Overview

- Today: this lecture introduces real-time scheduling theory

- **To learn more on real-time scheduling terminology**:
- **see basic concepts on "Hard Real-Time Computing Systems" book from G. Buttazzo** (useful chapters are in the Lab!)
- **see chapter 4 on "Hard Real-Time Computing Systems" book from G. Buttazzo**

- Basic tutorial at:

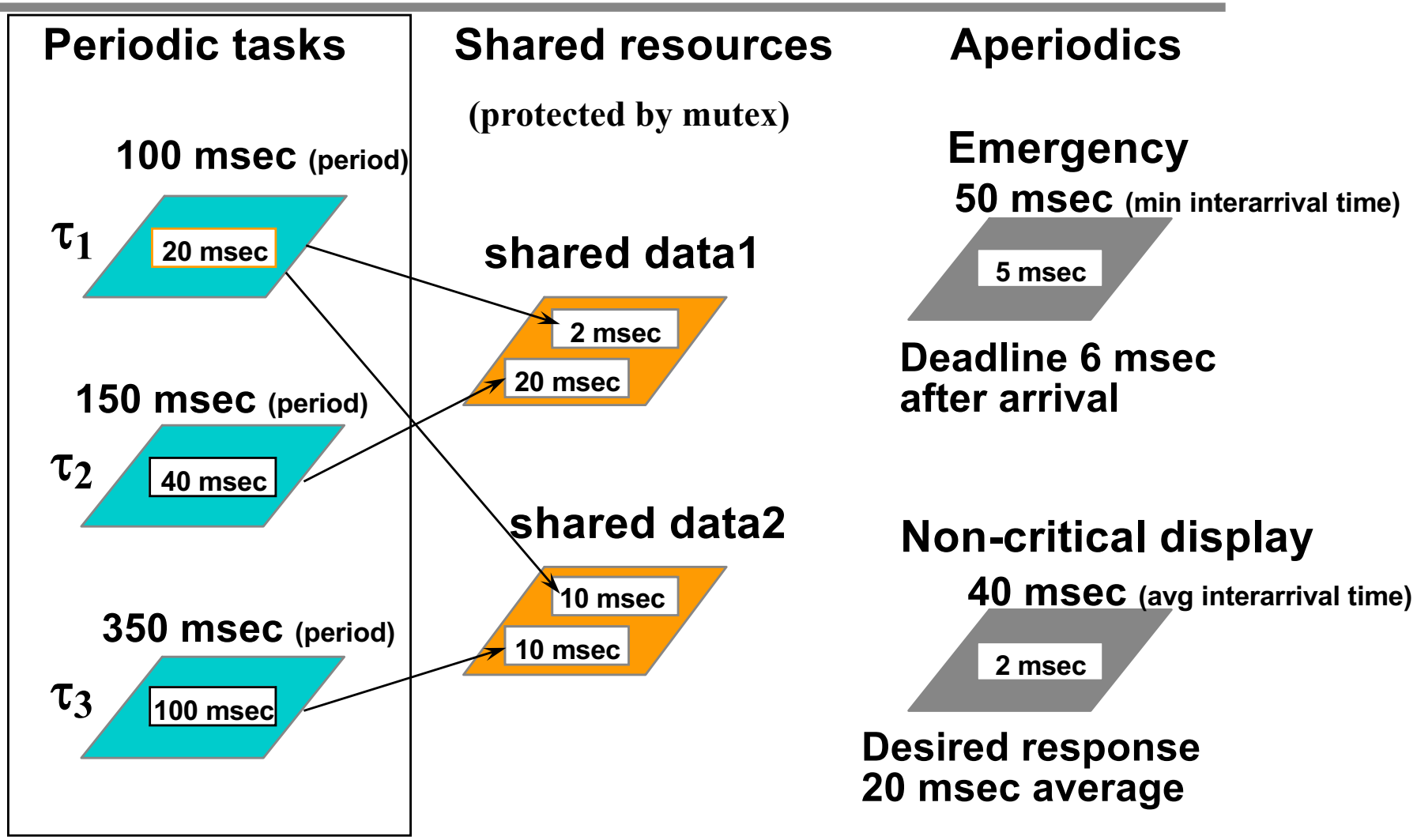
http://www.embedded.com/electronics-blogs/beginner-s-corner/4023927/Introduction-to-Rate-Monotonic-Scheduling#

So What is a Real-Time System?

