Assessing the realizability of Real-Time System Requirements

Werner Damm
Joint work with
H. Dierckx, A. Metzner, I. Stierand
OFFIS

Problem statement

- Given
  - a System Level Design S
    - E.g. a UML / Matlab-Simulink model expressed in the Speeds MetaModel
  - a Partitioning of S as sets of task-chains T
    - E.g. runnables in the Autosar jargon
  - a set of real-time requirements R (subsuming end-to-end latencies for all task chains in T)
  - Assumptions on arrival rates
  - Constraints on the target architecture
    - Type of bus systems
    - Type of ECUs
- Determine realizability of S under stated assumptions and constraints
- If realizable:
  - Find minimal number of ECUs
  - Find mapping which for minimal number of ECU optimizes slackness of solution

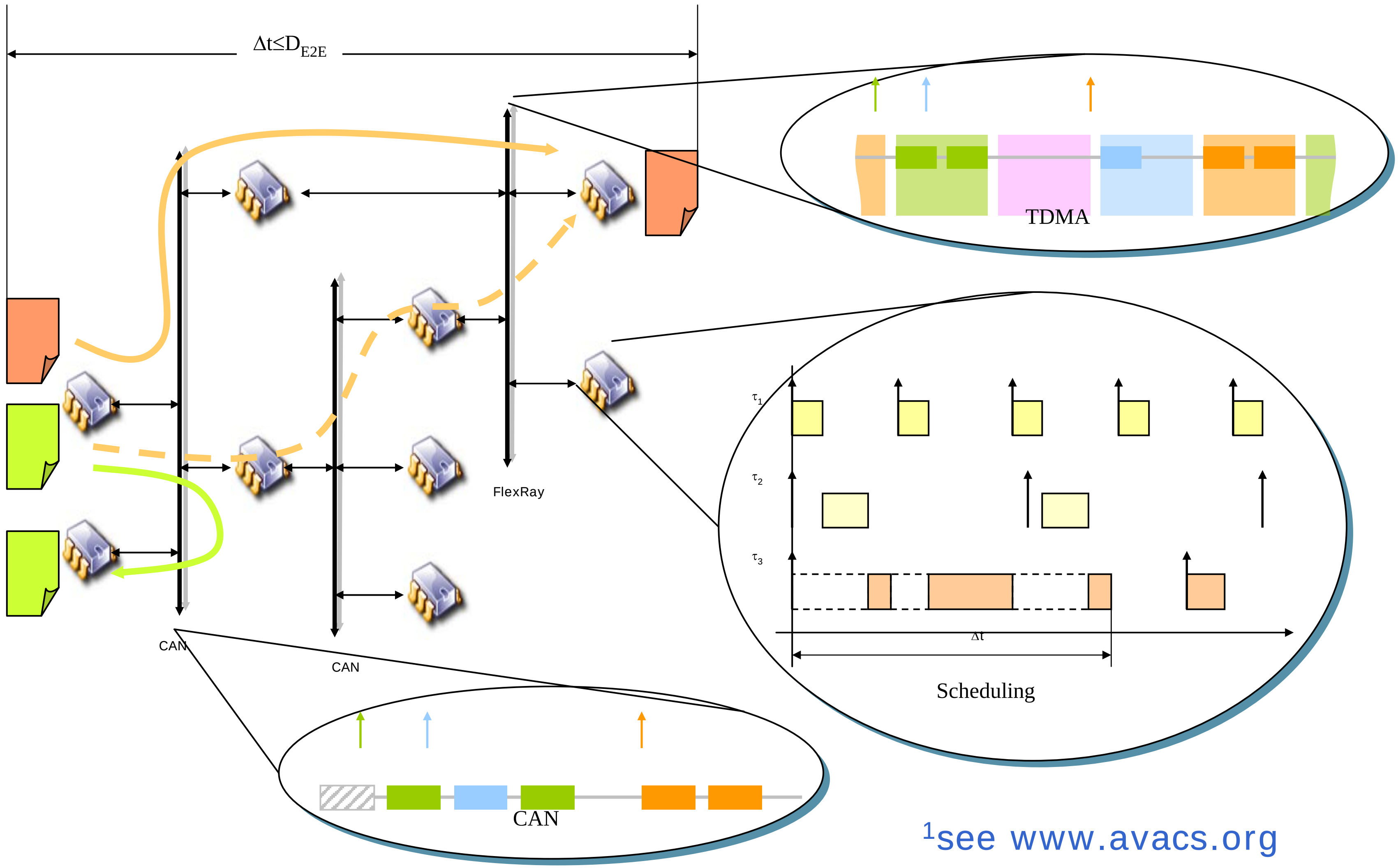
Capturing Requirements

- Using Timed LSCs
- Have formal semantics as Timed Büchi Automata
- Efficient techniques for verification of Timed LSCs against timed automata models available
  - Klose et al, CAV 2006
- Can in particular express end-to-end latencies for task chains
- Can express sub-budgeting such as local deadlines for intermediate communications in task chains
- Can express communication protocols such as e.g. inherited from Virtual Function Bus

Timed Automata only?

- Timed Automata based Scheduling Analysis is an active topic of research
  - Fixed Priority Scheduling, Single Processor Systems [FMPY03]
  - Fixed Priority Scheduling, Distributed Systems [HV06][MAS06]
