Improved Schedulability Tests for Global Fixed-Priority Scheduling

Risat Mahmud Pathan and Jan Jonsson

Chalmers University of Technology, Sweden

Outline

Introduction

- Multiprocessors, specifically CMPs, are considered for many embedded real-time systems (e.g., automotive)
- The application of real-time systems are often modeled as a collection of recurrent tasks (e.g., control applications)
- Hard real-time systems must meet all the deadlines of its application tasks during runtime
- Problem: How can we guarantee that all the tasks deadlines are met on m identical processors?

Task Model

- We consider a set of **sporadic** real-time task set

$$\Gamma = \{\tau_1, \tau_2, \ldots \tau_n\}$$

- Each task τ_i has three parameters (C_i, D_i, T_i)
  - ***Implicit-deadline*** if $D_i = T_i$
  - ***Constrained-deadline*** if $D_i \leq T_i$
  - ***Total utilization*** $U = \sum u_i = \sum \frac{C_i}{T_i}$
  - ***Total dendity*** $\delta = \sum \lambda_i = \sum \frac{C_i}{D_i}$

- Tasks are given fixed priorities
- Tasks are scheduled on *m* identical processors

Partitioned and Global Scheduling

- Partitioned Scheduling: task can execute in exactly one processor to which it is assigned
- Global Scheduling: task can execute on any processor even when resumed after preemption

Global Fixed-Priority Preemptive Scheduling

The highest priority ready task is always dispatched by preempting, if any, the execution of a low priority task

Two problems

Priority Assignment

How to assign the fixed priorities for a given task set?

Schedulability Test

How to guarantee the schedulability of a given task set?

Our Contributions

Priority Assignment and Utilization Bound Test

Proposed new fixed-priority assignment policy, called `ISM–US`, and derived the schedulability utilization bound

Priority Assignment and Iterative Test

Proposed improved fixed-priority assignment policy for two state-of-the-art iterative schedulability tests

Utilization Bound Test

Priority Assignment Policy ISM-US

- Implicit-deadline sporadic task systems
  - is also applicable for constrained-deadline

Hybrid (Slack-Monotonic) Priority Assignment (HPA)

A subset of the tasks are given slack-monotonic priority and the other tasks are given the highest fixed-priority

Slack-Monotonic (SM)

Task τ_i has higher SM priority than task τ_k if and only if $(T_i - C_i < T_k - C_k)$

Priority Assignment Policy ISM-US

Policy ISM-US

If $u_i > u_{ts}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given slack-monotonic priority

Threshold Utilization

$$u_{ts} = \frac{3m - 2 - \sqrt{5m^2 - 8m + 4}}{2m - 2}$$

Priority Assignment Policy ISM-US

Policy ISM-US

If $u_i > u_{ts}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given slack-monotonic priority

Threshold Utilization

$$u_{ts} = \frac{3m - 2 - \sqrt{5m^2 - 8m + 4}}{2m - 2}$$

Theorem (Utilization Bound)

If $\mathbf{U \leq m \cdot \min\{0.5, u_{ts}\}}$, then all the deadlines of task set Γ are met using global FP scheduling

State-of-the-art utilization bound

RM-US[$\frac{1}{3}$] M. Bertogna et. al., OPODIS 2005

If $u_i > \frac{1}{3}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given *rate-monotonic* priority

Utilization Bound: $\frac{m+1}{3}$

State-of-the-art utilization bound

RM-US[$\frac{1}{3}$] M. Bertogna et. al., OPODIS 2005

If $u_i > \frac{1}{3}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given *rate-monotonic* priority

Utilization Bound: $\frac{m+1}{3}$

SM-US[$\frac{2}{3+\sqrt{5}}$] B. Andersson, OPODIS 2008

If $u_i > \frac{2}{3+\sqrt{5}}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given *slack-monotonic* priority