- A **real-time system** is a system whose specification includes both logical and temporal correctness requirements.

  - ✓ *Logical Correctness*: produces correct outputs.
  - ✓ *Temporal Correctness*: produces outputs at the right time.

A Sample Problem



Goal: guarantee that no real-time deadline is missed!!!

Real-time scheduling algorithms

- Jobs can be scheduled according to the following scheduling algorithms:

  - **Rate Monotonic (RM)**: the faster the rate, the higher is the priority. All the jobs in a task have the same priority and hence the name "fixed priority" algorithm.

  - **Earliest Deadline First (EDF)**: the job with the earliest deadline has the highest priority. Jobs in a task have different priorities and hence the name, "dynamic priority" algorithm.

What is GRMS for?

As the hardware becomes much faster and the RTOS market is gaining wide spread usage, the key required skill is no longer the ability of writing assembly programs that execute on bare custom hardware.

The skills in demand now become the development of large and sophisticated multi-threaded real time applications.

GRMS is a practical theory that will enable you to design and analyze multi-threaded real time applications.

The L&L Bound

- A set of n periodic tasks is schedulable if:

$$\frac{c_1}{p_1} + \frac{c_2}{p_2} + \ldots + \frac{c_n}{p_n} \leq n\left(2^{1/n} - 1\right)$$

- 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. Otherwise, for large n, the bound converges to *ln 2* ~ 0.69.

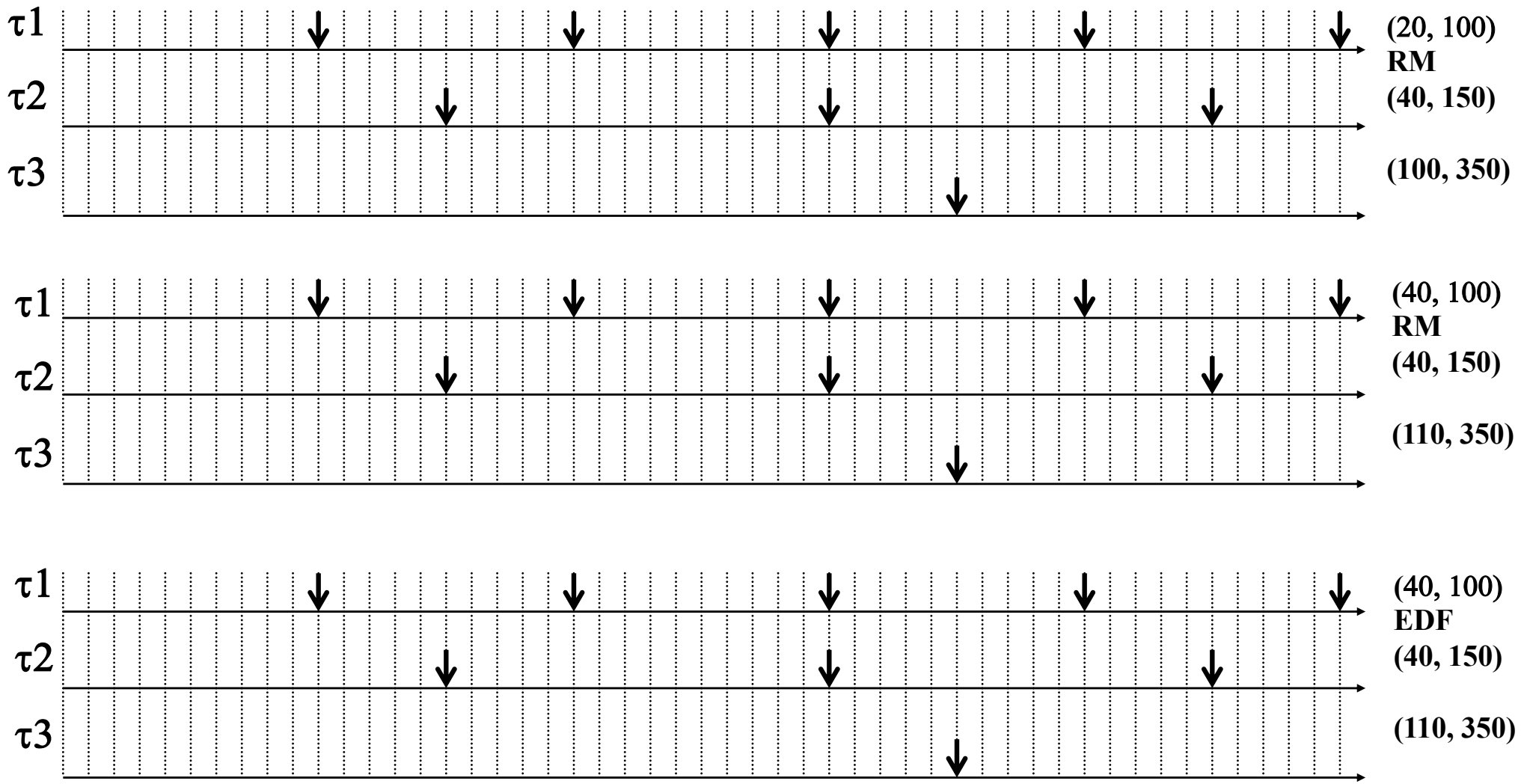
- The L&L bound for rate monotonic algorithm is one of the most significant results in real-time scheduling theory. It allows to check the schedulability of a group of tasks with a single test! It is a sufficient condition; hence, it is inconclusive if it fails!

