Lecture 6: Scheduling

Michael O'Boyle
Embedded Software

Overview

- Definitions of real time scheduling
- Classification
- Aperiodic no dependence
    - No preemption EDD
    - Preemption EDF
    - Least Laxity
- Periodic
    - Rate Monotonic
    - Earliest deadline first
- Summary

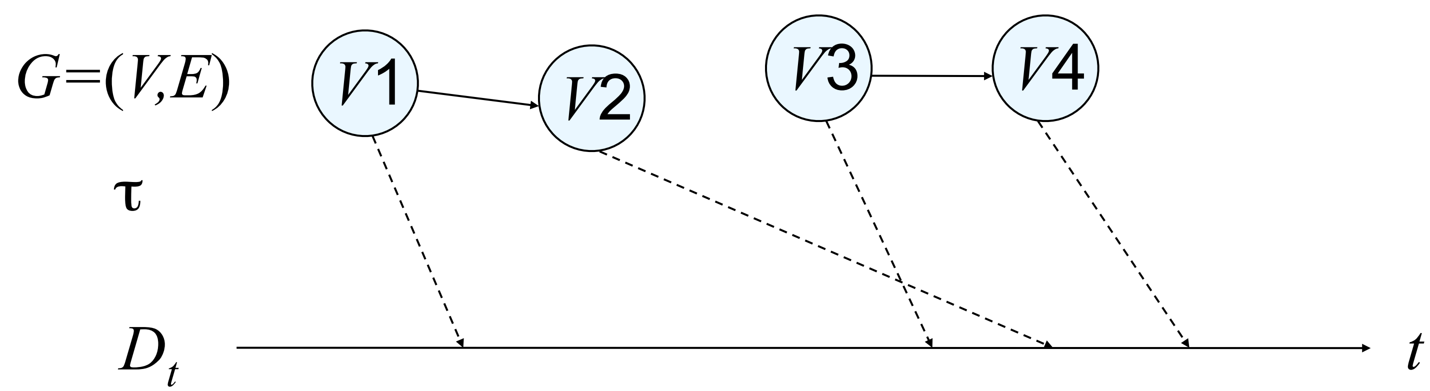
Real time

Assume that we are given a task graph $G=(V,E)$.

Def.: A **schedule** τ of G is a mapping

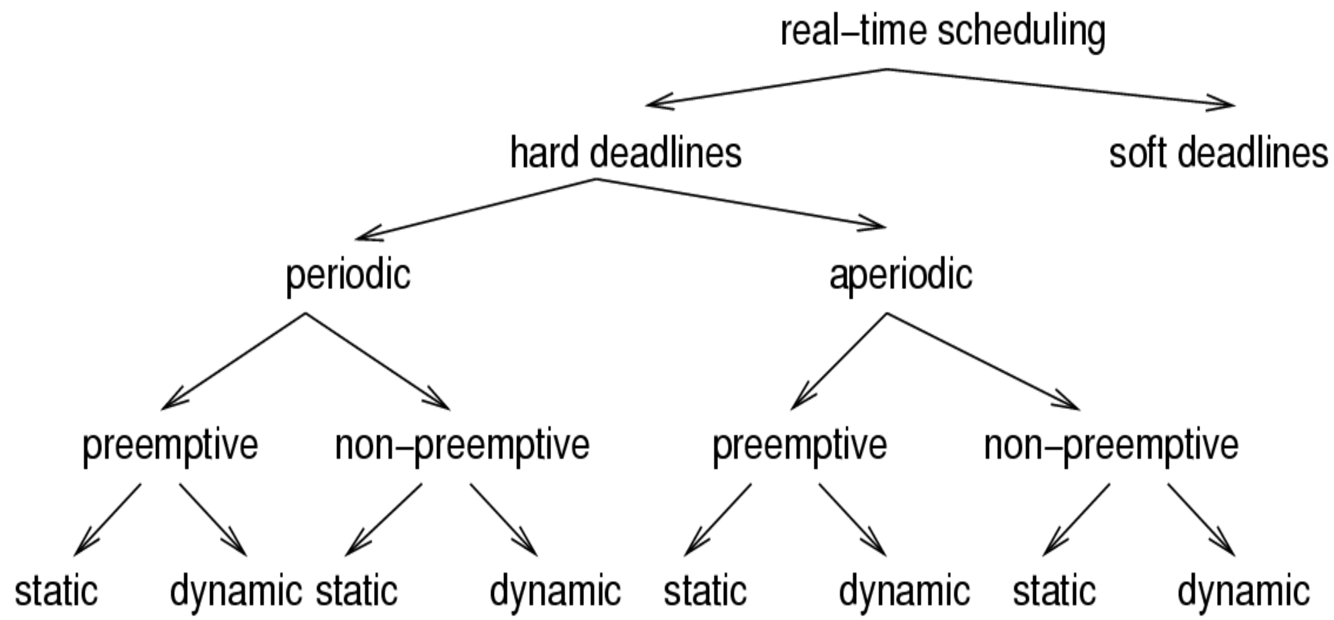
$$V \rightarrow D_t$$

of a set of tasks V to start times from domain D_t.



Typically, schedules have to respect a number of constraints,
incl. resource constraints, dependency constraints, deadlines.
Scheduling = finding such a mapping.

Classification



Def.: A time-constraint (deadline) is called **hard** if not meeting that constraint could result in a catastrophe [Kopetz, 1997].

