Scheduling and schedulers

Dr. Bystrov

School of Electrical Electronic and Computer Engineering

University of Newcastle upon Tyne

Introduction

- Remember concurrent programming? How is concurrency implemented in a system with a single processor?
- Remember concurrent models? How close to the “true concurrency” can we approach?
- An Operating System (OS) – is it incomprehensibly complex?
- Is it possible to guarantee compliance with the Real-Time constraints in some particular system?
  - What is Real-Time?
  - How to measure it?
  - Which scheduling algorithm to choose?
  - Is there the best scheduler for an application?

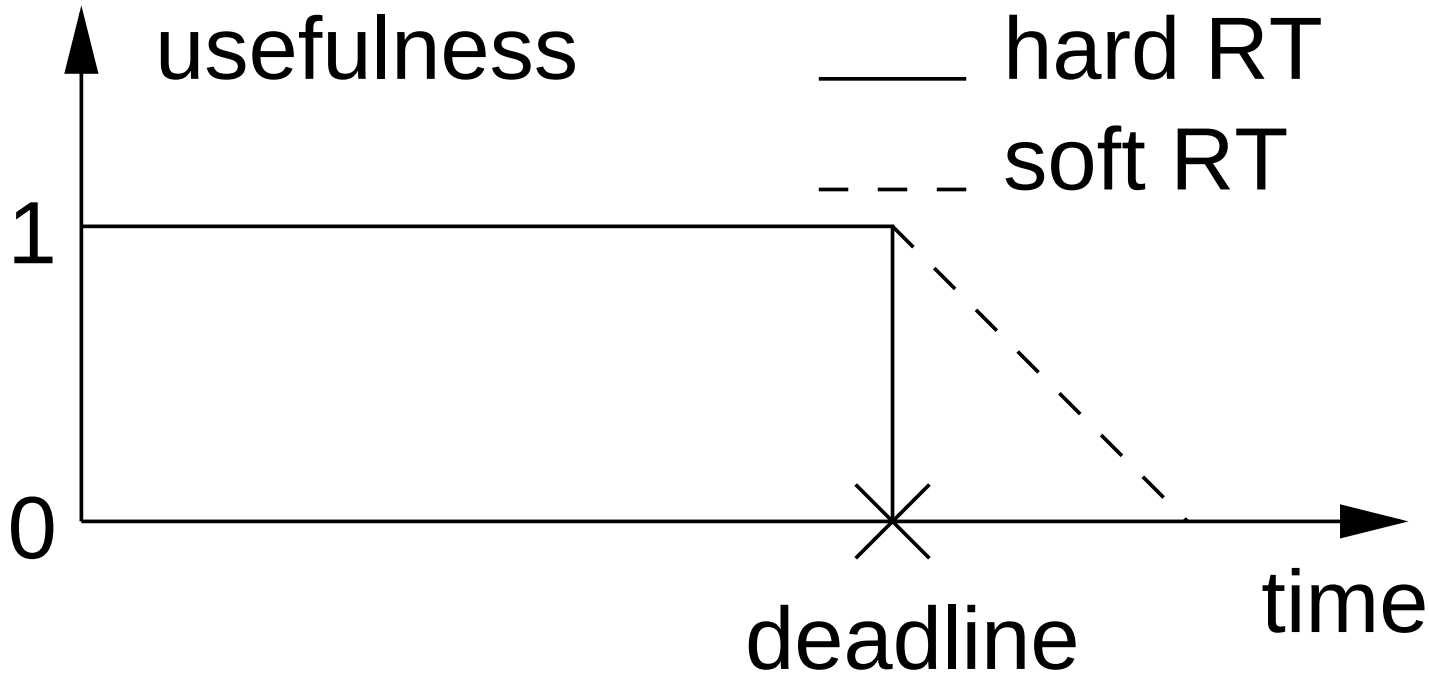
Taxonomy

Dimensions:

- Hard vs. Soft Real-Time
- Time-Triggered vs. Event-Triggered
- Cooperative vs. Preemptive
- Static vs. Dynamic (on/off-line)
- Optimal vs. heuristic

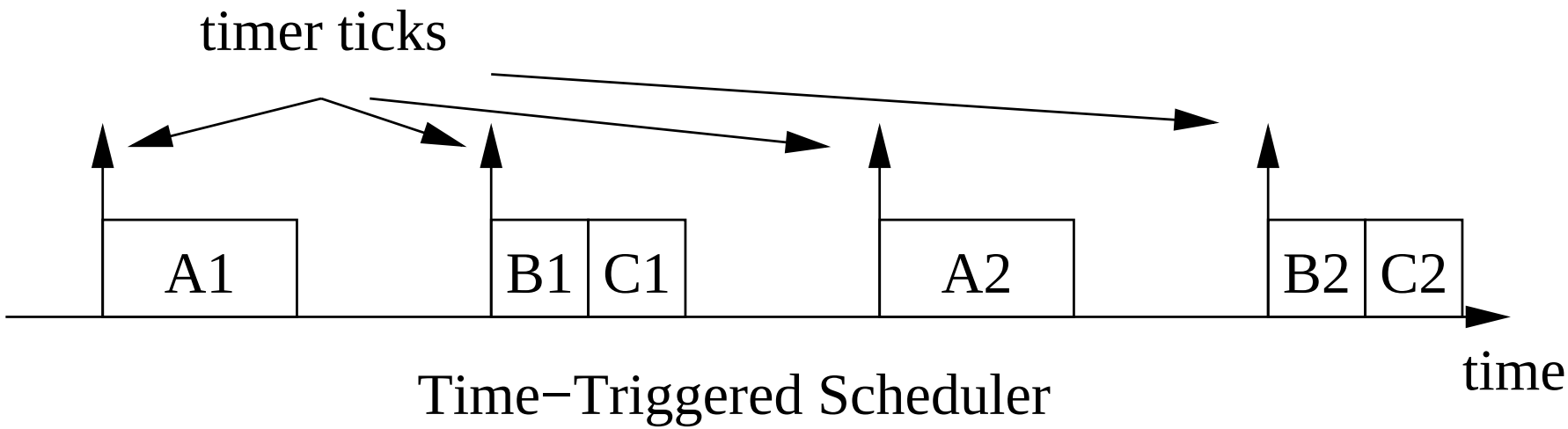
The permutations of the above dimensions give us the taxonomy of schedulers

