Scheduling Transient Overload with the TAFT Scheduler

M. Gergeleit, E. Nett

Institute for Distributed Systems (IVS)
Otto-von-Guericke Universität Magdeburg

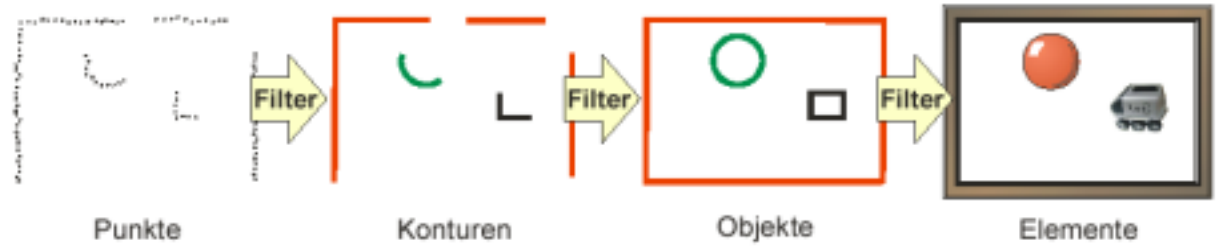
{gergeleit, nett}@ivs.cs.uni-magdeburg.de

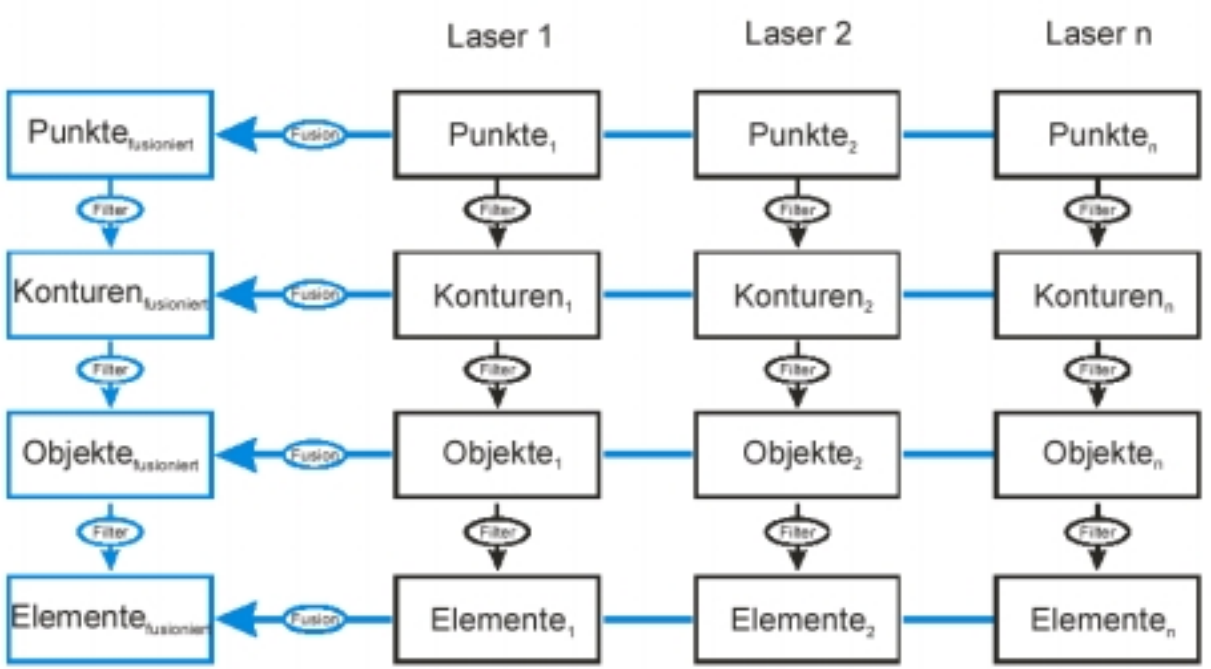
Overview

- Motivation
- Concepts of the TAFT-Scheduler
- Scheduling-Policy for TAFT
- Implementation on RTLinux
- Evaluation
- Summary

Application: Distributed Sensor Fusion (1)