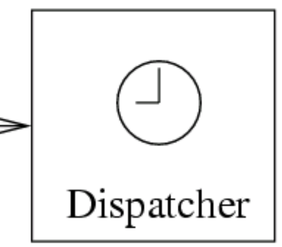
All other time constraints are called **soft**.

We will focus on hard deadlines.

Definitions

- Soft and hard deadlines
- Scheduling for periodic and aperodic tasks
  - sporadic tasks
- Prememptive vs non-preemptive
  - Suspend tasks. Can result in upredictable delays
- Static and dynamic scheduling
  - Static. Uses a priori knowledge about deadlines and arrival times
    - Timer triggers dispatch based on table. Predictable
  - Dynamic useful in reacting to sporadic events
    - Based on only what know so far
- Dependent vs independent tasks

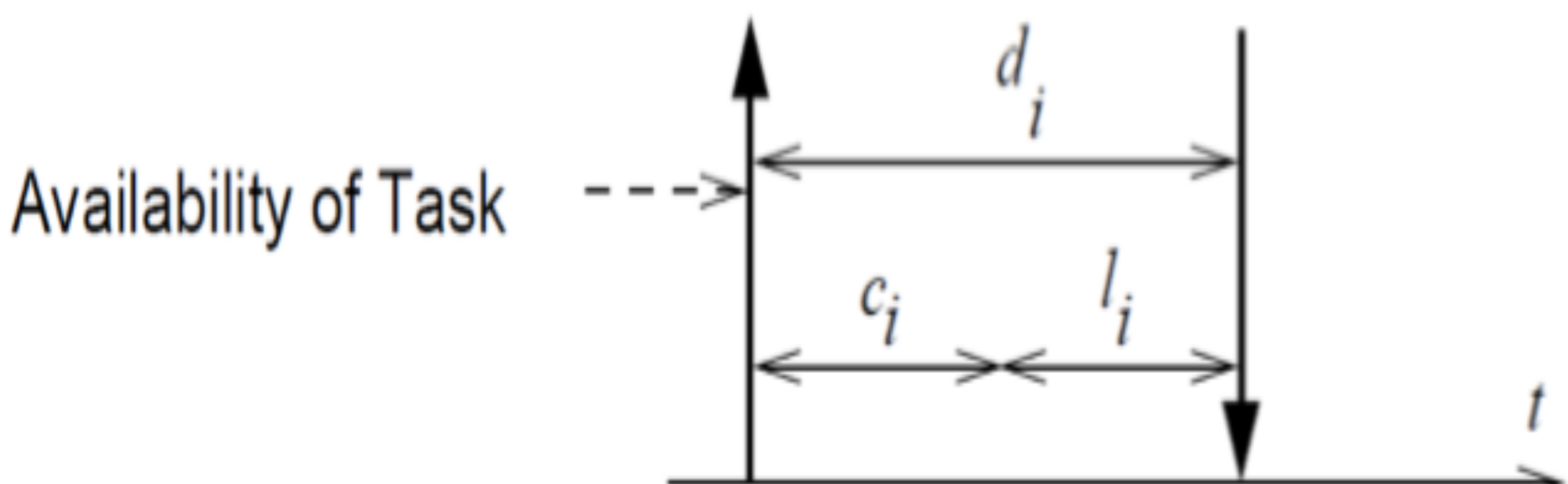
Time	Action	WCET
10	start T1	12
17	send M5	
22	stop T1	
38	start T2	20
47	send M3	



Aperiodic no predecessors

Let $\{T_i\}$ be a set of tasks. Let:

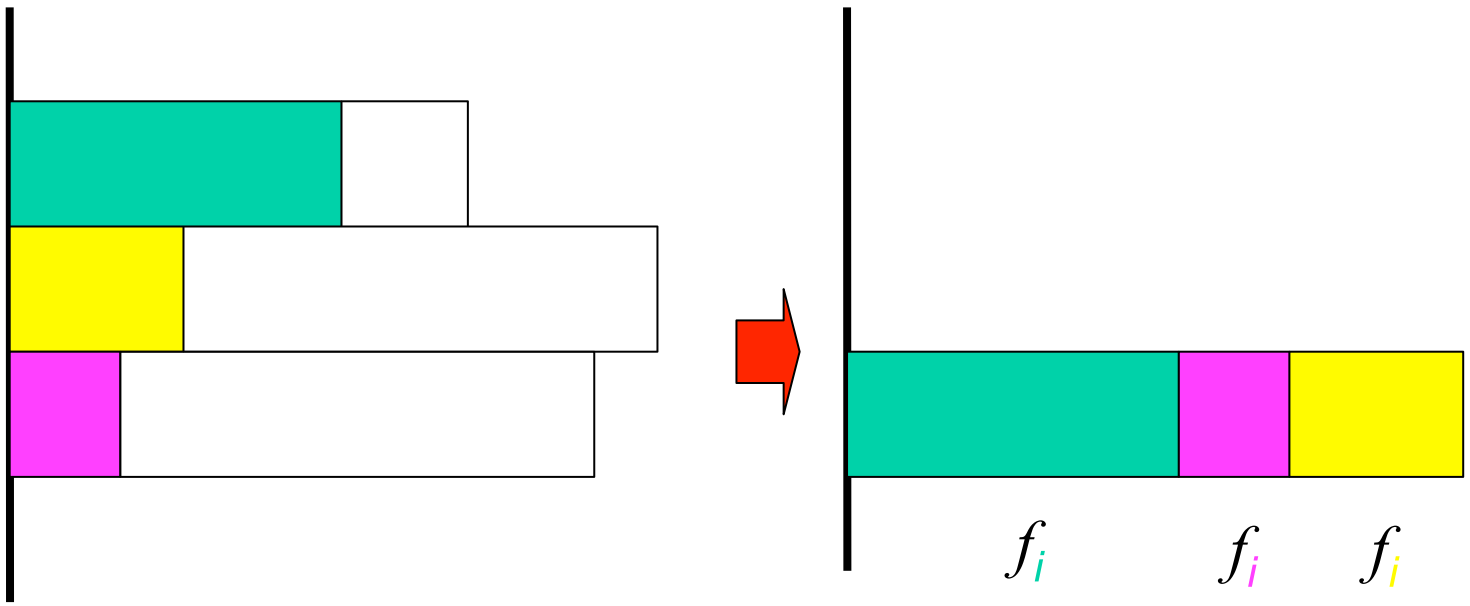
- c_i be the execution time of T_i ,
- d_i be the **deadline interva**l, that is,
  the time between T_i becoming available
  and the time until which T_i has to finish execution.
- l_i be the **laxity** or **slac**k, defined as $l_i = d_i - c_i$
- f_i be the finishing time.



