Safe Evolution in Real-Time Systems

INSTITUTE OF COMPUTER AND NETWORK ENGINEERING

TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG

Moritz Neukirchner, Steffen Stein, Harald Schrom, Rolf Ernst
{neukirchner|stein|schrom|ernst}@ida.ing.tu-bs.de

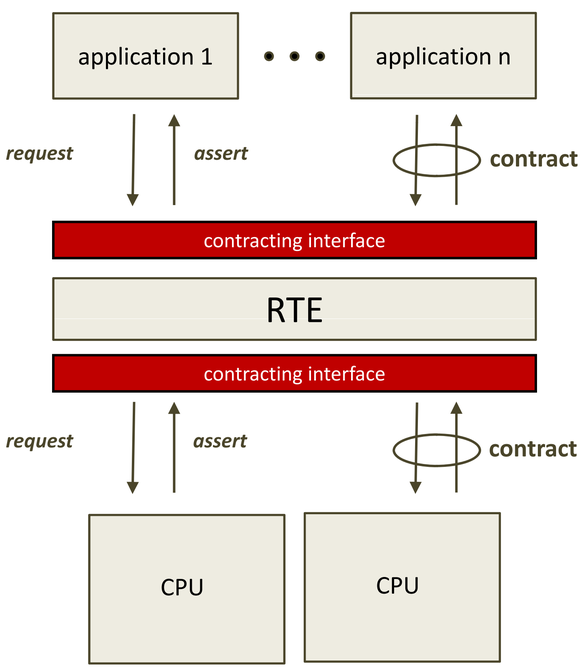
Challenges

- In-Field updates
  - Increasing share of invention (only) in software
  - Exploding configuration space
- Component/System Variability
  - Aging effects
  - Addition/Removal of System components
- Design Complexity
  - More functionality
  - Diverse system variants
  - Third-party integration
  - Increasingly networked systems

- Complex changing timing behaviour
- Timing properties specific to individual systems

Approach – Online Performance Contracting

Contracts for Specification and Guarantee



- A distributed System is extended by a sophisticated distributed RTE with formal performance analysis capabilities

Software Contracting:

- Applications specify their own timing behavior and request resources and timing guarantees from the RTE (provide a model)
- The RTE analyzes feasibility of the requesting application and asserts that all constraints are met
- Model, requirements and assertions form a contract

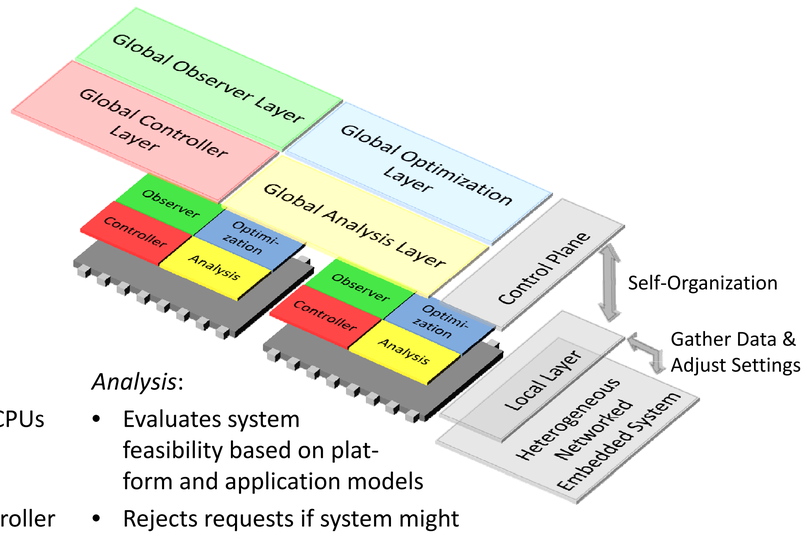
Platform Contracting:

- The RTE specifies a set of configuration parameters of a CPU
- The CPU guarantees a certain performance level for the given set of parameters
- Specification and assertions form a contract

Distributed self-organizing RTE

Distributed RTE:

- Local RTEs self-organize to form Distributed RTE
- Local RTEs are composed of Observer, Controller, Analysis and Optimization components
- Locally gather data and adjust settings