Real-Time constraints

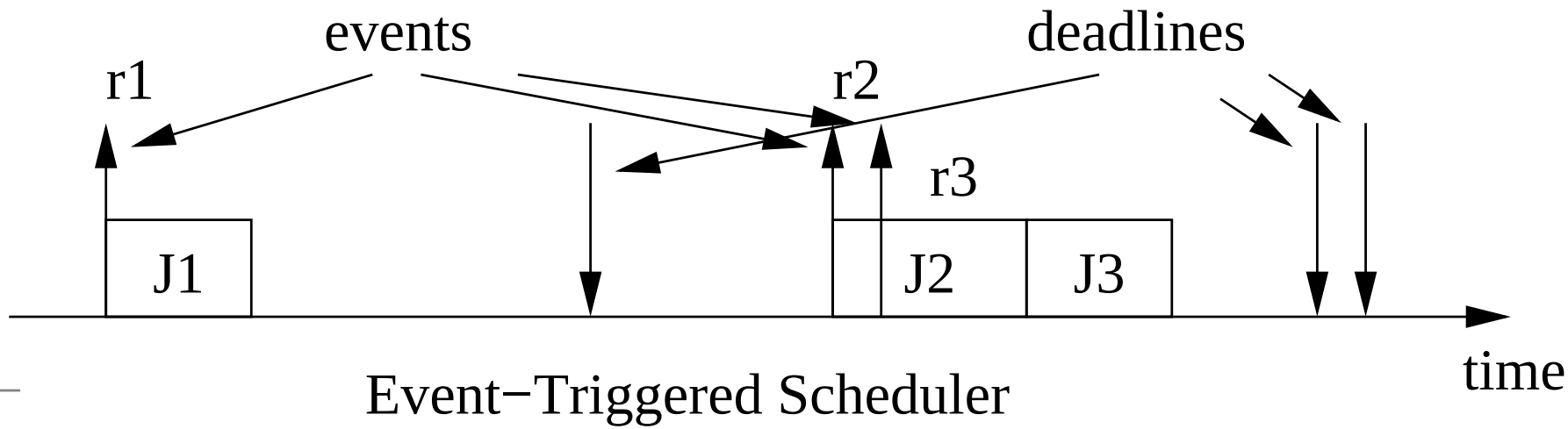


- Value or “usefulness” as a function of time
- It is important to meet the deadline
- High or low performance is not important – “just do it by the deadline”