EDD **for uniprocessor with equal arrival times**

Preemption is useless.

Earliest Due Date (EDD): Execute task with earliest due date (deadline) first.



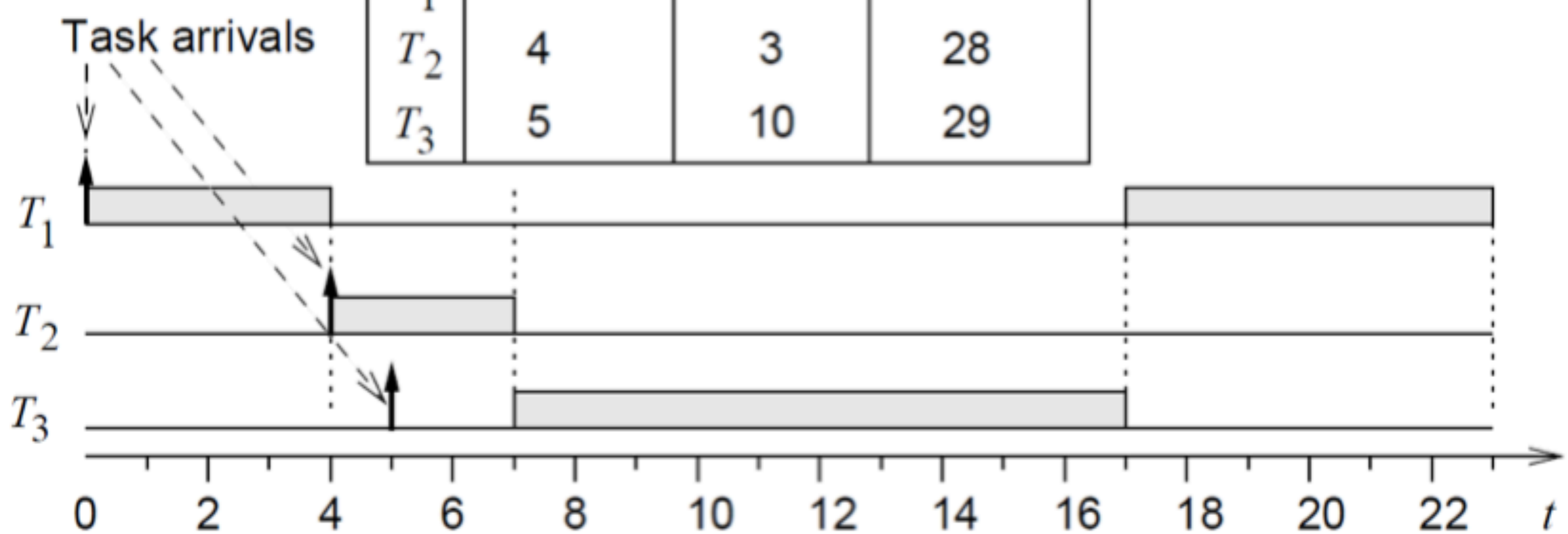
EDD requires all tasks to be sorted by their (absolute) deadlines. Hence, its complexity is $O(n \log(n))$.

EDD is optimal for this limited setting Proof Buttazzo 2002

EDF: earliest deadline first

- Different arrival times: Preemption potentially reduces lateness.
  - optimal with respect to minimizing the maximum lateness. Horn74
  - implement with sorted queue O(n^2)