- Would be a natural match for proving LSC requirements, but
  - Requires models of schedulers for tasks and communications
- Key issue is scalability

Previous Work in AVACS[1] (Metzner et al)
Scheduling distributed real-time systems



[1]see www.avacs.org

Using SAT Based Methods for Schedulability Analysis

- Task system:
  - Periodic, with deadlines, resource consumption and messages
- Architecture:
  - Set of ECUs connected in various topologies using different kinds of bus systems (CAN, Flexray, TTP, Token Ring)
- Goal:
  - Assign tasks to ECUs and assign priorities such that assignment is feasible and optimal w.r.t. given objective functions
- Reduced to SAT problem
  - Enhanced with dedicated heuristics

Results

- Specially tailored MILP approach [RTCSA'06]

	Feasible		Optimal	
	sec	nodes	sec	nodes
input00	2375	11860	> 10000	> 8000
input01	120	823	> 10000	> 20000
input02	> 10000	> 7000	> 10000	> 7000
input03	480	3199	> 10000	> 25000
input04	230	370	> 10000	> 7000
input05	595	2683	> 10000	> 7000
input06	> 10000	> 5000	> 10000	> 5000

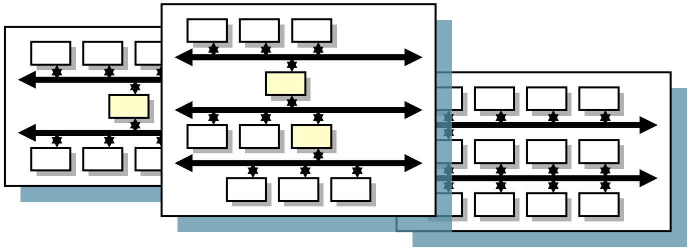
CAN, without improvements

- Incorporation of
  - Efficient primal heuristics
  - On-demand generation of cutting planes

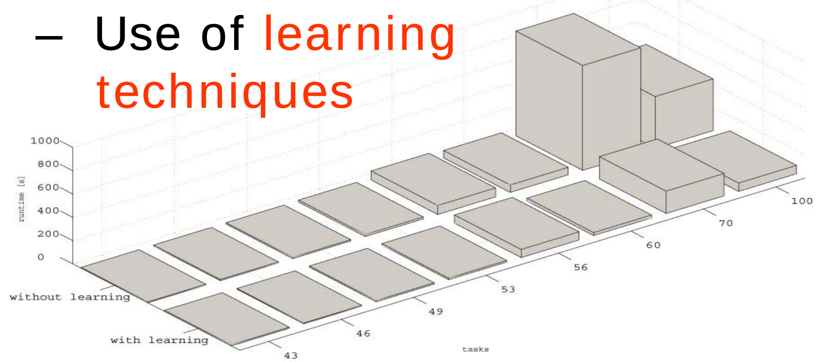
	Feasible		Optimal	
	sec	nodes	sec	nodes
input00	65	50	543	1478
input01	2	11	7553	71697
input02	289	2897	552	3675
input03	5	39	2011	16460
input04	22	14	> 10000	> 7500
input05	30	27	> 10000	> 7000
input06	30	25	> 10000	> 7300

