Design of Real-Time Software

Analysis of hard real-time tasks under fixed-priority pre-emptive scheduling

Reinder J. Bril
Technische Universiteit Eindhoven
Department of Mathematics and Computer Science
System Architecture and Networking Group
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
r.j.bril@tue.nl, http://www.win.tue.nl/~rbril

January 2014

Overview

- Context
- Schedulability conditions
- Basic response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
- Concluding remarks
- References

Overview

- Context
  - Basic scheduling model (Recap)
  - Motivation for FPPS
- Schedulability conditions
- Basic response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
- Concluding remarks
- References

A basic scheduling model – recap

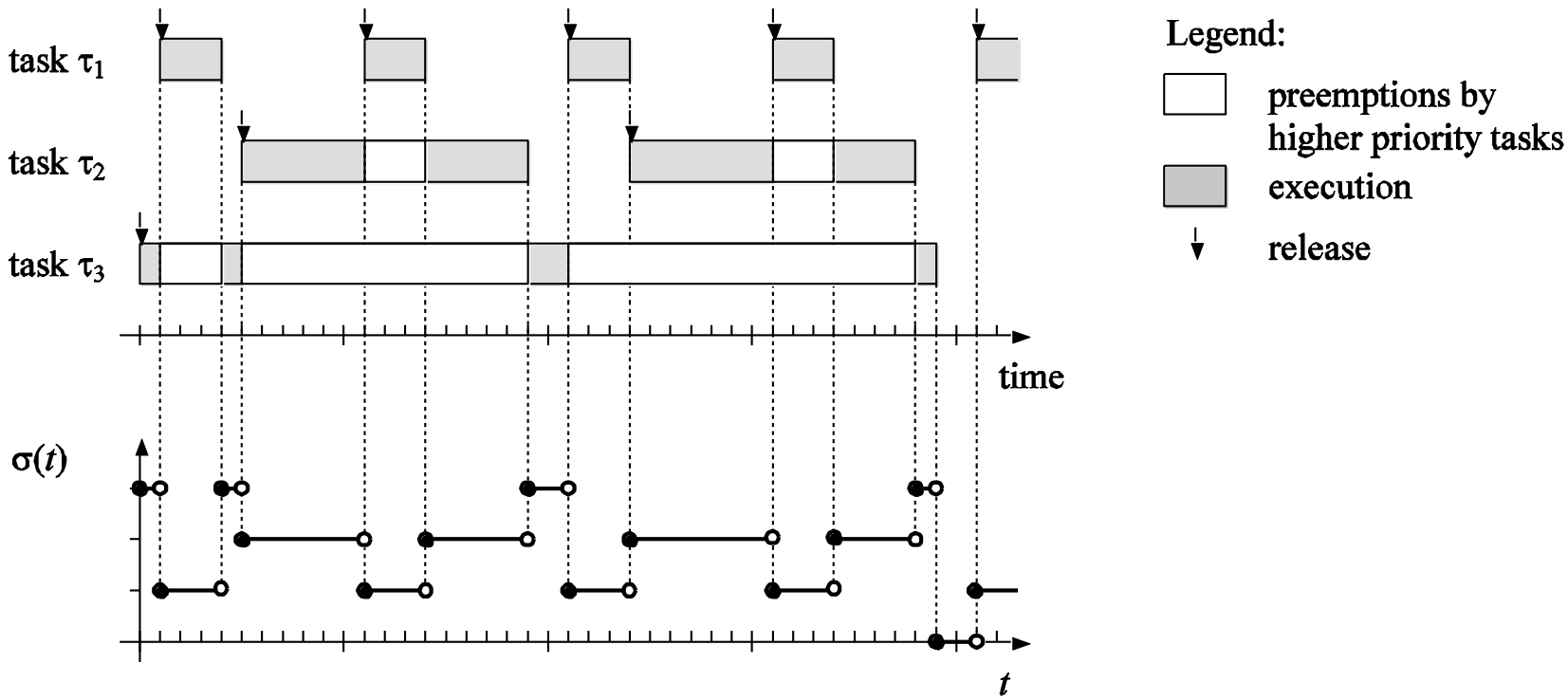
- Model (of a system in general):
  - Abstraction (of that system)
    - leaving out details irrelevant to a given set of criteria
    - preserving the properties of interest
- Scheduling model (for Real-Time SW)
  - explicitly addresses relevant issues in real-time systems
  - ...but must be mapped eventually onto an execution environment
    - OS, hardware, run-time system, ...

A basic scheduling model – recap

- Event:
  - indicates a state change requiring a *timely* response, i.e. neither too early nor too late;
  - *external* (e.g. at RT(C)S boundary), *internal* (e.g. a task triggering another), or *timed*.
- Task: actions in response of event.
- Processor: executes a single task at the time.
- Schedule:
  - assignment of tasks to processor;
  - set Γ on n tasks $\tau_1 \ldots \tau_n$;
  - $\sigma: \mathrm{R} \rightarrow \{0, 1, \ldots n\}$, where $\sigma(t) = 0$ means idle.

Schedule: example

Schedule $\sigma(t)$ of three independent periodic tasks τ_1, τ_2, τ_3, where τ_1 has highest priority and τ_3 lowest priority.



Initial basic assumptions

- Events: implicit
- Tasks:
  - released strictly periodically, elastically, or sporadically;
  - independent;
  - no self-suspension.
- Processor: only one
- Scheduling algorithm:
  - fixed-priority pre-emptive scheduling (FPPS),
    - i.e. processor is used to execute the *highest* priority task that has work pending;
    - non-idling;
  - tasks have *unique* priorities;
  - Overhead of scheduling and context switching is ignored.

Basic model for a task

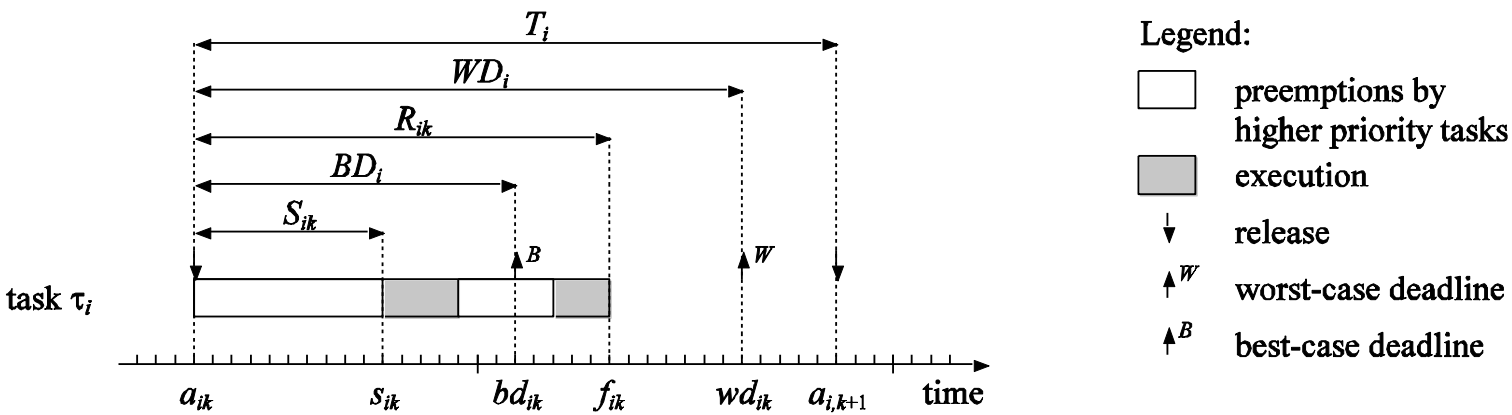
- Task τ_i: sequence of *jobs* ι_{ik} with $k \in \mathrm{Z}$.
- Basic timing notions (taken from R) for ι_{ik}
  - activation time a_{ik};
  - start time s_{ik};
  - finalization time f_{ik};
- Derived timing notions
  - response (or active) interval $[a_{ik}, f_{ik})$;
  - response time $R_{ik} = f_{ik} - a_{ik}$;
  - phasing $\varphi_i = a_{i,0}$;

Basic model for a task – continued

- (Best-case and worst-case) characteristics:
  - deadline: $R_{ik} \in [BD_i, WD_i]$, where $BD_i \in \mathrm{R}^+ \cup \{0\}$, $WD_i \in \mathrm{R}^+$;
  - computation time $C_{ik} \in [BC_i, WC_i]$, where $BC_i, WC_i \in \mathrm{R}^+$;
  - inter-arrival time (or period):
    - *strictly periodic* task (i.e. *without* activation jitter):
      - period $T_i \in \mathrm{R}^+$;
    - *elastic* task:
      - best-case period BT_i and worst-case period WT_i, where $WT_i < BT_i$;
    - *sporadic* task:
      - worst-case period WT_i (best-case period $BT_i \rightarrow \infty$)
  - deadlines and inter-arrival times:
    - $0 \leq BD_i \leq WD_i \leq WT_i$
- Notation:
  - we will also use WT_i and BT_i for periodic tasks.