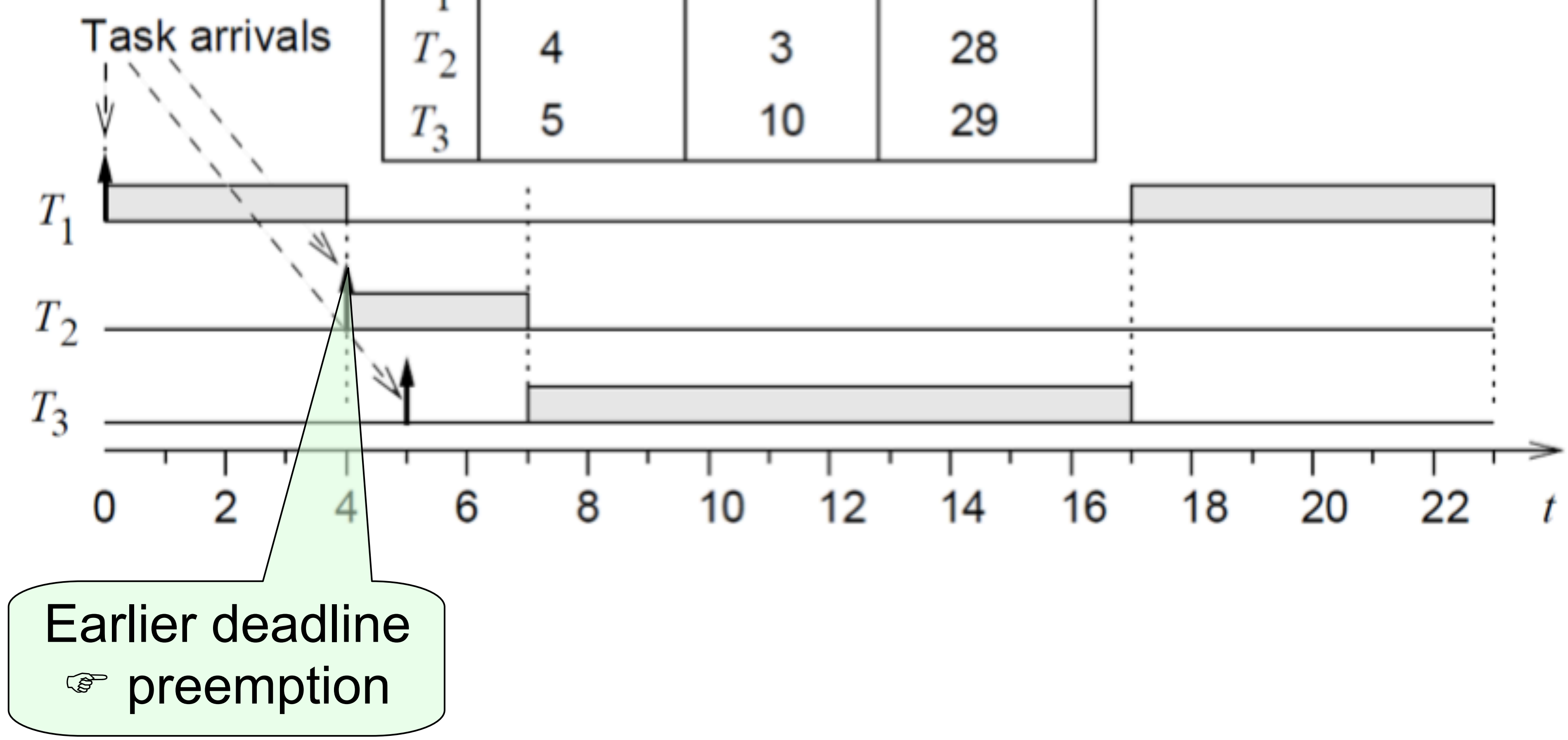
	arrival	duration	deadline
T_1	0	10	33
T_2	4	3	28
T_3	5	10	29



EDF: earliest deadline first

- Different arrival times: Preemption potentially reduces lateness.
  - optimal with respect to minimizing the maximum lateness. Horn74
  - implement with sorted queue O(n^2)

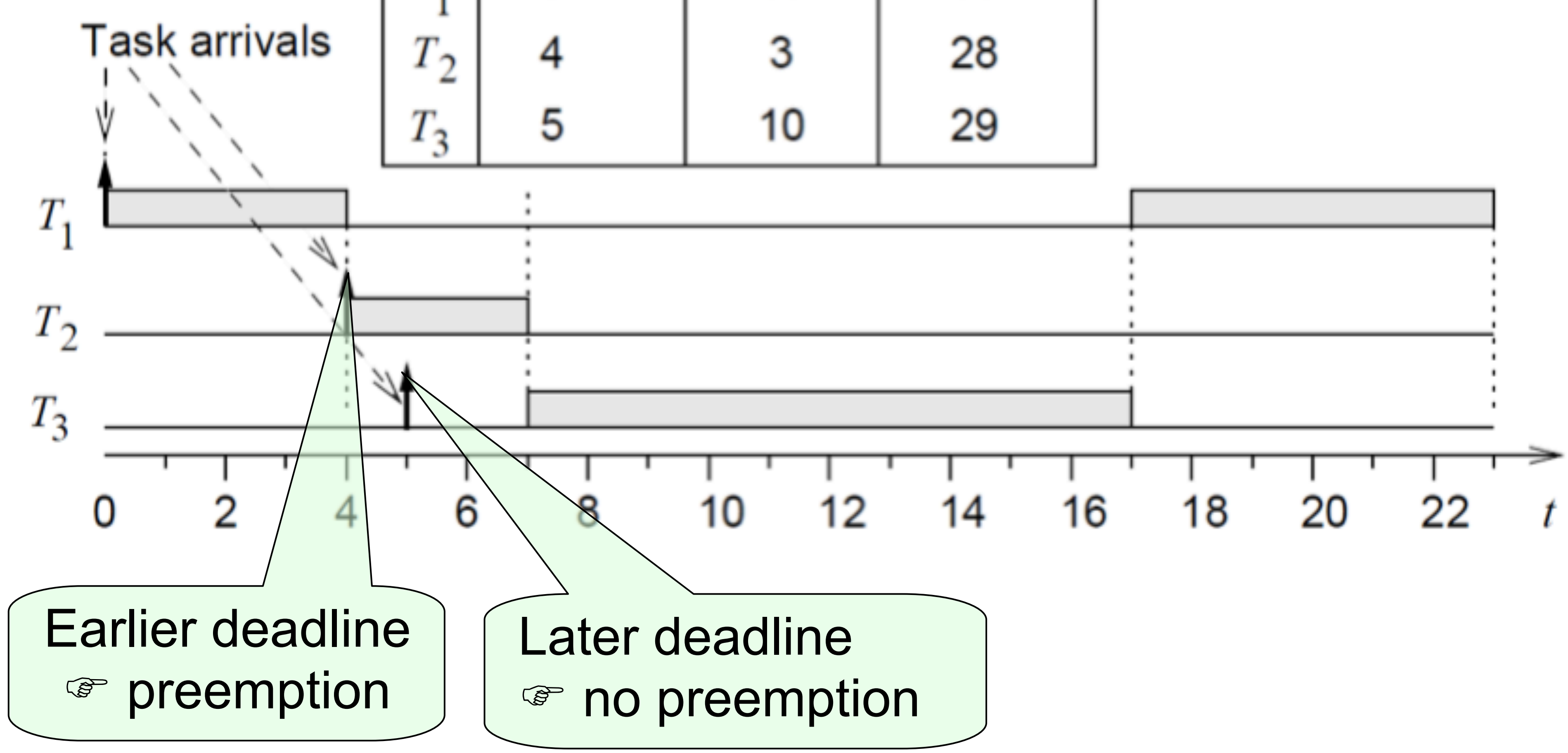
	arrival	duration	deadline
T_1	0	10	33
T_2	4	3	28
T_3	5	10	29



