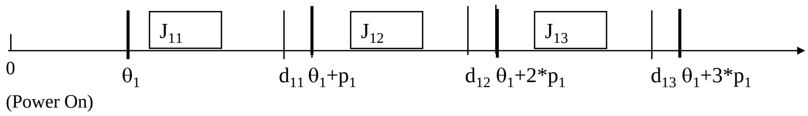
Priority-Driven Scheduling of Periodic Tasks (1)
- Chapter 6 -

Overview

- Reference Model Assumptions
- Fixed-priority vs. Dynamic Priority
- RM
  - schedulable utilization bound
  - time demand analysis
- EDF
  - schedulable utilization bound
  - time demand analysis
  - The stability problem of EDF

Periodic Task Model

- A periodic task T_i is characterized by
  - phase: θ_i
  - Period: p_i
  - Execution time : e_i
  - Relative deadline: D_i from the beginning of the period.

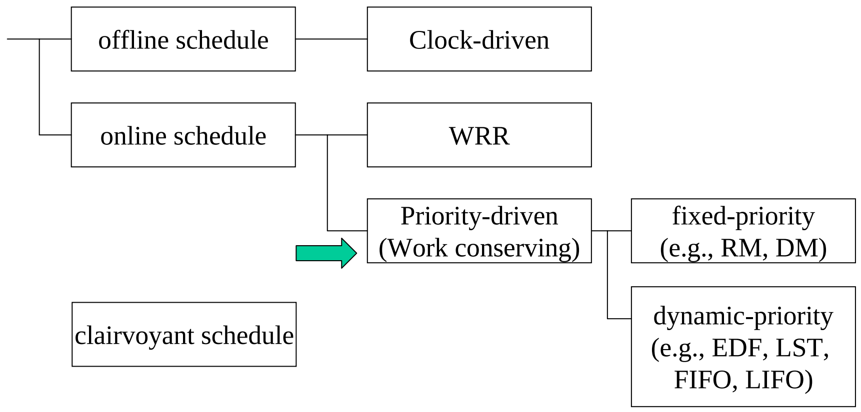


- Default assumption: $D_i = p_i$. That is, a periodic task deadline is located at the end of the period

Assumptions

- the tasks are independent
  - for resource contention, Chapter 8
- there are no aperiodic and sporadic tasks
  - for integrated scheduling, Chapter 7
- other assumptions
  - can be preempted at any time
  - context switch overhead is negligible

Classification of Scheduling Algorithms (Review)



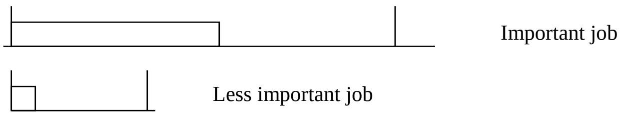
"Priority vs. Criticality"

- Priority: priority is the order we execute ready jobs.
- Criticality (Importance): represents the penalty if a task misses a deadline (one of its jobs misses a deadline).

- Quiz: Which task should have higher priority?
- Task 1: The most import task in the system: if it does not get done, catastrophic consequences will occur
- Task 2: An mp3 player: if it does not get done in time, the played song will have a glitch

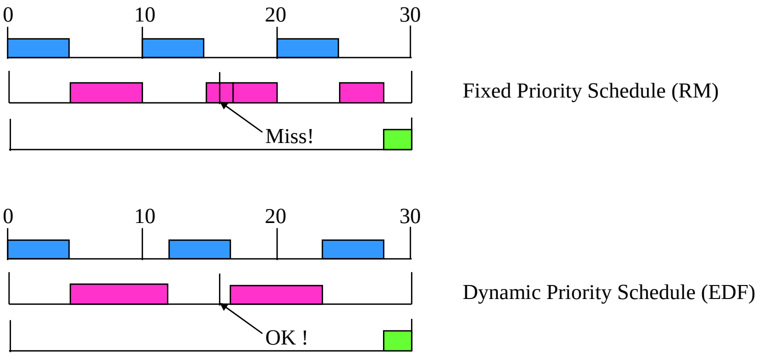
“Priority vs. Criticality”

- An important find in real-time computing theory is that ***importance*** may or may not correspond to ***scheduling priority***.
- In the following example, giving the less important task higher priority results in both tasks meeting their deadlines.
- Importance matters only when tasks cannot be scheduled (overload condition), not when they can be scheduled.