Deadlines

- Relative deadlines:
  - best-case BD_i (typically assumed to be zero);
  - worst-case WD_i;
- Absolute deadlines:
  - best-case $bd_{ik} = a_{ik} + BD_i$;
  - worst-case $wd_{ik} = a_{ik} + WD_i$;



Overview of basic assumptions

- Single processor;
- Set Γ on n tasks $\tau_1 \ldots \tau_n$:
  - (released strictly periodically: $a_{ik} = \varphi_i + kT_i$);
  - (independent: no resource sharing and no precedence relations);
  - arbitrary phasing;
  - `ready' to run upon activation;
  - no self-suspension;
  - a job does not *start* before previous job completed ($f_{i,k-1} \leq s_{ik}$);
  - *hard* deadlines (i.e. $BD_i \leq R_{ik} \leq WD_i$) and $WD_i \leq WT_i$.
- Scheduling:
  - (FPPS and unique priorities);
  - (instantaneous pre-emption and non-idling);
  - overhead of context switching and task scheduling is ignored;
- Notational convenience:
  - tasks are given in order of *decreasing* priority,
    - i.e. τ_1 has highest priority and τ_n has lowest priority.

Motivation for FPPS

- De-facto standard
- Supported by commercial RTOS
- Rate monotonic analysis (RMA):
  - Adopted by leading companies and institutions [Obenza 94]:
    - Boeing, Honeywell, IBM, McDonnell Douglas, NASA, …;
    - IBM research, CMU/SEI.
  - Usage:
    - From simple control applications …
    - to large defense and aero-space applications.
  - Documentation [Klein et al 93]:
    - Practitioner's Handbook by CMU/SEI (KAP).

Overview

- Context
- Schedulability conditions
  - Exact, necessary, and sufficient conditions;
  - Recapitulation of [Liu and Layland 73].
- Basic response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
- Concluding remarks
- References

Schedulability conditions

- Requirement Req:
  - all jobs of all tasks of Γ must meet their deadline constraints,
    - i.e. $BD_i \leq R_{ik} \leq WD_i$ for all i and all k.
- Derived notions for task τ_i
  - *worst-case* response time WR_i

$$WR_i \stackrel{\text{def}}{=} \sup_{\varphi,k} R_{ik}(\varphi)$$

    where φ is the phasing of the set Γ;
  - *critical* instant: a (hypothetical) instant that leads to WR_i;
  - *best-case* response time BR_i

$$BR_i \stackrel{\text{def}}{=} \inf_{\varphi,k} R_{ik}(\varphi)$$

  - *optimal* instant: a (hypothetical) instant that leads to BR_i.

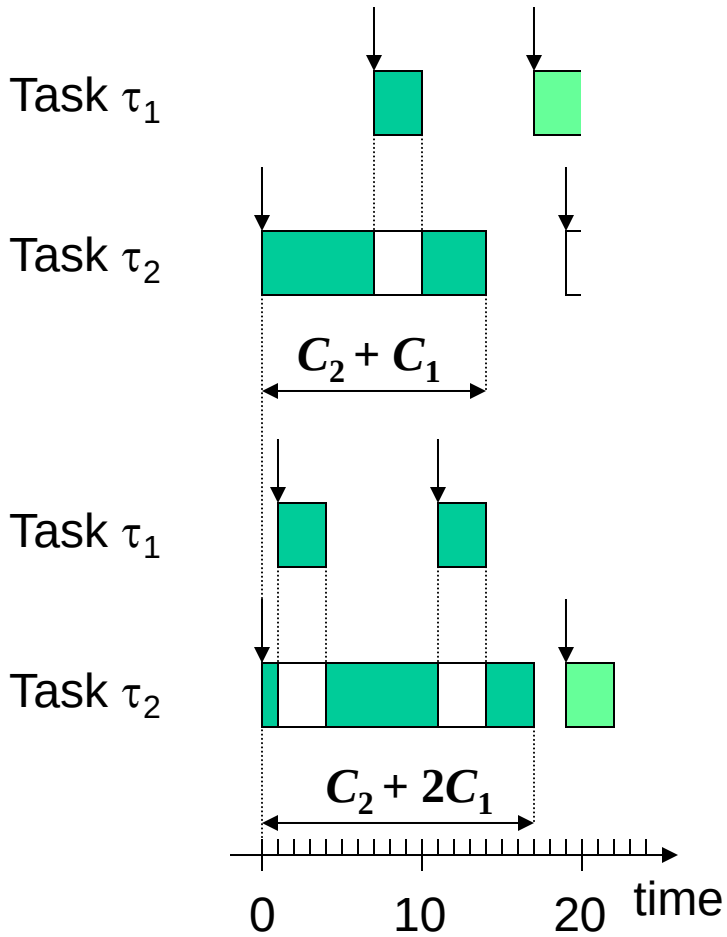
Schedulability conditions

- Re-phrased requirement Req:
  - $BD_i \leq BR_i \wedge WR_i \leq WD_i$ for all i and all k;
  - note:
    - a *best-case* part: $BReq = BD_i \leq BR_i$, and
    - a *worst-case* part: $WReq = WR_i \leq WD_i$.
- Types of conditions:
  - *Exact* condition EC: $EC \Leftrightarrow Req$
  - *Sufficient* condition SC: $SC \Rightarrow Req$
  - *Necessary* condition NC: $NC \Leftarrow Req$
- Specializations for best-case and worst-case, e.g.
  - *best-case* sufficient condition BSC: $BSC \Rightarrow BReq$
- Examples:
  - $BC_i \geq BD_i$: best-case *sufficient* condition;
  - $WC_i \leq WD_i$: worst-case *necessary* condition.

Recapitulation of [Liu and Layland 73]

- Assumptions (additional):
  - fixed computation time and inter-arrival time: i.e. $C_i = BC_i = WC_i$ and $T_i = BT_i = WT_i$;
  - best-case deadlines ignored: i.e. assume $BD_i = 0$;
  - worst-case deadlines equal to periods: i.e. $WD_i = T_i$.
- Utilization
  - $U_i^\tau = C_i / T_i$: utilization factor of task τ_i;
  - $U^\Gamma = \sum U_i^\tau$: (processor) utilization factor.
- *Necessary* condition: $U^\Gamma \leq 1$.
- Th. 1: A *critical instant* occurs upon a simultaneous release of a task with all its higher priority tasks.

Critical Instant



Task τ_2 is preempted by a single activation of the higher priority task τ_1.

The interference increases when the activation of task τ_1 is advanced.

Recapitulation of [Liu and Layland 73]

- Th. 2: *rate monotonic priority assignment* (RMA) is an *optimal* fixed priority assignment,
  - i.e. if a set Γ of tasks can be scheduled based on a fixed priority assignment, then Γ can be scheduled based on RMA.
- Note: Th. 2 only holds for *arbitrary* phasing; see [Goossens et al 97].
- *Sufficient* condition: $U^{\Gamma} \leq n(2^{1/n} - 1)$
  - also referred to as `Liu and Layland bound´ $LL(n)$;
  - adding tasks increases U^{Γ} and decreases `RHS´;
  - converges to $ln(2)$ (≈ 0.69) for $n \rightarrow \infty$;
  - for the proof, see [Devillers et al 00].

Alternative utilization bound

- *Sufficient* condition
  - `Hyperbolic bound´ $HB(n)$ [Liu 00]*

$$HB(n): \prod_{i=1}^{n} (U_i^{\tau} + 1) \leq 2$$

  - improves $LL(n)$, i.e. $LL(n) \Rightarrow HB(n)$;
  - the bound is `tight´;
  - Note:
    - adding tasks increases `LHS´.
    - * independently conceived by [Bini et al 01].

Exercises

- Consider the exercises C.1, C.2, and C.3 of the module “Implementing the real-time task model”.
  - Determine the utilization of the task-sets.
  - Determine whether or not the *necessary* condition $U^{\Gamma} \leq 1$ holds for the task-sets.
  - Determine whether or not the *sufficient* conditions for the task-sets holds:
    - LL-bound;
    - Hyperbolic bound.

An example (leading)

- Task set Γ consisting of 3 tasks:

Task	T	C	U
τ_1	10	3	0.3
τ_2	19	11	0.58
τ_3	56	5	0.09

- **Notes**:
  - RM priority assignment and $D_i = T_i$ (RMS);
  - *Necessary* condition:
    - $U_1^\tau + U_2^\tau + U_3^\tau = 0.97 \leq 1$, hence Γ *could* be schedulable;
  - *Sufficient* condition:
    - $LL(n)$: $U^\Gamma \leq n\,(2^{1/n} - 1)$: $U_1^\tau + U_2^\tau = 0.88 >$ `$LL(2)$´ $\approx 0.83$, therefore $U_1^\tau + U_2^\tau + U_3^\tau >$ `$LL(3)$´, hence another test required;
    - $HB(n)$: $(U_1^\tau + 1)(U_2^\tau + 1) \approx 2.05 > 2$, therefore `$HB(3)$´ > 2, hence another test required.