Sample Problem: Applying UB Test

	C	P	U
Task $\tau 1$:	**20**	**100**	**0.200**
Task $\tau 2$:	**40**	**150**	**0.267**
Task $\tau 3$:	**100**	**350**	**0.286**

- Are all the tasks schedulable?
- What if we double the execution time of task $\tau 1$?

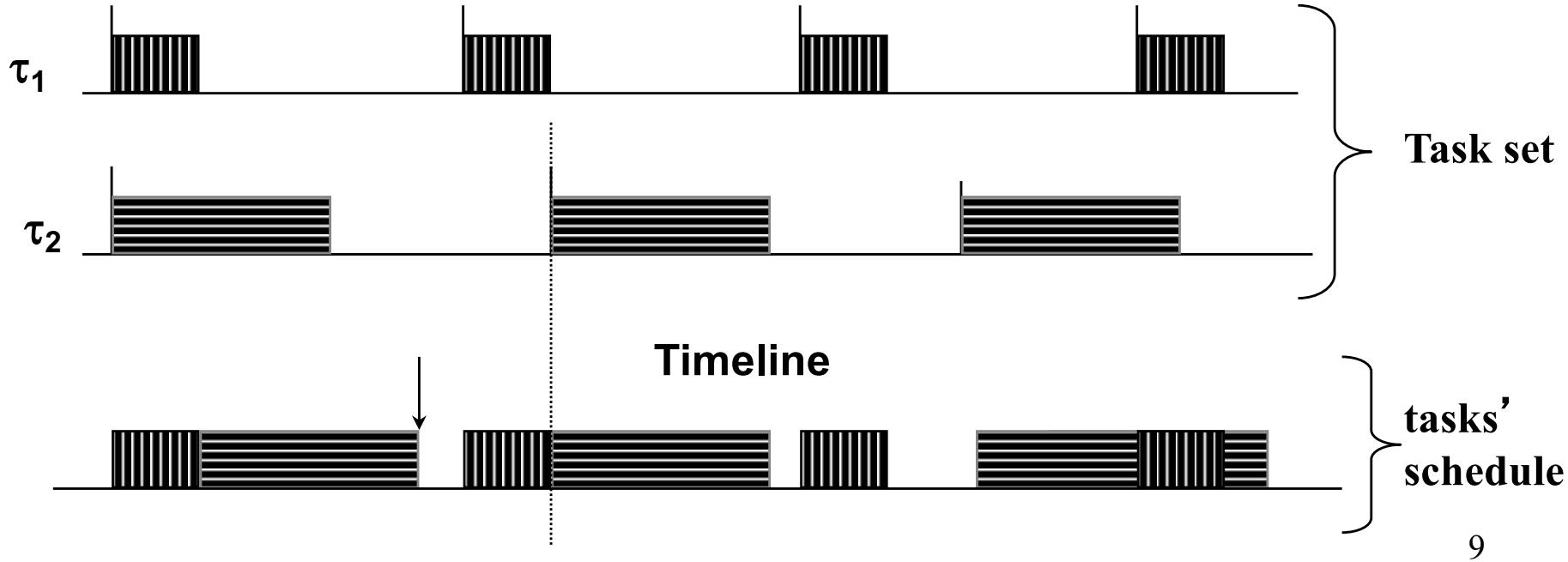
Sample Problem: draw the schedule by using RM and EDF



The Exact Schedulability Test

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].

