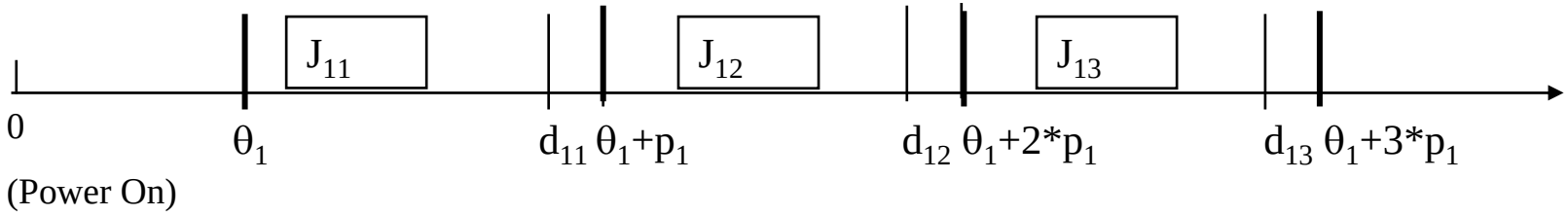
Fixed-Priority Scheduling

Overview

- Reference Model and Assumptions
- Fixed-priority vs. Dynamic Priority
- Fixed-Priority Scheduling (e.g., RM)
  - Priority Assignment
  - Schedulability Analysis
    - schedulable utilization bound
    - time demand analysis

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.

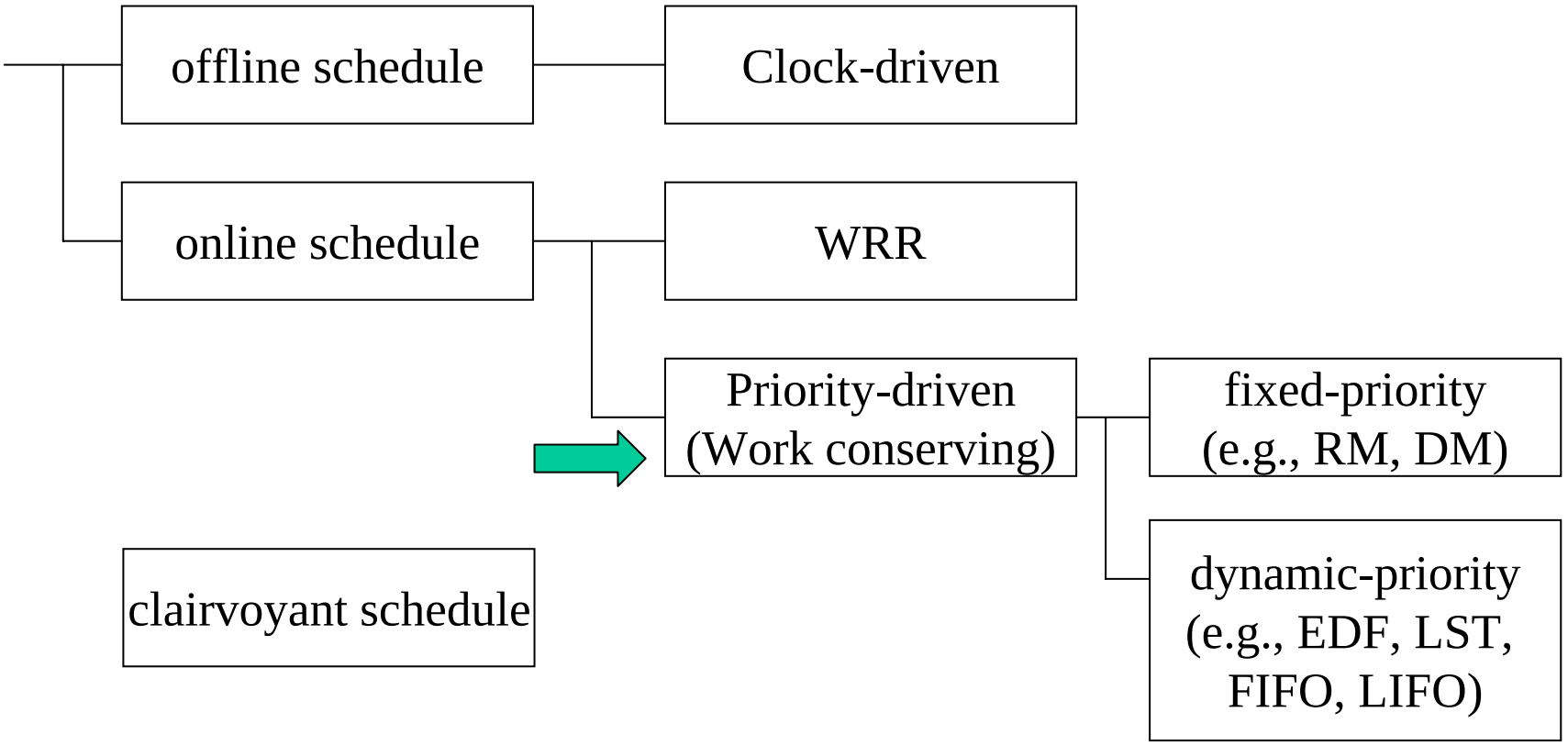


- Default assumption: $D_i = p_i$. That is, a periodic task deadline is located at the end of the period

Assumptions

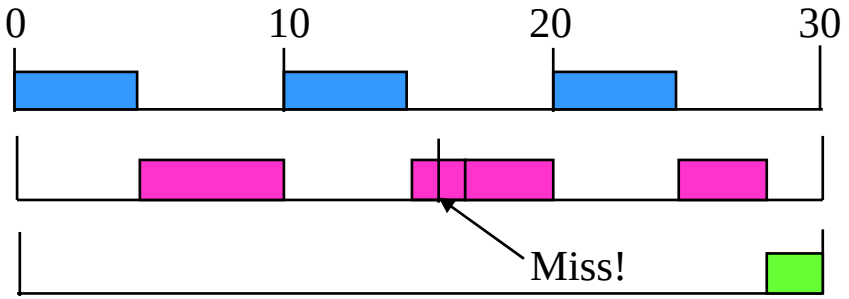
- the tasks are independent
- there are no aperiodic and sporadic tasks
- other assumptions
  - can be preempted at any time
  - context switch overhead is negligible

Classification of Scheduling Algorithms (Review)

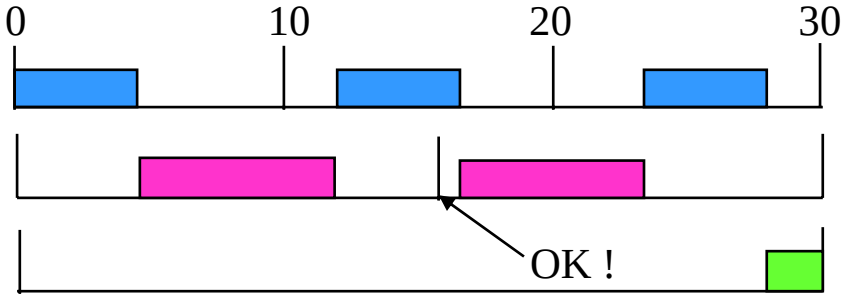


Dynamic Priority vs. Fixed Priority

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



Fixed Priority Schedule (RM)



Dynamic Priority Schedule (EDF)

