Real-Time Architectures 2003/2004

Response times for FPPS

Reinder J. Bril

Response times for FPPS

- Motivation for FPPS
- Leading example
- Worst-case response times
  - Recapitulation
  - Techniques
- Best-case response times
  - Introduction
  - Techniques
- Jitter analysis
  - Completion jitter
  - Release jitter
  - Completion and release jitter

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).

Leading example

- Task set Z consisting of 3 tasks:

Task	Period T_j	Execution time C_j	Utilization U_j
τ_1	10	3	0.3
τ_2	19	11	0.58
τ_3	56	5	0.09

Notes:

- RM priority assignment and $D_j = T_j$ (RMS);
- $U = 0.97 \leq 1$, hence Z *could* be schedulable;
- Utilization bound: $U(n) \leq LL(n) = n\,(2^{1/n} - 1)$:
  - $U_1 + U_2 = 0.88 > LL(2) \approx 0.83$,
  - therefore $U(3) > LL(3)$, hence another test required.

Worst-case response times for FPPS

- Recapitulation
  - Formalization
  - Critical instant
- Techniques
  - Time line
  - Calculation
    - Recursive equation
    - Iterative procedure

Recapitulation

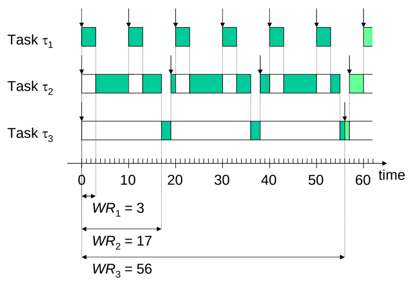
- Formalization:
  - Worst-case response time WR_j of a periodic task τ_j:

$$WR_j = \sup_{\varphi,i} R_{j,i}$$

  where φ is the phasing of the set Z.
- *Critical* instant:
  - Task τ_j is released simultaneously with all tasks with a higher priority.
  - The *highest* amount of preemption of a task is found *after* a simultaneous release of higher priority tasks.
  - General for a set of tasks.

Techniques

- **Time line**:



Techniques

- **Calculation** (recapitulation):
  – Recursive equation for task τ_j:

$$x = C_j + \sum_{i<j} \left\lceil \frac{x}{T_i} \right\rceil C_i$$

  WR_j is the *smallest* positive solution

  – Assume a task τ_i with a higher priority than τ_j;
    - $\lceil x/T_i \rceil$ denotes the *maximal* number of preemptions of task τ_j in an interval $[0, x)$ by task τ_i;
    - $\lceil x/T_i \rceil C_i$ denotes the *maximal* preemption time of task τ_j in an interval $[0, x)$ by task τ_i.

Techniques

- **Calculation** (recapitulation):
  – Iterative procedure:

$$WR_j^{(0)} = C_j + \sum_{i<j} C_i$$

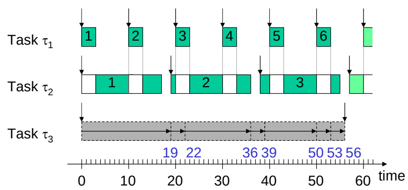
$$WR_j^{(k+1)} = C_j + \sum_{i<j} \left\lceil \frac{WR_j^{(k)}}{T_i} \right\rceil C_i$$

  – Stopped when either:
    - the same value is found for two successive iterations; or
    - the deadline D_j is exceeded (hence, not schedulable).

  – All intermediate values are *at most* equal to WR_j;
  – Terminates when $U_{j-1} \leq 1$;
  – Z is schedulable when $WR_j \leq D_j$ for all tasks in Z.

