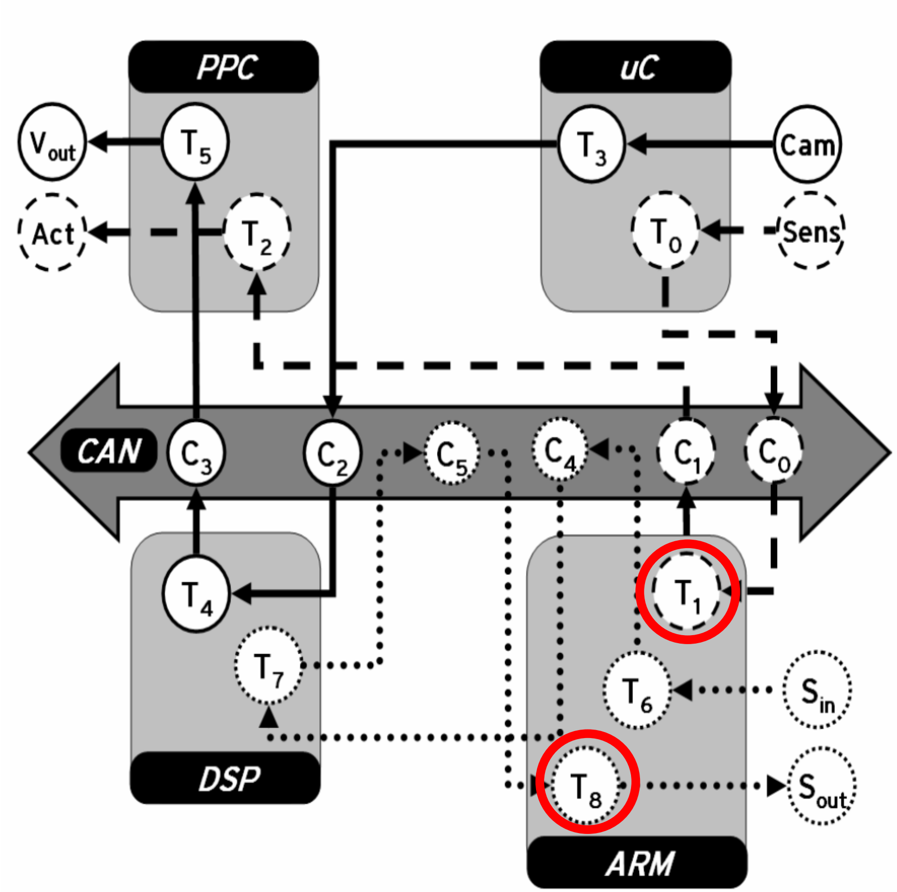
- **Idea: Pareto-optimize MGR and MPR**
- **Advantages**
  - **Stochastic sensitivity analysis is scalable**

  →Little computational effort necessary to reasonably bound robustness potential of given configuration
  - **In-depth analysis can be performed once interesting configurations are identified (i.e. high MGR or high MPR)**

  →Perfectly suited for robustness optimization

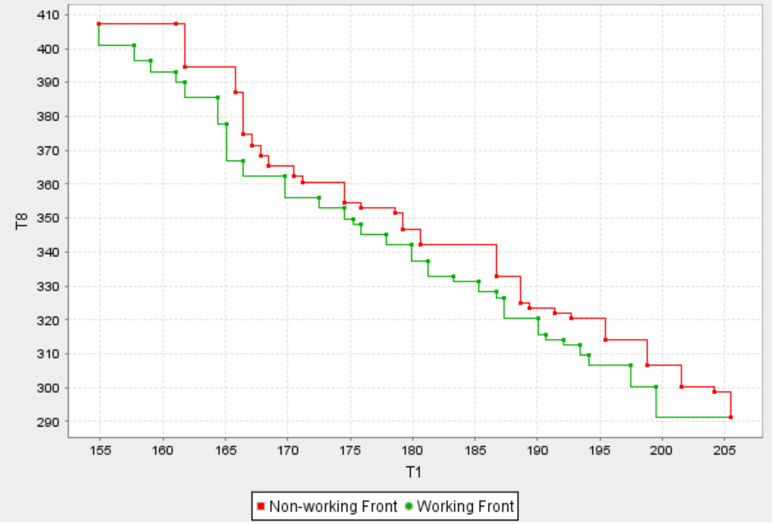
Example System

- **Distributed embedded system**
- **4 computational resource …**
- **… connected via CAN bus**
- **3 applications**
  - **Sens→Act**
  - **S_{in}→S_{out}**
  - **Cam→V_{out}**



Approximation Quality (1)

- **Approximation after 100 evaluations (20 sec)**
- **MGR = 2447**
- **MPR = 2937**