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

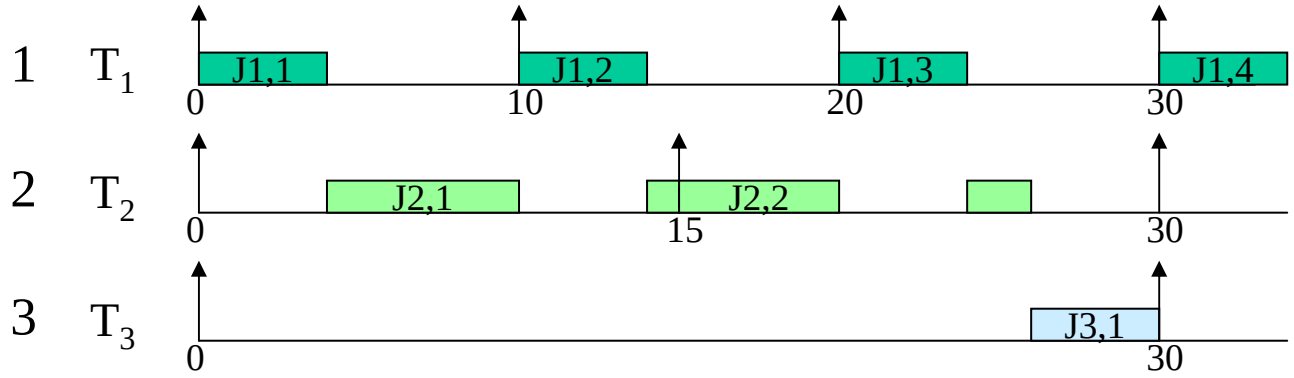
Fixed-Priority Scheduling

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

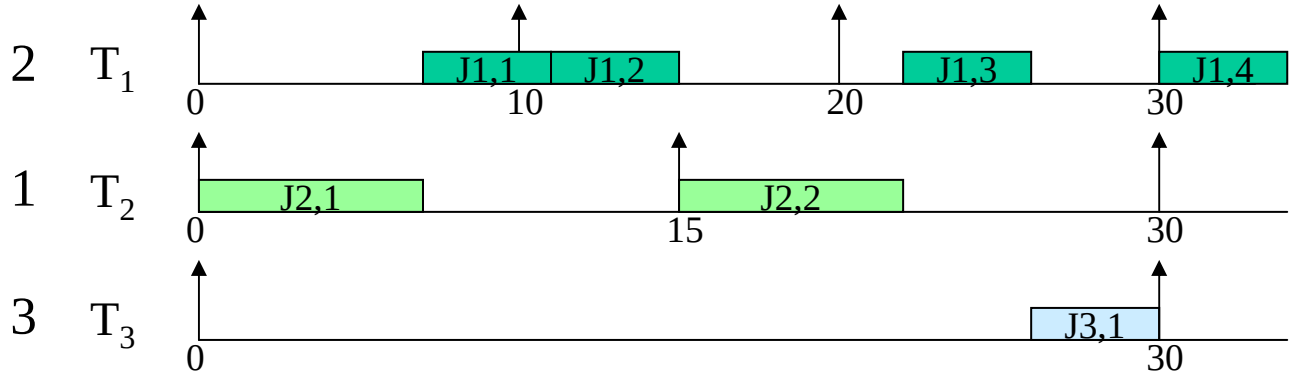
Priority Assignment

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

Assignment 1



Assignment 2



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 (Deadline 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

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**:

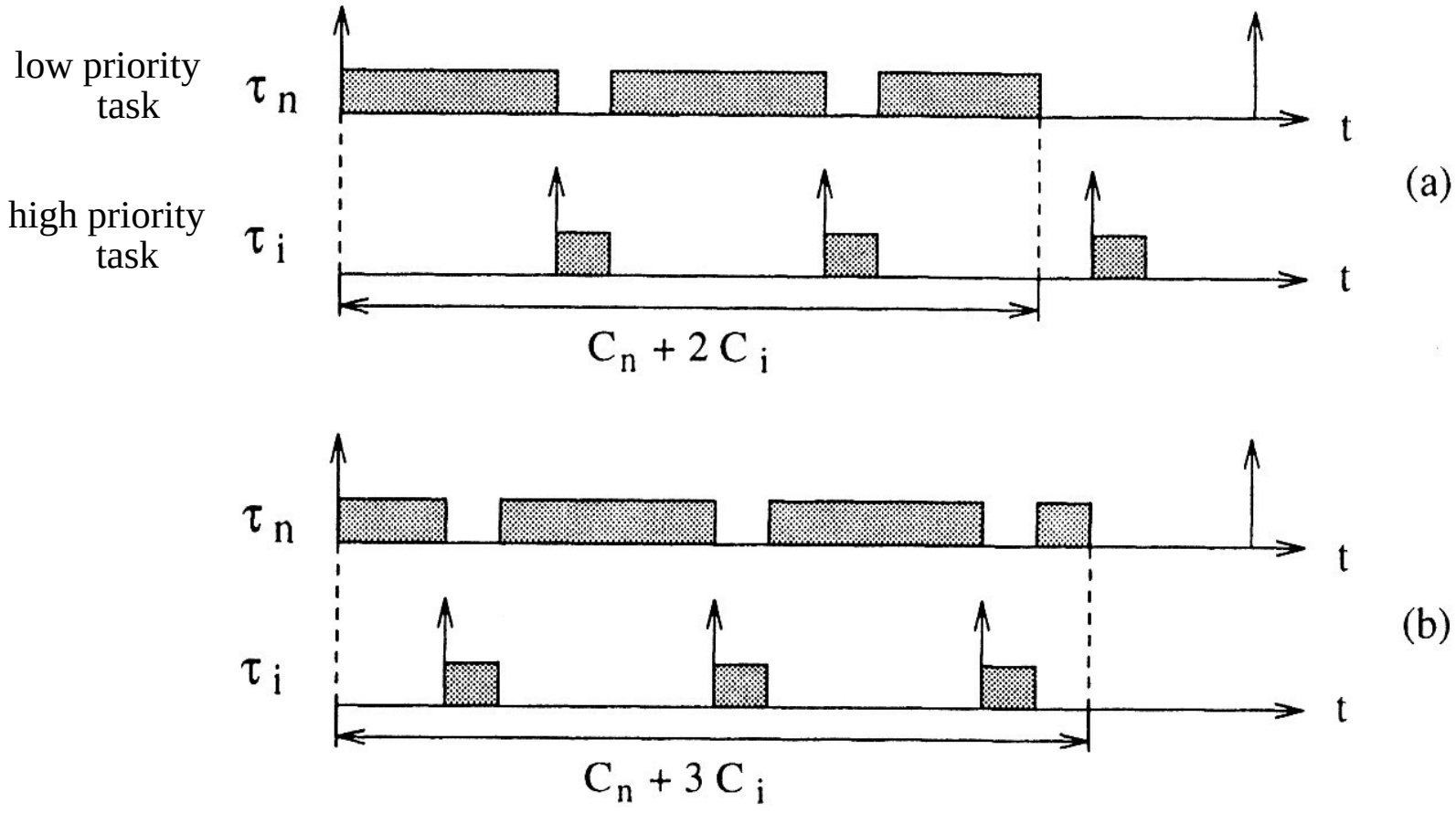
  Consider a set of periodic tasks ordered according to static priorities. For the sake of simplicity, let's consider RM.

  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



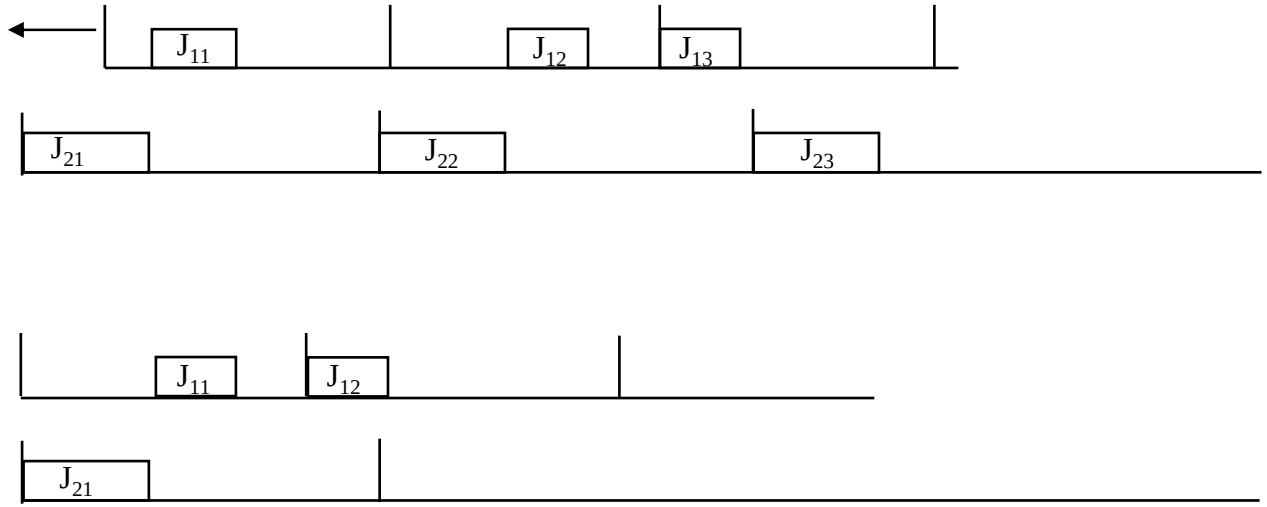
Critical Instant Theorem

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.

