Rate-Monotonic Analysis

Theorem 16.1 (Bound Test). *A task set $\mathbb{T}$, when scheduled by the RMA principle, can always meet the task deadlines if:*

$$\sum_{i=1}^{k}\left(\frac{e_i}{p_i}\right) = \frac{e_1}{p_1} + \cdots + \frac{e_k}{p_k} \leq k(2^{1/k} - 1). \tag{16.1}$$

k	1	2	3	4	$\cdots$	∞
$k(2^{1/k} - 1)$	1.00	0.83	0.78	0.76	$\cdots$	$log_e(2) = 0.69$

The bound given in Theorem 16.1 is only a *sufficient* condition. A task set might be still schedulable by RMA even though its accumulative utilization is above the bound.

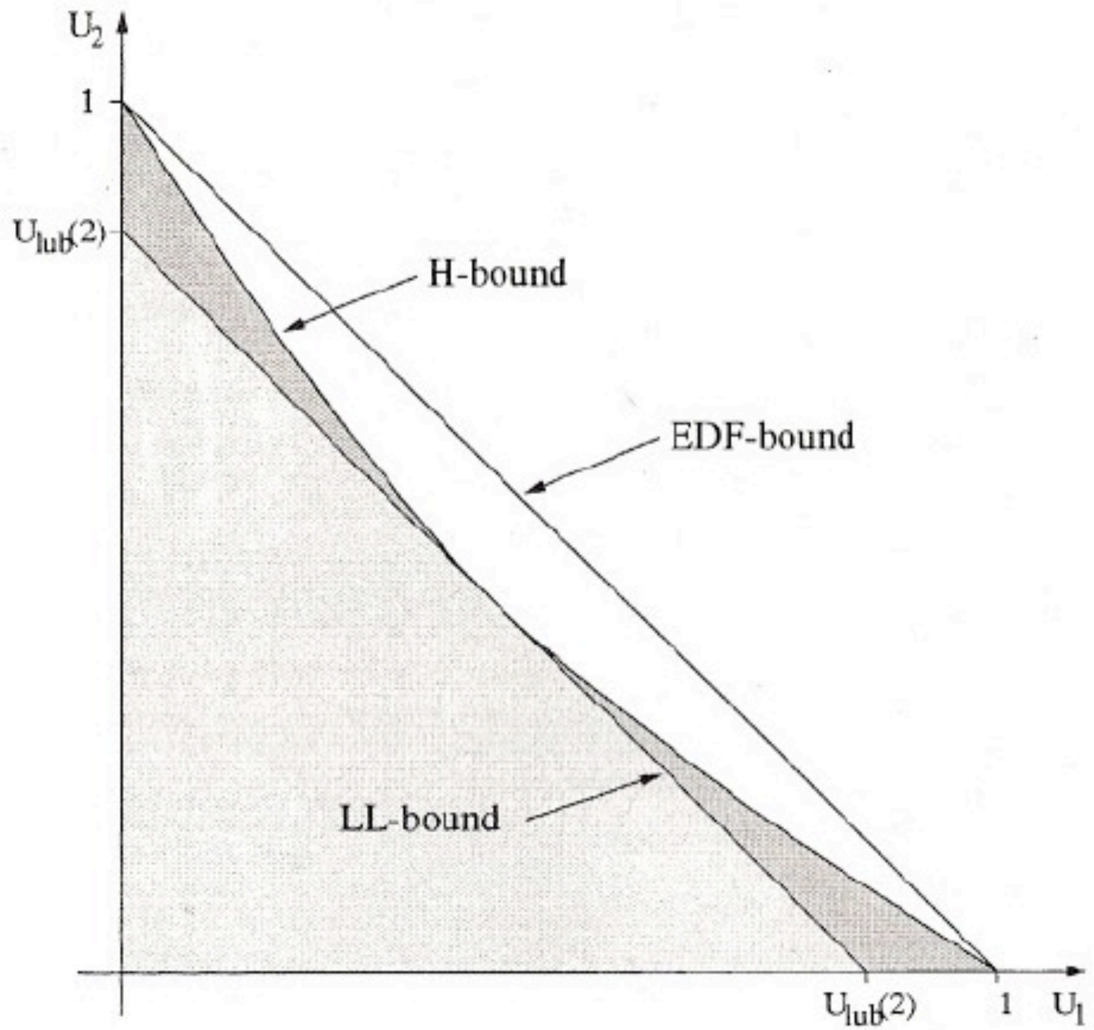
Hyperbolic Bound

- For years, people have been fascinated by the RM bound and try to improve it.
- Hyperbolic bound was proved in 2001

Theorem 4.1 *Let $\Gamma = \{\tau_1, \ldots, \tau_n\}$ be a set of n periodic tasks, where each task τ_i is characterized by a processor utilization U_i. Then, Γ is schedulable with the RM algorithm if*

$$\prod_{i=1}^{n} (U_i + 1) \leq 2. \qquad (4.10)$$

Hyperbolic Bound vs LL Bound



Exact Time Analysis

- Calculate response (finish) times of all jobs in the 1st period
- Start from the lowest priority (although it's not necessary)
  - $J_{n,1}$ implies all jobs $J_{1,1}$, $J_{2,1}$ … $J_{n-1,1}$ must finish
  - $t^0 = e_n + e_1 + e_2 + \ldots + e_{n-1}$
    - But some other jobs may arrive again