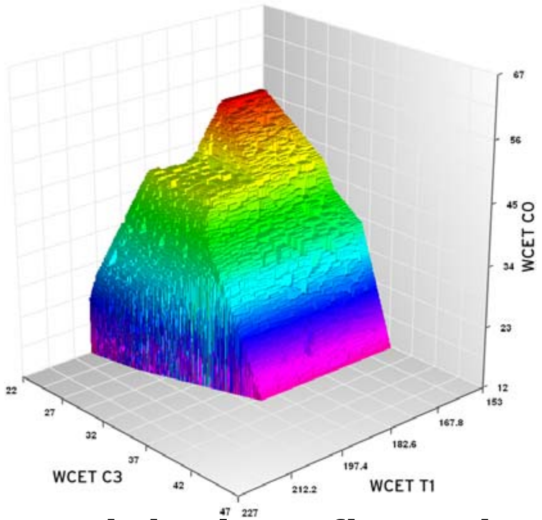


- **Approximation after 200 evaluations (40 sec)**
- **MGR = 2580**
- **MPR = 2813**

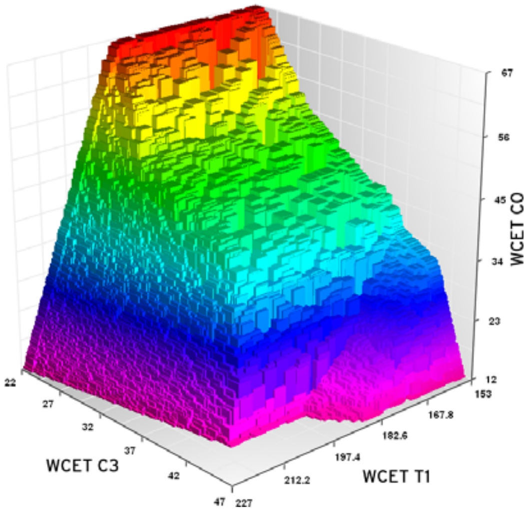
Approximation Quality (2)

- **Approximation after 300 evaluations (60 sec)**
- **MGR = 2632**
- **MPR = 2777**

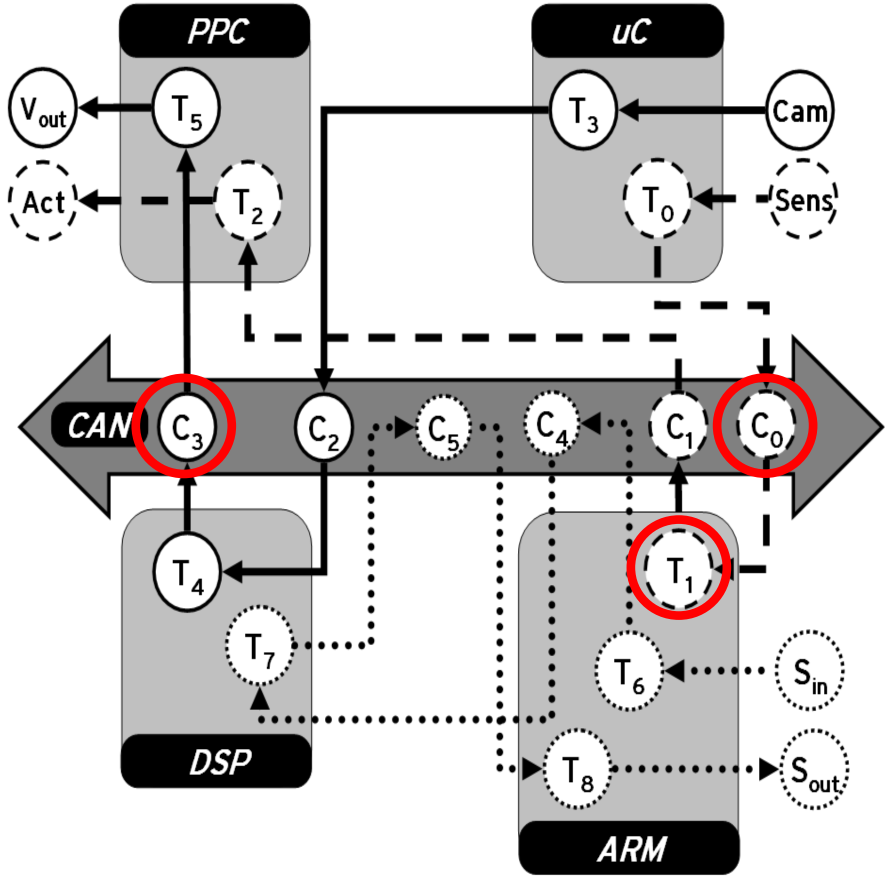
3D - Robustness Maximization (1)