□

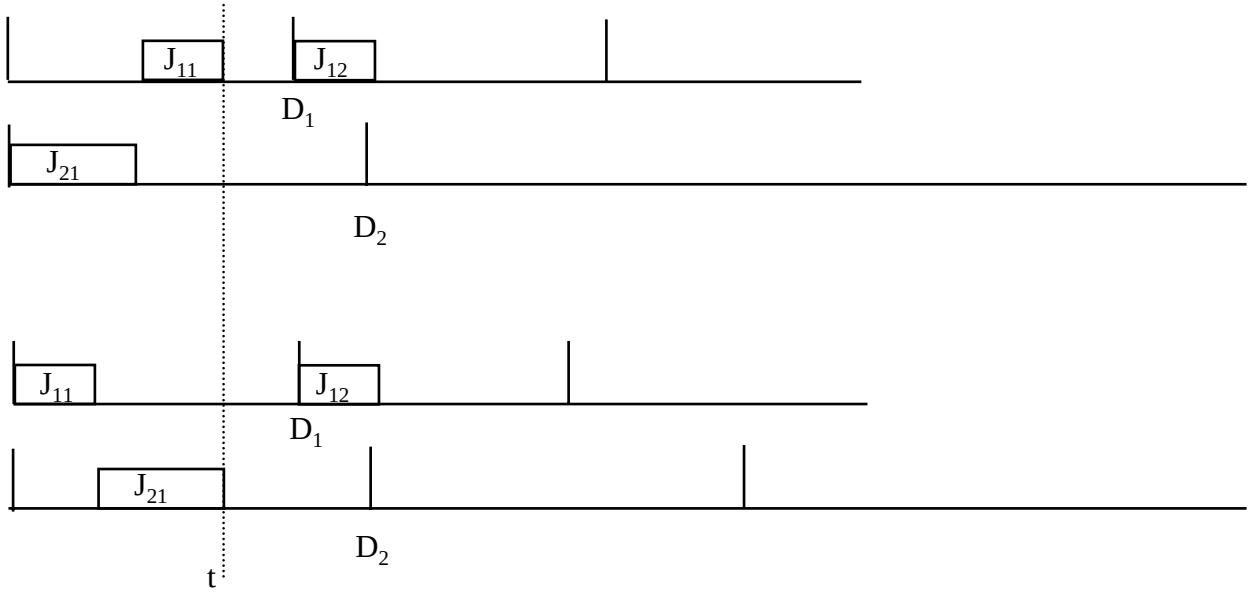
- What's an important consequence of this result?

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 1^{st} deadline under rate monotonic scheduling
  - Then, all T_3 future deadlines can be met.

Formulation (Exact Analysis)

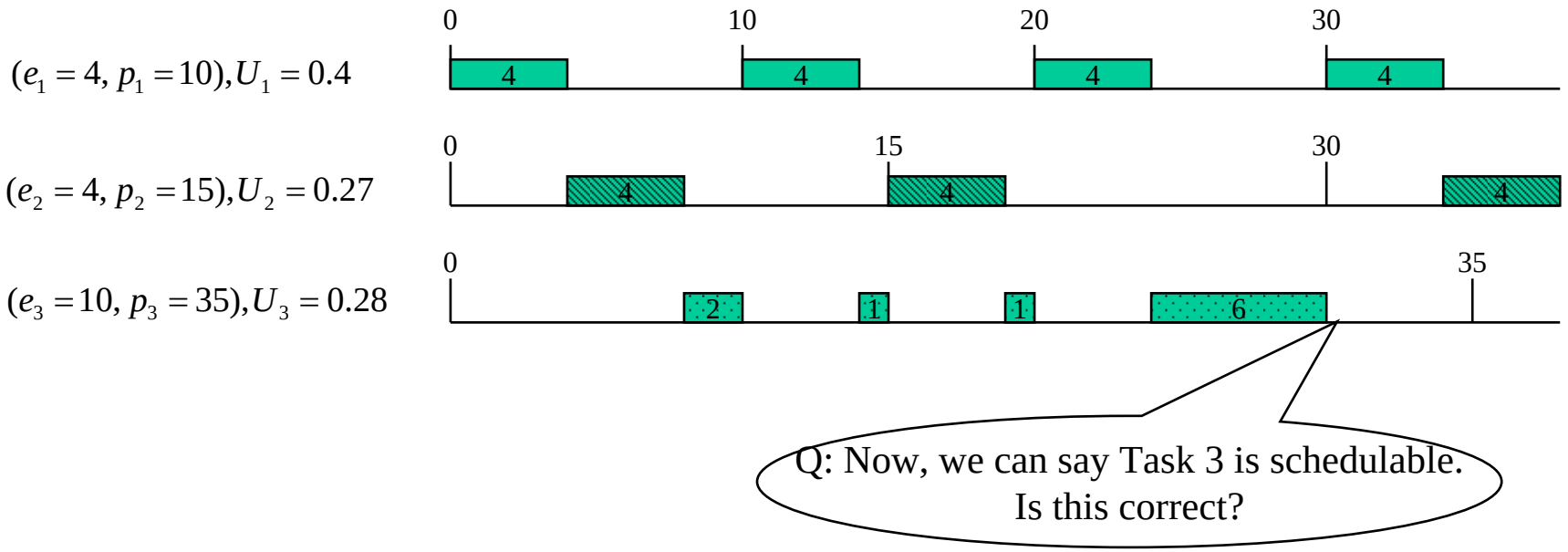
$$r_i^{k+1} = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil e_j, \quad \text{where } r_i^0 = \sum_{j=1}^{i} e_j$$

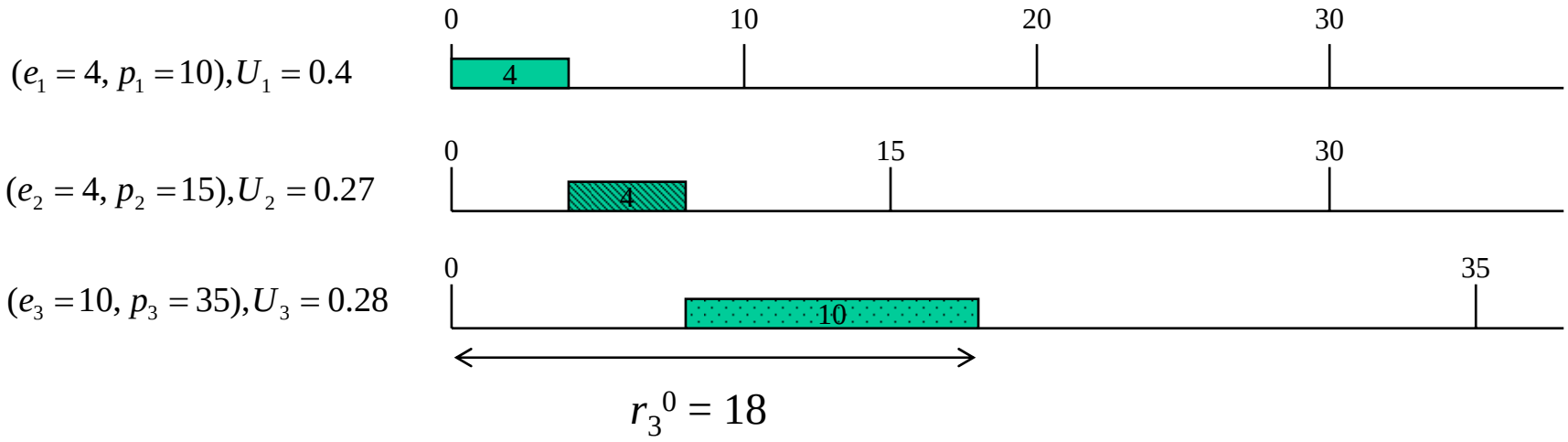
Test terminates when $r_i^{k+1} > p_i$ (not schedulable)
or when $r_i^{k+1} = r_i^k \leq p_i$ (schedulable).