Time-Triggered vs. Event-Triggered

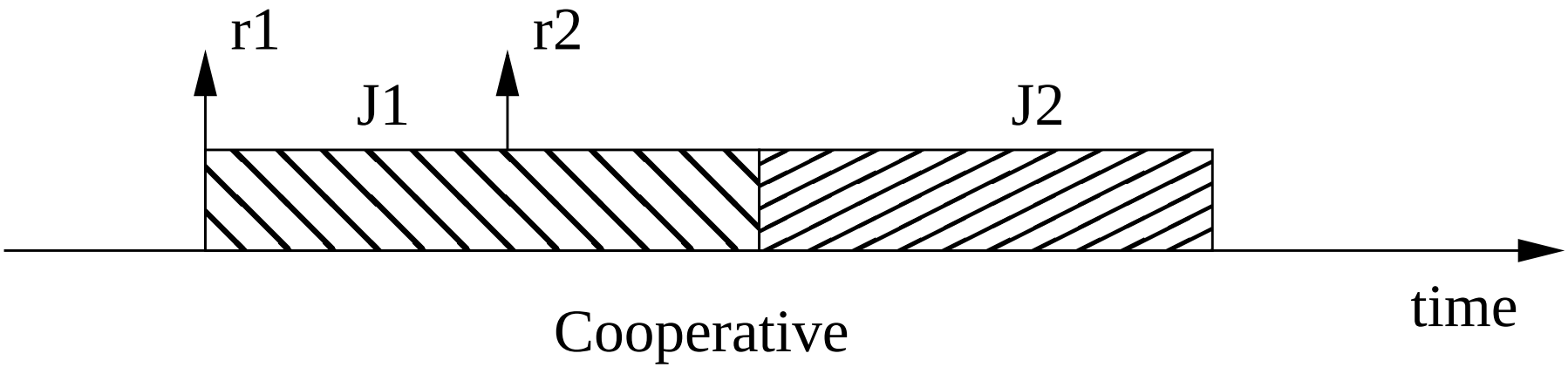


Time−Triggered Scheduler

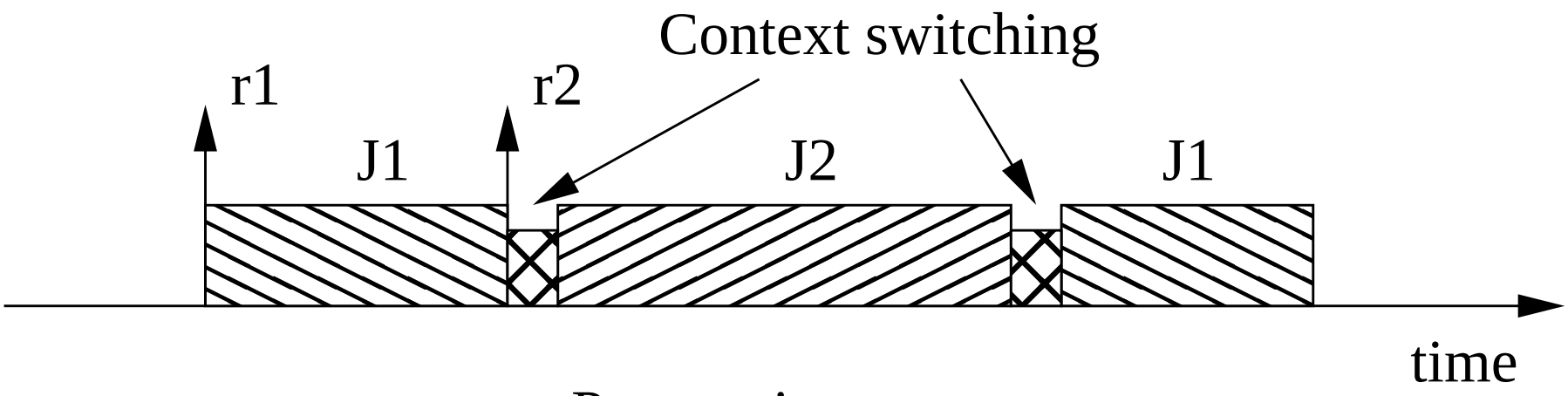


Event−Triggered Scheduler

Cooperative vs. Preemptive



Cooperative

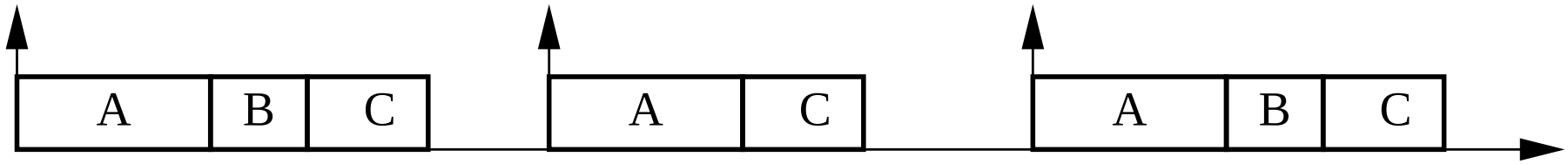


Preemptive

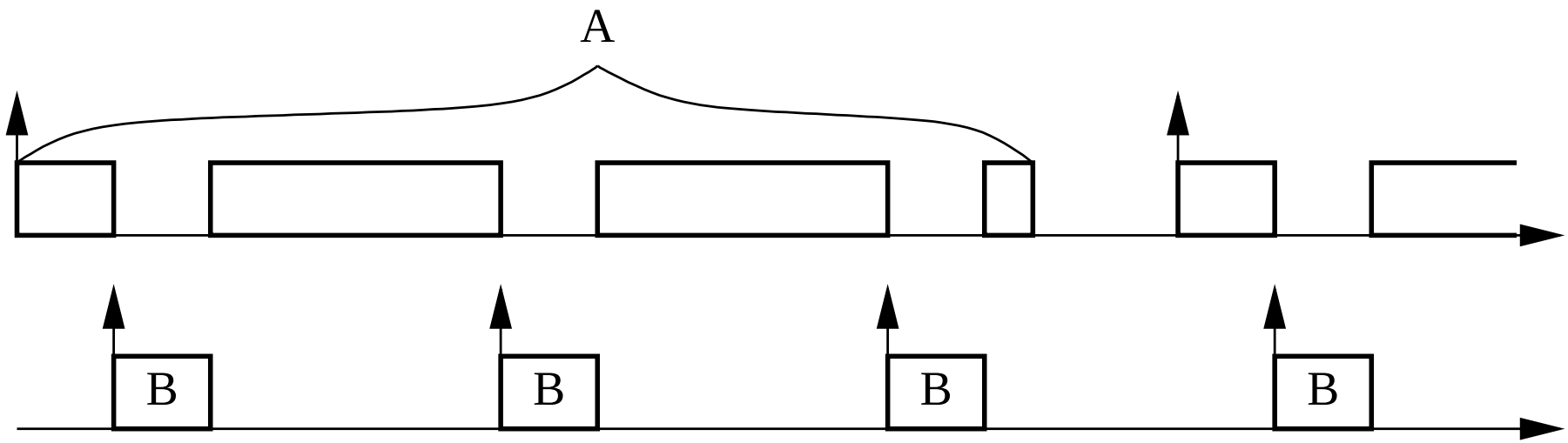
Example: Taxonomy of TTS or ETS

- Preemptive – non-preemptive: preemption means interrupting of one task with the other. Task priorities may be employed.
- Static – dynamic: scheduling decision are based on parameters which can be fixed or not before task activation
- Offline – online: scheduling decisions are made in advance or in the course of system operation
- Optimal – heuristic: An optimal algorithm minimises some given cost function defined for the task set. Feasibility is a simple example of the cost function. Heuristic algorithms use rules, which do not guarantee optimality.

