THE UNIVERSITY of York

RTS York

Quantifying the sub-optimality of uniprocessor fixed priority pre-emptive scheduling

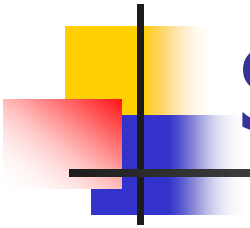
Robert Davis[1], Thomas Rothvoß[2], Sanjoy Baruah[3], Alan Burns[4]

[1]Real-Time Systems Research Group, University of York

[2]Institute of Mathematics, Ecole Polytechnique Federale de Lausanne

[3]Dept. of Computer Science, University of North Carolina

[4]Real-Time Systems Research Group, University of York



Speedup factor

- **QUESTION:**

  What is the **speedup factor** by which the processing speed of a single processor would need to be increased, so that any taskset that was previously schedulable according to an optimal scheduling algorithm (i.e. any feasible taskset), can be scheduled using fixed priority pre-emptive scheduling, assuming optimal priority assignment?

Problem scope

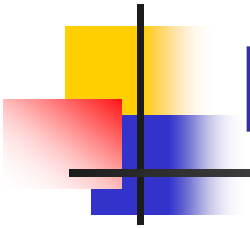
- **Single processor systems**
  - Pre-emptive scheduling
  - Execution time of all tasks scales linearly with processor speed
- **Sporadic task model**
  - Static set of *n* tasks τ_i with priorities 1..*n*
  - Bounded worst-case execution time C_i
  - Sporadic/periodic arrivals: minimum inter-arrival time T_i
  - Relative deadline D_i
  - Utilisation $U_i = C_i / T_i$
  - Independent execution

Outline of presentation

- **Different speedup factors for different classes of taskset**
  - Implicit-deadline tasksets ($D_i=T_i$) [1]
  - Constrained-deadline tasksets ($D_i \leq T_i$) [1]
  - Arbitrary-deadline tasksets ($D_i \leq T_i$, $D_i > T_i$) [2]

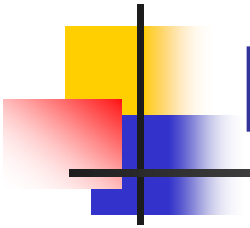
[1] R.I. Davis, T. Rothvoß, S.K. Baruah, A. Burns "Exact Quantification of the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling". *Real-Time Systems*, Volume 43, Number 3, pages 211-258, November 2009. (Published online 17th July 2009).

[2] R.I. Davis, T. Rothvoß, S.K. Baruah, A. Burns "Quantifying the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling for Sporadic Tasksets with Arbitrary Deadlines". *RTNS'09*, October 26-27th, 2009.



Background

- **Feasibility and Optimality**
  - A taskset is said to be feasible if there exists some scheduling algorithm that can schedule the taskset without missing a deadline
  - A scheduling algorithm is said to be optimal if it can schedule all feasible tasksets
- **EDF is optimal**
  - Dertouzos (1974), proved that EDF is an optimal uniprocessor pre-emptive scheduling algorithm for arbitrary-deadline tasksets that comply with the sporadic task model
  - EDF can schedule all feasible tasksets that comply with our model
  - So we can use a comparison with EDF to determine the speedup factor for fixed priority pre-emptive scheduling



Background

- **FP scheduling: Optimal Priority Assignment**
  - A priority assignment policy Q is said to be optimal if there are no tasksets that are schedulable using some other priority assignment policy P which are not also schedulable using policy Q.

- **Optimal priority assignment policies**
  - Implicit-deadline tasksets – Rate-Monotonic (Liu & Layland, 1973)
  - Constrained-deadline tasksets – Deadline Monotonic (Leung & Whitehead, 1982)
  - Arbitrary-deadline tasksets – Optimal Priority Assignment algorithm, (Audsley, 1993)

Speedup factor

- **Two perspectives and definitions**
  - **#1 Speedup factor** is the maximum factor by which it is necessary to increase the processor speed so that any taskset that was schedulable under EDF becomes schedule under FP.
  - **#2 Speedup factor** is the maximum factor by which the execution times of a set of tasks, that are only just schedulable under FP can be increased and the taskset remain schedulable under EDF.
  - A taskset is said to be **speedup-optimal** if it exhibits the (maximum) speedup factor.