Overview

- Context
- Schedulability conditions
- Basic response time analysis
  - Worst-case response time analysis
  - Best-case response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
- Concluding remarks
- References

Basic response time analysis

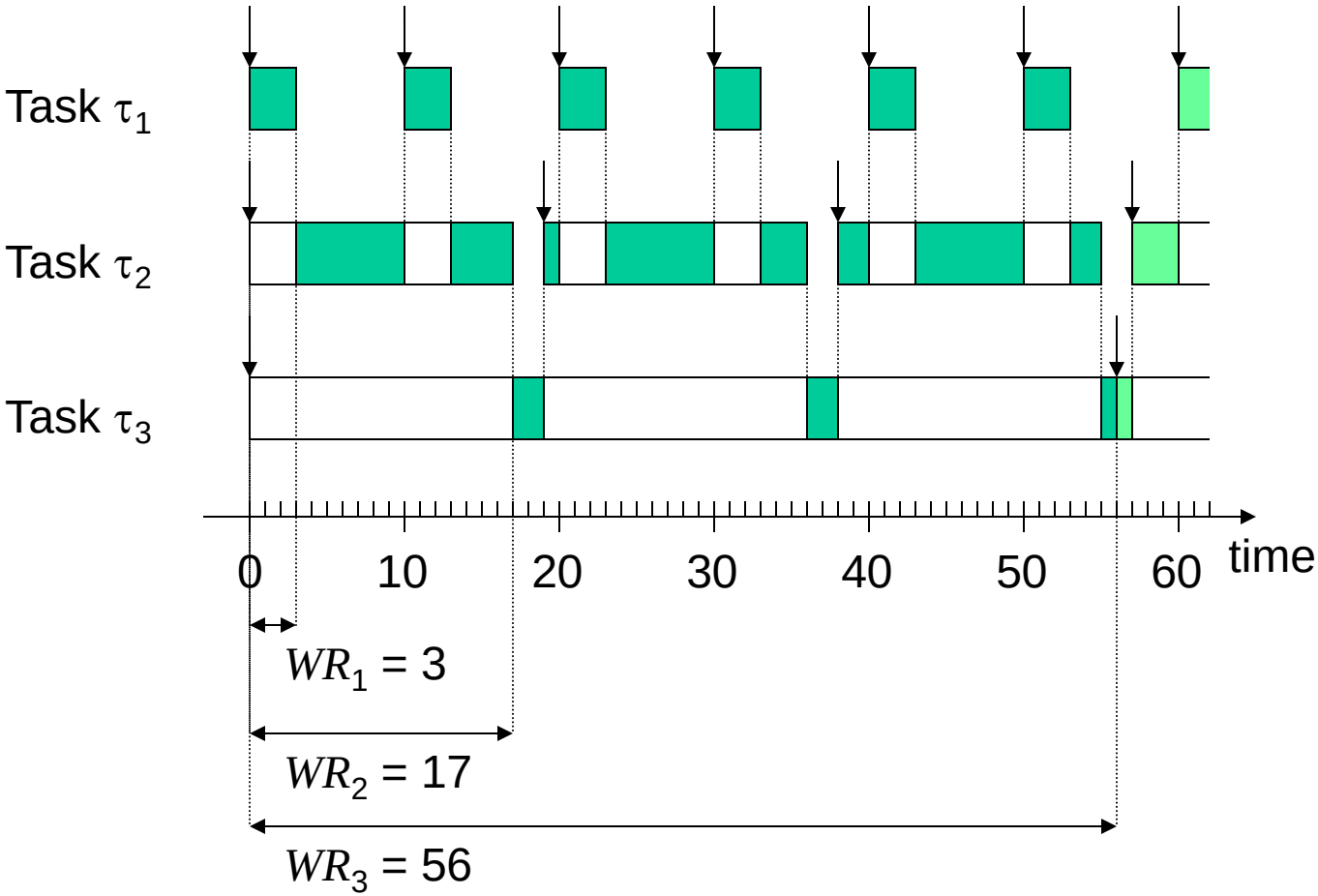
- Assumptions ('reset')
  - deadlines at most equal to periods: $WD_i \leq WT_i$;
  - worst-case and best-case characteristics;
  - arbitrary (but fixed and unique) priority assignments.
- Worst-case response time analysis
  - (Critical instant);
  - Techniques: timeline + calculation.
- Best-case response time analysis
  - Introduction;
  - Optimal instant;
  - Techniques: timeline + calculation.

Worst-case response time analysis

- *Critical* instant for task τ_i: τ_i "assumes" its WR_i.
- A critical instant for task τ_i:
  - Task τ_i is released simultaneously with all tasks with a higher priority.
  - The *highest* amount of pre-emption of a task is found *after* a simultaneous release of higher priority tasks.
  - General for a set of tasks.
- Note: "A" rather than "the", because there may be more instants for which τ_i "assumes" its WR_i.

WCRT – Techniques

Time line:



WCRT – Techniques

Calculation:

- Recursive equation for task τ_i:

$$x = WC_i + \sum_{j<i} \left\lceil \frac{x}{WT_j} \right\rceil WC_j$$

WR_i is the *smallest* positive solution

- Assume a task τ_j with a higher priority than τ_i;
  - $\lceil x/WT_j \rceil$ denotes the *maximum* number of preemptions of task τ_i in an interval [0, x) by task τ_j;
  - $\lceil x/WT_j \rceil WC_j$ denotes the *maximal* preemption time of task τ_i in an interval [0, x) by task τ_j.
- Intuition:
  - LHS: amount of time *available* (or *provided*) in [0, x);
  - RHS: *max*. amount of time *requested* in [0, x) by τ_i and $\forall_{j<i}\ \tau_j$.

WCRT – Techniques

Calculation:

- Iterative procedure:

$$WR_i^{(0)} = WC_i + \sum_{j<i} WC_j$$

$$WR_i^{(k+1)} = WC_i + \sum_{j<i} \left\lceil \frac{WR_i^{(k)}}{WT_j} \right\rceil WC_j$$

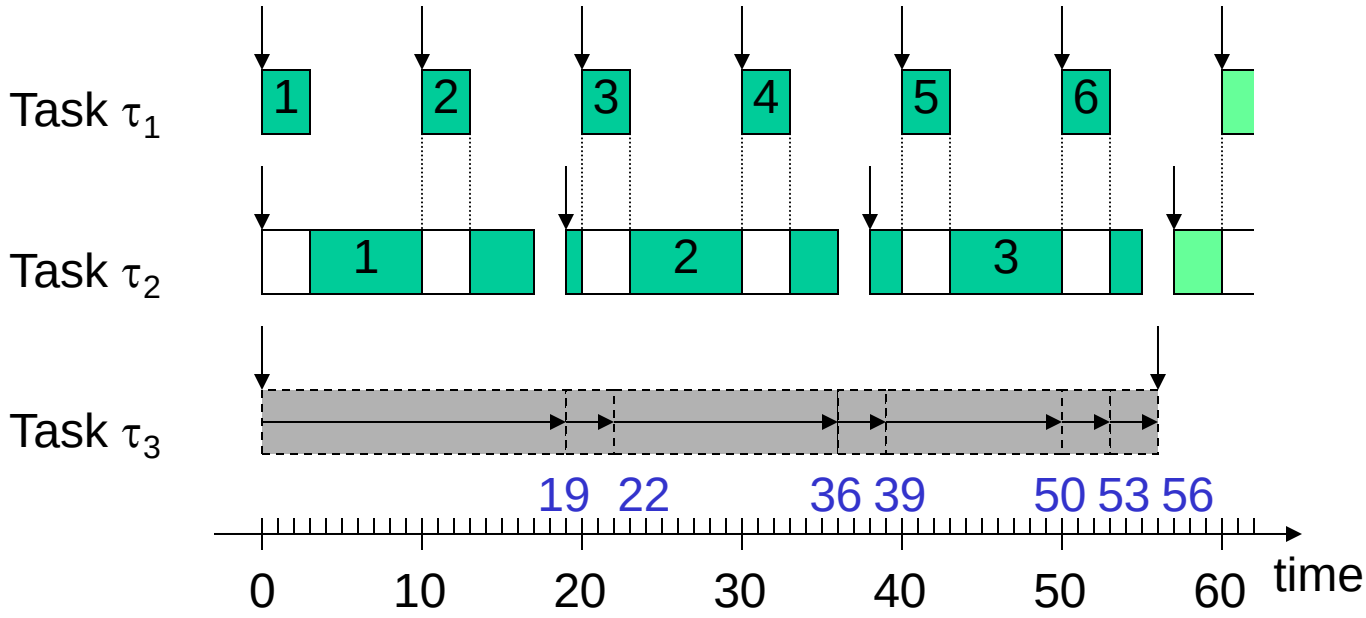
- Stopped when either:
  - the same value is found for two successive iterations; or
  - the deadline WD_i is exceeded (hence, not schedulable).
- All intermediate values are *at most* equal to WR_i;
- Terminates when $\Sigma_{j<i}\ U_j^{\tau} < 1$.
- See [Harter 84], [Joseph et al 86] or [Audsley et al 91].