Observer:

- Supervises adherence of CPUs and applications to their respective contracts
- Reports violations to Controller component

Controller:

- Sets execution and configu-ration parameters (e.g. priorities, processor speed)
- Reacts on contract violations to isolate misbehavior from the system

Analysis:

- Evaluates system feasibility based on plat-form and application models
- Rejects requests if system might become infeasible

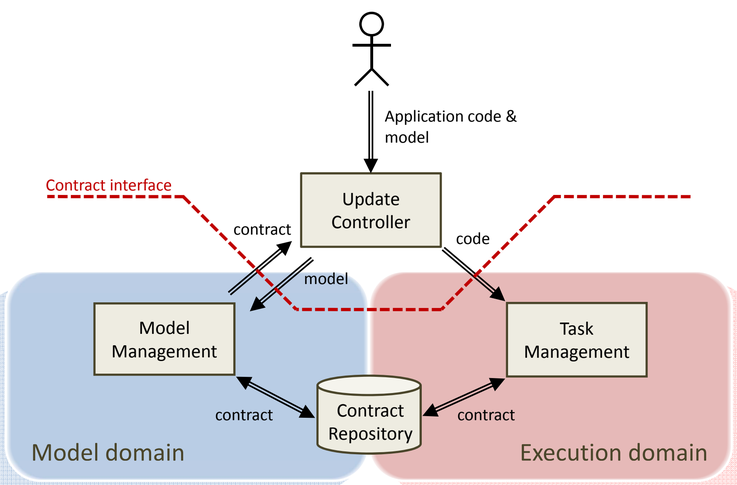
Optimization:

- Alters system models to make a configuration feasible/optimize its performance (e.g. w.r.t. timing, robustness)
- Employs Analysis component for evaluation of modified configuration

System Architecture

Contracting Procedure:

- An Update Controller, which may reside at an arbitrary position in the System, distributes the application model to all CPUs that are affected by the request
- Model Management evaluates feasibility of the request based on its model and existing contracts
- In case of acceptance the contract is closed and stored in the Contract Repository
- The Update Manager distributes the program code to the corresponding CPUs
- Task Management schedules the code for execution based on their contracts in the repository

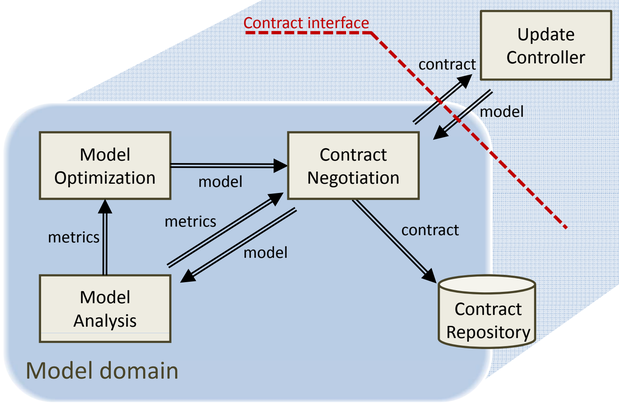


Separation of domains:

- System architecture is separated into two domains: *Model domain* & *Execution domain*
- Model domain performs worst-case performance analysis based on formal methods to ensure system feasibility at all times
- Execution domain enforces parameter settings based on contract information and detects deviation from specified behavior

Advantages:

- Decoupling of analysis and execution → analysis has minimal influence on application execution
- System optimization and analysis can be performed during processor idle times



Model domain:

Contract Negotiation:

- Protocol Handler for communication with Update Controller
- Inserts model into Analysis component and evaluates results

Model Analysis:

- Performs model based distributed performance analysis

Model Optimization:

- Modifies system model based on analysis metrics to optimize configurations
- Multi-objective otimization