CAN, with improvements

- Use SAT checking for scheduling analysis and optimization [RTCSA'05]
- Encoding of complex distributed architectures [WPDRTS'06]



- Specially tailored SAT approach [RTSS'06]
  - Satisfiability modulo scheduling theory
  - Use of learning techniques

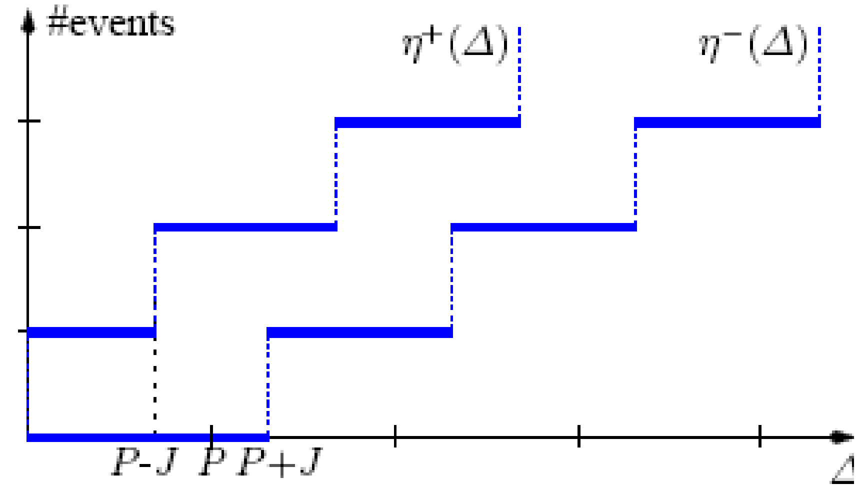


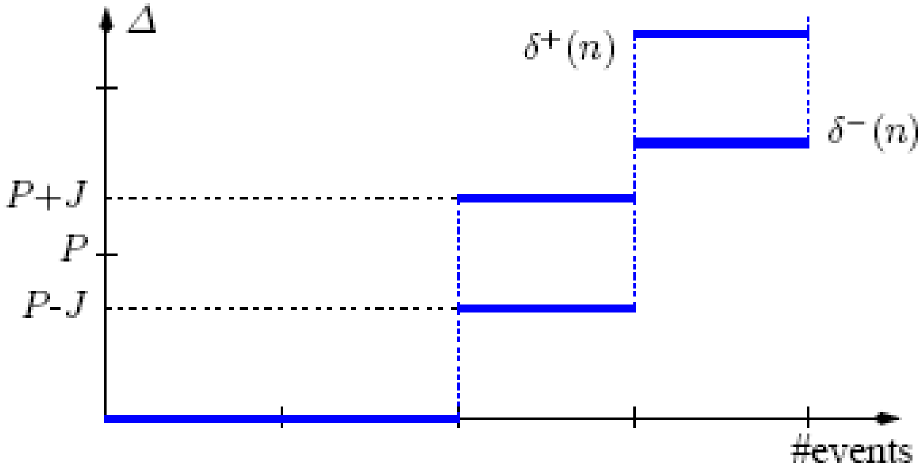
This talk

- Lets get the best of both worlds
  - Let SAT based methods decide realizability and find optimal solution
    - Minimal number of ECUs, optimized for slackness
  - Lift results (e.g. computed response times for tasks and communication) to timed automata level
  - Verify LSC-based Requirements against lifted results
- Leads to drastic reductions of verification complexity
- Superior to timed automata based approach