Dynamic Priority vs. Fixed Priority

- $\{T_1=(p_1=10, e_1=4), T_2=(p_2=15, e_2=8), T_3=(p_3=30, e_3=2)\}$



What are advantages of priority-driven schedule over clock-driven?

- Scheduling decision is made online, and hence ***flexible***
  - Jobs of a task doesn't need to be released at the fixed time (exact periodic)
    - period = minimum inter-release time
  - Tasks can dynamically enter and leave the system
- Good! BTW, how can we validate the timing behavior?
  - Predictability: can we say the system is schedulable a priori?
  - Fortunately, we have sound theory on the schedulability of priority-driven schedule

OK, Let's study such theory

- Is it enough to simply memorize important theorems? -

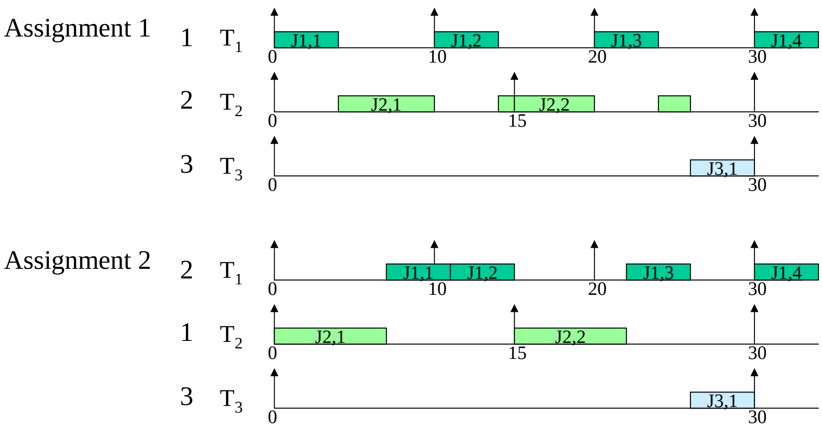
- Facts
  - "RM is optimal"
  - "DM is optimal"
  - "The system is schedulable if $U < n(2^{1/n}-1)$ according to RM schedule"
  - "EDF is optimal"
  - "The system is schedulable if $U < 1$ according to EDF schedule"
- Not that useful!
  - Most facts are true under some limited conditions
  - Our problem does not exactly meet those conditions
  - Most of time, we cannot directly apply the fact to our problem
- Deep understanding
  - how people developed the facts?
  - how to prove the facts?
  - how to change the facts for our problem?

Fixed-Priority Scheduling

- How to assign Priorities?
- How to check the schedulability?

Priority Assignment

- $\{T_1=(p_1=10, e_1=4), T_2=(p_2=15, e_2=7), T_3=(p_3=30, e_3=4)\}$



Intuitive priority assignments

- Random – mostly perform poorly
- Functional Criticality (Semantic importance)
  - T_1 is a video display task
  - T_2 is a task monitoring and controlling patient's blood pressure
- Urgency
  - If all tasks are feasibly schedulable, the critical task doesn't have to be the highest priority task
  - RM and DM are examples

Optimal Static Priority Algorithm

- ***RM (Rate Monotonic)*** is an optimal static priority assignment for periodic tasks with deadlines at the end of the period.
  - Higher priority is assigned to a task with higher rate (inverse of period)
- ***DM (Rate Monotonic)*** is an optimal static priority assignment for periodic tasks with arbitrary relative deadlines.
  - Higher priority is assigned to a task with shorter relative deadline

What does optimality mean?