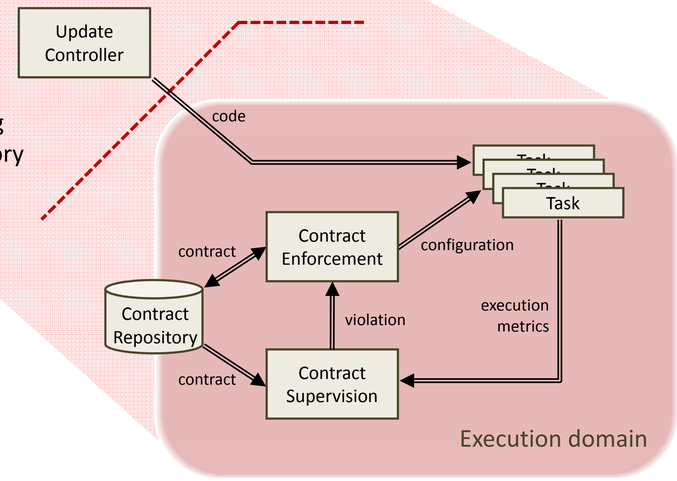
Execution domain:

Contract Enforcement:

- Sets execution parameters according to contracts in the Contract Repository
- In case of contract violations alters system configuration to isolate misbehavior and protect remaining applications
- if applicable alters contract information for new analysis

Contract Supervision:

- detects anomalous application behaviour
- monitors hardware performance



supported by DFG SPP 1183 Organic Computing

Safe Evolution in Real-Time Systems

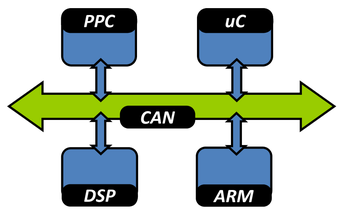
INSTITUTE OF COMPUTER AND NETWORK ENGINEERING

TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG

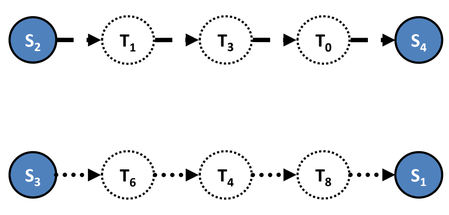
Moritz Neukirchner, Steffen Stein, Harald Schrom, Rolf Ernst
{neukirchner|stein|schrom|ernst}@ida.ing.tu-bs.de

System Modeling in SymTA/S

Platform

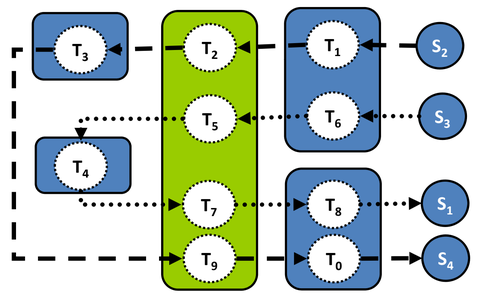


Applications

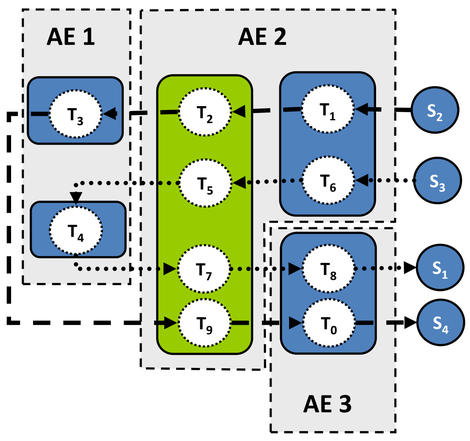


- Platform Components are modeled as Resources
  - Represented by Boxes in the image
- Applications are modeled as task graphs
- Task Graphs are mapped onto resources by assigning tasks to resources
- If two tasks communicate over a shared medium (Bus), a task for the bus representing the communication process needs to be inserted