Utilization Bound: $\frac{2m}{3+\sqrt{5}}$

State-of-the-art utilization bound

RM-US[$\frac{1}{3}$] — M. Bertogna et. al., OPODIS 2005

If $u_i > \frac{1}{3}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given *rate-monotonic* priority

Utilization Bound: $\frac{m+1}{3}$

SM-US[$\frac{2}{3+\sqrt{5}}$] — B. Andersson, OPODIS 2008

If $u_i > \frac{2}{3+\sqrt{5}}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given *slack-monotonic* priority

Utilization Bound: $\frac{2m}{3+\sqrt{5}}$

State-of-the-art Utilization Bound

- **If $m \leq 6$, then RM–US[$\frac{1}{3}$] is the best**
- **If $m > 6$, then SM–US[$\frac{2}{3+\sqrt{5}}$] is the best**

Comparison with our bound

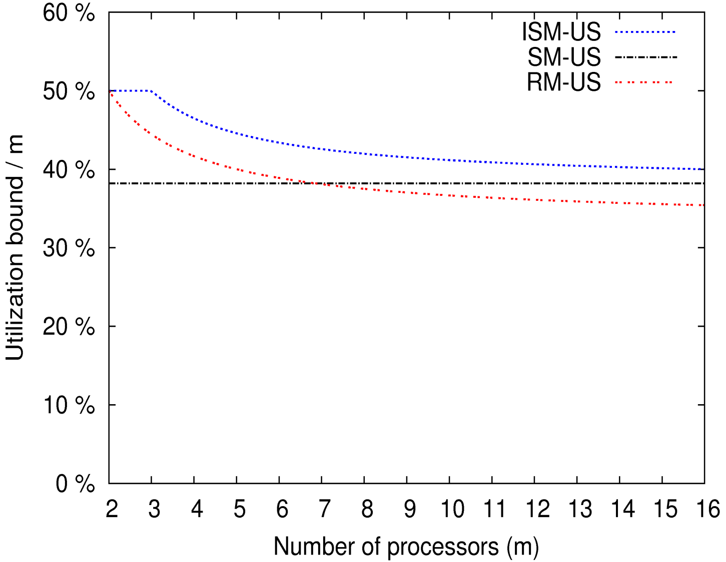


Figure: Utilization bounds of RM−US[$\frac{1}{3}$], SM−US[$\frac{2}{3+\sqrt{5}}$] and proposed ISM−US

HPA policy and Global Scheduling

- **Predictability [Ha and Liu, ICDCS 1994]:** If task τ_i is schedulable with WCET T_i, then τ_i is also schedulable with WCET C_i.

HPA policy and Global Scheduling

- **Predictability [Ha and Liu, ICDCS 1994]:** If task τ_i is schedulable with WCET T_i, then τ_i is also schedulable with WCET C_i.

Separation of Concern

- During schedulability analysis, each highest priority task τ_i's WCET is set to T_i and one processor is (virtually) dedicated to τ_i ***without any concern***.
- The problem now ***reduces*** to the schedulability of the other (lower) priority tasks on $(m - m')$ processors (m' is the number of ***heavy*** tasks)

Iterative Schedulability Test

Iterative Schedulability test

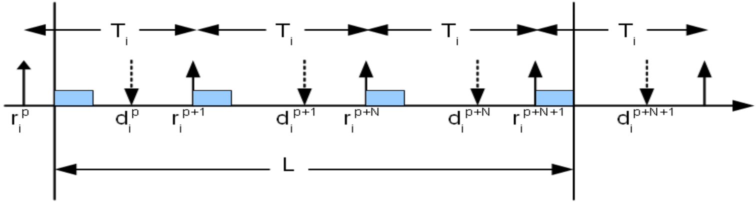
- We consider ***constrained-deadline*** sporadic task systems
- We propose an improved fixed-priority assignment policy for two state-of-the-art ***iterative tests***:
  - the `DA-LC` test proposed by Davis et al. (RTSJ, 2011)
  - the `RTA-LC` test proposed by Guan et al. (RTSS, 2009).
- **Iterative Test:** one schedulability condition is tested for each of the lower priority tasks