Techniques

- **Calculation** (visualization):



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

Exercise

- Task set Z' consisting of 3 tasks:

Task	Period T_j	Execution time C_j	Utilization U_j
τ_1	5	2	0.4
τ_2	7	3	0.43
τ_3	29	3	0.10

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

Best-case response times for FPPS

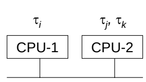
- Introduction
  - Motivation
  - Formalization
  - Optimal instant
- Techniques
  - Time line
  - Calculation
    - Recursive equation
    - Iterative procedure

Introduction

- Motivation
  - Best-case and worst-case notions are duals:
    - Best-case: earliest, shortest, or minimal;
    - Worst-case: latest, longest, or maximal;
  - Needed for jitter analysis:
    - Typically for multi-processor systems, but also applies to single-processor systems;
    - Completion (or end) jitter *EJ*: variation in end times;
    - Release (or activation) jitter *AJ*: variation in release times (e.g. output of one task triggers a next task);
    - Example: multimedia in a networked environment.

Introduction

- Example (continued):



τ_i triggers τ_j
EJ_i causes AJ_j, which
influences response times
of *both* τ_j and τ_k

- Goal jitter analysis:
  - Determine schedulability in the context of jitter,
  - hence, response times.

Introduction

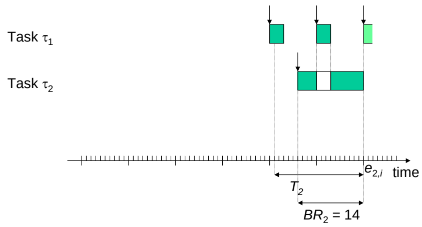
- Formalization:
  - Best-case response time BR_j of a periodic task τ_j:

$$BR_j = \inf_{\varphi, i} R_{j,i}$$

  where φ is the phasing of the set *Z*.
- *Optimal* instant:
  - Job $\tau_{j,i}$ *ends* simultaneously with the *release* of all tasks with a higher priority, and $\tau_{j,i}$'s release time is equal to its start time.
  - The *lowest* amount of preemption of a task is found *before* a simultaneous release of higher priority tasks.
  - Specific for each task !

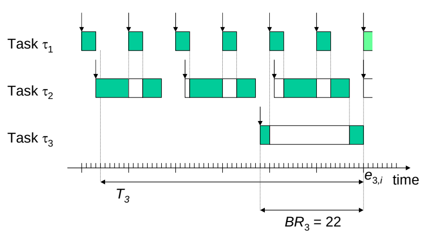
Techniques

- **Time line:** optimal instant for τ_2



Techniques

- **Time line:** optimal instant for τ_3



Techniques

- **Calculation**:
  - Recursive equation for task τ_j:

$$x = C_j + \sum_{i<j} \left(\left\lceil \frac{x}{T_i} \right\rceil - 1 \right) C_i$$

BR_j is the *largest* positive solution

  - Assume a task τ_i with a higher priority than τ_j;
    - $\lceil x/T_i \rceil - 1$ denotes the *minimal* number of preemptions of task τ_j in an interval $(0, x)$ by task τ_i;
    - $(\lceil x/T_i \rceil - 1)C_i$ denotes the *minimal* preemption time of task τ_j in an interval $(0, x)$ by task τ_i.

Techniques

- **Calculation**:
  - Iterative procedure:

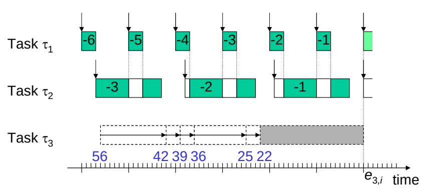
$$BR_j^{(0)} = WR_j$$

$$BR_j^{(k+1)} = C_j + \sum_{i<j} \left(\left\lceil \frac{BR_j^{(k)}}{T_i} \right\rceil - 1 \right) C_i$$

  - Stopped when:
    - the same value is found for two successive iterations.
  - All intermediate values are *at least* equal to BR_j;

Techniques

- **Calculation** (visualization)**:**



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

Exercise

- Task set Z' consisting of 3 tasks:

Task	Period T_j	Execution time C_j	Utilization U_j
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_j for all three tasks of Z'.
  - 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.