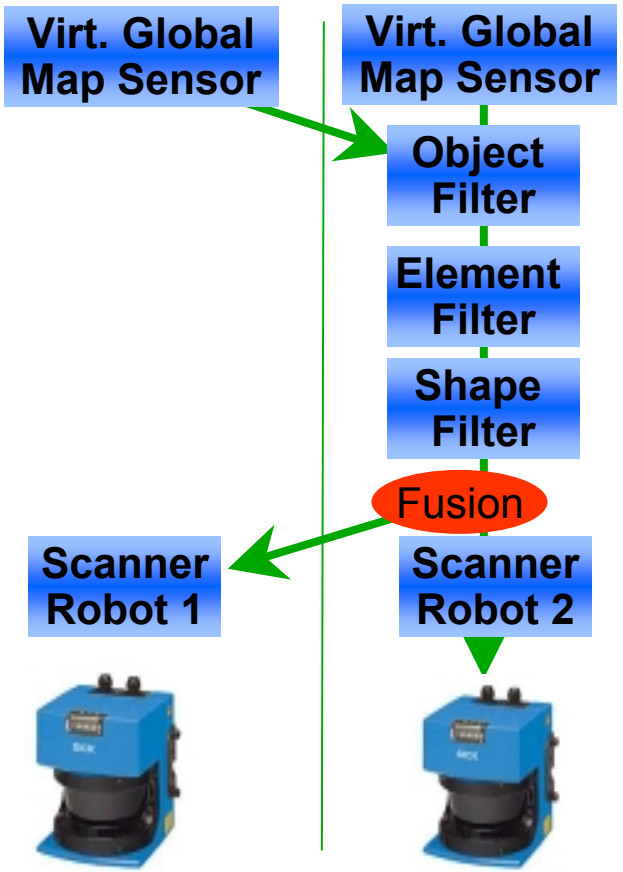
- In a team of RoboCup robots
- Fusion of Laser-Scanner Data
- Fusion at different Levels





Application: Distributed Sensor Fusion (2)

- **Problems:**
  - Execution time of each filter module depends on unpredictable complexity of the scene
  - WCET much too large but actually never reached
  - Variable other workload on the robots (overload)
  - Global deadline for the results of the fusion process
  - Load Distribution



Approach

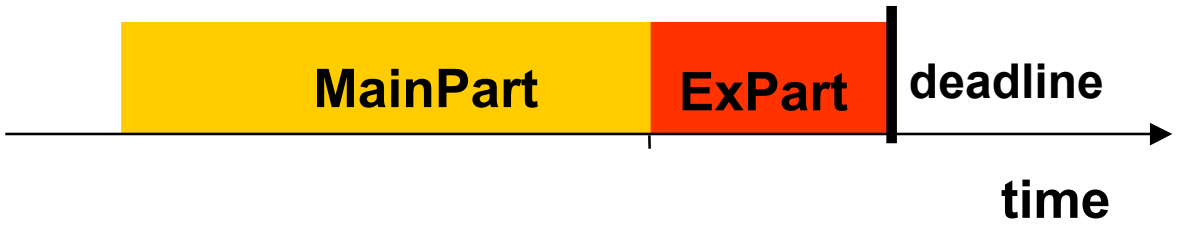
- **If possible use of Anytime-Algorithms**
- **Local encapsulation of the unpredictability**
  - By the use of adequate scheduling scheduling

Idea of TAFT-Scheduling

- **Handling of tasks with unknown or too pessimistic WCETs**
  - Introduction of Expected Case Execution Time (ECET)
- **Still with timing guarantees**
  - Scheduled exception handling **before** the deadline
- **Fault-tolerance with respect to timing errors**
  - Graceful degradation in overload situations
  - Tradeoff between functionality and timing