Speedup-optimal tasksets

- **Speedup optimal tasksets are key to finding speedup factors**
- **Properties of speedup-optimal tasksets for implicit-deadline and constrained-deadline cases [1]**

  [1] R.I. Davis, T. Rothvoß, S.K. Baruah, A. Burns "Exact Quantification of the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling". *Real-Time Systems*, Volume 43, Number 3, pages 211-258, November 2009. (Published online 17th July 2009).

  - Lemma 1: τ_n must be a constraining task, with the longest deadline and the lowest priority.
  - Lemma 2: τ_n must have the longest possible period (infinite in the constrained-deadline case).
  - Lemma 3: D_n must be the start of an idle period.
  - Lemma 4: All tasks $\tau_i \neq \tau_n$ must have $D_i < T_n$
  - Lemma 5: All tasks $\tau_i \neq \tau_n$ must have $D_i = T_i$

Speedup-optimal tasksets

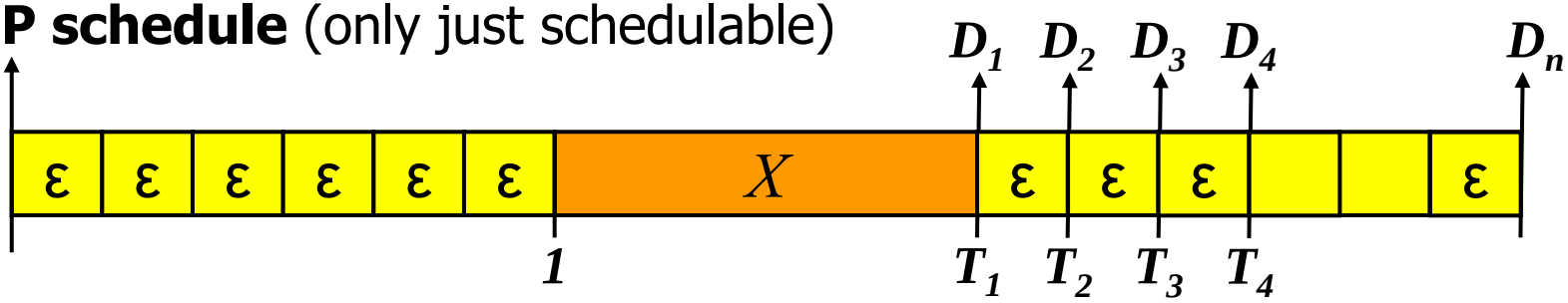
- **Properties of speedup-optimal tasksets for implicit-deadline and constrained-deadline cases**
  - Lemma 6: All tasks $\tau_i \neq \tau_n$ must have $T_i > D_n/2$
  - Lemma 7: Following a critical instant, τ_n executes continuously from when it first starts execution until it completes.
  - Lemma 8: The task parameters must comply with the following equation

$$\forall i \neq n \quad D_i = T_i = \sum_{\forall j} C_j + \sum_{\forall j \in hp(i)} C_j$$

  - Lemma 9: High priority task execution time is divided into an infinite number of tasks each with an infinitesimal execution time.

Normalised speedup-optimal taskset

FP schedule (only just schedulable)



- **Speedup-optimal taskset *V***
  - Limit as $n \to \infty$ of:

    $\forall i \neq n \quad D_i = T_i = 1 + X + (i-1)/(n-1)$

    $C_i = 1/(n-1)$

    $C_n = X \qquad D_n = 2 + X$
  - X is as yet an unknown value
  - Implicit-deadline case: $T_n = D_n$
  - Constrained-deadline case: $T_n = \infty$

Normalised speedup-optimal taskset

- **Utilisation of high priority tasks:**

$$U^V = \lim_{n-1\to\infty}\left(\frac{1}{(n-1)}\sum_{i=1}^{n-1}\frac{1}{(1+X+(i-1)/(n-1))}\right)$$

  - Substituting $k=n-1$

$$U^V = \lim_{k\to\infty}\left(\frac{1}{k}\sum_{i=1}^{k}\frac{1}{(1+X+(i-1)/k)}\right)$$

  - This is a left Riemann sum of $y=1/z$ over the partition $[(1+X), (2+X)]$
  - Limit as $k\to\infty$ is given by the integral:

$$U^V = \int_{1+X}^{2+X}\frac{1}{z}dz = \ln\left(\frac{2+X}{1+X}\right)$$

- **Utilisation of lowest priority task:**
  - Constrained-deadline case: $U_n=0$
  - Implicit-deadline case: $U_n = \frac{X}{2+X}$