WCRT – Techniques

- **Example for task τ_3:**
  - $WR_3^{(0)} = C_3 + \Sigma_{j<3} C_j = 5 + 3 + 11 = 19$
  - $WR_3^{(1)} = C_3 + \Sigma_{j<3} \lceil WR_3^{(0)}/T_j \rceil C_j =$
    $5 + \lceil 19/10 \rceil \cdot 3 + \lceil 19/19 \rceil \cdot 11 = 5 + 2 \cdot 3 + 1 \cdot 11 = 22$
  - $WR_3^{(2)} = 5 + \lceil 22/10 \rceil \cdot 3 + \lceil 22/19 \rceil \cdot 11 = 5 + 3 \cdot 3 + 2 \cdot 11 = 36$
  - $WR_3^{(3)} = 5 + \lceil 36/10 \rceil \cdot 3 + \lceil 36/19 \rceil \cdot 11 = 5 + 4 \cdot 3 + 2 \cdot 11 = 39$
  - $WR_3^{(4)} = 5 + \lceil 39/10 \rceil \cdot 3 + \lceil 39/19 \rceil \cdot 11 = 5 + 4 \cdot 3 + 3 \cdot 11 = 50$
  - $WR_3^{(5)} = 5 + \lceil 50/10 \rceil \cdot 3 + \lceil 50/19 \rceil \cdot 11 = 5 + 5 \cdot 3 + 3 \cdot 11 = 53$
  - $WR_3^{(6)} = 5 + \lceil 53/10 \rceil \cdot 3 + \lceil 53/19 \rceil \cdot 11 = 5 + 6 \cdot 3 + 3 \cdot 11 = 56$
  - $WR_3^{(7)} = 5 + \lceil 56/10 \rceil \cdot 3 + \lceil 56/19 \rceil \cdot 11 = 5 + 6 \cdot 3 + 3 \cdot 11 = 56$
  - Because $WR_3^{(6)} = WR_3^{(7)} = 56 \leq D_3 = T_3$, $WR_3 = 56$.

WCRT – Techniques

Calculation (visualization):



$$WR_3^{(6)} = WR_3^{(7)} = 5 + 6 \cdot 3 + 3 \cdot 11 = 56$$

WCRT - Techniques

- Note:
  - The number of iterations can be reduced by using
    - $WR_3^{(0)} = \frac{c_i}{1-U_{i-1}}$,
    - where $U_{i-1} = \sum_{1 \le j < i} \frac{c_j}{T_j}$.
  - We now get
    - $WR_3^{(0)} = 5/(1 - (\frac{3}{10} + \frac{11}{19})) \approx 41$
    - $WR_3^{(1)} = 5 + \lceil 41/10 \rceil \cdot 3 + \lceil 41/19 \rceil \cdot 11 = 5 + 5 \cdot 3 + 3 \cdot 11 = 53$
    - $WR_3^{(2)} = 5 + \lceil 53/10 \rceil \cdot 3 + \lceil 53/19 \rceil \cdot 11 = 5 + 6 \cdot 3 + 3 \cdot 11 = 56$
    - $WR_3^{(3)} = 5 + \lceil 56/10 \rceil \cdot 3 + \lceil 56/19 \rceil \cdot 11 = 5 + 6 \cdot 3 + 3 \cdot 11 = 56$
    - Hence, only 3 instead of 7 iterations.

WCRT – Exercises

- Consider the exercises C.1, C.2, and C.3 of the module “Implementing the real-time task model”.
  - Determine the worst-case response times of the tasks under FPPS.

WCRT – Exercise*

- Task set Γ' consisting of 3 tasks:

Task	$WT = WD$	WC	WU
τ_1	5	2	0.4
τ_2	7	3	0.43
τ_3	29	3	0.10

  - Determine `worst-case schedulability' of Γ'.
  - Draw a time line with a critical instant for Γ'.
  - Explain, using the time line, why 25 and 27 are also solutions for the recursive equation for WR_3.

Best-case response time analysis

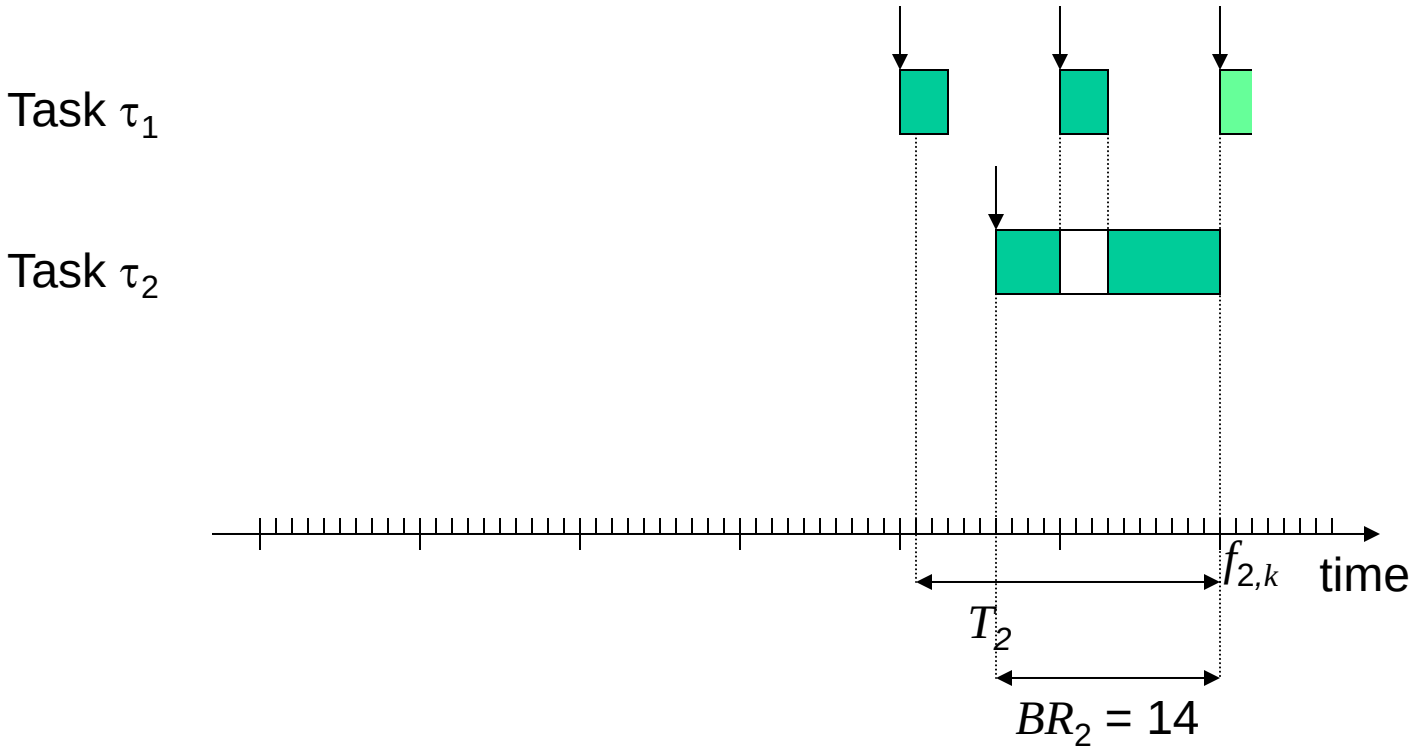
- See [Redell et al 02] or [Bril et al 04].
- Motivation
  - Best-case and worst-case notions are duals:
    - Best-case: *earliest*, *shortest*, or *minimal*;
    - Worst-case: *latest*, *longest*, or *maximal*;
  - Best-case deadline:
    - response no *earlier* than BD
  - May simplify design and reduce cost:
    - No need to “delay” (i.e. buffer) output artificially;
  - Desirable for analysis in distributed systems:
    - but also applies to single-processor systems;
    - examples: airbag, engine control.

BCRT – Optimal instant

- *Optimal* instant for task τ_i: τ_i "assumes" its BR_i.
- An optimal (or *favourable*) instant:
  - Job ι_{ik} *ends* simultaneously with the *release* of all tasks with a higher priority, and ι_{ik}'s release time is equal to its start time, i.e. $a_{ik} = s_{ik}$.
  - The *lowest* amount of pre-emption of a task is found *before* a simultaneous release of higher priority tasks.
  - Specific for each task !
- Note: "An" rather than "the", because there may be more instants for which τ_i "assumes" its BR_i.