Exact Time Analysis

$$t^{1} = e_n + \sum_{i=1}^{n-1} \left\lceil \frac{t^{0}}{P^{i}} \right\rceil e_i$$

$$t^{2} = e_n + \sum_{i=1}^{n-1} \left\lceil \frac{t^{1}}{P^{i}} \right\rceil e_i$$

- …
- Repeat until $t^k = t^{k-1}$, then check to see if $t^k \le P_n$

Example

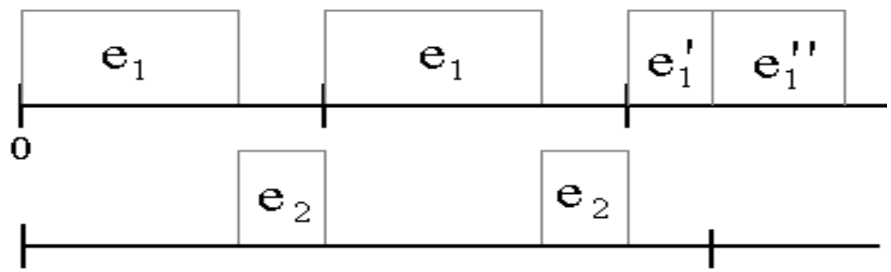
- 1.2/3, 3.6/7
- $t^0 = 1.2 + 3.6 = 4.8 \Rightarrow$
  - task 2 can never finish before 4.8
- $t^1 = 3.6 + \left\lfloor \frac{4.8}{3} \right\rfloor 1.2 = 6$ $\quad$ $t^2 = 3.6 + \left\lceil \frac{6}{3} \right\rceil 1.2 = 6$
- $t^1 = t^2$, so the actual response time for J_{21} is 6.
  - 6 ≤7, so it is schedulable

RM Schedulability Checks

- Liu and Layland Bound: 0.69 or $n(2^{1/n} - 1)$
- Hyperbolic bound: $\Pi_i(u_i + 1) \leq 2$
- Total execution time required: $e_2 + e_1 \left\lceil \frac{T_2}{T_1} \right\rceil$
- Exact time analysis: Fixed point calculation

Execution Times Ratio Implications

- In a system with two jobs, let $3P_l$ be slightly less than P_2
  - $e_l' + e_l'' = e_l$
  - In order for the second job to meet its deadline, $P_2 = e_2 + 3e_l' + 2e_l''$
- Making e_l' larger makes e_2 drop 3 times as fast



Comparisons Between RM and EDF

- The major advantage of the fixed priority approach is that it is simpler to implement.
  - If the ready queue is implemented as a multi-level queue with *P* priority levels (where *P* is the number of different priorities), both task insertion and extraction can be achieved in O(1).
  - In a deadline driven scheduler, the best solution for the ready queue is to implement it as a heap (i.e., a balanced binary tree), where task management requires an O(*log n*) complexity.

Schedulability Between RM and EDF

- In terms of schedulability analysis, an exact guarantee test for RM requires a pseudopolynomial complexity, even in the simple case of independent tasks with relative deadlines equal to periods
- It can be performed in $O(n)$ for EDF.
- In the general case in which deadlines can be less than or equal to periods, the schedulability analysis becomes pseudo-polynomial for both algorithms.
  - Under fixed-priority assignments, the feasibility of the task set can be tested using the response time analysis, whereas under dynamic priority assignments it can be tested using the processor demand criterion.

Processor Utilization of RM and EDF

- As for the processor utilization, EDF is able to exploit the full processor bandwidth, whereas the RM algorithm can only guarantee feasibility for task sets with utilization less than 69%, in the worst case.
- In the average case, a statistical study performed by Lehoczky, Sha, and Ding [LSD89] showed that for task sets with randomly generated parameters the RM algorithm is able to feasibly schedule task sets with a processor utilization up to about 88%.
- However, this is only a statistical result and cannot be taken as an absolute bound for performing a precise guarantee test.

Deadline Monotonic (DM) Priority

- The Deadline Monotonic (DM) priority assignment weakens the "*period equals deadline*" constraint.
- This algorithm was first proposed in 1982 by Leung and Whitehead [LW82] as an extension of Rate Monotonic where tasks can have a relative deadline less than their period.
- According to the DM algorithm, each task is assigned a static priority inversely proportional to its relative deadline. As RM, DM is preemptive.
- Thus, at any instant, the task with the shortest relative deadline is executed. Since relative deadlines are constant, DM is a static priority assignment.

Processor Utilization of DM

- The Deadline-Monotonic priority assignment is optimal, meaning that if any static priority scheduling algorithm can schedule a set of tasks with deadlines unequal to their periods, then DM will also schedule that task set.
- The feasibility of a set of tasks with deadlines unequal to their periods could be guaranteed using the Rate-Monotonic schedulability test, by reducing tasks' periods to relative deadlines

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{1/n} - 1).$$

Conclusion about RM, EDF and DM

- Fixed Priority vs. Dynamic Priority
- Pessimistic vs. Exact Analysis
- Simple vs. Complex Analysis
- Implementation Simplicity
- Runtime Overheads

Least Slack Time First (LST)

- LST = $(D_i - C_i)_t$ where D_i is the relative deadline
- For each time-slice, compute the new slack time for each task in the queue: $((D_i - t) - C_i)_t$
- Slack of the task being executed is a constant. Slack for the rest of the tasks decreases

Least Slack Time First (LST) cont.

- If 2 jobs have the same slack, 2/4, 2/4, at $t = 1$ the second task will preempt, and a racing condition will begin.
- LST needs a time-slice to avoid system overhead
- LST cannot handle over-load, similar to EDF.