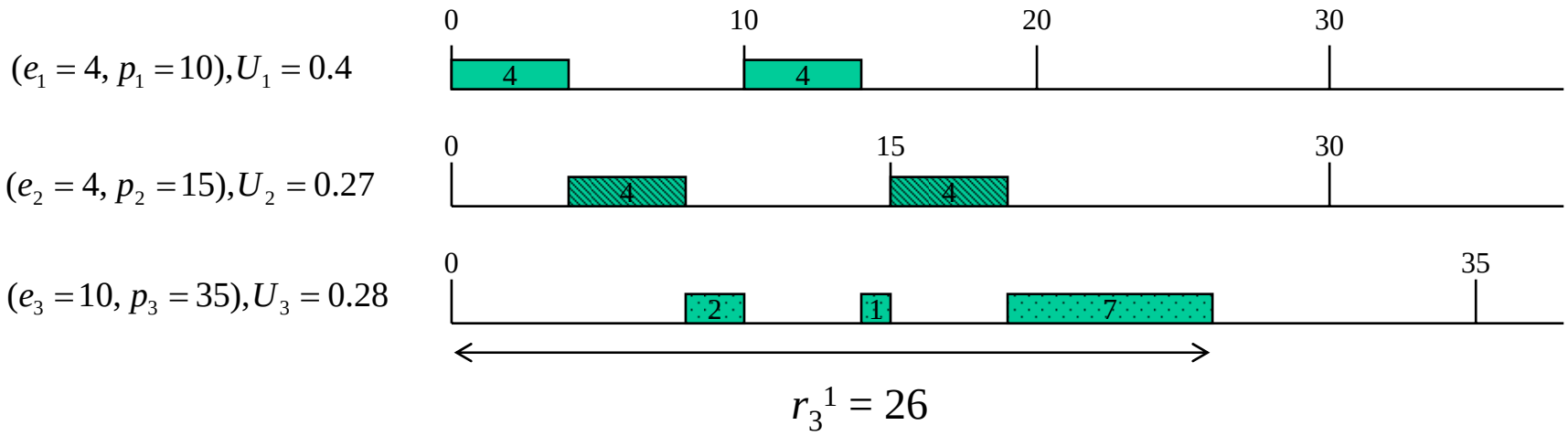
- Tasks are ordered according to their priority: T_1 is the highest priority task.

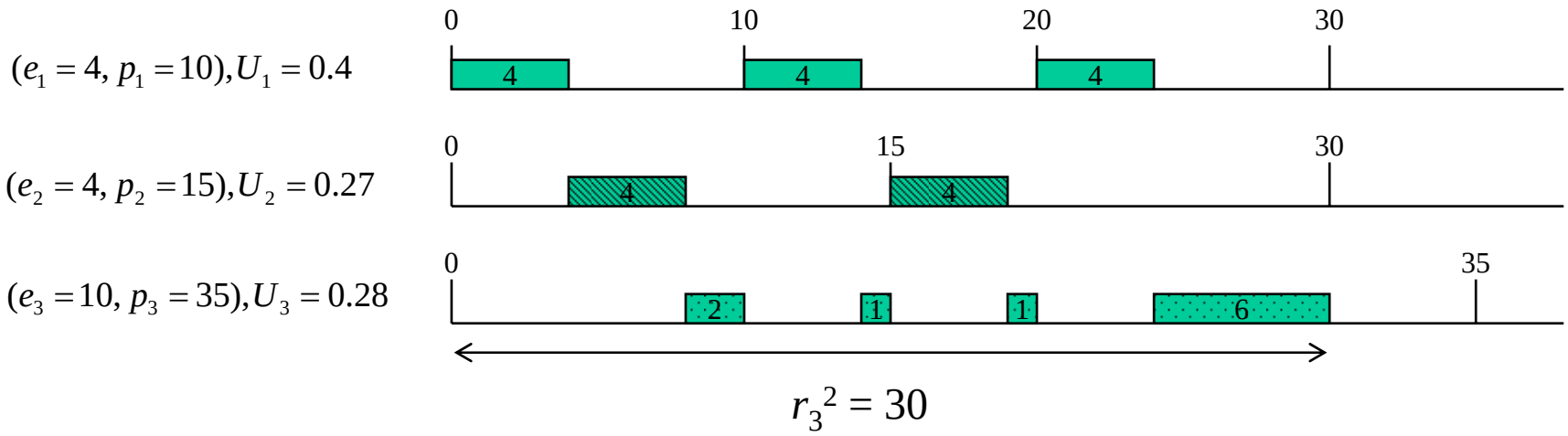
The Exact Schedulability Test

- 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
$r_3^1 = 26$


$(e_1 = 4, p_1 = 10), U_1 = 0.4$
$(e_2 = 4, p_2 = 15), U_2 = 0.27$
$(e_3 = 10, p_3 = 35), U_3 = 0.28$
0
10
20
30
4
4
4
0
15
30
4
4
0
35
2
1
1
6
$r_3^2 = 30$

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

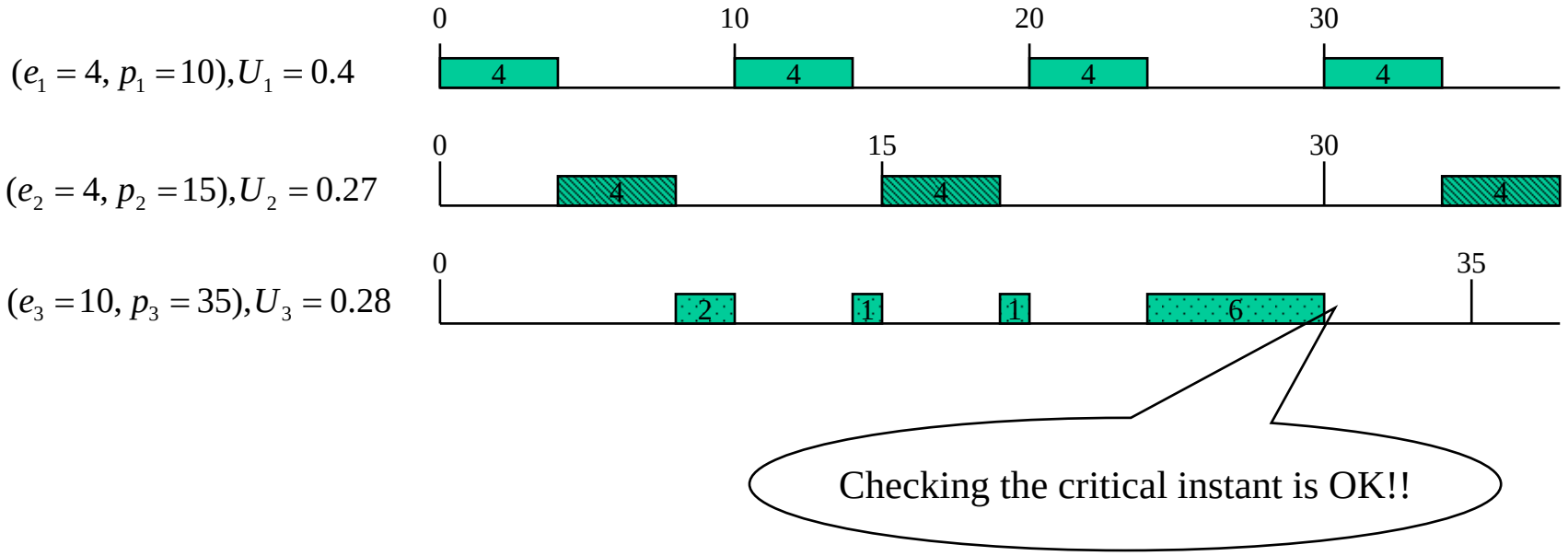
- Keep doing this until either r_3^k no longer increases or $r_3^k > p_3$