- Optimality: I am an optimal algorithm …..
  - If I cannot find a feasible schedule, nobody else can!
- Quiz: EDF is optimal, RM is optimal too…….. Is RM as powerful as EDF (why or why not)?
- RM is optimal under limited conditions
  - fixed-priority domain
  - deadlines are the end of periods

Proof of RM optimality

- Recall the swapping trick
  - Any feasible schedule (static-priority) can be transformed to RM feasible schedule!
- When saying a periodic task schedulable, we mean that every job of this task will meet its deadline. Since a periodic task can repeat itself endlessly, checking every job is impractical, if not impossible.
- Fortunately, there is a shortcut.

The Critical Instant Theorem

- In static priority scheduling, the completion time of a job is the sum of its own execution time plus the sum of preemptions from higher priority tasks.
- Critical instant theorem claims that maximum preemption occurs when all higher priority tasks line up at time 0. So if a job can make it under maximum preemption, it can certainly make it when preemption is lighter.

• **Critical instant theorem:** in static priority scheduling, a task is schedulable if its first job meets its deadline, under the condition that all the higher priority tasks and this task start at the same time, e.g., t = 0.

Critical Instant Theorem

- **Proof**:

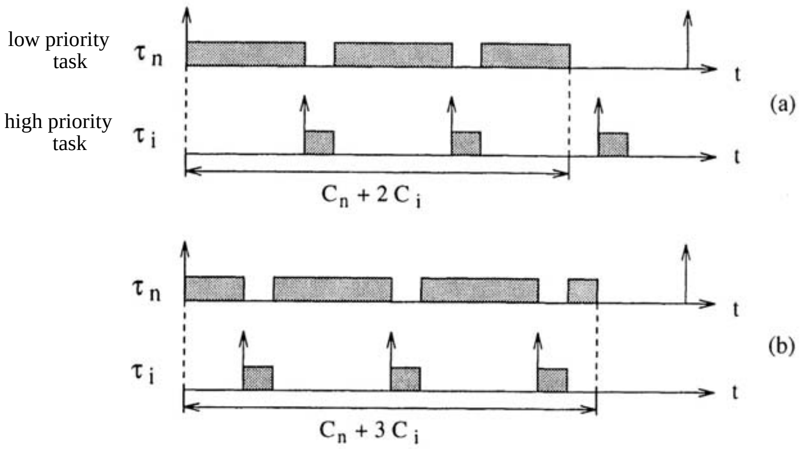
  Consider a set of periodic tasks ordered according to static priorities. For the sake of simplicity, let's consider RM.

  Let $\Gamma=\{T_1,\ldots,T_n\}$ be a set of tasks ordered by increasing periods, with T_n being the task with the longest period. According to RM, T_n will also be the task with the lowest priority.

  Notice that (see figure) the response time of T_n is delayed by the interference of T_i with higher priority. Moreover, it is clear that advancing the release time of T_i may increase the completion time of T_n.

  It follows that the response time of T_n is largest when it is released simultaneously with T_i.

Critical Instant Theorem



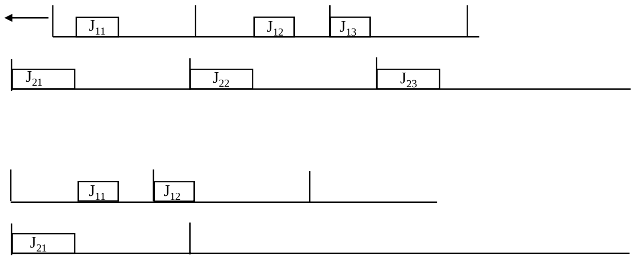
Critical Instant Theorem

Repeating the argument for all T_i, i = 2,…,n-1, we prove that the worst response time of a task occurs when it is released simultaneously with all higher-priority tasks.

□

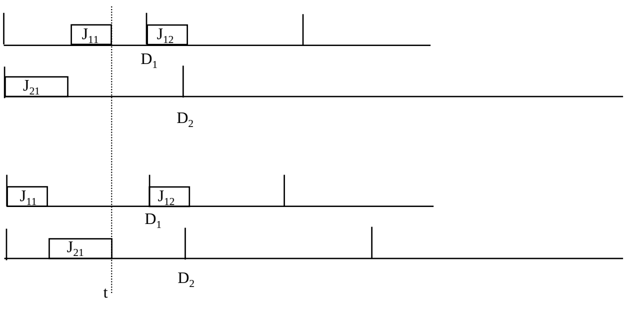
- What’s an important consequence of this result?