Interference and Workload

When considering the schedulability of a lower priority task τ_k within the ***problem window***, both `RTA-LC` and `DA-LC` tests consider

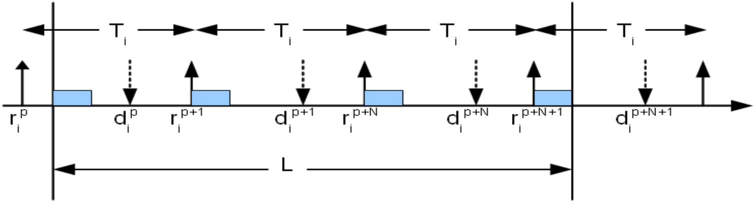
- the ***interference*** of each higher priority task $\tau_i \in hp(k)$
- based on the ***workload*** of each higher priority task τ_i in set $hp(k)$
- where each higher priority task τ_i is considered either a ***carry-in*** or a ***non carry-in*** task

Carry-in and Non Carry-in Interference

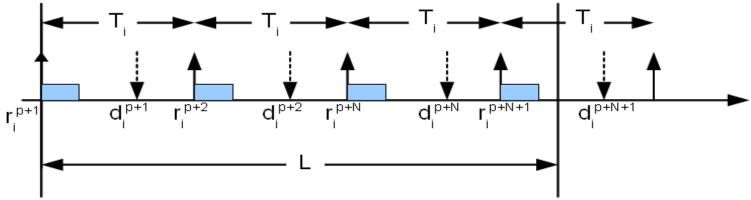


$I_i^C(L, C_k)$ = **carry-in interference** of task τ_i on τ_k

Carry-in and Non Carry-in Interference



$I_i^C(L, C_k) =$ **carry-in interference** of task τ_i on τ_k



$I_i^{NC}(L, C_k) =$ **non carry-in interference** of task τ_i on τ_k

DA−LC and RTA−LC test

- The DA−LC test (Davis et al. RTSJ 2011) for task τ_k is given as follows:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

DA−LC and RTA−LC test

- The DA−LC test (Davis et al. RTSJ 2011) for task τ_k is given as follows:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

- The RTA−LC test (Guan et al. RTSS 2009) for task τ_k is given as follows:

$$R_k^{t+1} \leftarrow C_k + \left\lfloor \frac{I_k(R_k^t)}{m} \right\rfloor$$

DA−LC and RTA−LC test

- The DA−LC test (Davis et al. RTSJ 2011) for task τ_k is given as follows:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

- The RTA−LC test (Guan et al. RTSS 2009) for task τ_k is given as follows:

$$R_k^{t+1} \leftarrow C_k + \left\lfloor \frac{I_k(R_k^t)}{m} \right\rfloor$$

- The function $I_k(L)$ is calculated as follows:

$$I_k(L) = \sum_{i \in hp(k)} I_i^{NC}(L, C_k) + \sum_{i \in Max(k, m-1)} I_i^{DIFF}(L, C_k)$$

DA−LC and RTA−LC test

- The DA−LC test (Davis et al. RTSJ 2011) for task τ_k is given as follows:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

- The RTA−LC test (Guan et al. RTSS 2009) for task τ_k is given as follows:

$$R_k^{t+1} \leftarrow C_k + \left\lfloor \frac{I_k(R_k^t)}{m} \right\rfloor$$

- The function $I_k(L)$ is calculated as follows:

$$I_k(L) = \sum_{i \in hp(k)} I_i^{NC}(L, C_k) + \sum_{i \in Max(k,m-1)} I_i^{DIFF}(L, C_k)$$