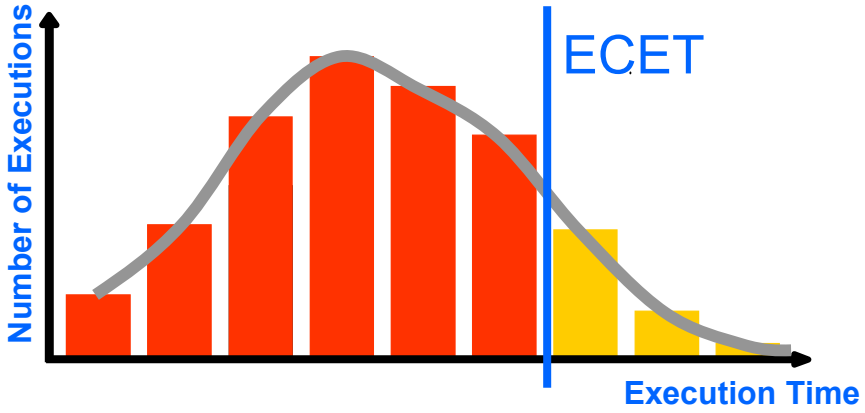
TAFT Scheduling

- **Each Task consists of a MainPart and an ExceptionPart**
- **MainPart**
  - real-time computation with expected execution times (ECET)
  - best effort approach, from no guarantee to total guarantee
- **ExceptionPart**
  - provides exception handling before the deadline
    - prevention of domino effect, fail-safe and consistent behavior
  - guaranteed execution by reserving its WCET



ECET – Expected Case Execution Time

- **$ECET_{t,p}$ of task-instance *t* of task τ with probability *p***
  - CPU-time required to complete task-instance *t* with probability *p*
  - *p*-quantile of the probabilistic density function of *T*'s execution time
- **$ECET_{t,k,n}$ - The minimal execution time that was needed to successfully complete at least *k* out of the last *n* most recent executions of τ before *t*.**
  - A statistic quantity



TaskPair Programming

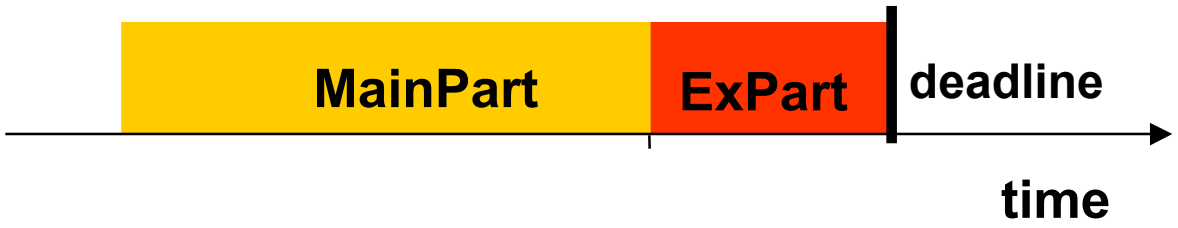
```
void Procedure_TP (struct taskpair TP){
        if(guarantee (TP.T, TP.D, TP.C, TP.E)){
        while (1) {
                pthread_wait_np_ex();
                TP_SAFE_REGION

                BEGIN_MAIN_PART
                        MP();
                END_MAIN_PART

                BEGIN_EXCEPTION_PART
                        EP();
                END_EXCEPTION_PART
                }
        }
```

Scheduling Strategy

- **Two-level Scheduling**
- **Level One – ExceptionParts**
  - Highest dispatching priorities
  - LRT (Latest Release Time - Reverse-EDF)
  - Tries to do everything as late as possible
- **Level Two - MainParts**
  - Lower dispatching priorities
  - EDF
  - Tries to do everything as soon as possible



Acceptance Test

- **Harmonic task set:**
  - A task set of TPs
    $\Pi = \{\tau_i = (T_i, D_i, C_i, E_i), i = 1 \text{ to } n\}$
    ordered by crescent deadlines is schedulable with the TAFT scheduler
    (level 1: EPs with LRT, level 2: MPs with EDF)
    if the Maximum Utilization Factor $\Omega_i \leq 1$ for each τ_i.

$$\Omega_i = \sum_{j=1}^{i} \frac{C_j + E_j}{T_j} + \frac{1}{T_i} \sum_{i<j\leq n} E_j$$

  - Similar condition to EDF or LRT

Implementation (1)