It holds only for fixed priority scheduling!



Exact Schedulability Test (Exact Analysis)

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

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.

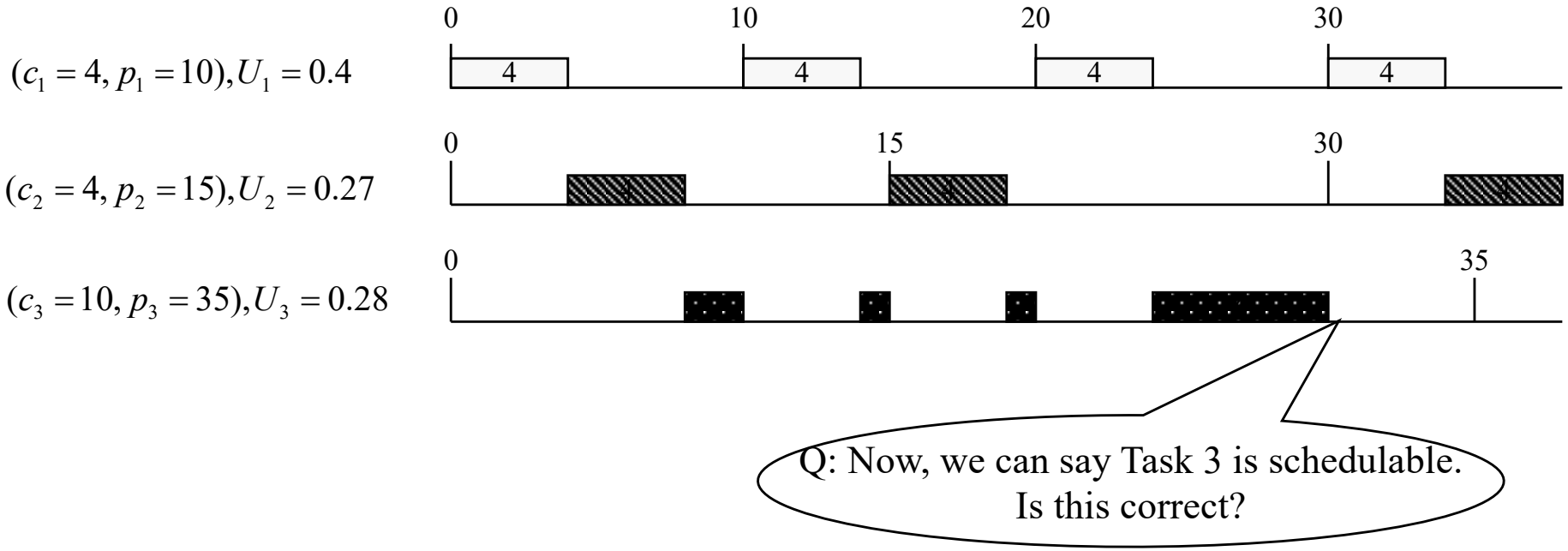
The superscript *k* indicates the number of iterations in the calculation.
The index *i* indicates it is the *ith* task being checked.

The index *j* runs from 1 to i-1, i.e. all the higher priority tasks. Recall from the convention - task 1 has a higher priority than task 2 and so on.