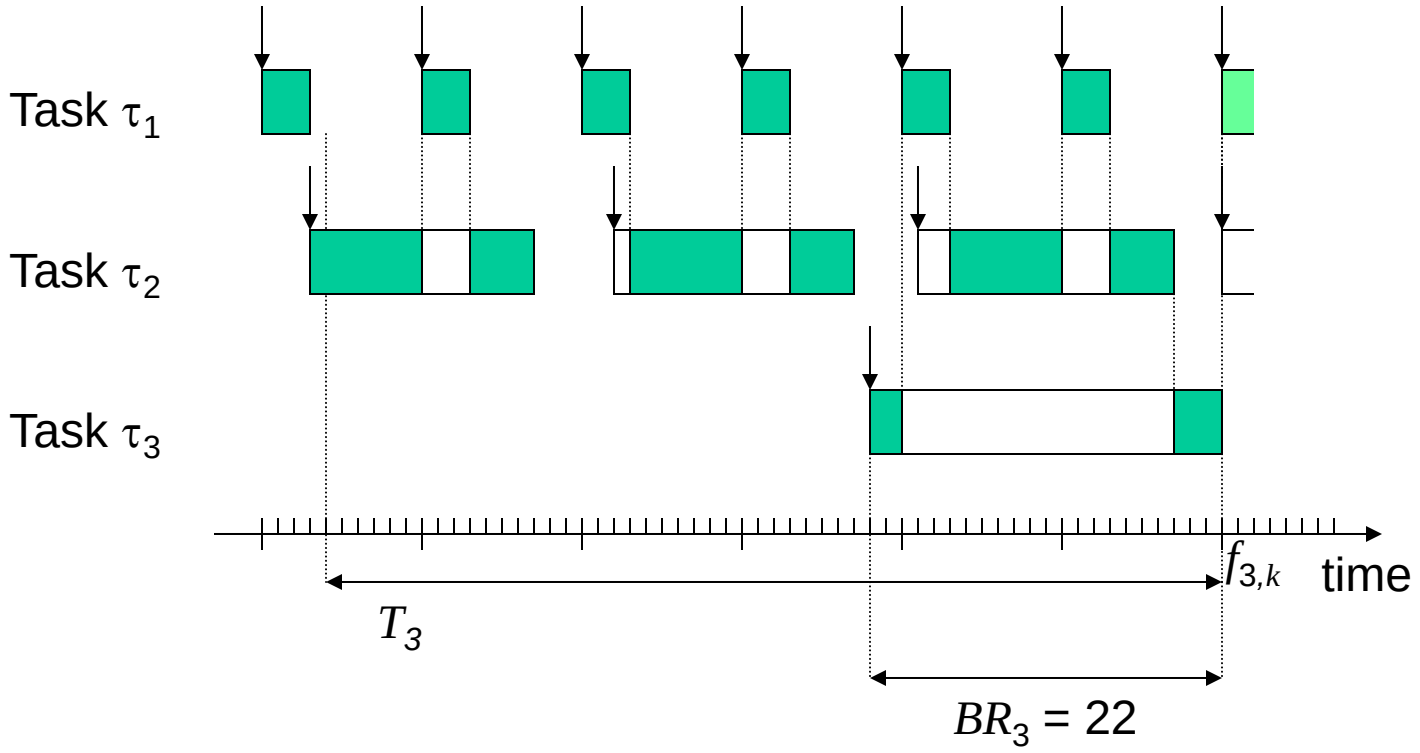
BCRT – Techniques

Time line: optimal instant for τ_2



BCRT – Techniques

Time line: optimal instant for τ_3



BCRT – Techniques

Calculation:

- Recursive equation for task τ_i:

$$x = BC_i + \sum_{j<i} \left(\left\lceil \frac{x}{BT_j} \right\rceil - 1 \right) BC_j$$

BR_i is the *largest* positive solution

- Assume a task τ_j with a higher priority than τ_i;
  - $\lceil x/BT_j \rceil - 1$ denotes the *minimal* number of preemptions of task τ_i in an interval $(0, x)$ by task τ_j;
  - $(\lceil x/BT_j \rceil - 1)BC_j$ denotes the *minimal* preemption time of task τ_i in an interval $(0, x)$ by task τ_j.
- Intuition:
  - LHS: amount of time *available* (or *provided*) *in* $(0, x)$;
  - RHS: *min*. amount of time *requested* in $(0, x)$ by τ_i and $\forall_{j<i}\ \tau_j$.

BCRT – Techniques

Calculation:

- Iterative procedure:

$$BR_i^{(0)} = WR_i$$

$$BR_i^{(k+1)} = BC_i + \sum_{j<i} \left(\left\lceil \frac{BR_i^{(k)}}{BT_j} \right\rceil - 1 \right) BC_j$$

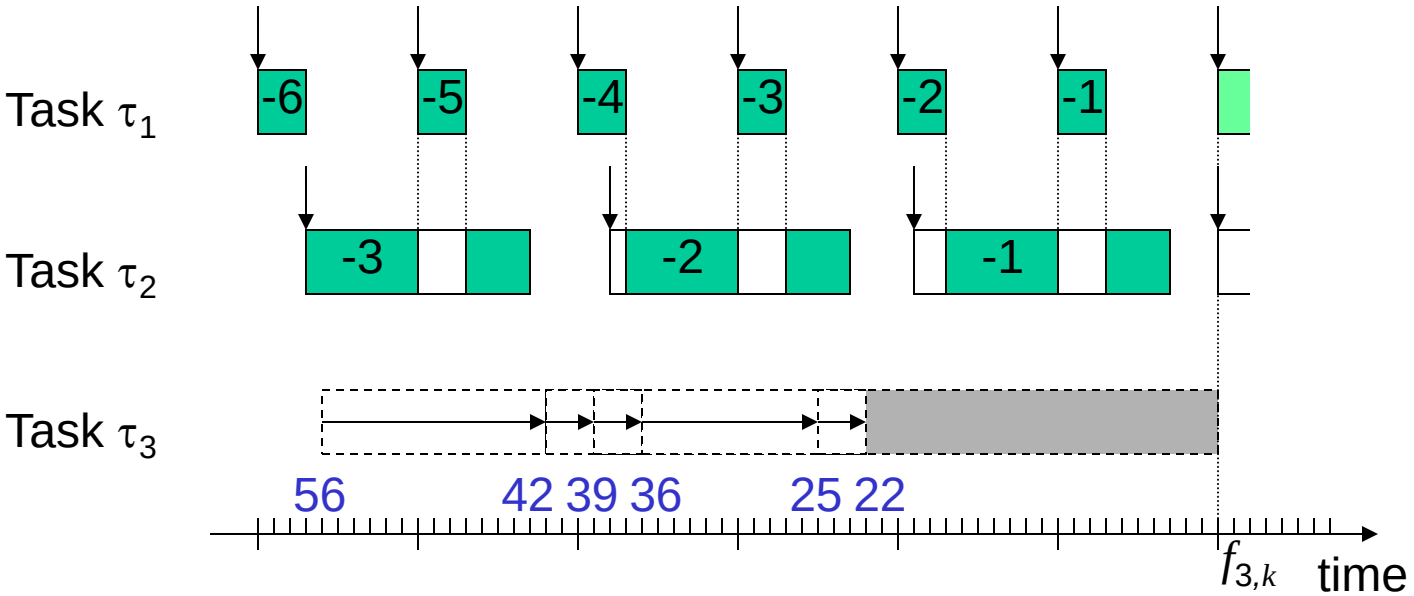
- Stopped when:
  - the same value is found for two successive iterations; or
  - the deadline BD_i is exceeded (hence, not schedulable).
- All intermediate values are *at least* equal to BR_i.

BCRT – Techniques

- **Example for task τ_3:**
  - Assume $BD_i = 0$.
  - $BR_3^{(0)} = WR_3 = 56$
  - $BR_3^{(1)} = C_3 + \Sigma_{j<3} (\lceil BR_3^{(0)}/T_j \rceil - 1)C_j =$
    $5 + (\lceil 56/10 \rceil - 1)\cdot 3 + (\lceil 56/19 \rceil - 1)\cdot 11 = 5 + 5\cdot 3 + 2\cdot 11 = 42$
  - $BR_3^{(2)} = 5 + (\lceil 42/10 \rceil - 1)\cdot 3 + (\lceil 42/19 \rceil - 1)\cdot 11 = 5 + 4\cdot 3 + 2\cdot 11 = 39$
  - $BR_3^{(3)} = 5 + (\lceil 39/10 \rceil - 1)\cdot 3 + (\lceil 39/19 \rceil - 1)\cdot 11 = 5 + 3\cdot 3 + 2\cdot 11 = 36$
  - $BR_3^{(4)} = 5 + (\lceil 36/10 \rceil - 1)\cdot 3 + (\lceil 36/19 \rceil - 1)\cdot 11 = 5 + 3\cdot 3 + 1\cdot 11 = 25$
  - $BR_3^{(5)} = 5 + (\lceil 25/10 \rceil - 1)\cdot 3 + (\lceil 25/19 \rceil - 1)\cdot 11 = 5 + 2\cdot 3 + 1\cdot 11 = 22$
  - $BR_3^{(6)} = 5 + (\lceil 22/10 \rceil - 1)\cdot 3 + (\lceil 22/19 \rceil - 1)\cdot 11 = 5 + 2\cdot 3 + 1\cdot 11 = 22$
  - Because $BR_3^{(5)} = BR_3^{(6)} = 22 \geq BD_3 = 0$, $BR_3 = 22$.

BCRT – Techniques

Calculation (visualization)**:**



$$BR_3^{(5)} = BR_3^{(6)} = 5 + 2\cdot 3 + 1\cdot 11 = 22$$

BCRT - Techniques

- Note:
  - The number of iterations can be reduced by using
    - $BR_3^{(0)} = \frac{c_i}{1-U_{i-1}}$,
    - where $U_{i-1} = \sum_{1 \le j < i} \frac{c_j}{T_j}$.
  - We now get
    - $BR_3^{(0)} = 5/(1 - (\frac{3}{10}+\frac{11}{19})) \approx 41$
    - $BR_3^{(1)} = 5 + (\lceil 41/10 \rceil - 1)\cdot 3 + (\lceil 41/19 \rceil - 1)\cdot 11 = 5 + 4\cdot 3 + 2\cdot 11 = 39$
    - $BR_3^{(2)} = 5 + (\lceil 39/10 \rceil - 1)\cdot 3 + (\lceil 39/19 \rceil - 1)\cdot 11 = 5 + 3\cdot 3 + 2\cdot 11 = 36$
    - $BR_3^{(3)} = 5 + (\lceil 36/10 \rceil - 1)\cdot 3 + (\lceil 36/19 \rceil - 1)\cdot 11 = 5 + 3\cdot 3 + 1\cdot 11 = 25$
    - $BR_3^{(4)} = 5 + (\lceil 25/10 \rceil - 1)\cdot 3 + (\lceil 25/19 \rceil - 1)\cdot 11 = 5 + 2\cdot 3 + 1\cdot 11 = 22$
    - $BR_3^{(5)} = 5 + (\lceil 22/10 \rceil - 1)\cdot 3 + (\lceil 22/19 \rceil - 1)\cdot 11 = 5 + 2\cdot 3 + 1\cdot 11 = 22$
    - Hence, only 5 instead of 6 iterations.