Speedup factor: Implicit-deadline tasksets

- **Total utilisation of speedup-optimal taskset**

$$U = \left(\ln\left(\frac{2+X}{1+X} \right) + \frac{X}{2+X} \right)$$

- **Exact EDF schedulability test for implicit-deadline tasksets:**

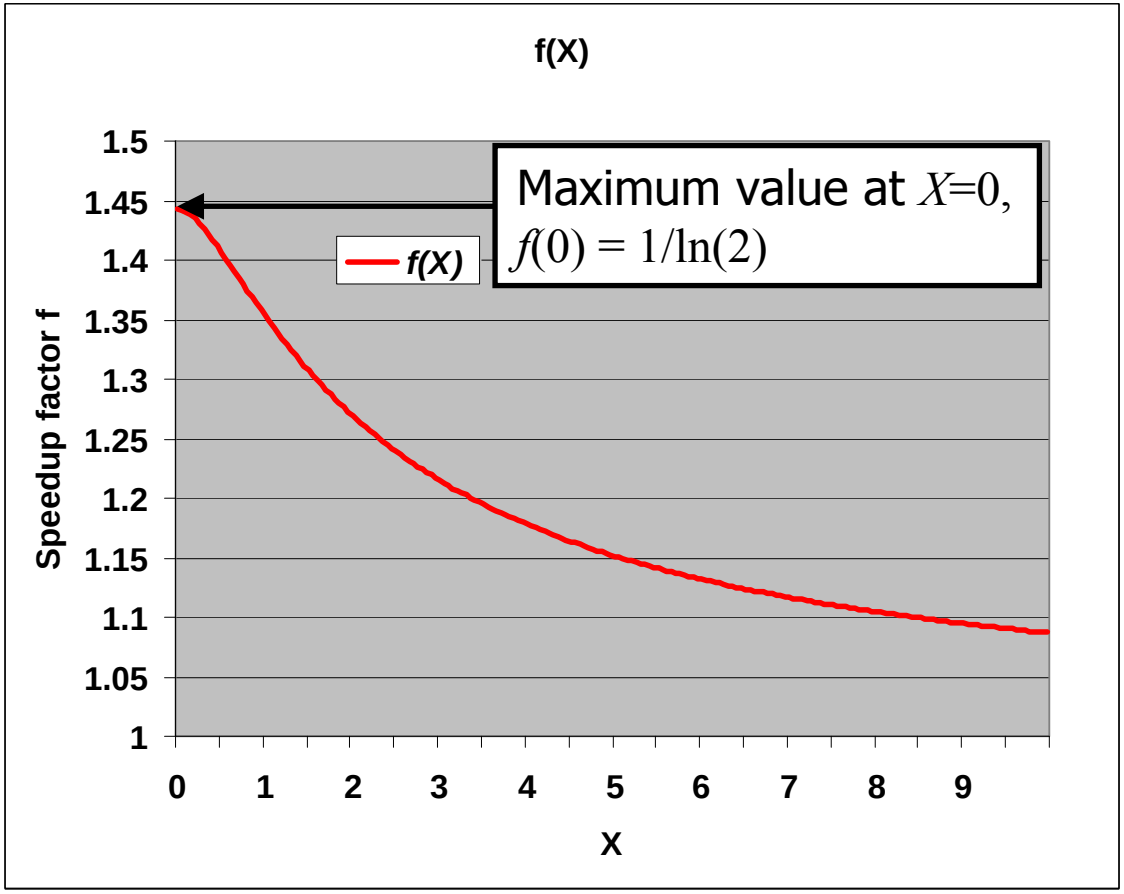
$$U \leq 1$$

- **(Maximum) speedup factor as a function of X**

$$f(X) = \frac{1}{\left(\ln\left(\frac{2+X}{1+X} \right) + \frac{X}{2+X} \right)}$$