Example: TTS (1,2)

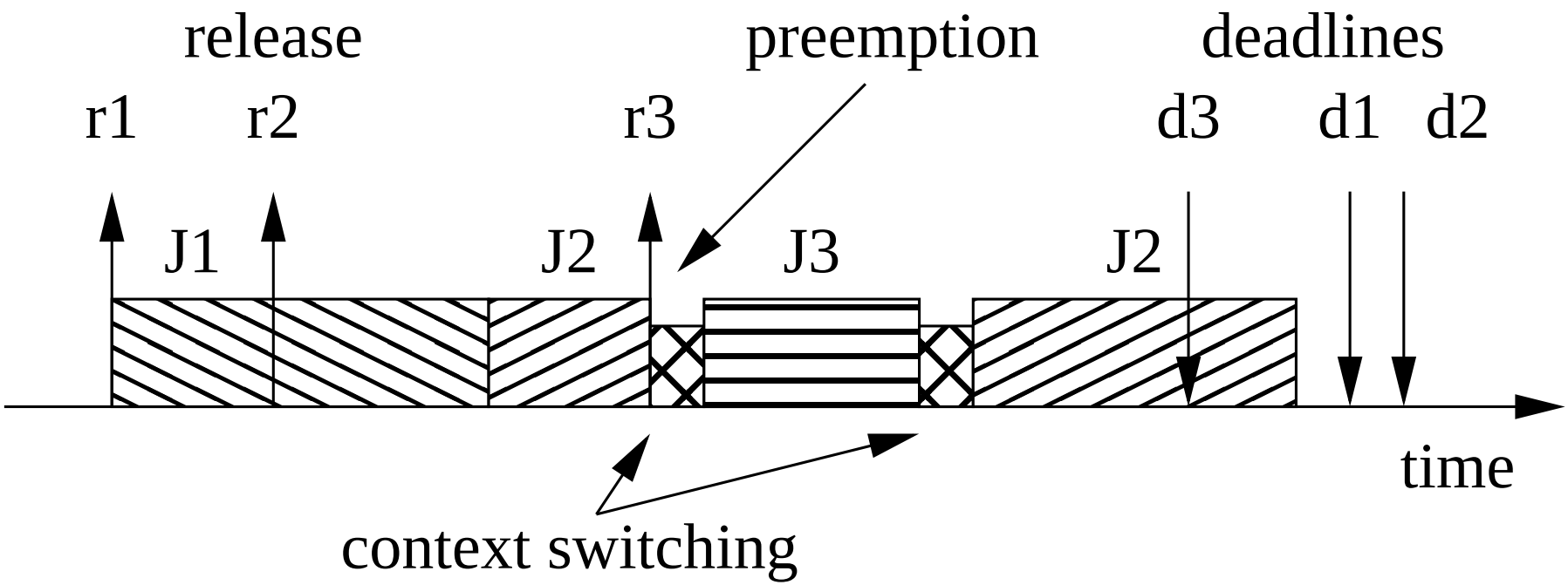


(a) Timeline scheduler



(b) Rate Monotonic scheduler

Example: ETS, EDF, preemptive



Task parameters (1)

- Task J_i or τ_i
- Arrival time a_i or r_i (Release Time, Request) $\uparrow$
- Computation Time C_i (exec. without interruption)
- Abs. Deadline d_i $\downarrow$
- Rel. Deadline D_i, $D_i = d_i - r_i$ $\downarrow$
- Start Time s_i
- Finishing Time f_i

Task parameters (2)

- Response Time R_i, $R_i = f_i - r_i$
- Criticality: hard/soft
- Value v_i
- Lateness L_i, $L_i = f_i - d_i$ (negative is good!)
- Tardiness or Exceeding time E_i, $E_i = max(0, L_i)$
- Laxity or Slack Time X_i, $X_i = d_i - a_i - C_i$
  (the max time a task can be delayed on its activation to complete within its deadline)

Task parameters (3)