Optimality of RM
(by using the Critical Instant Theorem)



- Given two tasks, suppose that priorities are not assigned according to RM and that the task set is feasible

Swapping



•Move J_{11} to t = 0, meets release time requirement and J_{11} still meets its deadline

•Since $J_{21} + J_{11} = J_{11} + J_{21} = t < D_1 < D_2$, J_{21} meets its deadline at D_2.

•Since J_{11} meets its deadline and J_{21} meets its deadline, all the jobs in both tasks will always meet their deadline (Why?)

Schedulability Check!

- Important for
  - Offline design phase
    - period selection
    - algorithm selection
    - identifying modules to be optimized
  - Online admission phase (in dynamic real-time systems)
    - periodic tasks are dynamically created by external events
      - In case that the system becomes unschedulable by adding the new task, we cannot admit it. Instead, we have to ring a warning alarm ASAP for alternative action.
    - control frequency and algorithm negotiation
    - frame rate and QoS parameter negotiation in multimedia

Using Critical Instant Theorem

- A direct use of the critical instant theorem is the exact schedulability test. It is also known as the time demand analysis.
- We shall illustrate this by an example of 3 tasks
- $\{(e_1 = 4, p_1=10), (e_2=4, p_2=15), (e_3=10, p_3=35)\}$ and we are interested to know if task T_3 can meet its 1st deadline under rate monotonic scheduling
  - Then, all T_3 future deadlines can be met.

Formulation (Exact Analysis)

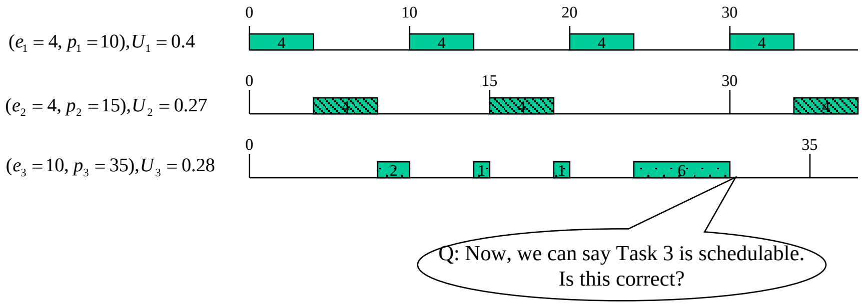
$$r_i^{k+1} = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil e_j, \quad \text{where } r_i^0 = \sum_{j=1}^{i} e_j$$

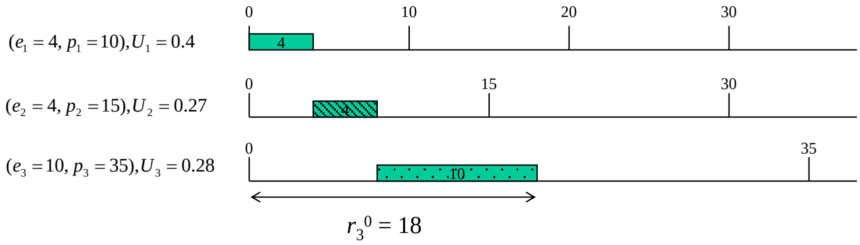
Test terminates when $r_i^{k+1} > p_i$ (not schedulable)
or when $r_i^{k+1} = r_i^k \le p_i$ (schedulable).

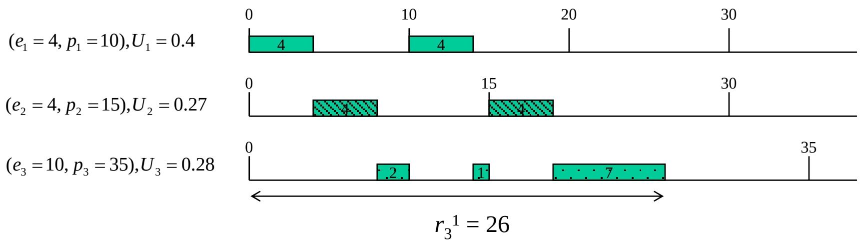
- Tasks are ordered according to their priority: T_1 is the highest priority task.