EDF: earliest deadline first

- Different arrival times: Preemption potentially reduces lateness.
  - optimal with respect to minimizing the maximum lateness. Horn74
  - implement with sorted queue O(n^2)

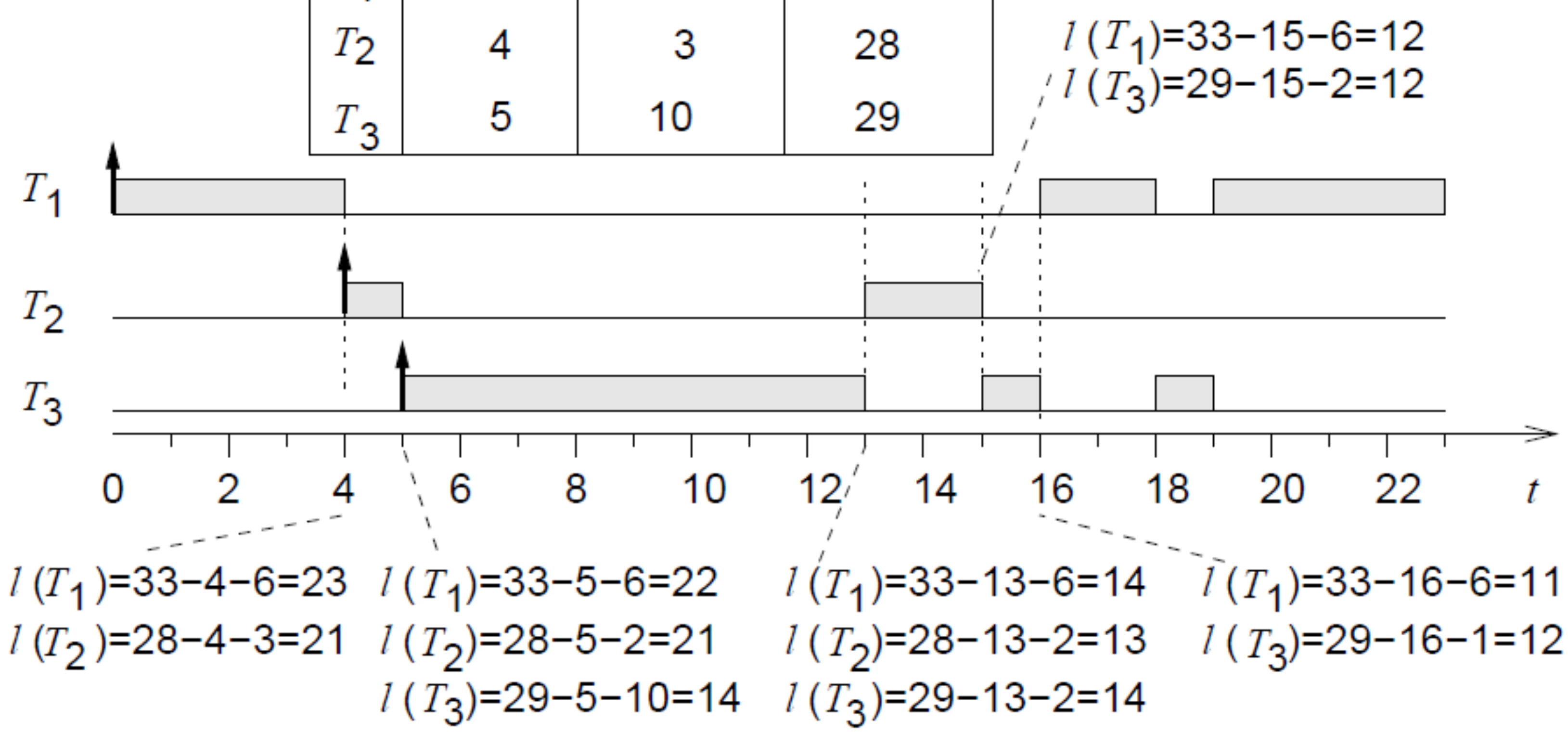
	arrival	duration	deadline
T_1	0	10	33
T_2	4	3	28
T_3	5	10	29