  - $f(X)$ is a monotonic non-increasing function of X

Speedup factor: Implicit-deadline tasksets

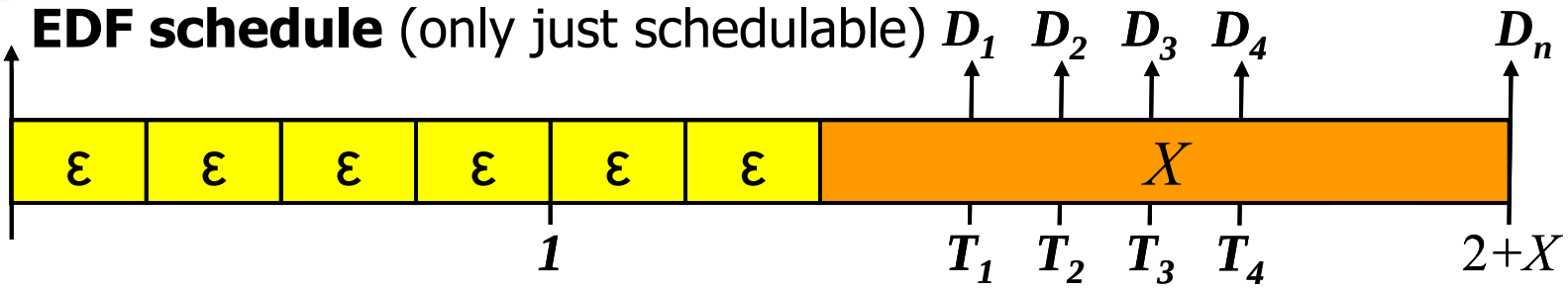


Exact speedup factor for FP scheduling of implicit deadline tasksets is 1/ln(2) ≈ 1.44270

As EDF can schedule any taskset with $U \leq 1$ Speedup factor implies FP can schedule any taskset with $U \leq 1/\ln(2)$

In agreement with and diverse proof of Liu & Layland's seminal result from 1973

Speedup factor: Constrained-deadline tasksets



- **Constraints on EDF schedulability when scaled by a factor of *f***

  (i) Lowest priority task τ_n and one invocation of each higher priority task i.e. $f(1+X)$ must complete by $2+X$:

$$f1(X) \le \frac{2+X}{1+X}$$

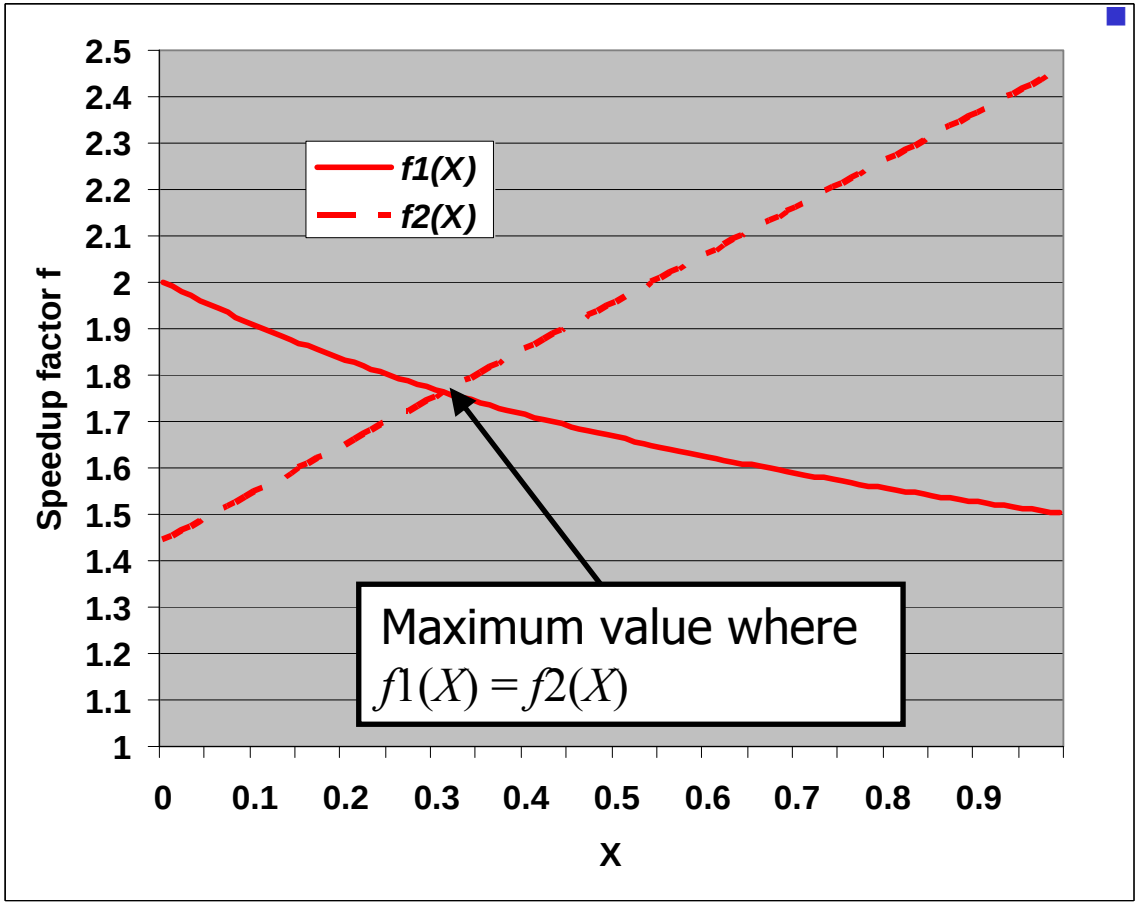
$f1(X)$ Monotonically non-increasing with $f1(0) = 2$