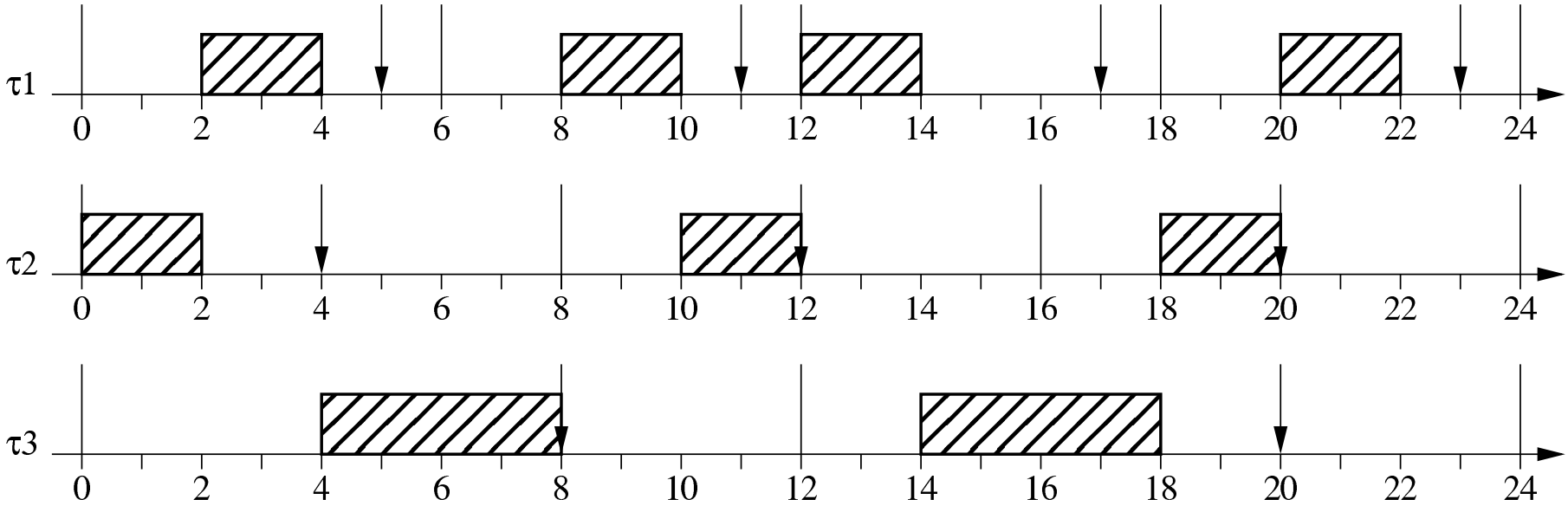
- Critical Instant – the time at which the release of the task will produce the largest response time
- Critical Time Zone – the interval between the critical instant and the response time of the corresponding request of the task
- Relative Release Jitter – the max deviation of the start time of two consecutive instances
  $RRJ_i = \max_k \left|(s_{i,k} - r_{i,k}) - (s_{i,k-1} - r_{i,k-1})\right|$
- Absolute Release Jitter – the max deviation of the start time among all instances
  $ARJ_i = \max_k(s_{i,k} - r_{i,k}) - \min_k(s_{i,k} - r_{i,k})$
- The same for the Execution and Finishing Jitter, Absolute or Relative.

Example: Construct a schedule (1)

Construct an EDF (Earliest Deadline first) schedule. In this table C_i denotes execution time, D_i – relative deadline, T_i – period and τ_i – task name. Use the non-preemptive execution model.

task	C_i	D_i	T_i
τ_1	2	5	6
τ_2	2	4	8
τ_3	4	8	12

Example: Construct a schedule (2)



Example: calculate task parameters

- Absolute Release Jitter *for* τ_1 *is 2*
- Absolute Finishing Jitter *for* τ_2 *is 2*
- Absolute Execution Jitter *for* τ_3 *is 0.*

RM optimality (1)

Read the book! Two "keys" to the proof:

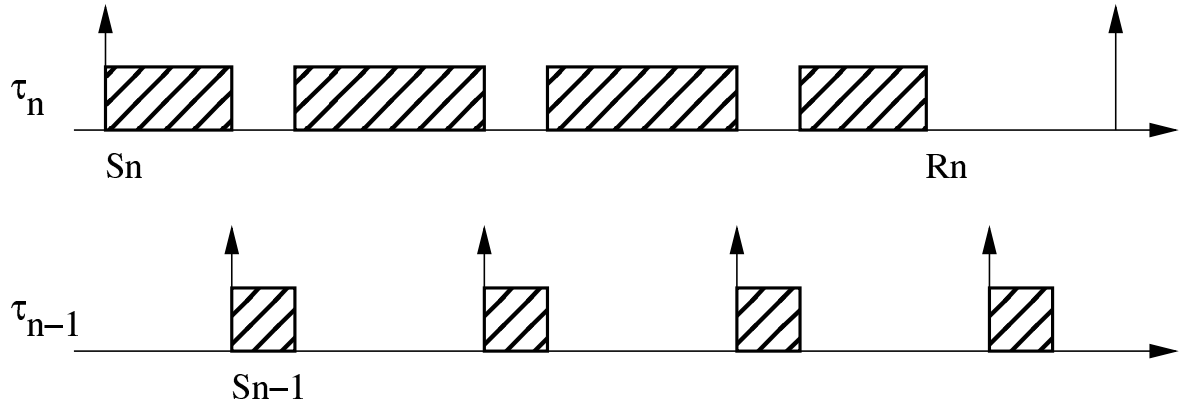
- "Given two periodic tasks τ_1 and τ_2, with $T_1 < T_2$, if the schedule is feasible by an arbitrary priority assignment, then it is also feasible by RM. That is, RM is optimal." [Buttazzo book]

First step: show that a critical instant for any task occurs whenever the task is released simultaneously with all higher-priority tasks.