- where
  - $Max(k, m-1)$ is the set of $(m-1)$ higher priority tasks in $hp(k)$ that have the largest value of $I_i^{DIFF}(L, C_k)$, and

DA−LC and RTA−LC test

- The DA−LC test (Davis et al. RTSJ 2011) for task τ_k is given as follows:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

- The RTA−LC test (Guan et al. RTSS 2009) for task τ_k is given as follows:

$$R_k^{t+1} \leftarrow C_k + \left\lfloor \frac{I_k(R_k^t)}{m} \right\rfloor$$

- The function $I_k(L)$ is calculated as follows:

$$I_k(L) = \sum_{i \in hp(k)} I_i^{NC}(L, C_k) + \sum_{i \in Max(k,m-1)} I_i^{DIFF}(L, C_k)$$

- where
  - $Max(k, m-1)$ is the set of $(m-1)$ higher priority tasks in $hp(k)$ that have the largest value of $I_i^{DIFF}(L, C_k)$, and
  - $I_i^{DIFF}(L, C_k) = I_i^{C}(L, C_k) - I_i^{NC}(L, C_k)$

RTA−LC and DA−LC test

R. Davis and A. Burns (RTSJ, 2011) have showed that

- For a given fixed-priority ordering, the RTA−LC test dominates the DA−LC test
- Audsley's Optimal Priority Assignment(OPA) algorithm is applicable to the DA−LC test but not to the RTA−LC test
- Empirically shown that DA−LC+OPA outperforms RTA−LC test

OPA+DA−LC is the state-of-the-art iterative schedulability test

Audsley's OPA for multiprocessors (RTSS, 2009)

Algorithm OPA (Taskset A, number of processors $\hat{m}$, Test S)

for each priority level k, lowest first
    for each unassigned task $\tau \in A$
        If τ is schedulable using S on $\hat{m}$ processors at priority k
            assign τ to priority k
            break (continue outer loop)
    return "unschedulable"
return "schedulable"

OPA+DA-LC (RTSJ, 2011)

Call OPA (Γ, m, DA-LC)

Interesting Observation

- `OPA +DA-LC` is proved optimal (RTSJ, 2011).

Interesting Observation

- OPA +DA−LC is proved optimal (RTSJ, 2011).
- This combination is optimal only under the assumption that it is applied to the entire task set and to all processors
  - i.e.,Call OPA(Γ, m, DA−LC)

Interesting Observation

- OPA +DA−LC is proved optimal (RTSJ, 2011).
- This combination is optimal only under the assumption that it is applied to the entire task set and to all processors
  - i.e.,Call OPA(Γ, m, DA−LC)

Scope for Improvement?

- Is it possible to obtain a more effective priority assignment if
  - OPA+DA−LC is applied to a **subset** of the entire task set and on a **lower** number of processors
  - while other tasks are assigned the highest priorities based on HPA and predictability?

Interesting Observation

- Recall the `DA-LC` test for task τ_k:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

- $I_k(L)$ depends on $(m-1)$ carry-in terms

$$I_k(L) = \sum_{i \in hp(k)} I_i^{NC}(L, C_k) + \sum_{i \in Max(k, m-1)} I_i^{DIFF}(L, C_k)$$

Interesting Observation

- Recall the DA−LC test for task τ_k:

$$D_k \geq C_k + \left\lfloor \frac{I_k(D_k)}{m} \right\rfloor$$

- $I_k(L)$ depends on $(m-1)$ carry-in terms

$$I_k(L) = \sum_{i \in hp(k)} I_i^{NC}(L, C_k) + \sum_{i \in Max(k,m-1)} I_i^{DIFF}(L, C_k)$$

Observation

- If we remove one task, say τ_h, from $hp(k)$ and
- reduce the number of processors from m to $(m-1)$, and
- apply the OPA+DA−LC test on $(\Gamma - \tau_h)$ and on $(m-1)$ processors,
- then $I_k(D_k)$ depends on $(m-2)$ carry-in tasks in $(hp(k) - \{\tau_h\})$