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

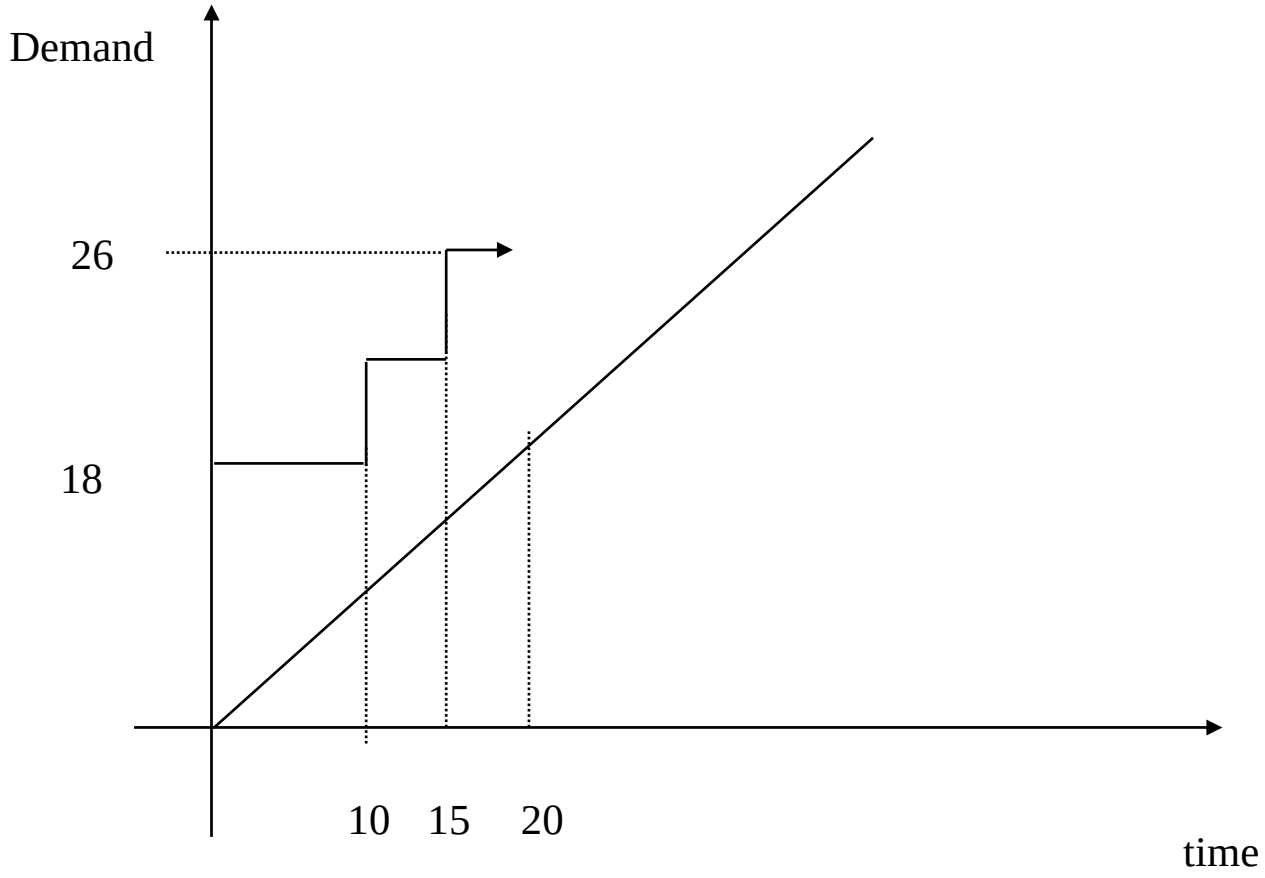
$$r_3^3 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{30}{10} \right\rceil 4 + \left\lceil \frac{30}{15} \right\rceil 4 = 30$$ **Done!**

How long should we enumerate the schedule?

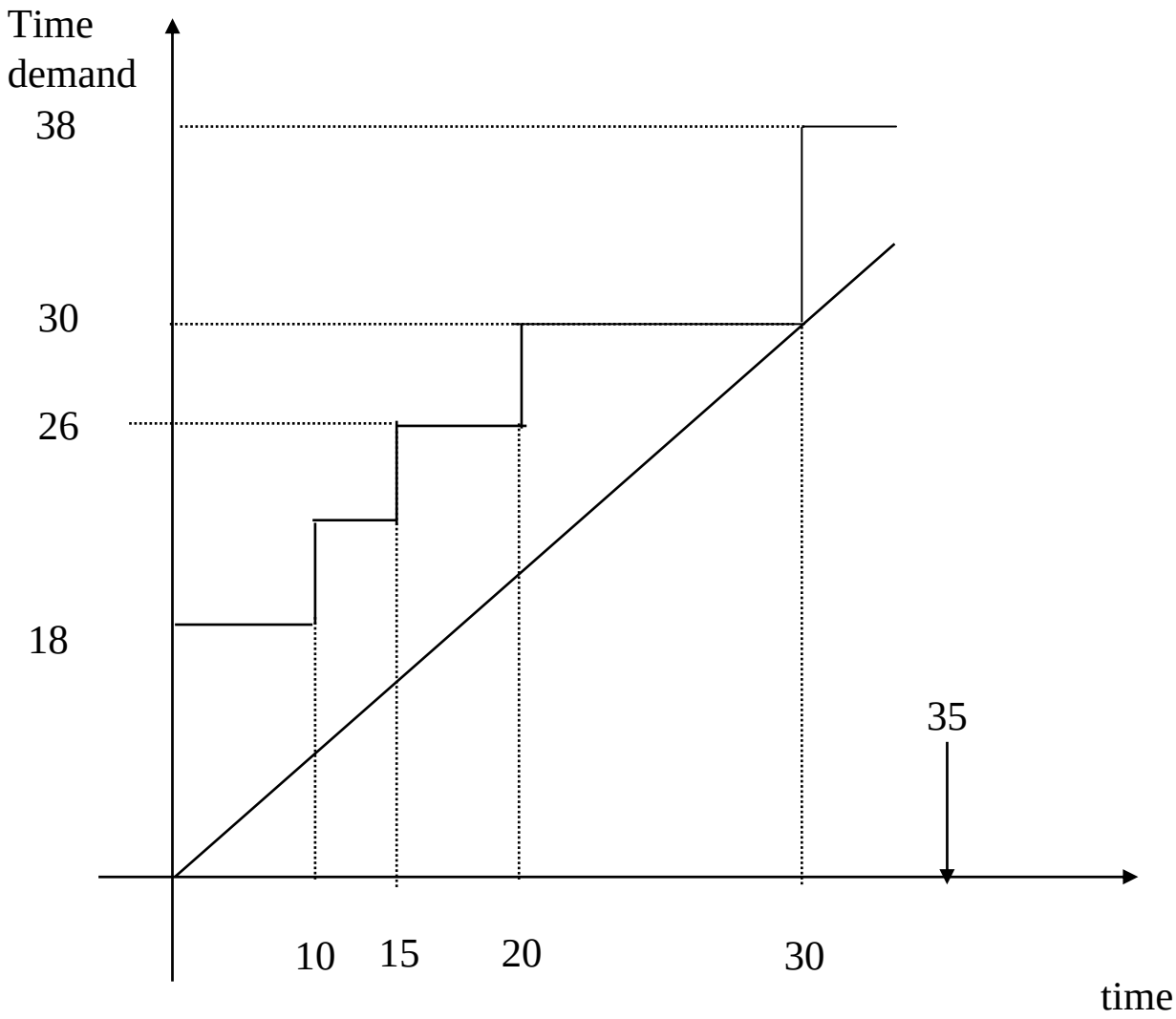
Critical instant theorem: If a task meets its first deadline when all higher priority tasks are started at the same time, then this task's future deadlines will always be met. The exact test for a task checks if this task can meet its first deadline[Liu73].