Least Laxity: detects missed deadlines early

Priorities = decreasing function of the laxity
(lower laxity implies higher priority); changing priority; preemptive.

	arrival	duration	deadline
T_1	0	10	33
T_2	4	3	28
T_3	5	10	29



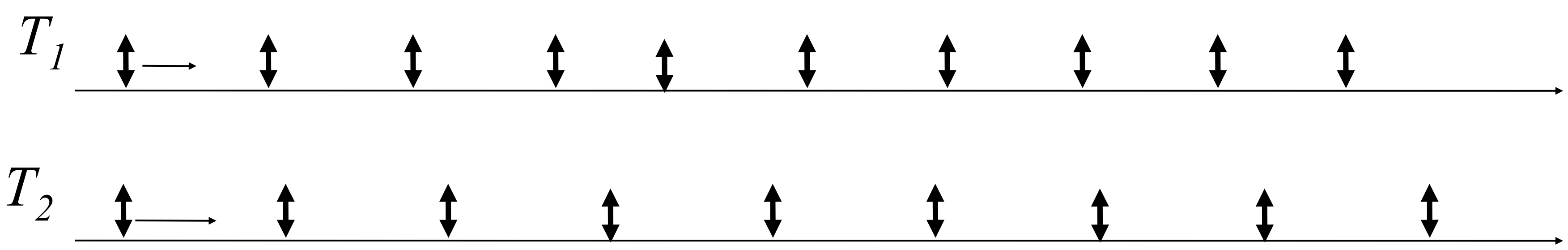
Scheduling without preemption

Lemma: If preemption is not allowed, optimal schedules may have to leave the processor idle at certain times.

Proof: Suppose: optimal schedulers never leave processor idle.

- Preemption not allowed: optimal schedules may leave processor idle to finish tasks with early deadlines arriving late.
  - Knowledge about the future is needed for optimal scheduling algorithms
  - No online algorithm can decide whether or not to keep idle.
- EDF is optimal among all scheduling algorithms not keeping the processor idle at certain times.
- If arrival times are known a priori, the scheduling problem becomes NP-hard in general. B&B typically used.

Periodic no predecessors



- Each execution instance of a task is called a **job**.
- Notion of optimality for aperiodic scheduling does not make sense for periodic scheduling.
- For periodic scheduling, the best that we can do is to design an algorithm which will always find a schedule if one exists.
    - A scheduler is defined to be **optimal** iff it will find a schedule if one exists.

Periodic Scheduling

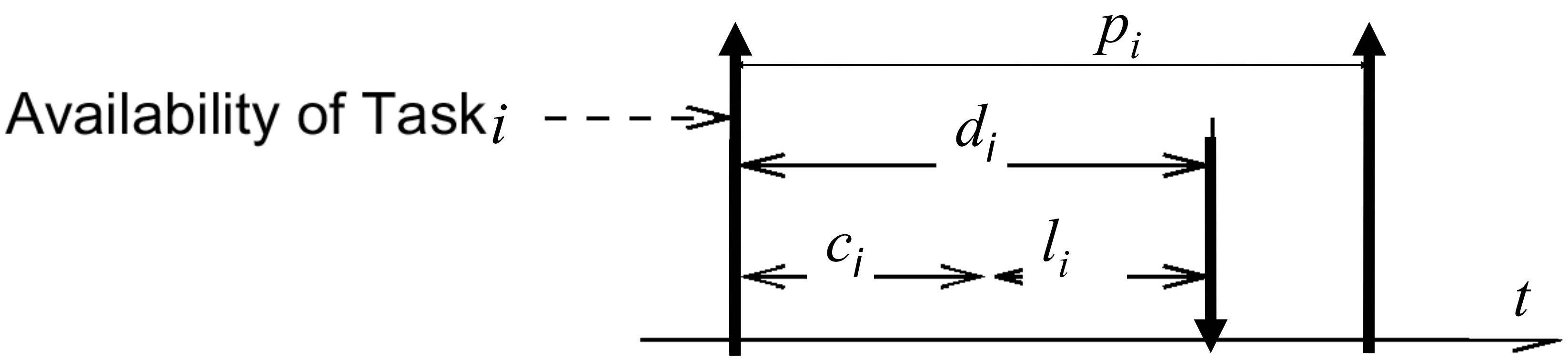
Let $\{T_i\}$ be a set of tasks. Let:

- p_i be the period of task T_i,
- c_i be the execution time of T_i ,
- d_i be the **deadline interval**, that is, the time between T_i becoming available and the time until which T_i has to finish execution.
- l_i be the **laxity** or **slack**, defined as $l_i = d_i - c_i$
- f_i be the finishing time.