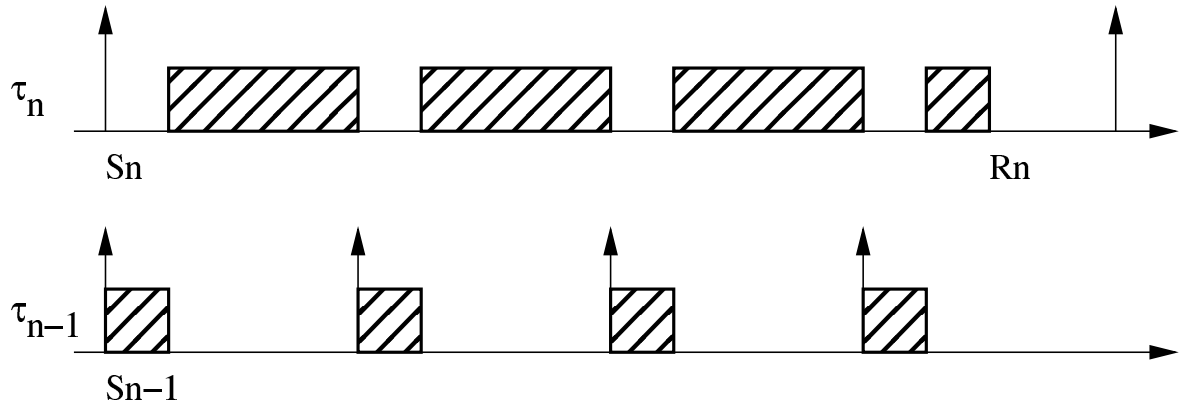
- Schedulability is checked on critical instants!

Second step: show that if a task set is schedulable by an arbitrary priority assignment, then it is also schedulable by RM.

RM optimality (2)



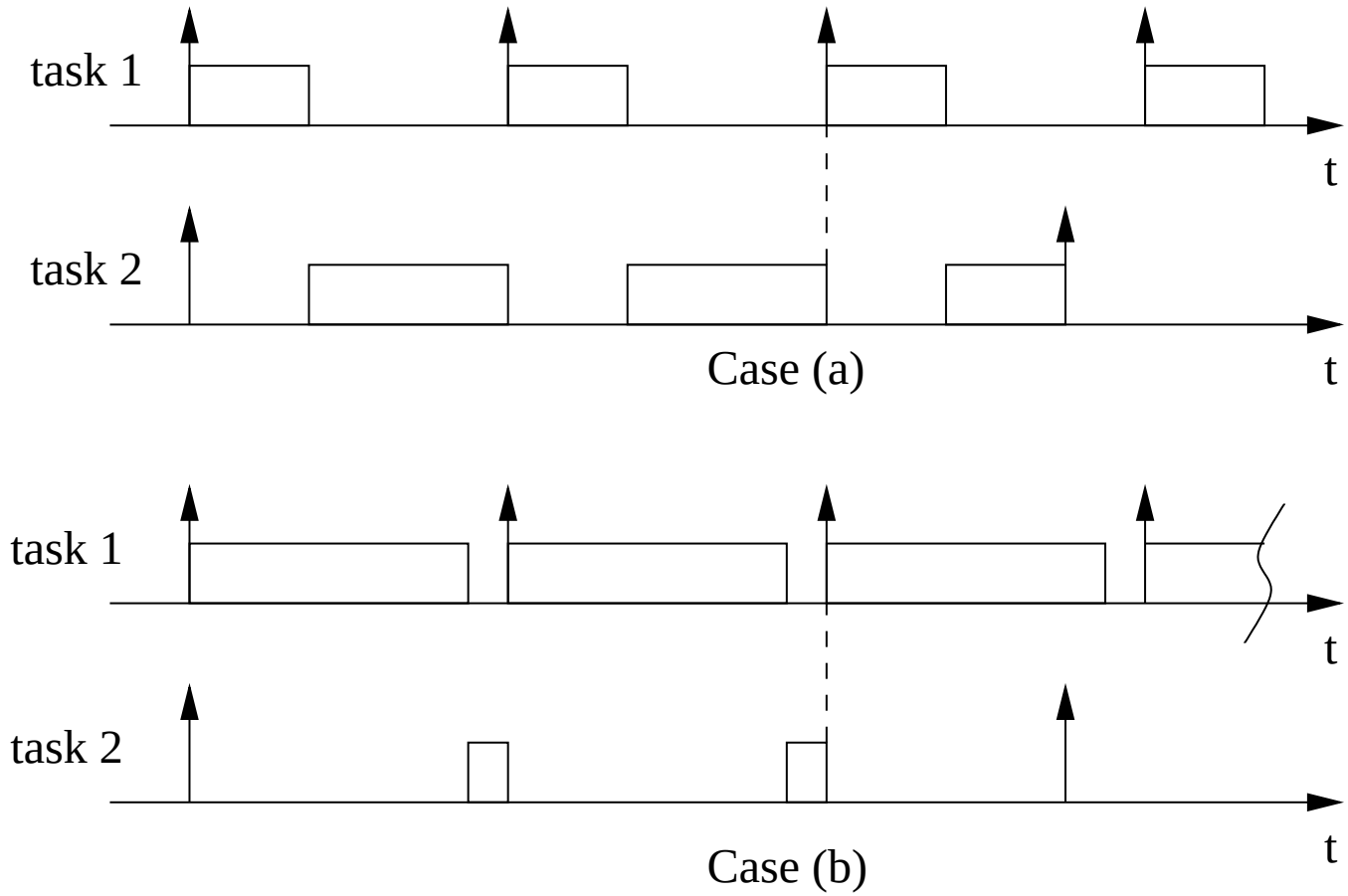
(a) Different release time Sn and Sn−1s — shorter response time Rn