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

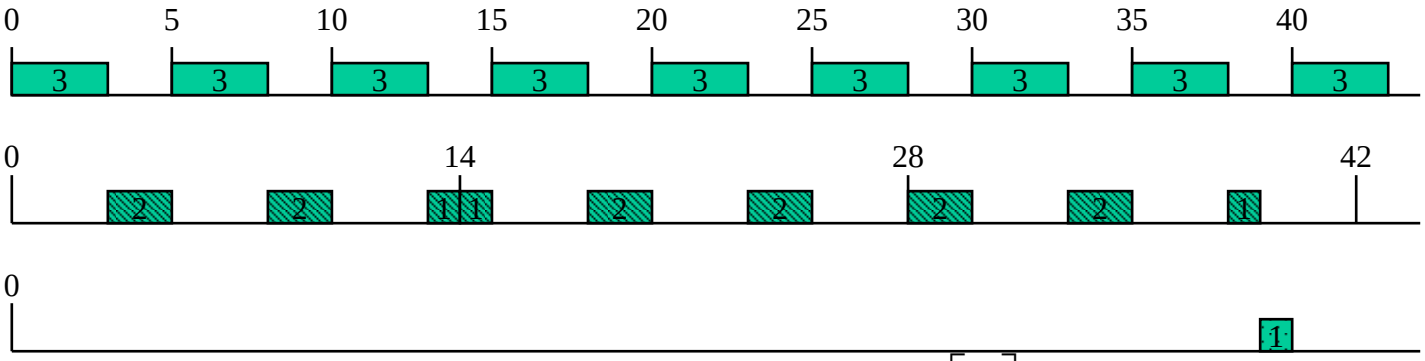
Class Exercise 1

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

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

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

Class Exercise 2: Task 2

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

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

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

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

Schedulable utilization bound

- Simpler method for the schedulabiity check

Utilization

- A periodic task's utilization U_i of an active resource is the ratio between its execution time and period: $U_i = C_i/p_i$
- Given a set of periodic tasks on an active resource, e.g. the CPU, the CPU's utilization is equal to the sum of periodic tasks' utilization:

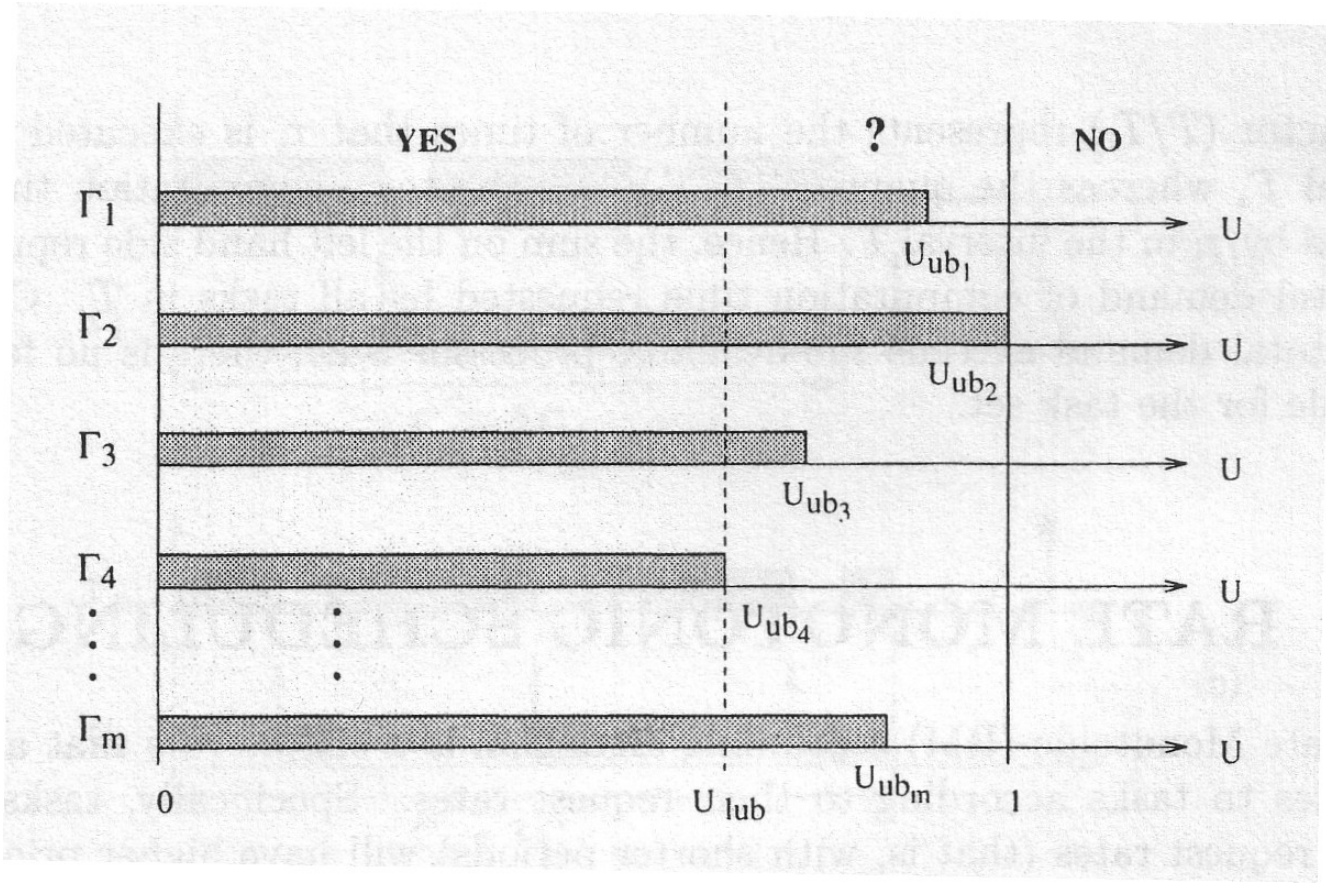
$$U = \sum_i \frac{C_i}{p_i}$$

- Can we find a bound called "schedulable utilization bound" under which a task set is guaranteed to be schedulable?