Resulting Model



Distributed Performance Analysis

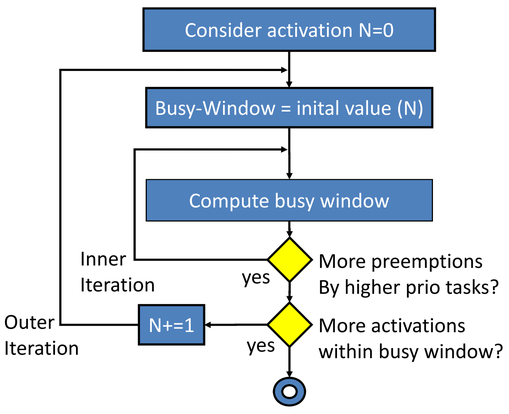


- Performance Model well known from Compositional Performance Analysis
- Resources can be analyzed independently
- Distribute Resources and tasks mapped on them over distributed Performance Analysis Engines
- Each Engine analyses its own task set, one of the Engines takes over the analysis of the bus
- Communication via Event Streams
  - Tunneled from model to model
- Event Model exchange only necessary between communicating processors, thus communication infrastructure must exist
- Scales and Evolves seamlessly with the system

- Major challenge: Performance Analysis in Bounded Time
- Bulk of computation time spent in local analysis runs (WCRT Analysis)
- Common Approach: Busy Window Algorithms
  - Static Priority schemes
  - Round Robin
- Formulated as fixed point problem: Find smallest solution to

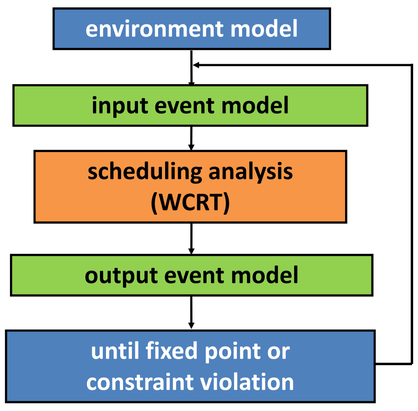
$$B_i(w) = w * C_i + \sum_{j \epsilon hp(i)} \eta_j^+(B_i(w)) * C_j$$

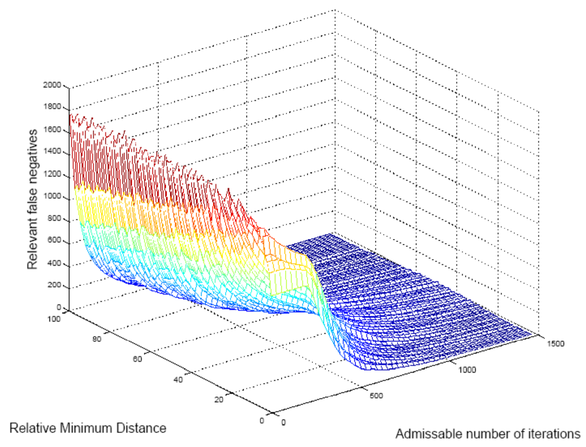
- Solution found using iterative approach (right)



- Runtime unbounded in Theory (NP Hard)
- Safe bound for run time
  - Limit number of inner iterations
  - Discard systems not analyzable in time as infeasible
- Produces false negatives
- Number of false negatives mainly dependent on number of admissible iterations
- Secondary indicator: Minimum inter arrival time between events
- For sufficiently large number of iterations, bounded Algorithm is not inferior to normal one

- Local Analysis Strategy (right) cannot be used
- Distributed strategy:
  - Monitor event stream changes at input ports of tasks
  - Analyze each Resource as soon as new data is available
- Does not require central coordination
- Termination (Convergence) Detection required
  - Approaches from Distributed Computing exist
  - "Quiescence Detection"