BCRT – Exercises

- Consider the exercises C.1, C.2, and C.3 of the module “Implementing the real-time task model”.
  - Determine the best-case response times of the tasks under FPPS.
  - Can all tasks assume their best-case response times for a “single” phasing?

BCRT – Exercise*

- Task set Γ' consisting of 3 tasks:

Task	BT	$BC = BD$	BU
τ_1	5	2	0.4
τ_2	7	3	0.43
τ_3	29	3	0.10

  - Determine the best-case response times BR_i for all three tasks of Γ'.
  - Draw a time line with an optimal instant for τ_3.
  - Explain, using the time line, why 3 is also a solution for the recursive equation for BR_3.

Overview

- Context
- Schedulability conditions
- Basic response time analysis
- Jitter analysis for periodic tasks
  - Distributed systems analysis
  - Response jitter
  - Activation jitter
  - Finalization and activation jitter
- Resource sharing
- Practical factors
- Concluding remarks
- References

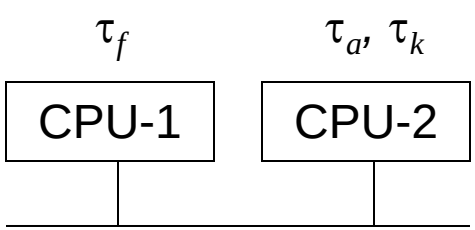
Distributed systems analysis

Finalization of a task τ_f on one processor *activates* a task τ_a on another processor:

- Periodic tasks:
  - *Finalization jitter* FJ: variation in finalization times;
  - *Activation jitter* AJ: variation in activation times (e.g. output of one task triggers a next task);
  - Example: multimedia in a networked environment.
- Elastic tasks:
  - *Minimal inter-finalization time* of τ_f determines WT_a of task τ_a.
  - *Maximal inter-finalization time* of τ_f determines BT_a of task τ_a.
- Sporadic tasks:
  - *Minimal inter-finalization time* of τ_f determines WT_a of task τ_a.

Distributed systems analysis

– Example:



τ_f triggers τ_a

FJ_f causes AJ_a, which

influences response times

of *both* τ_a and τ_k

– Goal jitter analysis:
  - Determine schedulability in the context of jitter,
  - hence, determine response (or finalization) times.

– Note: strictly spoken, the variations in the delay of the messages on the bus should be taken into account as well…

Advanced exercises

- Proof that the "type" of τ_a is determined by τ_f, e.g. τ_a is an elastic task if τ_f is an elastic task.
- For both an elastic and a sporadic task τ_f, express WT_a in terms of WT_f, WR_f, and BR_f.
- For an elastic task, express BT_a in terms of BT_f, WR_f, and BR_f.
- How about BT_a for a sporadic task?

Jitter analysis for periodic tasks

Types of (absolute) jitter (recap):

- *Response jitter* RJ_i:
  - variations in *response* times;
- *Activation* (or *release*) *jitter* AJ_i:
  - variation in *release* times (e.g. output of one task triggers a next task);
- *Finalization* (or *end*) *jitter* FJ_i:
  - variation in *end* times;

Response jitter

- Response jitter RJ_i of a task τ_i:

$$RJ_i \overset{\text{def}}{=} \sup_{\varphi,k,l}(R_{ik}(\varphi) - R_{il}(\varphi))$$

- A bound on response jitter

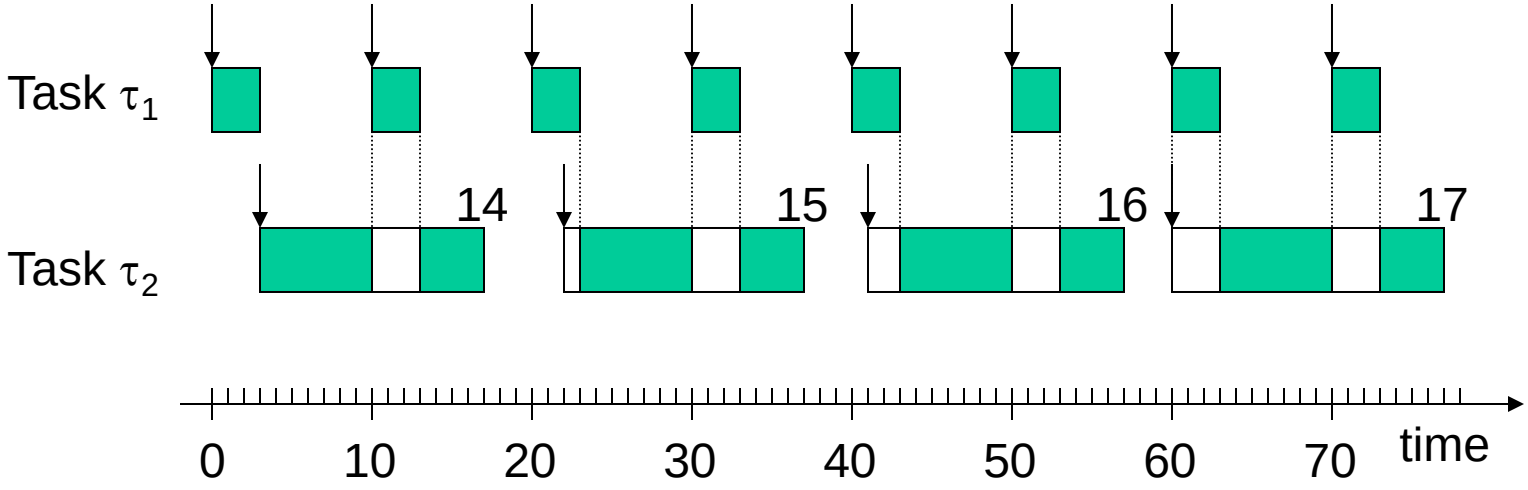
$$RJ_i \leq WR_i - BR_i$$

  – "$\leq$" because WR_i and BR_i are not necessarily assumed for the same phasing

Response jitter

Example (leading):

- $WR_2 = 17$, $BR_2 = 14$;
- WR_2 and BR_2 both assumed for a single phasing,
- hence, $FJ_2 = WR_2 - BR_2 = 3$,
- and the bound for RJ_2 is therefore *tight*.



Activation jitter of periodic tasks

- Assumptions:
  - revised assumption: $WD_i \leq WT_i - AJ_i$.
  - lifted restrictive assumption:

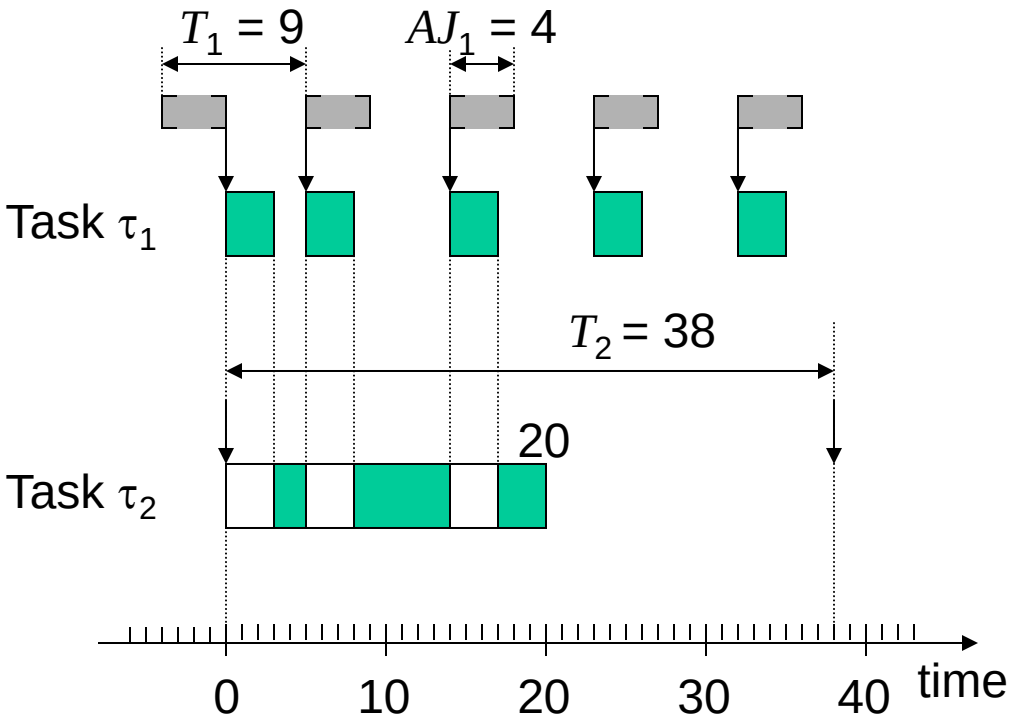
$$\sup_{k,l}(a_{ik} - a_{il} - (k-l)T_i) \leq AJ_i$$

- Worst-case response times:
  - See [Audsley et al 93] or [Tindell et al 94];
  - Critical instant revisited:
    - Task τ_i is released simultaneously with all tasks with a higher priority and
    - all tasks with a higher priority experience
      - a maximal release delay at that simultaneous release, and a minimal release delay at subsequent releases.
  - Hence, a *maximal* pre-emption of τ_i occurs.