Jitter analysis

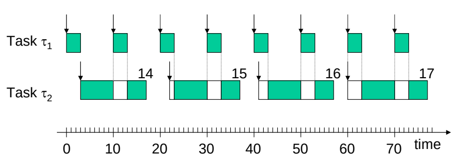
- Types of jitter (recap):
  – Completion jitter: variation in end times;
  – Release jitter: variation in release times (e.g. output of one task triggers a next task)
- Completion (or end) jitter EJ_j of a task τ_j:

$$EJ_j = \sup_{\varphi,i,k}(R_{j,i} - R_{j,k}) \leq WR_j - BR_j$$

  – "≤" because WR_j and BR_j are not necessarily assumed for the same phasing

Completion (or end) jitter

- Example (leading):
  – $WR_2 = 17$, $BR_2 = 14$;
  – WR_2 and BR_2 both assumed for a single phasing,
  – hence, $EJ_2 = WR_2 - BR_2 = 3$,
  – and the bound for EJ_2 is therefore tight.



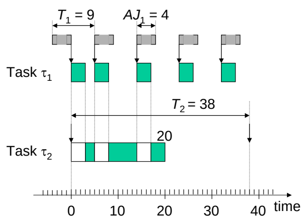
Release (or activation) jitter

- Worst-case response times:
  – Critical instant revisited:
    - Task τ_j 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* preemption of τ_j occurs.

Release (or activation) jitter

- New example:

Task	T_j	C_j	AJ_j
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.

Reinder J. Bril, r.j.bril@tue.nl
TU/e Informatica, System Architecture and Networking

Release (or activation) jitter

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

$$x = C_j + \sum_{i<j} \left\lceil \frac{x + AJ_i}{T_i} \right\rceil C_i$$

    - *Where* AJ_i is the activation jitter of τ_i.
    - WR_j is the *smallest* positive solution of the equation.
  - Iterative procedure:
    - Similar to the case without jitter.

Reinder J. Bril, r.j.bril@tue.nl
TU/e Informatica, System Architecture and Networking

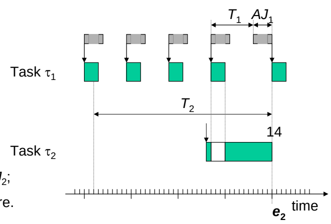
Release (or activation) jitter

- Best-case response times:
  - Optimal instant revisited:
    - Job $\tau_{j,i}$ *ends* simultaneously with the *release* of all tasks with a higher priority, and $\tau_{j,i}$'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* preemption of τ_j occurs.

Release (or activation) jitter

- Same (new) example:

Task	T_j	C_j	AJ_j
τ_1	9	3	4
τ_2	38	11	7



Notes:

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

Release (or activation) jitter

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

$$x = C_j + \sum_{i<j} \left(\left\lceil \frac{x - AJ_i}{T_i} \right\rceil - 1 \right)^+ C_i$$

    - *Where* AJ_i is the activation jitter of τ_i, and $w^+ = \max\{w,0\}$.
    - BR_j is the *largest* positive solution of the equation.
  - Iterative procedure:
    - Similar to the case without jitter.

Completion and release jitter

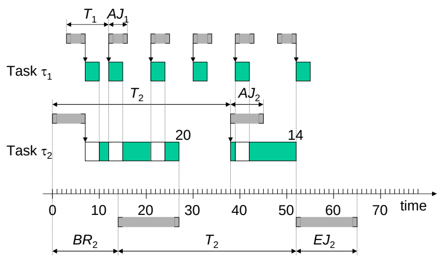
- Completion jitter including release jitter:

$$EJ_j \leq AJ_j + WR_j - BR_j$$

- This bound is also tight; see next slide.

Completion and release jitter

- Example:



References

- [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.
- [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.
- [Obenza 94] R. Obenza, *Guaranteeing real-time performance using RMA*, Embedded Systems Programming, pp. 26-40, 1994.