Tasks as timed automata I: the task model

- We use the task model from Ernst et al [ERJ04]
  - Based on event streams
  - Event Model given by
    - upper and lower bounds on number of arrived events **η(Δ)** within Δ seconds (Thiele et al)
    - upper and lower bounds δ**(n)** on separation time between **n** events
  - Supports compositional analysis





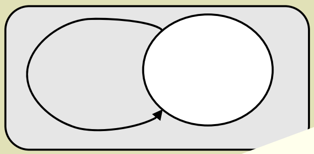
Event Model	$\eta^+(\Delta)$	$\eta^-(\Delta)$	$\delta^+(n)$	$\delta^-(n)$
Periodical	$\lceil \frac{\Delta}{P} \rceil$	$\lfloor \frac{\Delta}{P} \rfloor$	$(n-1)P$	$(n-1)P$
Sporadical	$\lceil \frac{\Delta}{P} \rceil$	0	$(n-1)P$	∞
P. + Jitter	$\lceil \frac{\Delta+J}{P} \rceil$	$\lfloor \frac{\Delta-J}{P} \rfloor$	$(n-1)P+J$	$(n-1)P-J$

Tasks as Timed Automata II

Trigger/Sink

- Models interface
- Encapsulation of env.

$$S/T\left(P_e^{min}, P_e^{max}, J_e, O_e\right)$$

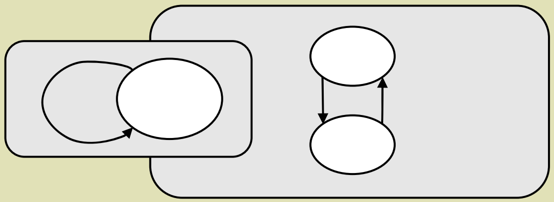


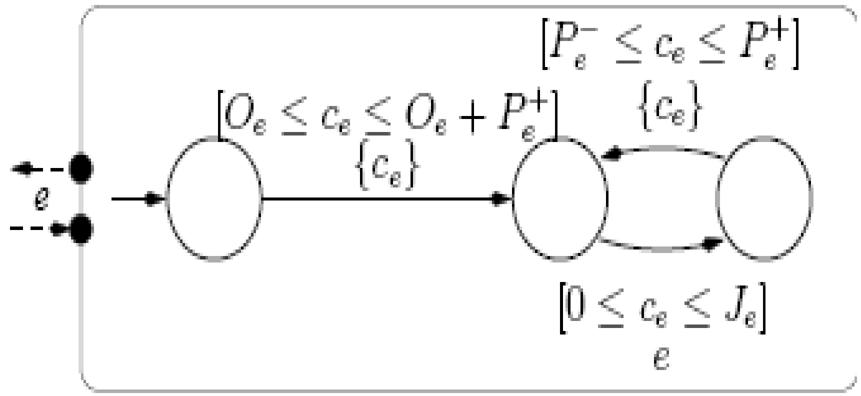
Cyclic Timed Automata

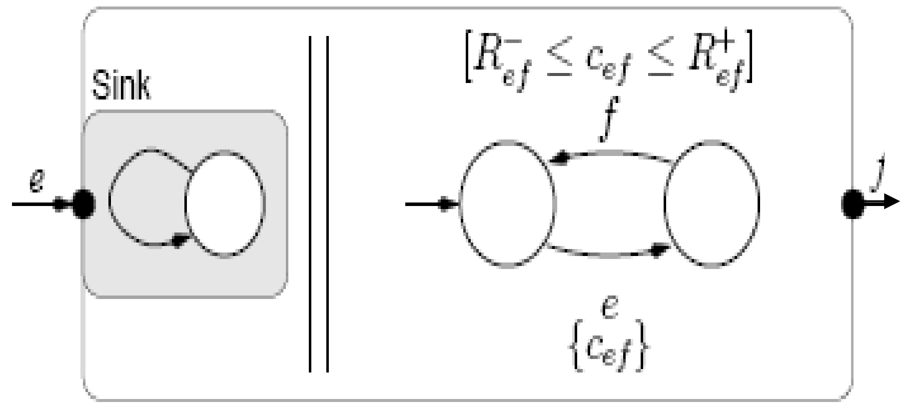
Execute

- I... for tasks
- Used for messages

$$E\left(P_e^{min}, P_e^{max}, J_e, O_e, R_{ef}^{min}, R_{ef}^{max}\right)$$







Tasks as timed automata III: characterization of task model as timed automata

- CTIs are parameterised classes of timed automata, where parameters are determined by parameters of schedulability analysis (P, J, R^{min}, R^{max})
- Executers are equivalent to event stream transformers for arbitrary R^{min}, R^{max}
- Inexpensive interface compatibility check

$$L\left(S\left(P_{1,}^{min} P_{1,}^{max}, J_1\right)\right) \subseteq L\left(T\left(P_{2,}^{min} P_{2,}^{max} J_2\right)\right) \Leftrightarrow \left(P_1^{min} \geq P_2^{min}\right) \wedge \left(P_1^{max} \leq P_2^{max}\right) \wedge \left(J_1 \leq J_2\right)$$

Thm

Composition of TA representation of task network T and task network accept same set of timed language over events