<u>We check the schedulability of a single task at the time!!!</u>

The Exact Schedulability Test

- Basically, "*Enumerate*" the schedule
- "Task by Task" schedulability test

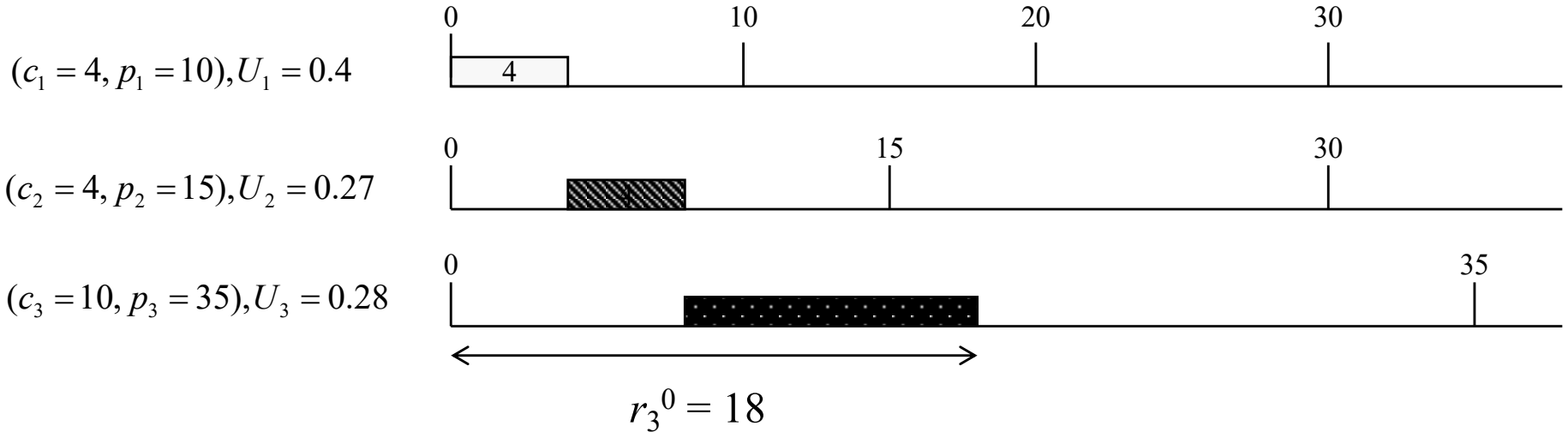


Intuitions of Exact Schedulability Test

- Obviously, the response time of task 3 should be larger than or equal to $c_1+c_2+c_3$

$$r_3^0 = \sum_{j=1}^{3} c_j = c_1 + c_2 + c_3 = 4 + 4 + 10 = 18$$

Intuitions of Exact Schedulability Test



Intuitions of Exact Schedulability Test

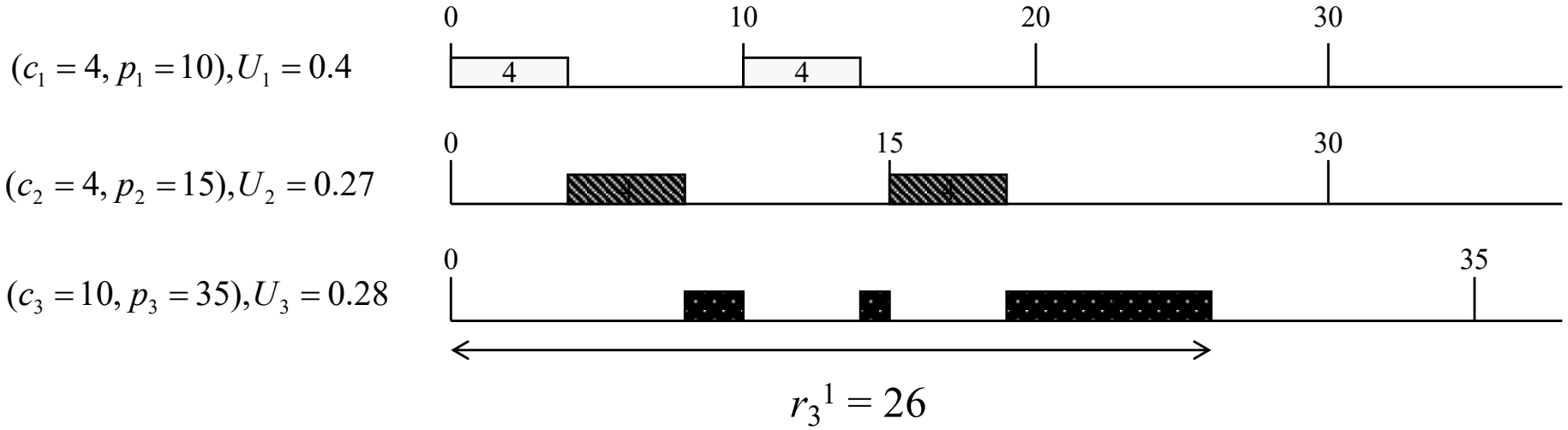
- Obviously, the response time of task 3 should be larger than or equal to $c_1+c_2+c_3$

$$r_3^0 = \sum_{j=1}^{3} c_j = c_1 + c_2 + c_3 = 4 + 4 + 10 = 18$$

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

$$r_3^1 = c_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^0}{p_j} \right\rceil \cdot c_j = 10 + \left\lceil \frac{18}{10} \right\rceil 4 + \left\lceil \frac{18}{15} \right\rceil 4 = 26$$