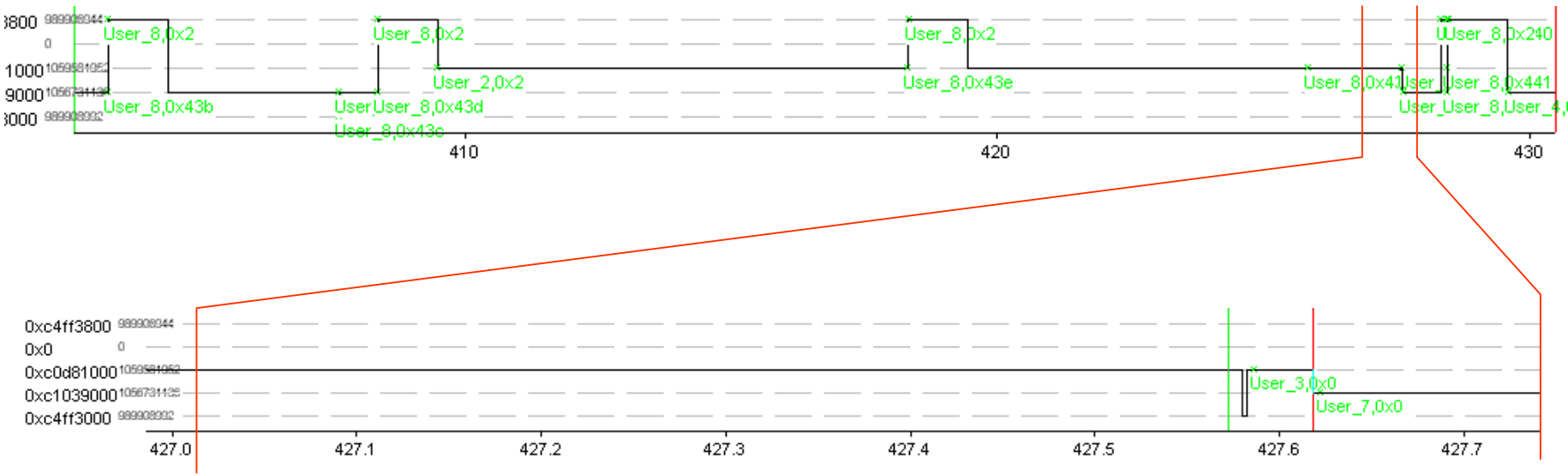
- **Implemented in RTLinux V3.0**
- **Changes in the original scheduler:**
  - support the two-level scheduler
    - LRT for EPs and EDF for MPs
  - mechanism for triggering deadlines and handling exceptions
  - execution times account
  - task acceptance condition
- **Available for download at:**

http://mosel.cs.uni-magdeburg.de/taft/index.html

Implementation (2)

- **Monitored by a RTLinux-Version of MagicZoom**

EP Activation:



Experiments (1)

- **Conducted experiment: *Hartstone Benchmark***
  - 5 periodic tasks
  - Harmonic and non-harmonic series

	PH Series		PN Series	
Task ID	*Frequency (F)*	*Execution time (ET)*	*Frequency (F)*	*Execution time (ET)*
0	1 Hz	160,00 ms	2 Hz	80,00 ms
1	2 Hz	80,00 ms	3 Hz	53,28 ms
2	4 Hz	40,00 ms	5 Hz	32,00 ms
3	8 Hz	20,00 ms	7 Hz	22,85 ms
4	16 Hz	10,00 ms	11 Hz	14,54 ms

Experiments (2)

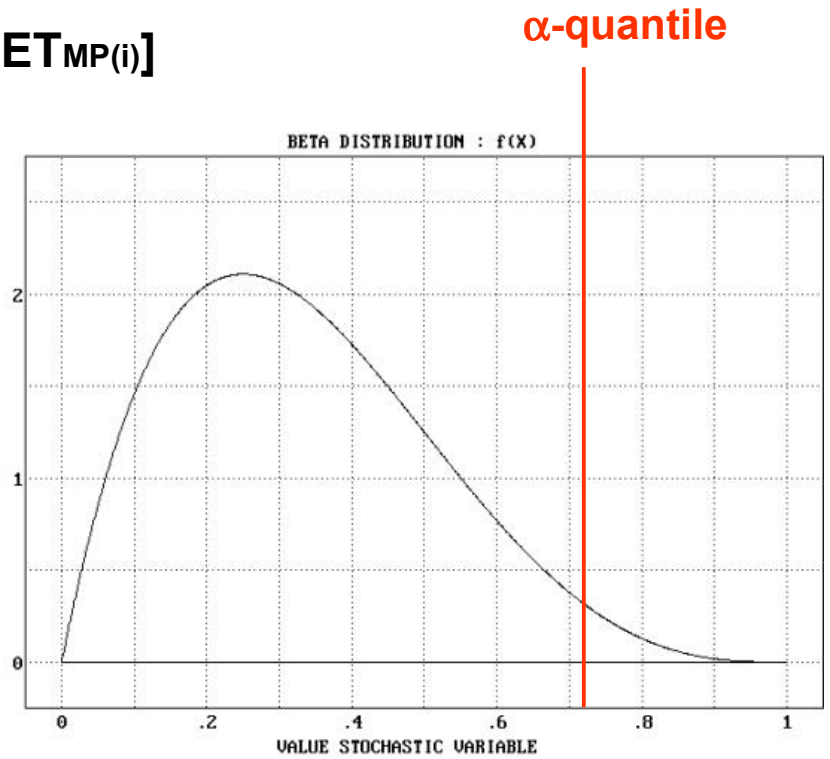
- **Used timing distributions**
  - Beta and Uniform PDF