  (ii) The total utilisation must not exceed 1:

$$f2(X) \le \frac{1}{\ln\left(\frac{2+X}{1+X}\right)}$$

$f2(X)$ Monotonically non-decreasing with $f1(0) = \sqrt{2} \approx 1.4142$

Speedup factor: Constrained-deadline tasksets



Intersection of the lines

$$\left(\frac{2+X}{1+X}\right) = \frac{1}{\ln\left(\frac{2+X}{1+X}\right)}$$

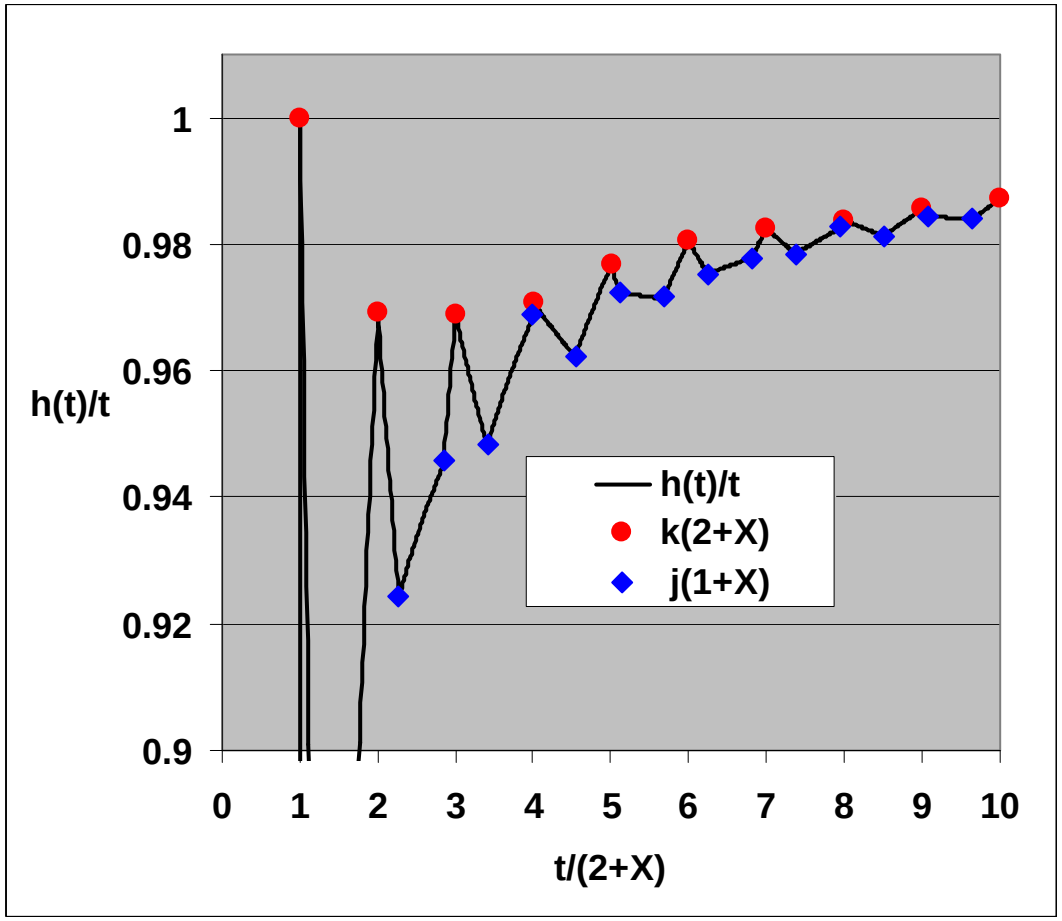
Speedup factor: Constrained-deadline tasksets