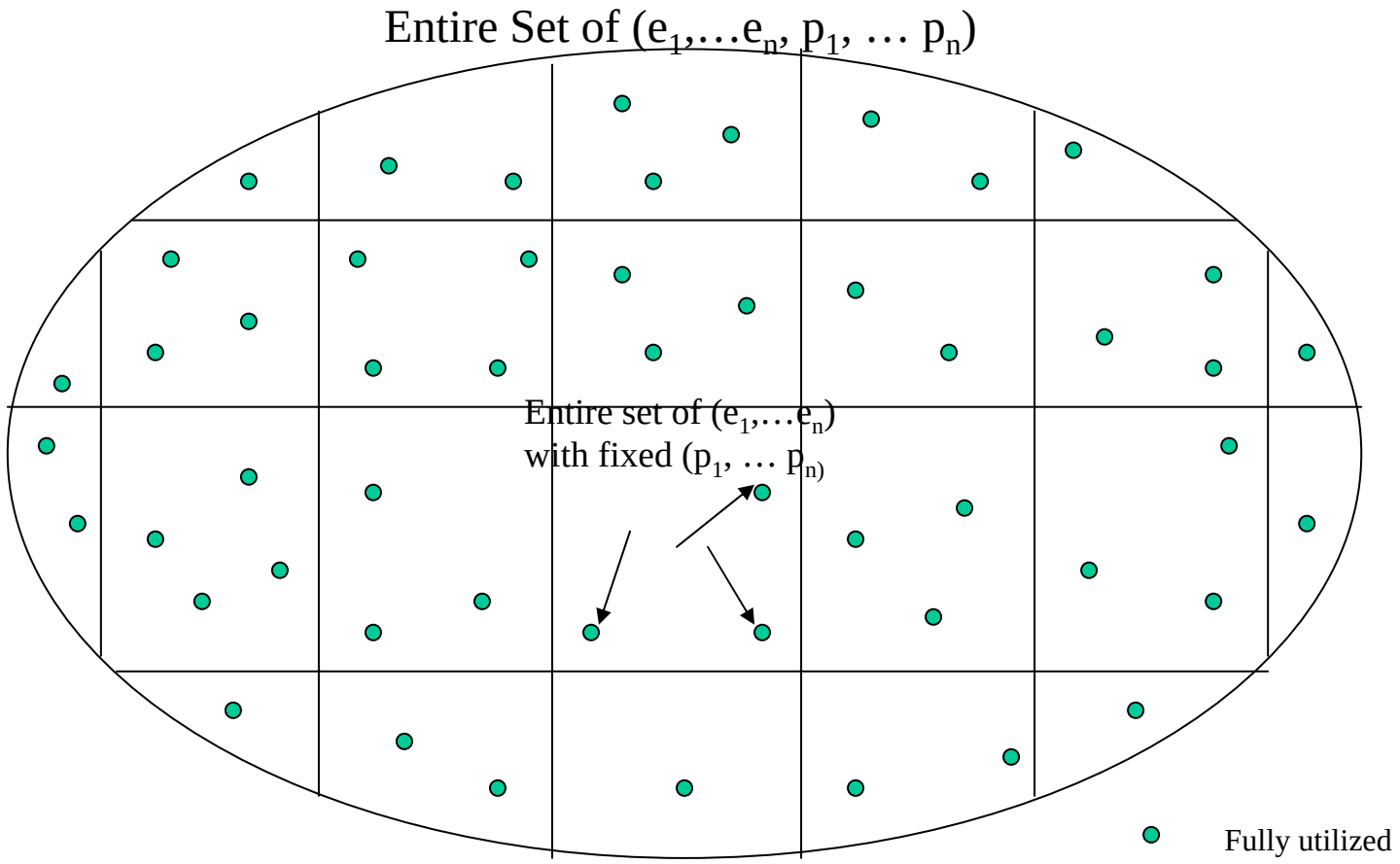
$$\text{if } U = \sum_i \frac{C_i}{p_i} \leq U_{bound}\text{, task set is schedulable}$$

Processor utilization factor

- For a given algorithm A, we are interested in finding its schedulability bound (e.g., the schedulability bound of EDF is 1)



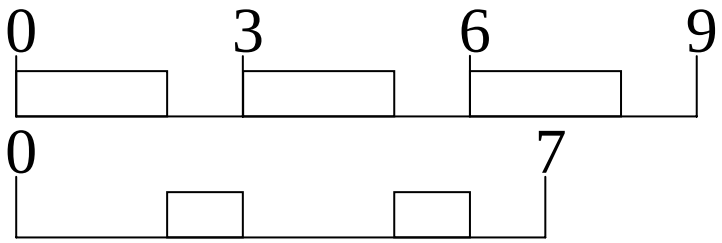
Processor utilization factor



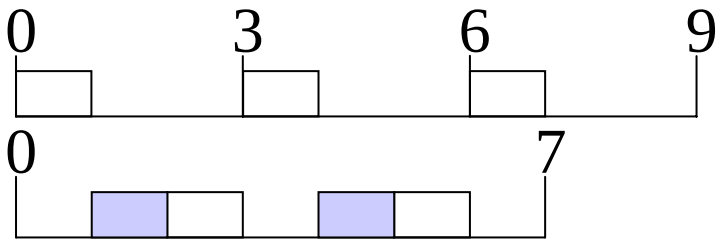
Find the minimum utilization factor among all fully utilized dots (barelly schedulable task set)

Which pattern of *e* and *p* values?

- Now, we can consider only (e and p) combinations that make the system barely schedulable.
- How to find a (e and p) combination that has the minimal utilization factor?
- Always start with examples→ intuition→ generic theorem



U = 2/3 + 2/7



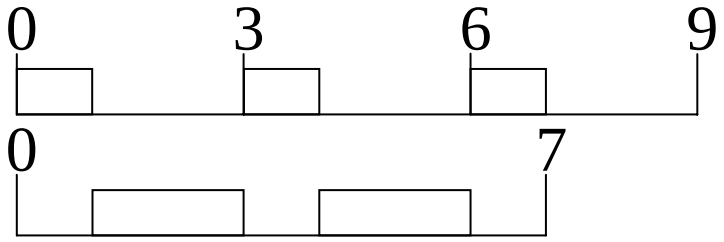
U = (2-Δ)/3 + (2+2Δ)/7

decrease : $\Delta/3$

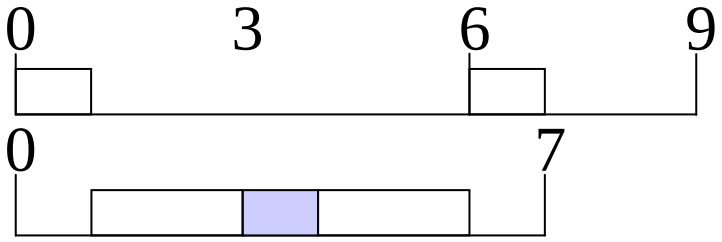
increase : $\Delta\lfloor 7/3 \rfloor /7 < \Delta(7/3)/7 = \Delta/3$

Which pattern of *p* values?

- For e values: “No-overflow theorem”
- What about p values?



U = 1/3 + 4/7



U = 1/(3*2) + (4+1)/7

Decrease: 1/3 – 1/(2*3)=1/(2*3)

Increase: 1/7

Since 7 > 2*3, U decreases

Transform (Ratio 3) to 2

$\left\lceil \frac{p_2}{p_1} \right\rceil e_1 + e_2 = p_2$; $where \left\lceil \frac{p_2}{p_1} \right\rceil = 3$, $e.g.$ $\left\lceil \frac{10}{4} \right\rceil 2 + 4 = 10$

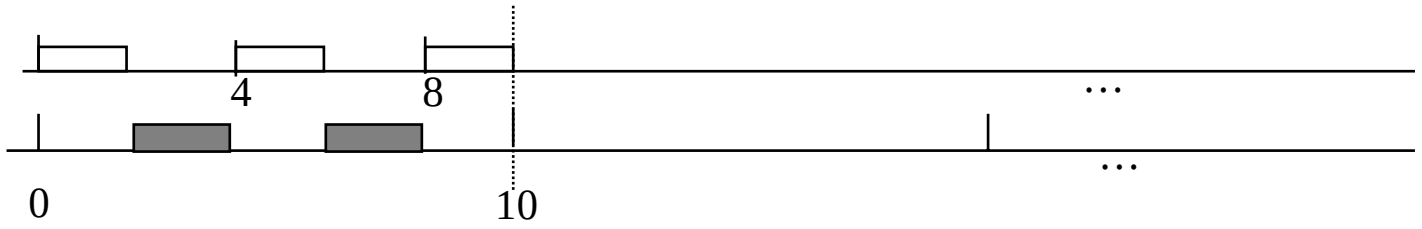
$\left\lceil \frac{p_2}{2p_1} \right\rceil e_1 + e_2 + e_1 = p_2$, $if \left\lceil \frac{p_2}{2p_1} \right\rceil = 2$ $\left\lceil \frac{10}{2*4} \right\rceil 2 + 4 + 2 = 10$

p_1 is doubled, so we obtain **ratio 2** among the periods

$(\frac{e_1}{p_1} + \frac{e_2}{p_2}) - (\frac{e_1}{2p_1} + \frac{e_2 + e_1}{p_2})$ $(\frac{2}{4} + \frac{4}{10}) - (\frac{2}{2*4} + \frac{4+2}{10}) > 0$