Example

- Consdier $\Gamma = \{\tau_1, \dots \tau_4\}$ and $m = 3$
- $(C_i, D_i, T_i) = \{(23, 33, 33), (106, 210, 214), (58, 216, 217), (46, 60, 64)\}$
- **`OPA` (Γ, $m = 3$, `DA-LC`) returns "unschedulable"**
- $I_k(D_k)$ considers $(m - 1) = 2$ as carry-in task

Example

- Consdier $\Gamma = \{\tau_1, \ldots \tau_4\}$ and $m = 3$
- $(C_i, D_i, T_i) = \{(23, 33, 33), (106, 210, 214), (58, 216, 217), (46, 60, 64)\}$
- **`OPA` (Γ, $m = 3$, `DA-LC`) returns "unschedulable"**
- $I_k(D_k)$ considers $(m-1) = 2$ as carry-in task

- The highest density task τ_4 is given the highest priority
- ***`OPA` ($\{\tau_1, \tau_2, \tau_3\}$, $m = 2$, `DA-LC`)* returns "schedulable"**
- $I_3(D_3)$ considers $(m-1) = 1$ task as carry-in task

HPA policy applied to OPA+DA-LC

The HPA policy (due to the predictability) can improve OPA +DA-LC as follows:

- OPA+DA-LC is applied to the $(n - m')$ lowest-density tasks to be scheduled on $(m - m')$ processors, and
- the remaining m' highest-density tasks are assigned the highest fixed priority

for some m', $0 \leq m' < m$.

HPA+OPA +DA−LC

Algorithm HybridOPA **(Γ, m)**

1. **for** $m' = 0$ **to** $(m - 1)$
2. remove m' highest desnity tasks from given task set Γ
3. **if** OPA (Γ, $m - m'$, DA−LC) returns "schedulable" then
4. **return** "schedulable"
5. **end for**
6. **return** "unschedulable"

We call this test HP−DA−LC test

HPA+RTA−LC

- RTA−LC is OPA-incompatible
- But HPA is applicable to the RTA−LC test as follows:
  - assign the m' highest-density tasks the highest fixed priority and
  - the fixed-priority ordering of the remaining $(n - m')$ lowest-density tasks remains the same as given for the entire task set Γ

  for some m', $0 \leq m' < m$

Experimental Results

Improvement of `HP-DA-LC` over `DA-LC`

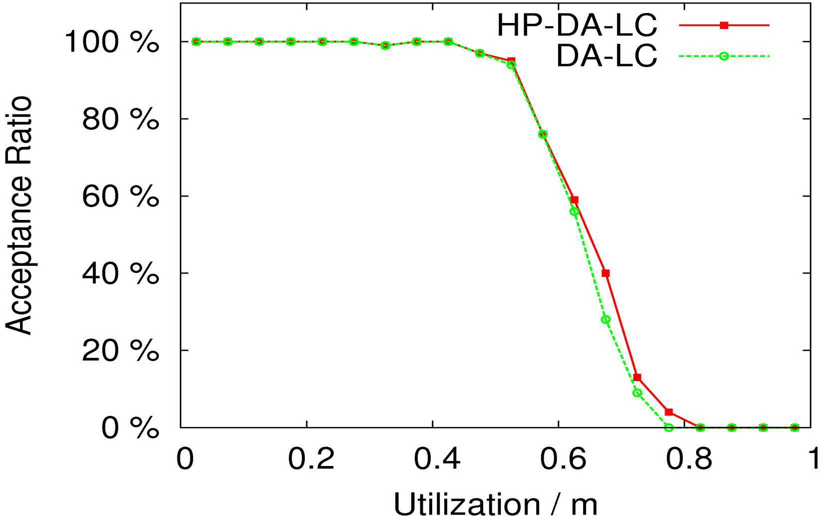


Figure: Acceptance Ratio ($m = 4, n = 16$)