- **Maximum speedup factor** $$f = \left(\frac{2+X}{1+X}\right) = \frac{1}{\ln\left(\frac{2+X}{1+X}\right)}$$
- **Can be written as:** $$\frac{1}{f} = \ln\left(\frac{1}{\left(\frac{1+X}{2+X}\right)}\right) = \left(\frac{1+X}{2+X}\right)$$
- **Similar to the transcendental equation:** $\ln(1/\Omega) = \Omega$
  defining the mathematical constant $\Omega \approx 0.567143$
- **Upper bound on speedup factor is $1/\Omega \approx 1.76322$**
  - Note upper bound as constraints used are necessary for EDF schedulability, but not sufficient

Speedup factor: Constrained-deadline tasksets

- **Exact speedup factor**
- **Need to prove that the speedup-optimal taskset scaled by a factor of $1/\Omega \approx 1.76322$ is schedulable under EDF**
  - Rather elegant, but lengthy proof in the RTS paper
  - Uses Exact schedulability analysis for EDF (Baruah 1990)
    - Processor demand bound $$h(t) = \sum_{i=1}^{n} \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) C_i$$
    - Processor LOAD = $\max_{\forall t} \left(\frac{h(t)}{t} \right) \leq 1$

Speedup factor: Constrained-deadline tasksets



Proof:

- Represents $h(t)$ by an infinite series of piecewise linear functions
- Shows that maxima in processor load occur at one end of these functions $t = k(2+X)$ and minima at the other end $t = j(1+X)$
- Shows that maxima are non-decreasing for $k \geq 6$, tend to 1 as $k \rightarrow \infty$, and are ≤ 1 for $k \leq 6 \Rightarrow h(t)/t \leq 1$

Exact speedup factor is $1/\Omega \approx 1.76322$ for constrained-deadline tasksets

Conclusions

- **Liu & Layland (1973):**
  - Characterised the maximum performance penalty (or sub-optimality) of using FP scheduling rather than an optimal algorithm for implicit-deadline tasksets
- **This research**
  - Characterises a similar maximum performance penalty (or sub-optimality) based on processor LOAD for the constrained-deadline case
  - Also provides a disparate proof of the Liu and Layland result
- **Other work**
  - Upper and lower bounds on the speedup factor for

    Arbitrary-deadline tasksets

    Non-pre-emptive scheduling

Speedup factor: Summary

Speedup factor: increase in processing speed required so that any feasible taskset (schedulable by an optimal algorithm) can be scheduled using Fixed Priority scheduling

Taskset Constraints [Priority ordering]	**FP-P Speedup factor** Lower bound	Upper bound	**FP-NP Speedup factor** Lower bound	Upper bound
Implicit-deadline [RM] [OPA]	$1/\ln(2) \approx 1.44269$		$1/\Omega \approx 1.76322$	2
Constrained-deadline [DM] [OPA]	$1/\Omega \approx 1.76322$		$1/\Omega \approx 1.76322$	2
Arbitrary-deadline [OPA] [OPA]	$1/\Omega \approx 1.76322$	2	$1/\Omega \approx 1.76322$	2

Future work / open questions

- **Determining exact speedup factors for FP-NP, and for FP-P with arbitrary deadline tasksets**
  - These are where optimal priority assignment requires Audsley's OPA algorithm – complicates proof of speedup optimal taskset attributes
- **Determining the exact speedup factor as a function of the number of tasks**
- **Empirical investigation**
  - Try to find tasksets that require a speedup factor > $1/\Omega$
  - Is $1/\Omega$ ultimately the limit ???

Questions

[1] R.I. Davis, T. Rothvoß, S.K. Baruah, A. Burns "Exact Quantification of the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling". *Real-Time Systems*, Volume 43, Number 3, pages 211-258, November 2009. (Published online 17th July 2009).