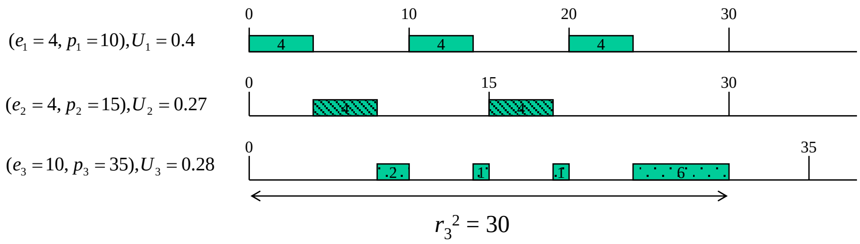
The Exact Schedulability Test

- Basically, “*Enumerate*” the schedule
- “Task by Task” schedulability test




$(e_1 = 4, p_1 = 10), U_1 = 0.4$
$(e_2 = 4, p_2 = 15), U_2 = 0.27$
$(e_3 = 10, p_3 = 35), U_3 = 0.28$
0
10
20
30
4
0
15
30
4
0
35
10
$r_3^0 = 18$


$(e_1 = 4, p_1 = 10), U_1 = 0.4$
$(e_2 = 4, p_2 = 15), U_2 = 0.27$
$(e_3 = 10, p_3 = 35), U_3 = 0.28$
0
10
20
30
4
4
0
15
30
4
4
0
35
2
1
7
$r_3^1 = 26$



Intuitions of Exact Schedulability Test

- Obviously, the response time of task 3 should larger than or equal to $e_1+e_2+e_3$

$$r_3^0 = \sum_{j=1}^{3} e_j = e_1 + e_2 + e_3 = 4 + 4 + 10 = 18$$

Intuitions of Exact Schedulability Test

- Obviously, the response time of task 3 should larger than or equal to $e_1+e_2+e_3$

$$r_3^0 = \sum_{j=1}^{3} e_j = e_1 + e_2 + e_3 = 4+4+10 = 18$$

- The high priority jobs released in r_3^0, should lengthen the response time of task 3

$$r_3^1 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^0}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{18}{10} \right\rceil 4 + \left\lceil \frac{18}{15} \right\rceil 4 = 26$$

Intuitions of Exact Schedulability Test

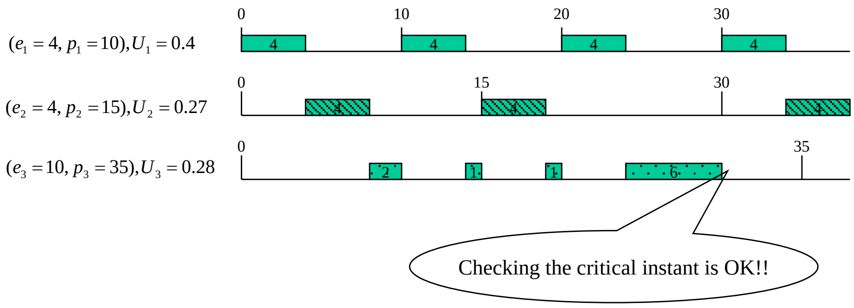
- Keep doing this until either r_3^k no longer increases or $r_3^k > p_3$

$$r_3^2 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^1}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{26}{10} \right\rceil 4 + \left\lceil \frac{26}{15} \right\rceil 4 = 30$$