$= (\frac{e_1}{2p_1} - \frac{e_1}{p_2}) > 0$, $if\ p_2 > 2p_1$

$Since \left\lceil \frac{p_2}{p_1} \right\rceil = 3, we\ have\ 3 > \frac{p_2}{p_1} > 2.\ Thus, 1.5 > \frac{p_2}{2p_1} > 1$ and $\left\lceil \frac{p_2}{2p_1} \right\rceil = 2$; $p_2 > 2p_1$



Transform (Ratio k > 2) to 2

$\left\lceil \frac{p_2}{p_1} \right\rceil = k$, we have $k > \frac{p_2}{p_1} > k-1$. Thus, $\frac{k}{k-1} > \frac{p_2}{(k-1)p_1} > 1$ and $\left\lceil \frac{p_2}{(k-1)p_1} \right\rceil = 2$

$\left\lceil \frac{p_2}{p_1} \right\rceil e_1 + e_2 = p_2$; original

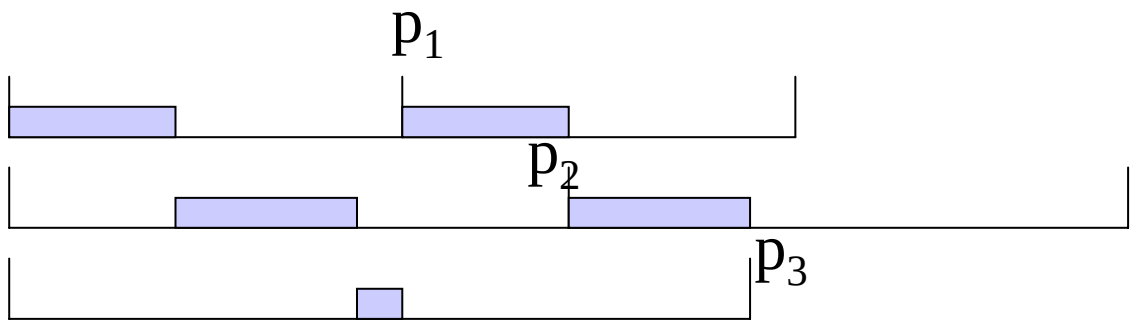
$\left\lceil \frac{p_2}{(k-1)p_1} \right\rceil e_1 + e_2 + (k-2)e_1 = 2e_1 + e_2 + (k-2)e_1 = p_2$; after transform

$$\left(\frac{e_1}{p_1} + \frac{e_2}{p_2}\right) - \left(\frac{e_1}{(k-1)p_1} + \frac{e_2 + (k-2)e_1}{p_2}\right)$$

$$= \left(\frac{(k-2)e_1}{(k-1)p_1} + \frac{(k-2)e_1}{p_2}\right) > 0, \text{ since } (k-1)p_1 < p_2$$

Putting Things Together

- The tasks' pattern that leads to minimal utilization is
  - the execution time shall not overflow
  - the period ratio between any pair of low priority task and high priority task should be less than 2 (and greater than 1)



The Remaining Free Variables

- The computation time for each task is as follows

$$e_1 = p_2 - p_1; e_2 = p_3 - p_2; \ldots; e_{n-1} = p_n - p_{n-1}$$

$$e_n = p_n - 2(e_1 + e_2 + \ldots + e_{n-1})$$

$$= p_n - 2(p_2 - p_1 + p_3 - p_2 + \ldots + p_n - p_{n-1})$$

$$= 2p_1 - p_n$$

- Quiz: what are the remaining free variables?

  what are the type of the free variables?

  what we want to minimize (or maximize)?

  what is the standard tool to get a maximum or minimal?

Solving the PDE

$$U = \frac{e_1}{p_1} + \ldots + \frac{e_n}{p_n} = \frac{p_2 - p_1}{p_1} + \ldots + \frac{p_n - p_{n-1}}{p_{n-1}} + \frac{2p_1 - p_n}{p_n}$$

$$= \frac{p_2}{p_1} + \frac{p_3}{p_2} + \ldots + \frac{p_n}{p_{n-1}} + \frac{2p_1 p_2 \ldots p_{n-1}}{p_2 \ldots p_{n-1} p_n} - n$$

$$= r_1 + r_2 + \ldots + r_{n-1} + \frac{2}{r_1 r_2 \ldots r_{n-1}} - n; \quad where \quad r_i = \frac{p_{i+1}}{p_i}$$

$$set \frac{\partial U}{\partial r_i} = 0; \; we\ have$$

$$r_1^2 r_2 \ldots r_{n-1} = 2 \quad (1); \; r_1 r_2^2 \ldots r_{n-1} = 2 \quad (2); \ldots; \; r_1 r_2 \ldots r_{n-1}^2 = 2 \quad (n-1)$$

$(1)/(2) = 1 \Rightarrow r_1 = r_2$

Dividing them successively, we have $r_1 = r_2 = \ldots = r_{n-1}$

Plug it back to (1) we have $r = 2^{1/n}$

$$U = (n-1)2^{1/n} + \frac{2}{2^{(n-1)/n}} - n = n(2^{1/n} - 1)$$

The L&L Bound

A set of n periodic task is schedulable if :

$$\frac{e_1}{p_1} + \frac{e_2}{p_2} + \ldots + \frac{e_n}{p_n} \leq n(2^{1/n} - 1)$$

- U(1) = 1.0 U(4) = 0.756 U(7) = 0.728
- U(2) = 0.828 U(5) = 0.743 U(8) = 0.724
- U(3) = 0.779 U(6) = 0.734 U(9) = 0.720

- For harmonic task sets, the utilization bound is U(n)=1.00 for all n. For large n, the bound converges to *ln 2* ~ 0.69.

- The L&L bound for rate monotonic algorithm is one of the most significant results in real-time scheduling theory. Its derivation also shows a wealth of analysis techniques that are useful in many new situations when considering static priority scheduling.

Summary of Utilization Bound

- The minimum utilization factor among all barely schedulable task sets is a sufficient bound for the schedulability
- One time check with simple comparison
- 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
  - Only sufficient condition
  - we cannot say anything if utilization is higher than the bound – safe choice is to assume it is not schedulable

Practical Issues

- Practical Issues
  - What if there is a non-preemptable code section (e.g., system call)?
  - What if the context switch overhead is not negligible?
  - Tick scheduling?
  - The deadline is earlier than the period?

Non-preemptable code section

- a non-preemptable code section (NPS) of a low priority task ***blocks*** high priority task
  - How to take this into account in time-demand analysis?

$$b_i = \max_{j=i+1}^{n} NPS_j$$

$$r_i = e_i + b_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i}{p_j} \right\rceil e_j$$

Non-preemptable code section

- a non-preemptable code section (NPS) of a low priority task ***blocks*** high priority task
  - How to take this into account in utilization bound check?

$$b_i = \max_{j=i+1}^{n} NPS_j$$

$$\sum_{j=1}^{i-1} \frac{e_j}{p_j} + \frac{e_i + b_i}{p_i} \leq U(i);\ \text{task by task check}$$

$$\sum_{j=1}^{n} \frac{e_j}{p_j} + \max_{j=1}^{n} \frac{b_j}{p_j} \leq U(n);\ \text{single check}$$

Deadline earlier than period?

- Time-demand analysis naturally works for this case
- What about utilization bound check?
  - Two utilization inflation methods

$$D_i < p_i$$

$$\sum_{j=1}^{i-1} \frac{e_j}{p_j} + \frac{e_i + p_i - D_i}{p_i} \le U(i);\ \text{increase execution time}$$

$$\sum_{j=1}^{i-1} \frac{e_j}{p_j} + \frac{e_i}{D_i} \le U(i);\ \text{decrease period}$$