Improvement of HP-DA-LC over DA-LC

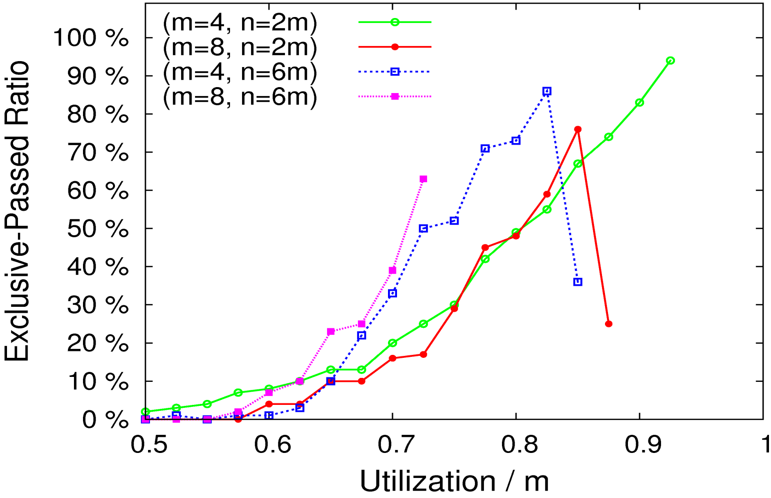


Figure: Exclusive-Passed Ratio

Conclusion

- Improved utilization bound for global fixed-priority scheduling based on ISM-US priority assignment.
- Improved two iterative schedulability tests by proposing better priority assignment policy and schedulability tests.
- HPA policy and predictability, originally used to circumvent Dhall's effect in RM-US [$\frac{m}{3m-2}$], provides
  - separation of concern for schedulability analysis
  - effective priority assignment

  for global fixed-priority scheduling.

Thank You

Backup Slides

Special Task Systems

Special Task Systems

An implicit-deadline sporadic task system Γ is ***special*** on m processor if it satisfies the following two properties:

Property 1: $u_{max} \leq \frac{m}{2m-1}$

Property 2: $U \leq \min\{F_m(u_{min}), F_m(u_{max})\}$

$$\text{where} \qquad F_m(x) = \frac{m(1-x)}{2-x} + x$$

Special Task Systems

Special Task Systems

An implicit-deadline sporadic task system Γ is ***special*** on m processor if it satisfies the following two properties:

Property 1: $u_{max} \leq \frac{m}{2m-1}$

Property 2: $U \leq \min\{F_m(u_{min}), F_m(u_{max})\}$

$$\text{where} \qquad F_m(x) = \frac{m(1-x)}{2-x} + x$$

Theorem

Sporadic task system Γ that is ***special*** on m processors is feasible using global slack-monotonic scheduling on m processors

Constrained Deadline Task System and ISM-DS

Slack: $D_i - C_i$

Total Density: $\delta = \sum \lambda_i = \sum \frac{C_i}{D_i}$

Policy ISM-DS

If $d_i > d_{ts}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given slack-monotonic priority

Threshold Utilization

$$d_{ts} = \frac{3m - 2 - \sqrt{5m^2 - 8m + 4}}{2m - 2}$$

Constrained Deadline Task System and ISM−DS

Slack: $D_i - C_i$

Total Density: $\delta = \sum \lambda_i = \sum \frac{C_i}{D_i}$

Policy ISM−DS

If $d_i > d_{ts}$, then task τ_i is given the highest fixed-priority, otherwise, task τ_i is given slack-monotonic priority

Threshold Utilization

$$d_{ts} = \frac{3m - 2 - \sqrt{5m^2 - 8m + 4}}{2m - 2}$$

Theorem (Utilization Bound)

If $\delta \leq m \cdot min\{0.5, d_{ts}\}$, then all the deadlines of task set Γ are met using global FP scheduling