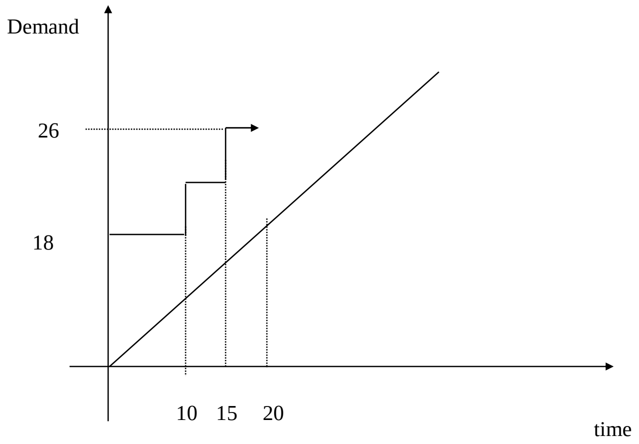
$$r_3^3 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{30}{10} \right\rceil 4 + \left\lceil \frac{30}{15} \right\rceil 4 = 30$$ **Done!**

How long should we enumerate the schedule?

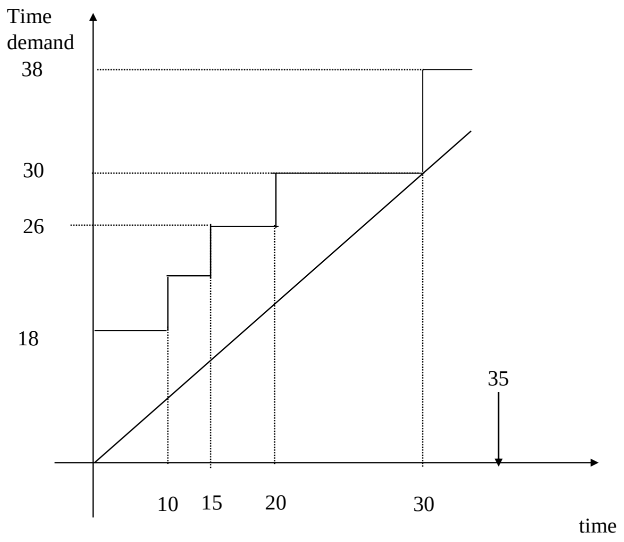
<u>Critical instant theorem</u>: If a task meets its first deadline when all higher priority tasks are started at the same time, then this task's future deadlines will always be met. The <u>exact test</u> for a task checks if this task can meet its first deadline[Liu73].



Time Demand Graph



Time Demand Graph



Class Exercise 1

Suppose that we have two tasks

- $e_1 = 3$, $p_1 = 5$
- $e_2 = 5$, $p_2 = 14$

- Use exact test to check the schedulability of task 2. Draw the schedule timeline to confirm that

- $r_2^0 = e_1 + e_2 = 3 + 5 = 8$

$$r_2^1 = e_2 + \left\lceil \frac{r_2^0}{p_1} \right\rceil e_1 = 5 + \left\lceil \frac{8}{5} \right\rceil 3 = 11$$

$$r_2^2 = e_2 + \left\lceil \frac{r_2^1}{p_1} \right\rceil e_1 = 5 + \left\lceil \frac{11}{5} \right\rceil 3 = 14$$

$$r_2^3 = e_2 + \left\lceil \frac{r_2^2}{p_1} \right\rceil e_1 = 5 + \left\lceil \frac{14}{5} \right\rceil 3 = 14$$

Done! ➔ the task set is schedulable

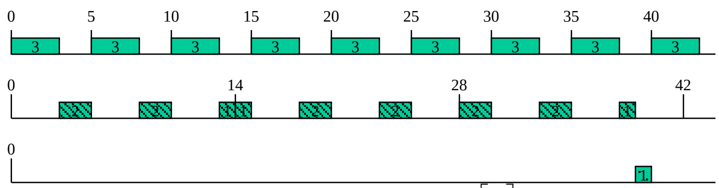
Class Exercise 1

Suppose that we have two tasks

- $e_1 = 3$, $p_1 = 5$
- $e_2 = 5$, $p_2 = 14$

- Can we add a task 3 with $e_3 = 1$ and $p_3 = 50$? What would be the shortest period of p_3 that it can still meet its deadlines? Apply the exact test formulation to confirm that.