Overall Methodology

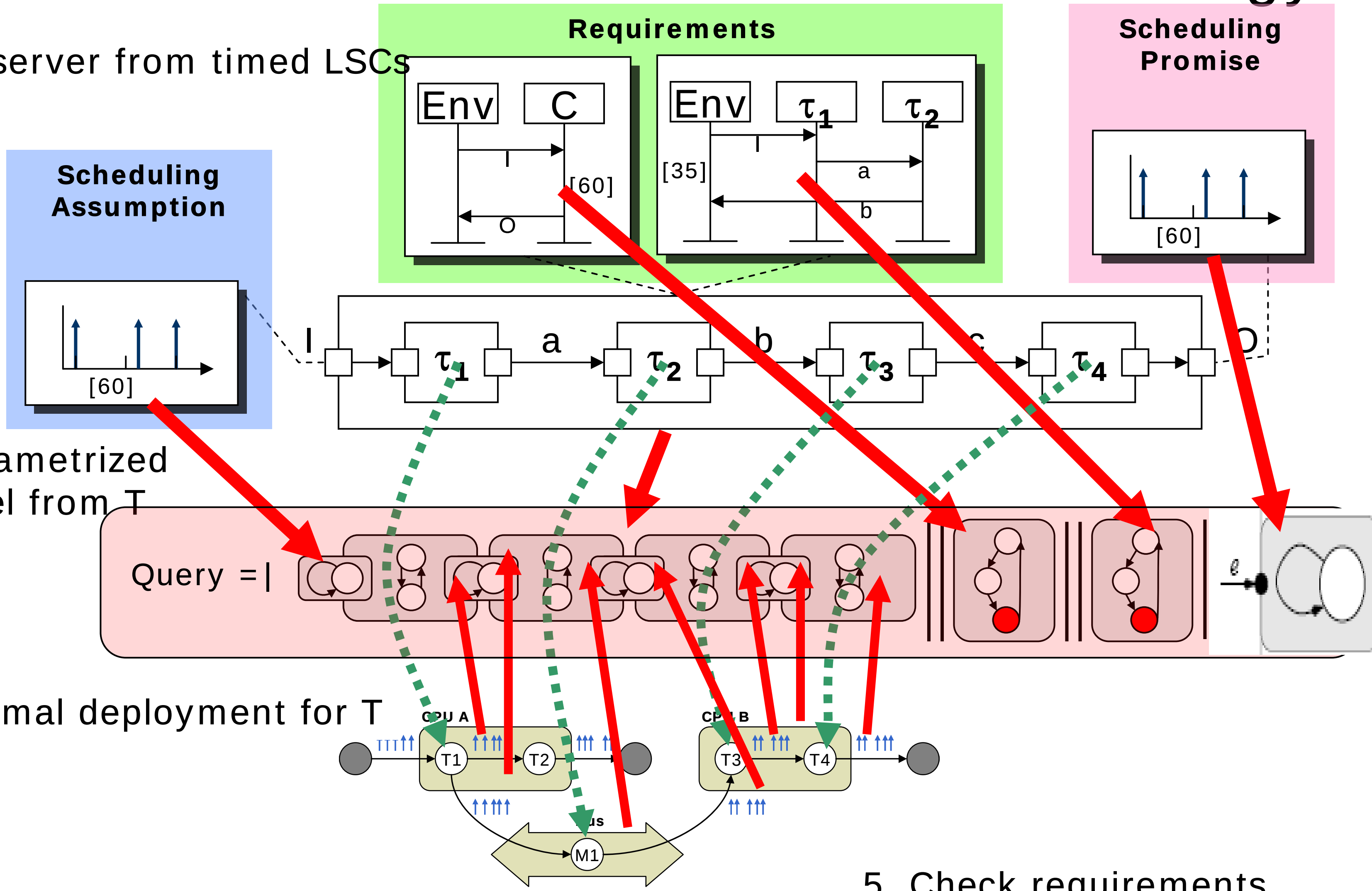
1. Build observer from timed LSCs

2. Build parametrized CTI model from T

3. Find optimal deployment for T

4. Analysis/propagation of CTI parameters

5. Check requirements (by reachability analyis of forbidden states)



Evaluation results

- Compared against *TIMES* [FMPY03]
  - 12 Tasks deployed on one ECU (TIMES Tool does not support distributed architectures), fixed priority scheduling
  - 4 task task chains, for each d_{e2e}=period
  - Requirements relate to intermediate deadlines
- Results:
  - *TIMES* Tool reported feasibility and response times
  - Schedulability test reported feasibility and response times and jitter
  - Both in few seconds
  - Requirement checking for CTIs with UPPAAL takes a few seconds

Task	WCET	Period	Priority	Response Time	Release Jitter
T1	13	400	6	37	0
T11	2		9	2	15
T12	3		8	3	15

Experiment	#Tasks	TIMES	RTSAT	Req.Check
D=P	12	30s	<20s	<1s
D<P	12	30s	<20s	<1s
D=P	13	>2h	<20s	<1s
D<P	13	>2h	<20s	<1s
D=P	14	>2h	<20s	<1s
D<P	14	>2h	<20s	<1s
D=P	15	>2h	<20s	<1s
D<P	15	>2h	<20s	<1s

- Adding one additional task leads to time-out of *TIMES*
- CTI based approach is unaffected, scalable

Conclusion

- Overall Methodology for assessing realizability of key system timing characteristics