$ET_{MP(i)}$ = beta(2, 3) [$WCET_{MP(i)}$ * 0.5, $WCET_{MP(i)}$] or

$ET'_{MP(i)}$ = uniform [$WCET_{MP(i)}$ * 0.5, $WCET_{MP(i)}$]

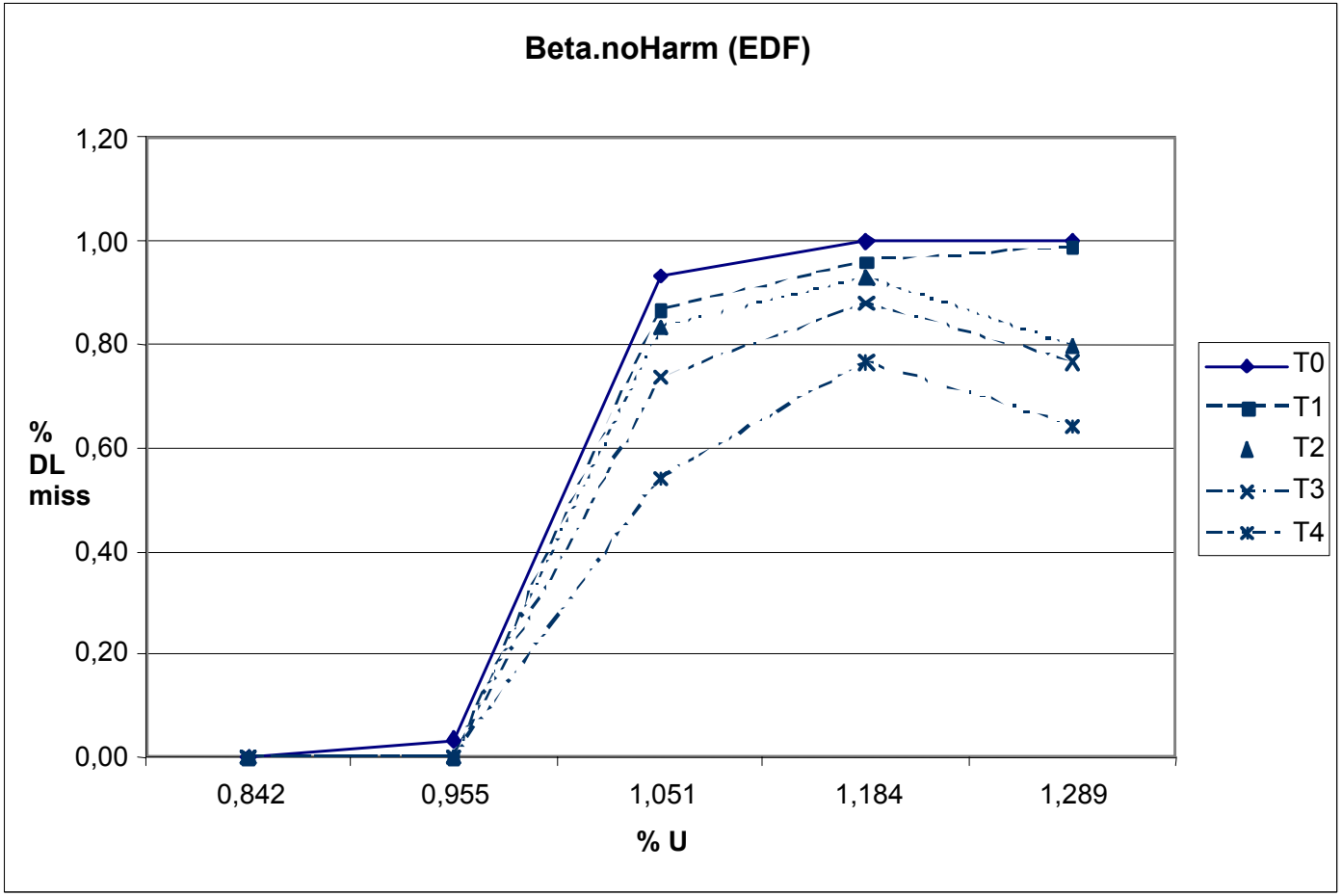
C_i = α-quantile of $ET_{MP(i)}$

E_i = $WCET_{MP(i)}$ * 0.05



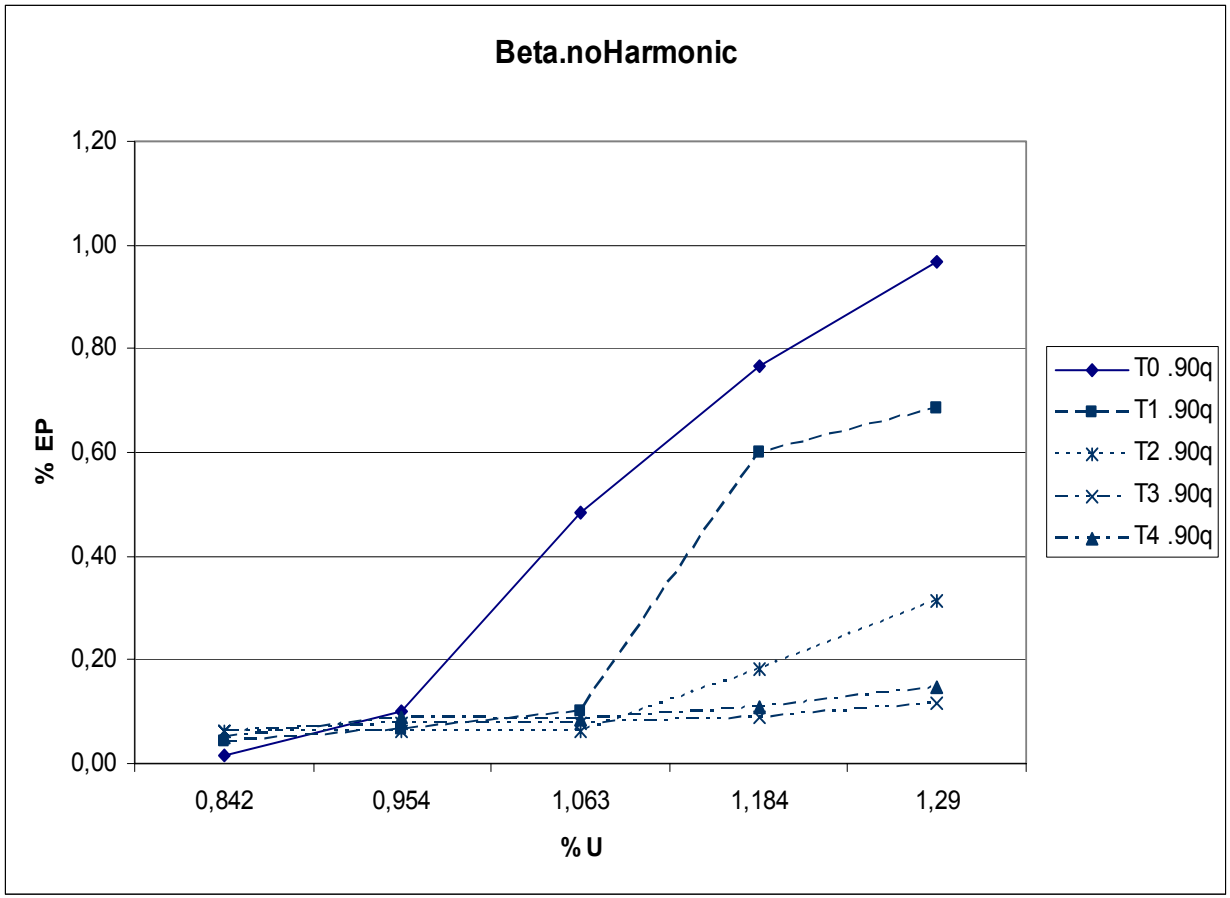