Intuitions of Exact Schedulability Test



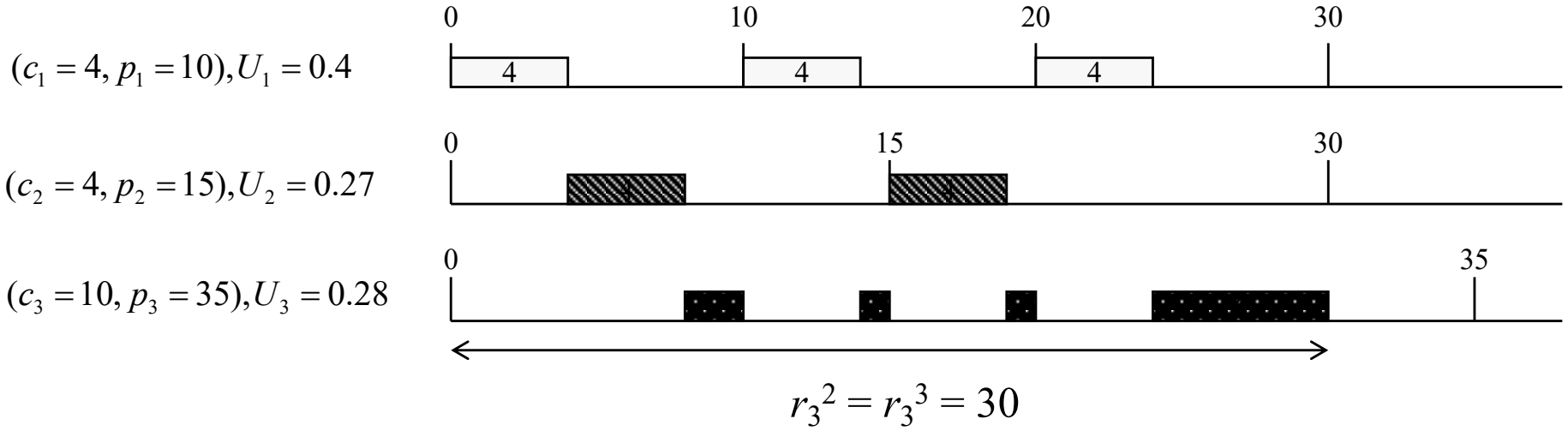
Intuitions of Exact Schedulability Test

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

$$r_3^2 = c_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^1}{p_j} \right\rceil \cdot c_j = 10 + \left\lceil \frac{26}{10} \right\rceil 4 + \left\lceil \frac{26}{15} \right\rceil 4 = 30$$

$$r_3^3 = c_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{p_j} \right\rceil \cdot c_j = 10 + \left\lceil \frac{30}{10} \right\rceil 4 + \left\lceil \frac{30}{15} \right\rceil 4 = 30 \qquad \textbf{Done!}$$

Intuitions of Exact Schedulability Test



Intuition for the Exact Schedulability Test

- Suppose we have n tasks, and we pick a task, say i, to see if it is schedulable.
- We initialize the testing by assuming all the higher priority tasks from 1 to i-1 will only preempt task i once.
- Hence, the initially presumed finishing time for task i is just the sum of C_1 to C_i, which we call r^0.
- We now check the actual arrival of higher priority tasks within the duration r^0 and then presume that it will be all the preemption task i will experience. So we compute r^1 under this assumption.
- We will repeat this process until one of the two conditions occur:
  - 1. The r^n eventually exceeds the deadline of task i. In this case we terminate the iteration process and conclude that task i is not schedulable.
  - 2. The series r^n converges to a fixed point (i.e., it stops increasing). If this fixed point is less than or equal to the deadline, then the task is schedulable and we terminate the schedulability test.

Assumptions under UB & Exact Analysis

- Both the Utilization Bound and the Exact schedulability test make the following assumptions:

  - All the tasks are periodic
  - Tasks are scheduled according to RMS
  - All tasks are independent and do not share resources (data)
  - Tasks do not self-suspend during their execution
  - Scheduler overhead (context-switch) is negligible