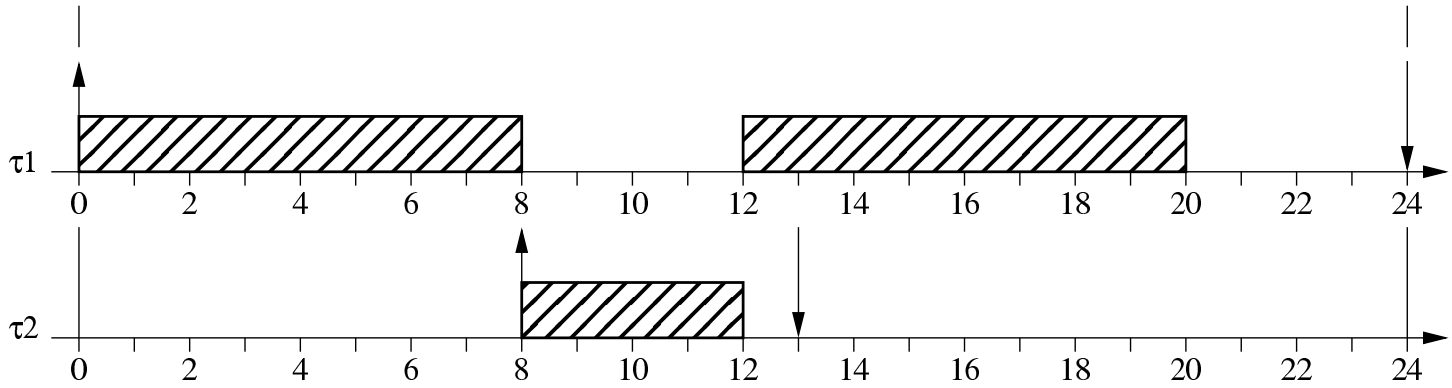
(b) Simultaneous release — longer response time Rn

RM optimality (3)

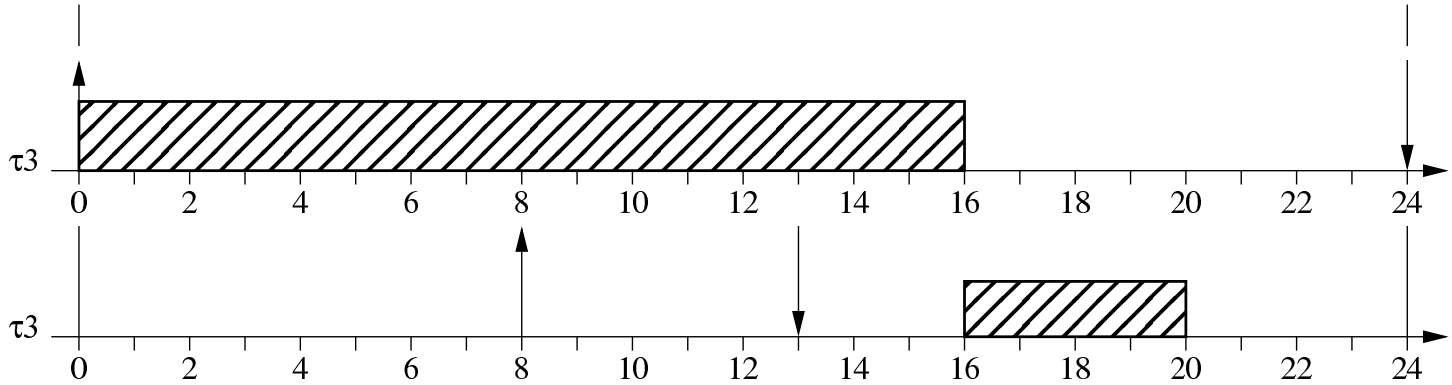
Consider two cases (the last r_i is in the critical zone of τ_j), try to change the priority, it will not be better



Periodic EDF optimality (1)



(a) Preemptive schedule –– optimal



(b) Cooperative schedule –– non–optimal and not feasible

Preemption is crucial for optimality!