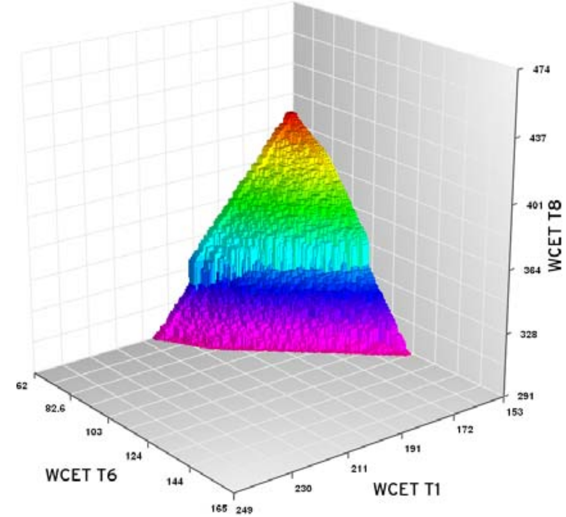
Original configuration



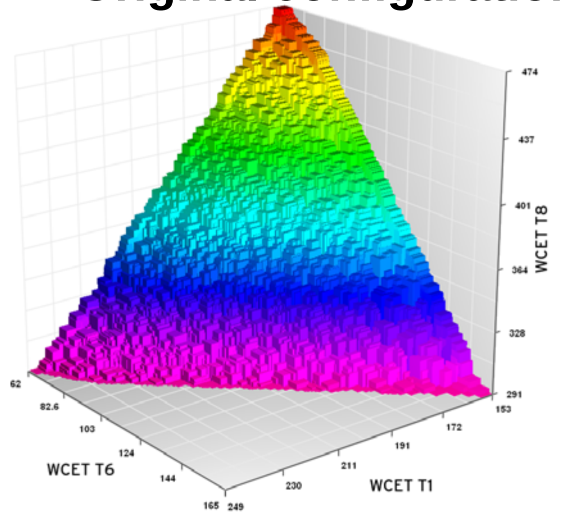
Optimized configuration



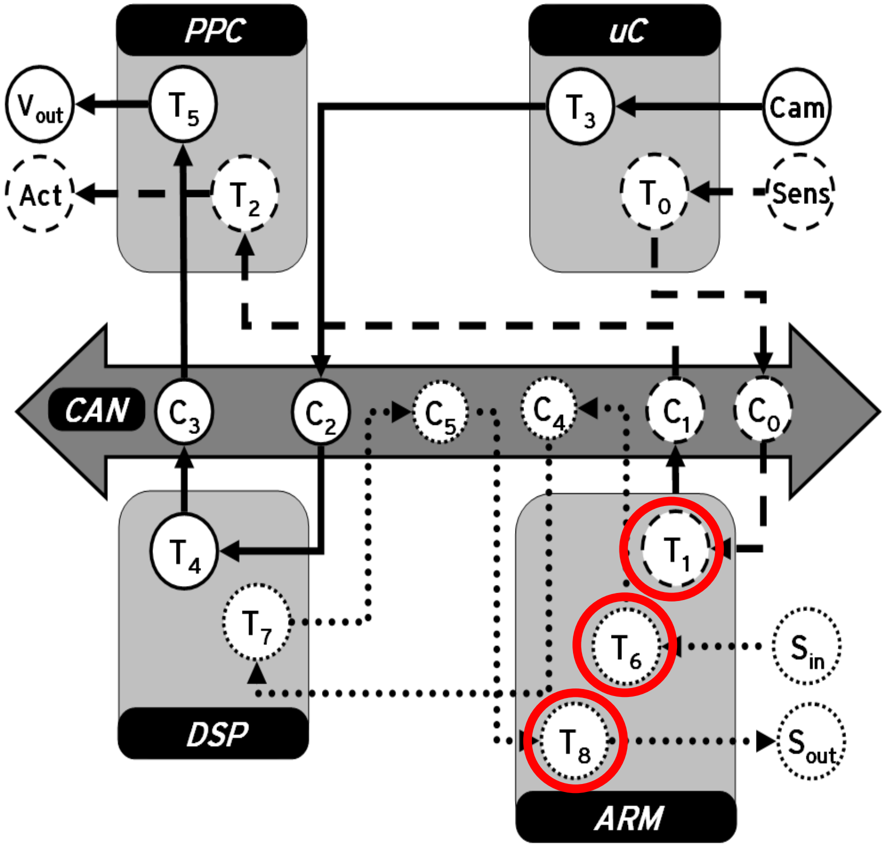
3D - Robustness Maximization (2)



Original configuration



Optimized configuration



Conclusion

- **Robustness to system property variations**
- **Scalable stochastic sensitivity analysis perfectly suited for robustness optimization**
- **Metrics expressing lower and upper system robustness bounds …**
- **… enable efficient integration of robustness criteria into design space exploration**