Average utilization: $\mu = \sum_{i=1}^{n} \frac{c_i}{p_i}$

Necessary condition for schedulability (with m=number of processors):

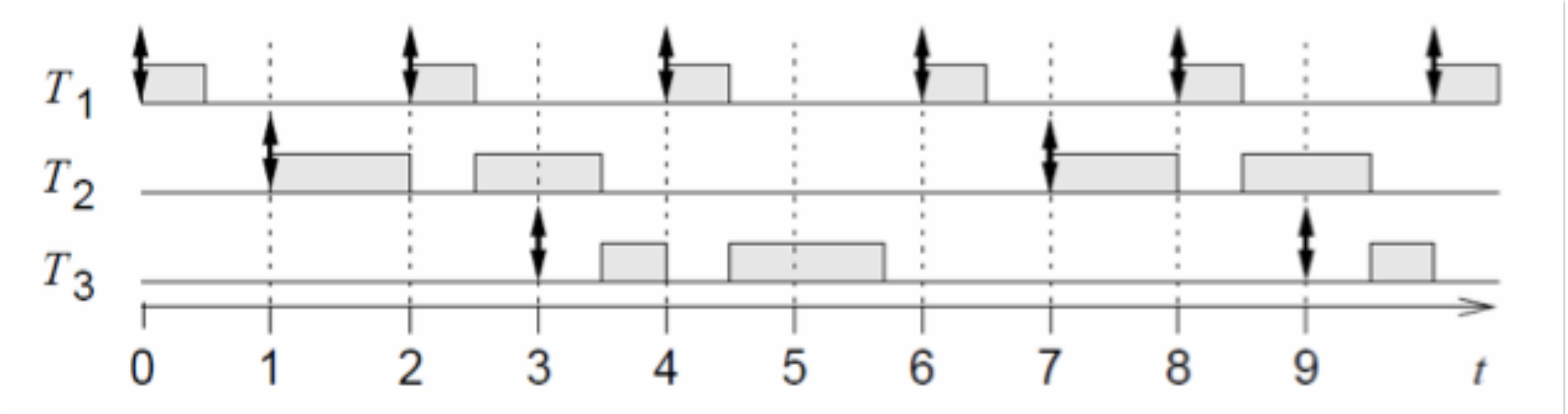
$$\mu \leq m$$



Rate Monotonic

RM policy: **The priority of a task is a monotonically decreasing function of its period.**

At any time, a highest priority task among all those that are ready for execution is allocated.



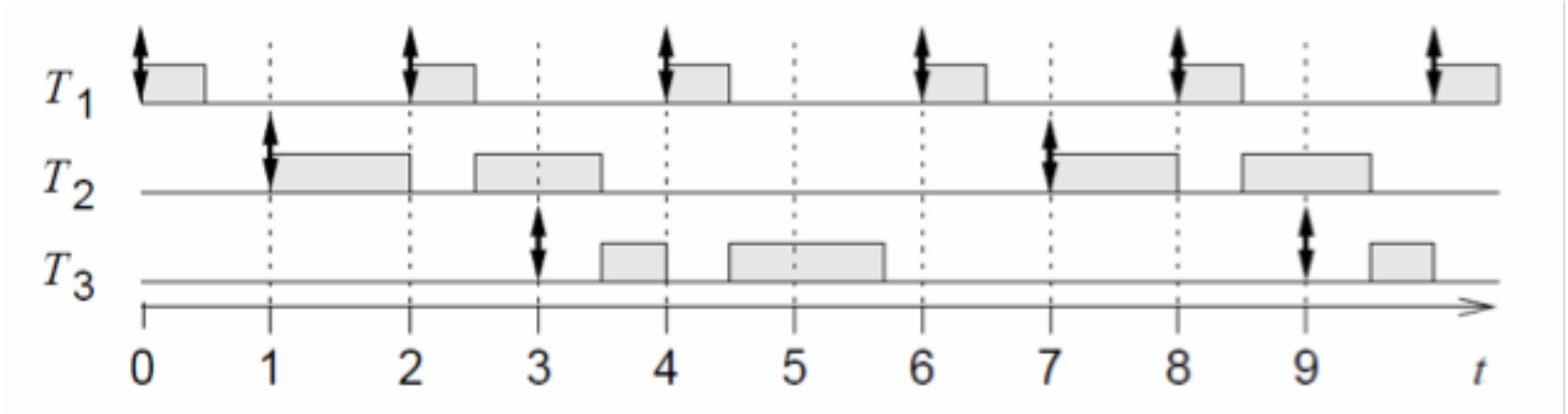
T_1 preempts T_2 and T_3.
T_2 and T_3 do not preempt each other.

Less than 0.7

Rate Monotonic

RM policy: **The priority of a task is a monotonically decreasing function of its period.**

At any time, a highest priority task among all those that are ready for execution is allocated.



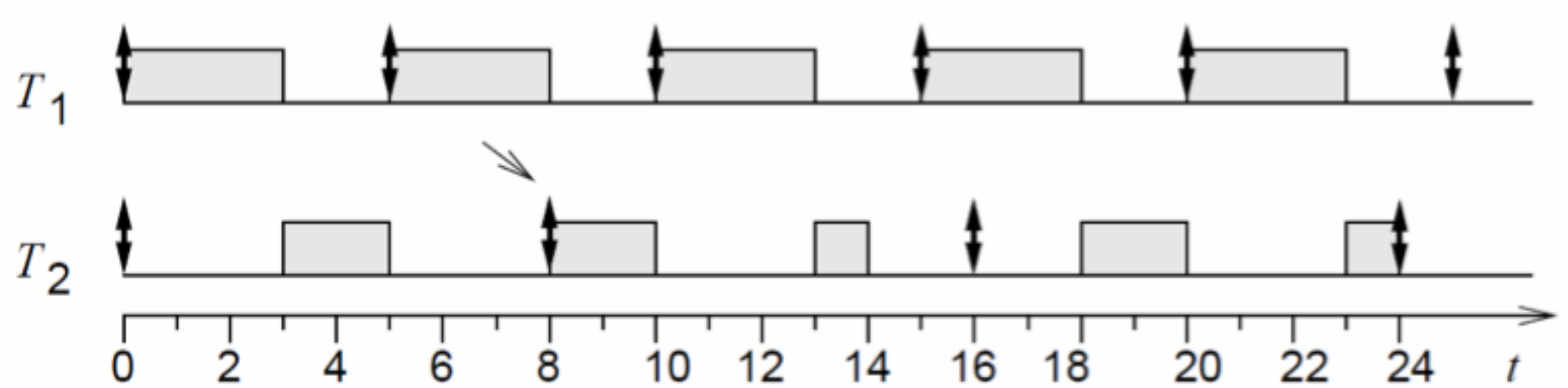
$$\mu = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{1/n} - 1)$$