  - Employing synthesis of potential architectures under constraints
  - Employing a SAT solver enhanced with a dedicated response time analysis engine: RT-SAT
  - Employing a semantic embedding of synthesized response times into “cyclic” timed automata
  - Using these for LSC based system requirement verification
  - Outperforms timed automata based approaches
- Further evaluation with larger benchmarks ongoing

Related Work

Thiele, Chakraborty, Naedele [TCN00],Wandler,Thiele [WT05]

Real-Time Calculus

- Demand Bound Functions (Network Calculus)
- Scheduling Analysis based on Functional Calculus

Henzinger, Matic [HM06]

Interface Algebra

- Compositional Algebra based on Demand Bound Functions
- Task Dependencies, Complex Components

Ernst, Richter, Jersak [ERJ04]

Scheduling Analysis based on Event Stream Functions

- Additional Bounding Functions (Event Distances)
- Complex Task Networks

Related Work

Fersman, Mokrushin, Pettersson, Yi [FMPY03]

Scheduling based on Timed Automata

- Extended Timed Automata (Task Automata)
- Single Processors
- TIMES Tool

Hendriks, Verhoef [HV06], Madl, Abdelwahed, Schmidt [MAS06]

Scheduling based on Timed Automata

- Distributed Systems

Altisen, Goessler, Sifakis [AGS02]

Scheduler Synthesis

- Verification based on Timed Systems