Results

- **EDF in a transient overload situation:**



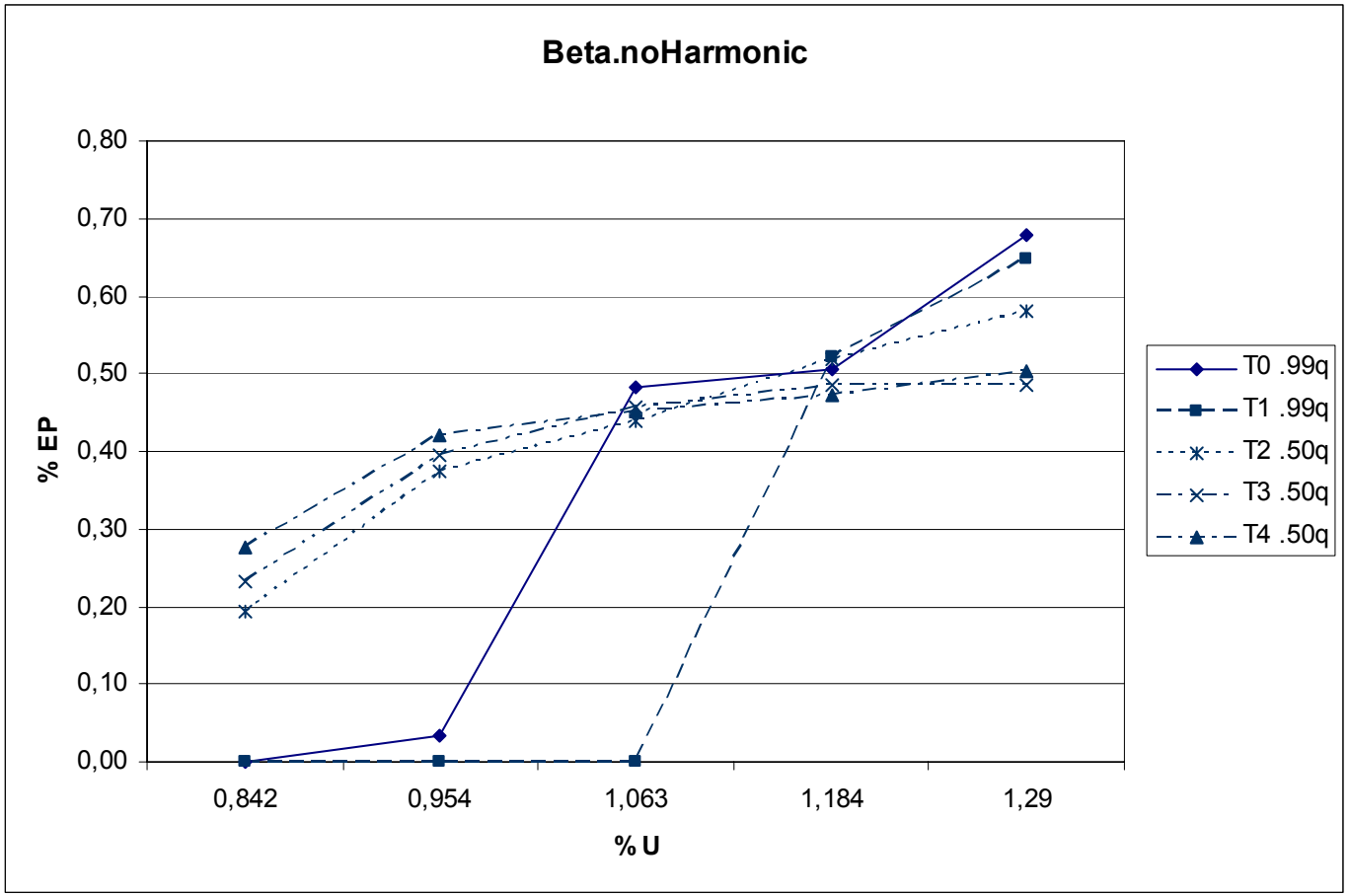
Results

- **TAFT in transient overload situation:**



Results

- **TAFT with different α-quantiles in transient overload situations:**



Summary

- **TAFT-Scheduling**
  - Can be used to handle unpredictability (incl. overload) in a fault-tolerant manner
  - Can be implemented using a combination of EDF-scheduling schemes
  - Has been implemented on RTLinux
  - Has been used to implement sensor-fusion of laser-scanners in a RoboCup scenario