Activation jitter of periodic tasks

New example:

Task	T	C	AJ
τ_1	9	3	4
τ_2	38	11	7



Notes:

- WR_2 is independent of AJ_2;
- WR_2 increases 3 due to AJ_1.

Activation jitter of periodic tasks

- Worst-case response times:
  - Recursive equation for task τ_i:

$$x = WC_i + \sum_{j<i} \left\lceil \frac{x + AJ_j}{WT_j} \right\rceil WC_j$$

    - *Where* AJ_j is the activation jitter of τ_j.
    - WR_i is the *smallest* positive solution of the equation.
  - Iterative procedure:
    - Similar to the case without jitter.
  - Note: *equation* also holds for elastic and sporadic tasks!
    - Because $AJ_j = 0$ for those tasks.

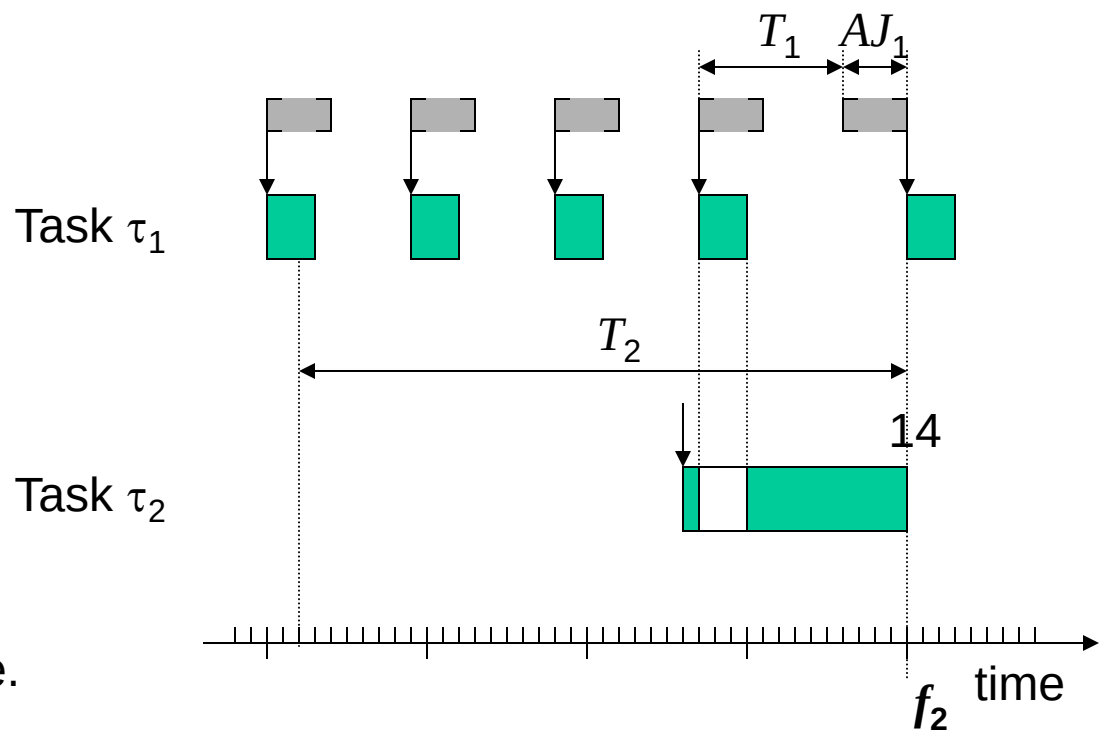
Activation jitter of periodic tasks

- Best-case response times:
  - Optimal instant revisited:
    - Job ι_{ik} *ends* simultaneously with the *release* of all tasks with a higher priority, and ι_{ik}'s release time is equal to its start time, and
    - all tasks with a higher priority experience
      - a maximal release delay at that simultaneous release, and
      - a minimal release delay at previous releases.
  - Hence, a *minimal* pre-emption of τ_i occurs.

Activation jitter of periodic tasks

Same (new) example:

Task	T	C	AJ
τ_1	9	3	4
τ_2	38	11	7



Notes:

- BR_2 is independent of AJ_2;
- BR_2 does not change here.

Activation jitter of periodic tasks

- Best-case response times:
  - Recursive equation for task τ_i:

$$x = BC_i + \sum_{j<i} \left(\left\lceil \frac{x - AJ_j}{BT_j} \right\rceil - 1 \right)^+ BC_j$$

    - *Where* AJ_j is the activation jitter of τ_j, and $w^+ = \max\{w,0\}$.
    - BR_i is the *largest* positive solution of the equation.
  - Iterative procedure:
    - Similar to the case without jitter.
  - Note: *equation* also holds for elastic and sporadic tasks!
    - Because $AJ_j = 0$ for those tasks.

Finalization and activation jitter

- Finalization jitter FJ_i of a periodic task τ_i:

$$FJ_i \overset{\text{def}}{=} \sup_{\varphi,k,l}\left(f_{ik}(\varphi) - f_{il}(\varphi) - (k-l)T_i \right)$$

- A bound on finalization jitter:

$$FJ_i \leq AJ_i + WR_i - BR_i$$

  – "≤" because WR_i and BR_i are not necessarily assumed for the same phasing.

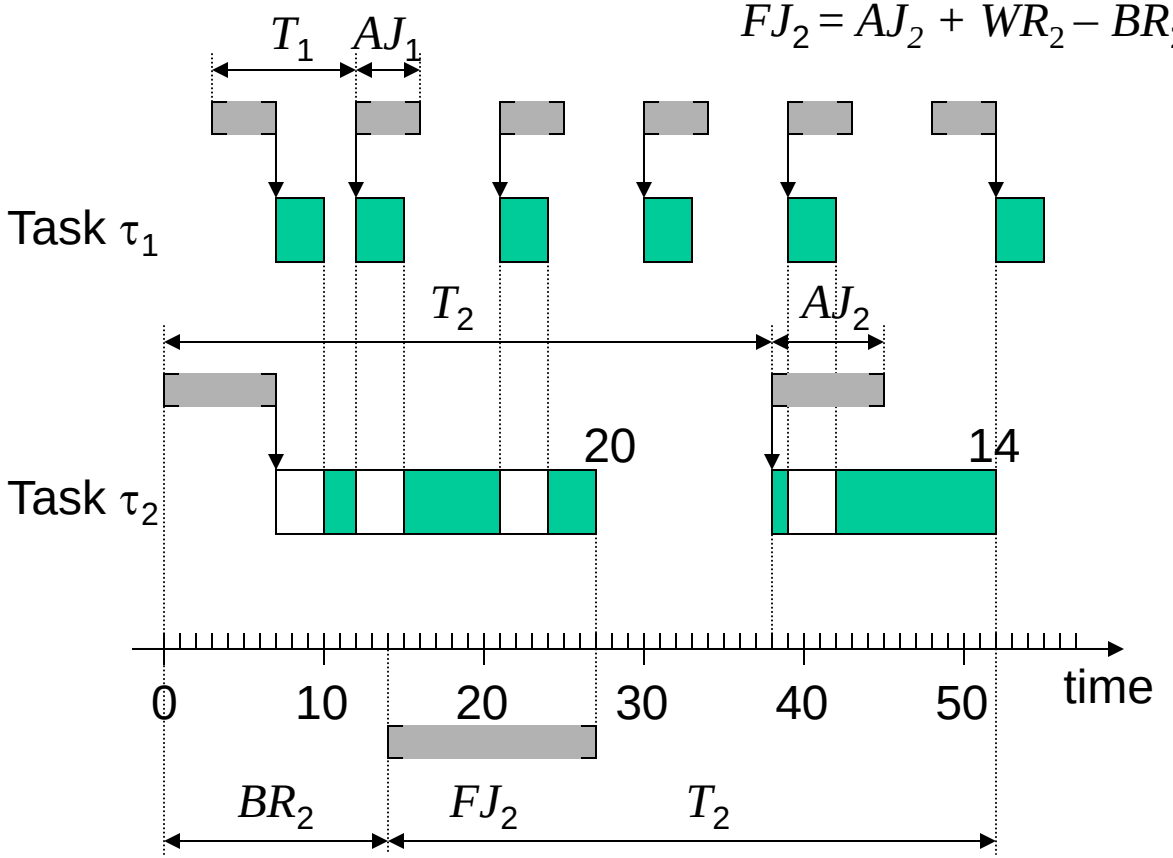
- Exercise: derive this latter bound

Response, finalization and activation jitter

Example:

$$RJ_2 = WR_2 - BR_2 = 20 - 14 = 6$$

$$FJ_2 = AJ_2 + WR_2 - BR_2 = 7 + 20 - 14 = 13$$



$$RJ_1 = WR_1 - BR_1 = 0$$

$$FJ_1 = AJ_1 = 4$$

Advanced exercise

- Define worst-case processor utilization WU in the context of periodic tasks with activation jitter.
  - Can you describe a (worst-case) necessary condition using WU?
  - What about the analysis?
- This lecture considered *elastic*, *periodic*, and *sporadic* tasks.
  - What is the effect of modeling
    - a *periodic task with jitter* as an *elastic task*?
    - a *periodic task with jitter* as a *sporadic task*?