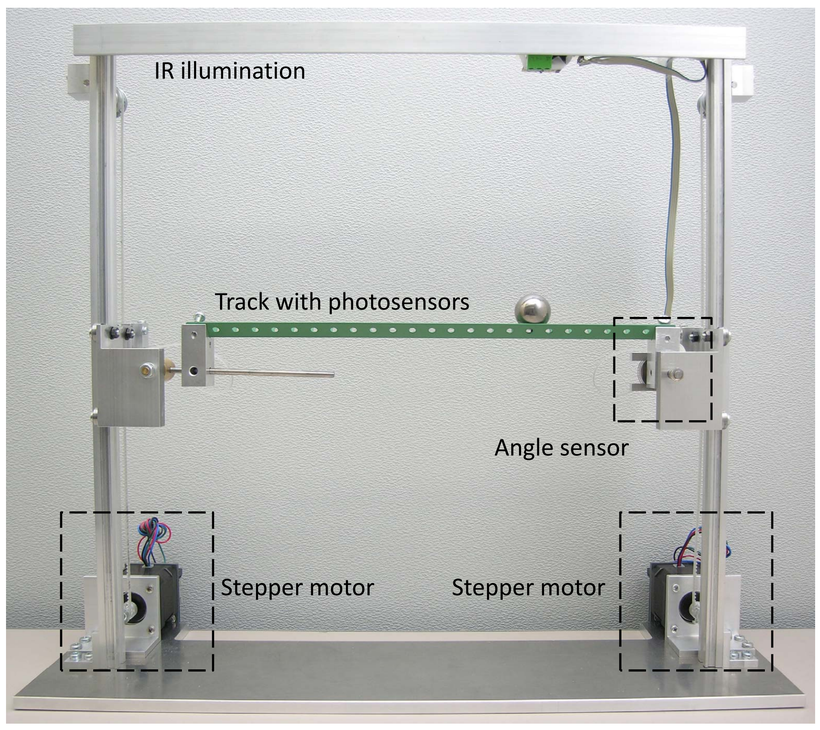
Demonstrator Use Case

Demonstrator mechanical setup:

- Green track is mounted moveable at both ends to the surrounding frame
- Both ends can be moved independently by separate stepper motors
- A ball is positioned on the rail and is to be kept at the center at all times
- The slope of the rail is measured by an incremental angle sensor
- The position of the ball on the rail is detected by an array of IR-photosensors

Demonstrator software and usage:

- Distributed RTE with online performance analysis protects a timing critical control loop
- One end of the green rail is moved by a user-defined disturbance
- Second end of the green rail is positioned by a control loop
- Depending on the frequency of the control loop it is capable to compensate the disturbance up to a specific frequency



Objectives of demonstrator:

First approach:

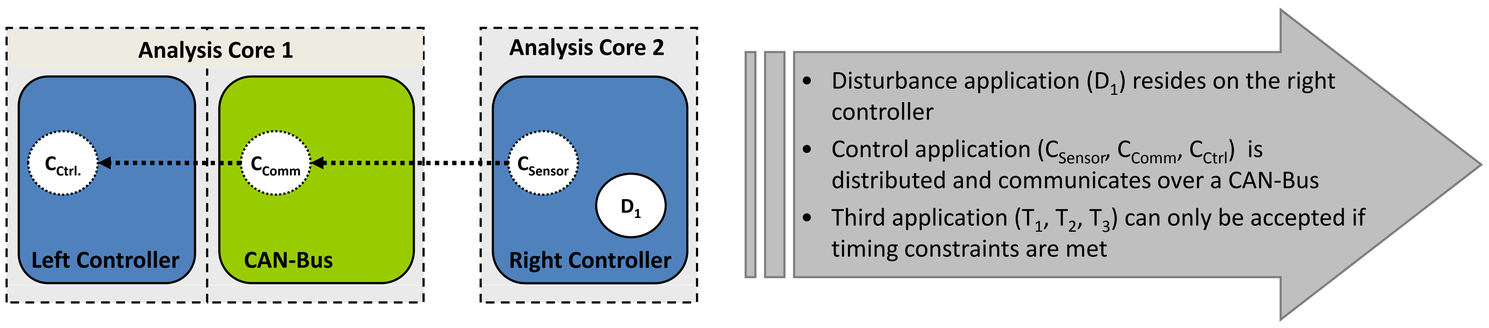
- Depending on the system utilization a third application can cause failure of the control loop by violation of timing constraints. This becomes visible as the ball cannot be kept at the center position
- This violation can be avoided by the RTE using the performance analysis to decide if the third application should be rejected to guarantee correct function of the control loop

Second approach:

- The period of the control loop can be shortened to enhance the maximum frequency that can be compensated by the control loop
- Depending on the period of the control loop and thus system utilization a third application can be accepted or is rejected

Third approach:

- In case of rejection of the third application an optimizer tries to find a feasible configuration



supported by DFG SPP 1183 Organic Computing