Periodic EDF optimality (2)

Theorem: A set of periodic tasks is schedulable with EDF iff

(1)
$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

The necessary condition

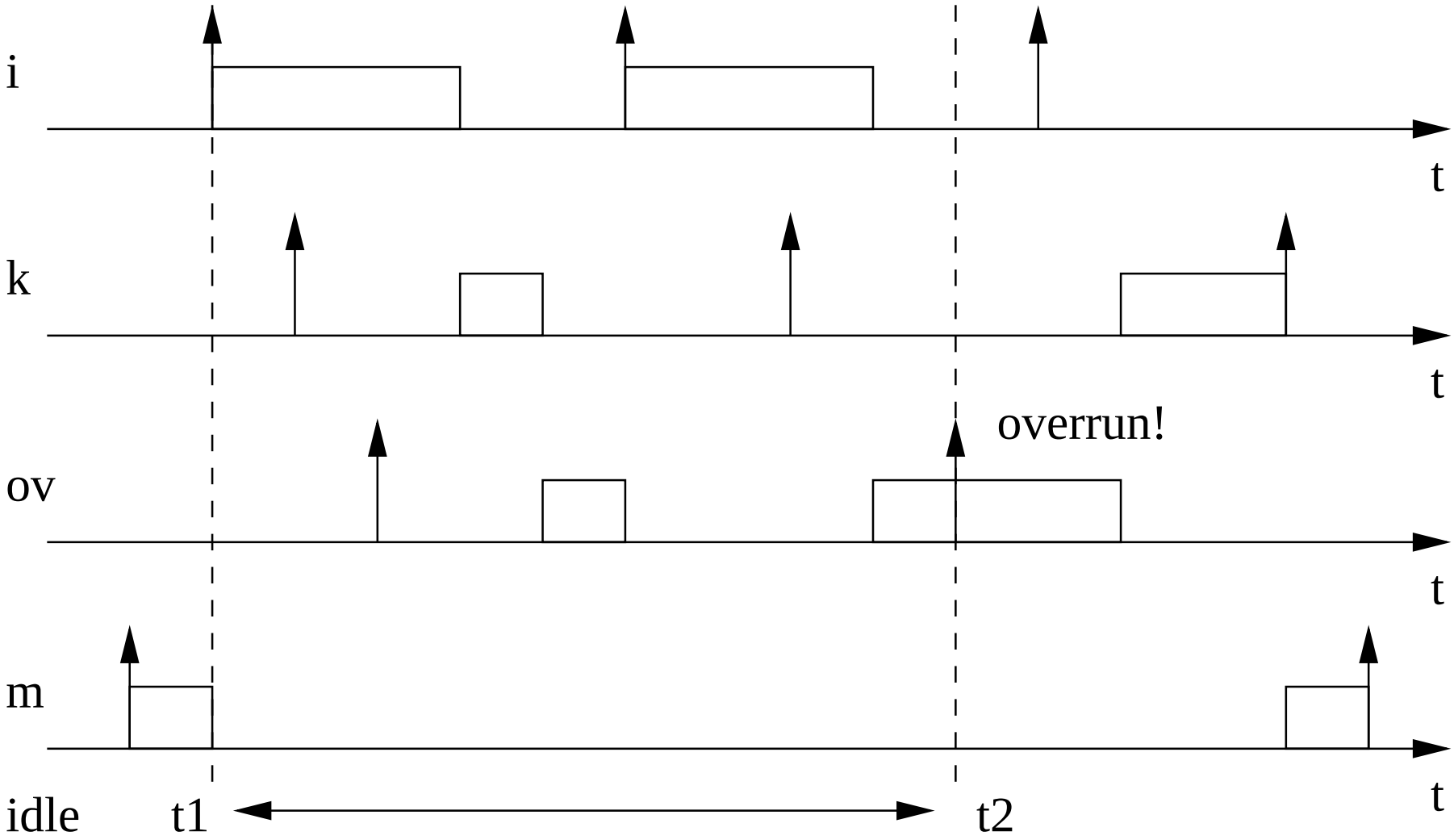
It is simple – the CPU utilisation cannot exceed 1.

- Create an example!

The sufficient condition

- Proof by contradiction:
  - Assume $U < 1$ and the set of tasks is not schedulable.
- Construct the schedule, find an overrun and find the longest interval $[t_1, t_2]$ of continuous utilisation directly preceding the overrun.
- Observe, that t_1 is the release time for some periodic instance.
- Calculate the total computational time of all tasks in $[t_1, t_2]$.
- Do maths (or just explain) and find the contradiction

The sufficient condition (diag.)



The sufficient condition (maths)

(2) $$C_p(t_1, t_2) = \sum_{r_k \geq t_1, d_k \leq t_2} C_k = \sum_{i=1}^{n} \left\lfloor \frac{t_2 - t_1}{T_i} \right\rfloor C_i.$$

The key to the proof is the following observation

(3) $$C_p(t_1, t_2) \leq \sum_{i=1}^{n} \frac{t_2 - t_1}{T_i} C_i = (t_2 - t_1)U.$$

Since a deadline is missed at t_2, $C_p(t_1, t_2)$ must be greater than the available CPU time $(t_2 - t_1)$. Together with (2) and (3) it gives the following inequality:

(4) $$(t_2 - t_1) < C_p(t_1, t_2) \leq (t_2 - t_1)U.$$

This is not be true under the initial assumption of $U < 1$.This concludes the proof.

What is next?

- Presentations
- Please read the book [Buttazzo] – all presentation topics, be prepared to discuss the presentations
- Presentations: 15 min. each, additional time will be booked the next week.
- Please email your presentation to me, use your last name as the name of the file.