Class Exercise 1 (continued)



$$r_3^0 = \sum_{j=1}^{3} C_j = 3+5+1 = 9$$

$$r_3^1 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^0}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{9}{5} \right\rceil 3 + \left\lceil \frac{9}{14} \right\rceil 5 = 12$$

$$r_3^2 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^1}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{12}{5} \right\rceil 3 + \left\lceil \frac{12}{14} \right\rceil 5 = 15$$

$$r_3^3 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{15}{5} \right\rceil 3 + \left\lceil \frac{15}{14} \right\rceil 5 = 20$$

$$r_3^4 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^3}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{20}{5} \right\rceil 3 + \left\lceil \frac{20}{14} \right\rceil 5 = 23$$

$$r_3^5 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^4}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{23}{5} \right\rceil 3 + \left\lceil \frac{23}{14} \right\rceil 5 = 26$$

$$r_3^6 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^5}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{26}{5} \right\rceil 3 + \left\lceil \frac{26}{14} \right\rceil 5 = 29$$

$$r_3^7 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^6}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{29}{5} \right\rceil 3 + \left\lceil \frac{29}{14} \right\rceil 5 = 34$$

$$r_3^8 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^7}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{34}{5} \right\rceil 3 + \left\lceil \frac{34}{14} \right\rceil 5 = 37$$

$$r_3^9 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^8}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{37}{5} \right\rceil 3 + \left\lceil \frac{37}{14} \right\rceil 5 = 40$$

$$r_3^{10} = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^9}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{40}{5} \right\rceil 3 + \left\lceil \frac{40}{14} \right\rceil 5 = 40$$

Formulation (Exact Analysis)

Quiz: Can we use the exact analysis formulation for non RM static priority scheduling?

Quiz: Can we extend the exact analysis to tasks with deadlines less than periods? How?

Quiz: Can we use the exact analysis for a task set where the critical instant never occurs?

Class Exercise 2

Suppose that three tasks are scheduled under RMS

- $e_1 = 4$, $p_1 = 10$
- $e_2 = 6.1$, $p_2 = 14$
- $e_3 = 1$, $p_3 = 70$

- Is task 2 schedulable?
- How about task 3?

Class Exercise 2: Task 2

- $e_1 = 4$, $p_1 = 10$
- $e_2 = 6.1$, $p_2 = 14$
- $e_3 = 1$, $p_3 = 70$

$$r_2^0 = 4 + 6.1 = 10.1$$

$$r_2^1 = \left\lceil \frac{10.1}{10} \right\rceil \cdot 4 + 6.1 = 14.1 > 14$$

Task 2 is not schedulable!

Class Exercise 2: Task 3

$$a_0 = 4 + 6.1 + 1 = 11.1$$

$$a_1 = \left\lceil \frac{11.1}{10} \right\rceil 4 + \left\lceil \frac{11.1}{14} \right\rceil 6.1 + 1 = 15.1$$

$$a_2 = \left\lceil \frac{15.1}{10} \right\rceil 4 + \left\lceil \frac{15.1}{14} \right\rceil 6.1 + 1 = 21.2$$

$$a_3 = \left\lceil \frac{21.2}{10} \right\rceil 4 + \left\lceil \frac{21.2}{14} \right\rceil 6.1 + 1 = 25.2$$

$$a_4 = \left\lceil \frac{25.2}{10} \right\rceil 4 + \left\lceil \frac{25.2}{14} \right\rceil 6.1 + 1 = 25.2 < 70$$

Even Task 2 is not schedulable, Task 3 is schedulable.

It is a common mistake to assume that if a higher priority task is not schedulable so are the lower priority tasks. Don't make this mistake!

Summary of Exact Test

- Exact test is sufficient and necessary condition for the schedulability!
  - when the critical instant actually occurs
  - execution times and periods are constant as given
  - applicable to non-RM priority assignment
  - applicable even when the deadlines are shorter than the periods
- Still sufficient condition
  - even if task phase never make critical instant
  - execution times are smaller than the given values
  - inter-release time is longer than the given periods
- Problems
  - applicable only when execution times *e* and periods *p* are known
  - high complexity – not practical for online admission control