Gives the Exact speedup factor for implicit- and constrained-deadline tasksets (pre-emptive scheduling)

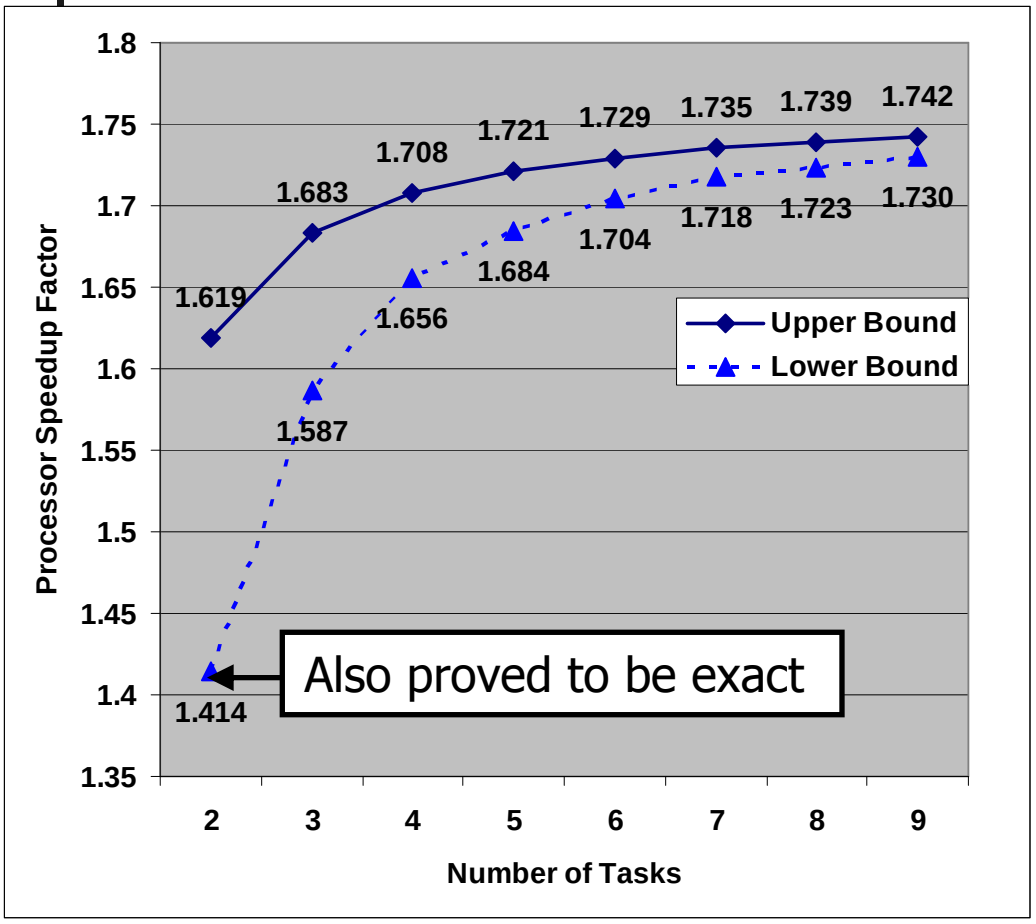
[2] R.I. Davis, T. Rothvoß, S.K. Baruah, A. Burns "Quantifying the Sub-optimality of Uniprocessor Fixed Priority Pre-emptive Scheduling for Sporadic Tasksets with Arbitrary Deadlines". *RTNS'09*, October 26-27th, 2009.

Gives upper and lower bounds for arbitrary-deadline tasksets (pre-emptive scheduling)

[3] R.I. Davis, L. George, P. Courbin "Quantifying the Sub-optimality of Uniprocessor Fixed Priority Non-pre-emptive Scheduling". *RTNS'10*, November 4-5th, 2010.

Gives upper and lower bounds for implicit, constrained, and arbitrary-deadline tasksets (non-pre-emptive scheduling)

Speedup factor: As a function of cardinality



For constrained-deadline tasksets

Upper bound

Based on proof using Hyperbolic Bound (Bini et al. 2003)

Lower bound

Based on generation of tasksets requiring these speedup factors

Note improvement over value for arbitrary n of $1/\Omega \approx 1.76322$