T_1 preempts T_2 and T_3.
T_2 and T_3 do not preempt each other.

Less than 0.7

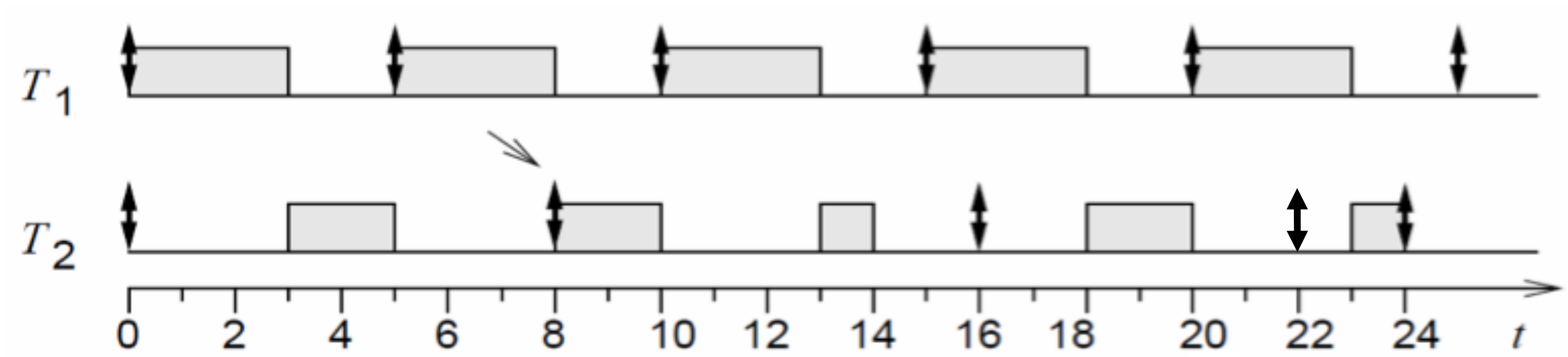
Failing RMS

Task 1: period 5, execution time 3
Task 2: period 8, execution time 3
μ=3/5+3/8=24/40+15/40=39/40 $\approx$ 0.975
$2(2^{1/2}-1) \approx 0.828$

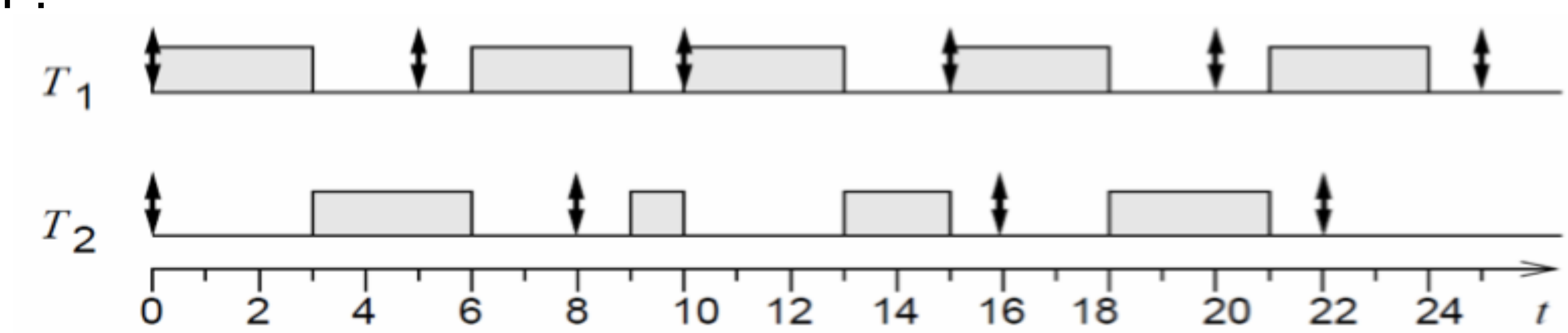


EDF can dynamically adjust priorities

RMS:



EDF:



Comparison between RMS and EDF

	RMS	EDF
Priorities	Static	Dynamic
Works with OS with fixed priorities	**Yes**	**No***
Uses full computational power of processor	No, just up till $\mu = n(2^{1/n}-1)$	Yes
Possible to exploit full computational power of processor without provisioning for slack	**No**	**Yes**

* Unless the plug-in by Slomka et al. is added.

Summary

- Definitions of real time scheduling
- Classification
- Aperiodic no dependence
  - No preemption EDD
  - Preemption EDF
  - Least Laxity
- Periodic
  - Rate Monotonic
  - Earliest deadline first