Overview

- Context
- Schedulability conditions
- Basic response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
- Concluding remarks
- References

Response-time analysis

- Worst-case response time analysis:
  - Blocking time B_i:
    - Longest time τ_i can be blocked by a task with a *lower* priority.
    - Depends on the resource access protocol.
    - *Includes* blocking due to non-interrupt-able code of system-calls.
  - Recursive equation for task τ_i:

$$x = B_i + WC_i + \sum_{j<i} \left\lceil \frac{x + AJ_j}{WT_j} \right\rceil WC_j$$

WR_i is the *smallest* positive solution

- Best-case response time analysis:
  - Yet unknown.

Overview

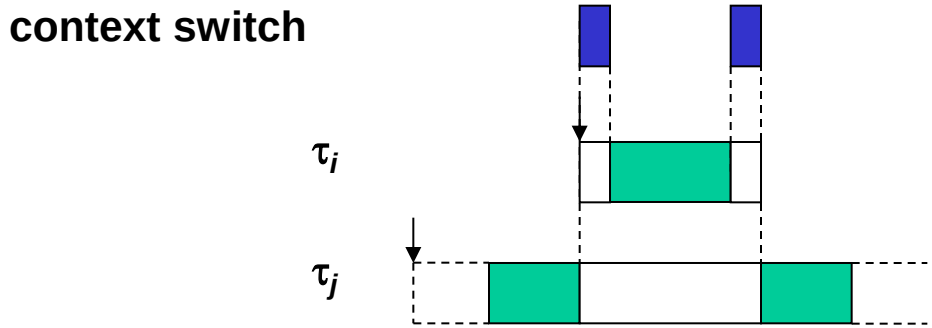
- Context
- Schedulability conditions
- Basic response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
  - From event to task activation
  - Context switches
  - Interrupts
- Concluding remarks
- References

From external event to task activation

- Steps:
  - External event occurs;
  - Detection by sensor;
  - Interrupt generated by sensor
    - may take some time before the bus is free
  - Interrupt arrival at CPU
    - may take some time before the interrupt is handled
  - Immediate interrupt service
    - may be pre-empted by higher priority interrupts
  - Activation of the scheduler
    - may be delayed to the next clock-tick
  - Activation of the task
- hence, both a *delay* and potential *jitter* between the arrival of the external event and activation of the task!

Context switches

- *Question*: how many *running* jobs can a job pre-empt?
- *Answer*: at most 1 (…but beware or resource sharing)!



- Let CS denote the *context-switch time of the system*, i.e.
  - max time the system spends on a context switch;
  - optionally including time of the scheduler to service the event interrupt that triggered the context switch
-

Context switches

- Extending the analysis:
  - Replace C_j by $C_j + 2CS$;
  - Replace C_i by $C_i + 2CS$;
- Questions:
  - Can these extensions be applied for the *necessary* condition, *sufficient* condition, and response-time analysis? Motivate your answer.
  - Can you ignore the context switch out-of a task for the response time analysis of that task, i.e. use $C_i + CS$ rather than $C_i + 2CS$? Motivate your answer.

External interrupts

- The *immediate interrupt service*
  - will pre-empt a running task τ;
  - even when the sporadic task handling the interrupt has a lower priority than τ.
- Let
  - T_k: the minimum inter-arrival time of the interrupt corresponding with sporadic task τ_k;
  - Γ_s: the set of sporadic tasks;
  - IH_k: the cost of handling that interrupt.
- Extension of the recursive equation $\sum_{\tau_k \in \Gamma_s} \left\lceil \frac{x}{T_k} \right\rceil IH_k$

Clock interrupt

- Simple (basic) approach:
  - similar to external interrupts, i.e.
  - extension of the recursive equation

$$\left\lceil \frac{x}{T_{\text{clk}}} \right\rceil IH_{\text{clk}}$$

- Refinement:
  - distinguish between
    - a *fixed* cost to serve the clock interrupt;
    - *additional* costs to “move” tasks from the waiting queue to the ready queue
- Question: How to model the *additional* costs?

Exercises

- Measure the overhead of the “tick” interrupt handler.

Overview

- Context
- Schedulability conditions
- Basic response time analysis
- Jitter analysis for periodic tasks
- Resource sharing
- Practical factors
- Concluding remarks
- References

Concluding remarks

- Many of the restrictive assumptions have been lifted, e.g.
  - other dependencies between tasks
    - e.g. precedence relations;
  - tasks with varying priorities [Harbour et al 91];
  - arbitrary deadlines [Klein et al 93];
  - tasks with offsets;
  - tasks that suspend themselves;
  - limited-pre-emptive scheduling [Regher 02], [Bril et al 07].
- Unlike *worst-case* response time analysis, *best-case* response time analysis has not been addressed yet for any of these lifted assumptions!
- Further elaboration falls outside the scope of this lecture, however.

Concluding remarks

- Be aware:
  - The analysis presented has the explicitly stated assumptions as *preconditions*!
  - For many situations, only looking at the first job upon a critical instant is *not* sufficient, even when deadlines are at most equal to periods. Instead, the response times of all jobs in a so-called *busy* (or *level-i active*) *period* have to be examined! Examples:
    - tasks with varying priorities [Harbour et al 91].
    - preemption thresholds [Regehr 02];
    - Controller Area Network (CAN), see [Davis et al 07];
    - fixed-priority scheduling with deferred preemption (FPDS), see [Bril et al 07].

References - I

- [Audsley et al 91] N.C. Audsley and A. Burns and M.F. Richardson and A.J. Wellings, *Hard Real-Time Scheduling: The Deadline Monotonic Approach*, In: Proc. 8th IEEE Workshop on Real-Time Operating Systems and Software (RTOSS), pp. 133-137, May 1991.
- [Audsley et al 93] N.C. Audsley and A. Burns and M.F. Richardson and K. Tindell and A.J. Wellings, *Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling*, Software Engineering Journal, 8(5): 284-292, 1993.
- [Bini et al 01] E. Bini and G. Buttazzo and G. Buttazzo, *A hyperbolic bound for the rate monotonic algorithm*, In: Proc. 13th Euromicro Conference on Real-Time Systems (ECRTS), pp. 59-66, June 2001.
- [Bril et al 04] R.J. Bril, E.F.M. Steffens, and W.F.J. Verhaegh, *Best-case response times and jitter analysis of real-time tasks*, Journal of Scheduling, 7(2): 133-147, 2004.
- [Bril et al 07] R.J. Bril, J.J. Lukkien, and W.F.J. Verhaegh, *Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited*, In: Proc. 19th Euromicro Conference on Real-Time Systems (ECRTS'07), pp. 269 – 279, July 2007.

References - II

- [Davis et al 07] R.I. Davis and A. Burns and R.J. Bril and J.J. Lukkien, *Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised*", Real-Time Systems, ISSN 1573-1383 (online), http://www.springerlink.com/content/8n32720737877071/, January 2007, ISSN 0922-6443 (print), 35(3): 239-272, April 2007.
- [Devillers et al 00] R. Devillers and J. Goossens, *Liu and Layland's schedulability test revisited*, Information Processing Letters, 73(5-6): 157-161, March 2000.
- [Goossens et al 97] J. Goossens and R. Devillers, *The non-optimality of the monotonic priority assignments for hard real-time offset free systems*, Real-Time Systems, 13(2): 107-126, September 1997.
- [Harbour et al 91] M. Gonzalez Harbour, M.H. Klein, J.P. Lehoczky *Fixed priority scheduling of periodic tasks with varying execution priority*. In: Proc. 12th IEEE Real-Time Systems Symposium, pp. 116–128, 1991.
- [Harter 84] P. Harter, *Response times in level-structured systems*, Department of Computer Science, University of Colorado, USA, Tech. Rep. CU-CS-269-84, 1984.
- [Joseph et al 86] M. Joseph and P. Pandya, *Finding Response Times in a Real-Time System*, The Computer Journal, 29(5): 390-395, 1986.

References - III

- [Klein et al 93] M.H. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour, *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*, Kluwer Academic Publishers (KAP), 1993.
- [Liu and Layland 73] C.L. Liu and J.W. Layland, *Scheduling Algorithms for Multiprogramming in a Real-Time Environment*, Journal of the ACM, 20(1): 46-61, January 1973.
- [Liu 00] J.W.S. Liu, *Real-Time Systems*, Prentice Hall, 2000.
- [Obenza 94] R. Obenza, *Guaranteeing real-time performance using RMA*, Embedded Systems Programming, pp. 26-40, 1994.
- [Redell et al 02] O. Redell and M. Sanfridson, *Exact best-case response time analysis of fixed priority scheduled tasks*, In: Proc. 14th Euromicro Conference on Real-Time Systems (ECRTS), pp. 165-172, June 2002.
- [Regehr 02] J. Regehr, *Scheduling tasks with mixed pre-emption relations for robustness to timing faults,* In: Proc. 23rd IEEE Real-Time Systems Symposium, pp 315–326, 2002.
- [Tindell et al 94] K.W. Tindell and A. Burns and A.J. Wellings, *An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks*, Real-Time Systems, 6(2): 133-151, 1994.