Real-Time Systems: concepts, task models, and uniprocessor scheduling

Prof. Dr. Giovani Gracioli
giovani@lisha.ufsc.br
http://www.lisha.ufsc.br/Giovani

Objectives

- Basic concepts of Real-Time Systems
  - Task models
    - the periodic task model
  - Schedulability analysis/schedulability tests
  - Uniprocessor real-time schedulers
    - RM
    - DM
    - EDF
    - LLF

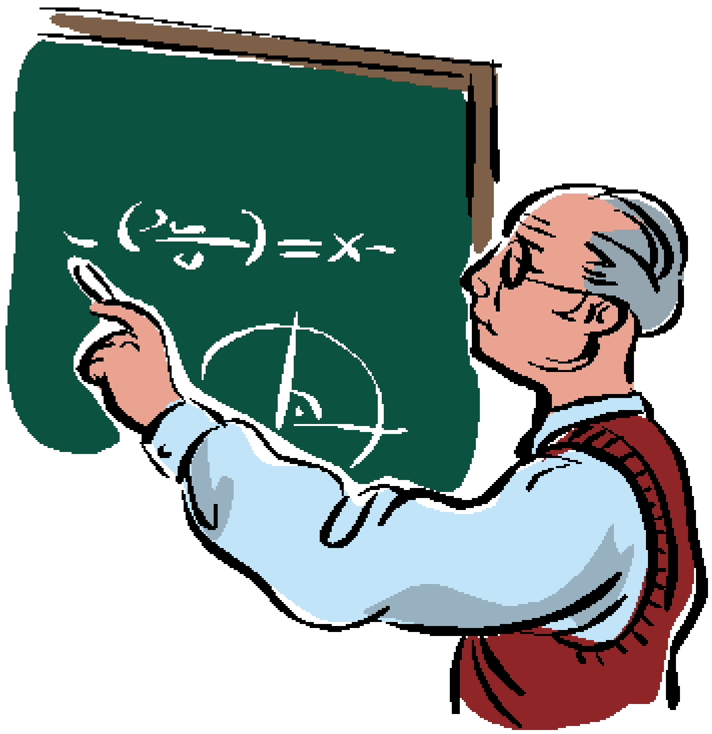
What you need to know to follow

- Experience with operating systems
  - Scheduling
  - Memory management
  - Resource management
- What a RTS is
- Soft/Hard real-time systems

What you will learn

- Periodic real-time model
- Uniprocessors real-time schedulers
  - Cyclic executive
  - RM
  - DM
  - EDF
  - LLF

Let's get started



Task Model

- Task: a real-time computation unit
  - Think of it as an execution thread with additional timing parameters
  - The job of a real-time scheduler is to schedule tasks
- Three main task models
  - Aperiodic
  - Periodic
  - Sporadic

Aperiodic Tasks

- Event-triggered computation
- Task is activated by an external event
- Task runs once to respond to the event
- Relative deadline D: available time to respond to the event
- Ex:
  - Event = loss of power
  - Task = drop control rods into nuclear reactor (this is actually happened in Fukushima)

Periodic Tasks

- Time-triggered computation
- Task is activated periodically every T time units
- Each periodic instance of the task is called a job
- Each job has the same relative deadline (usually = to period)
- Ex:
  - most digital controllers

Sporadic Tasks

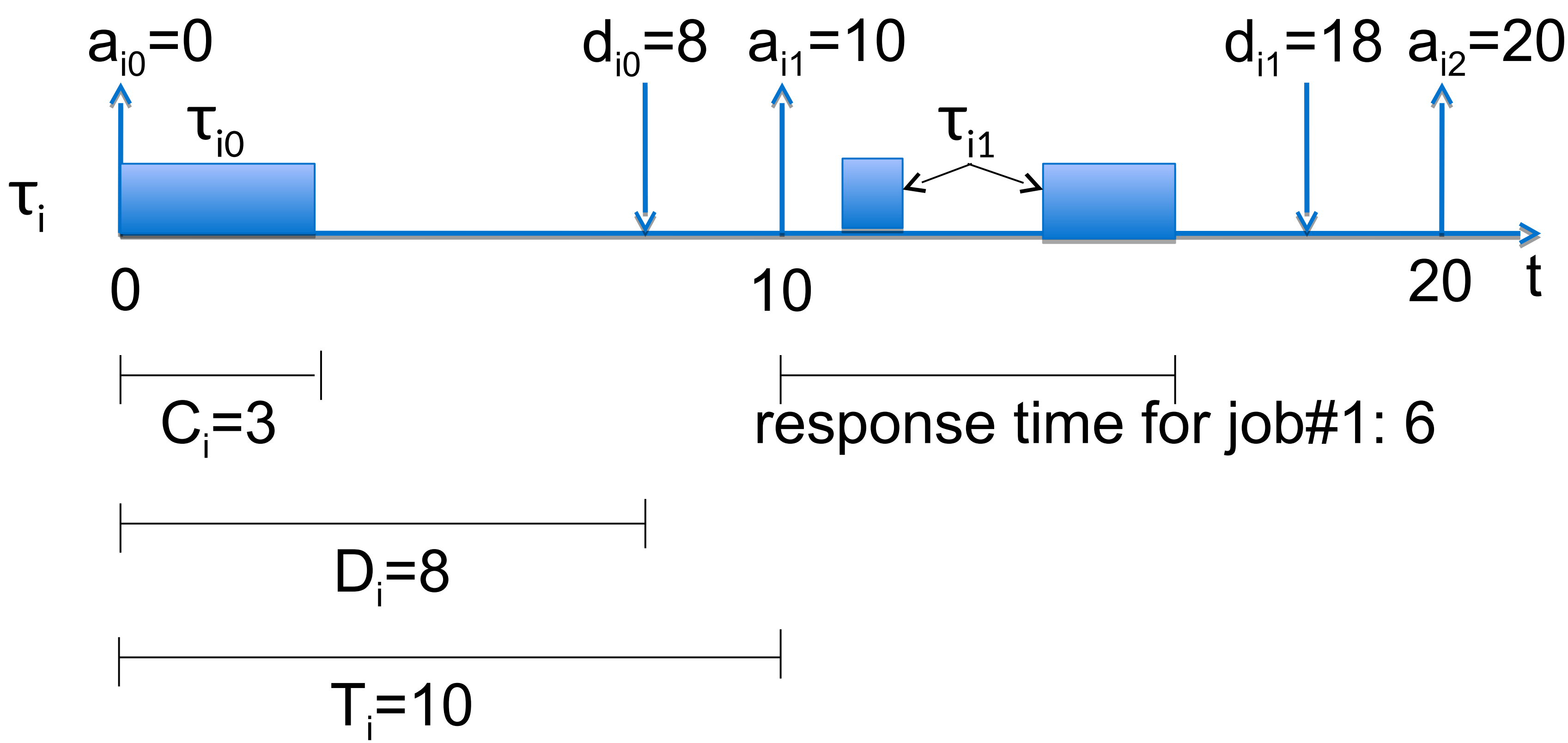
- Same as periodic task, but the task is activated at most every T time units (minimum inter-arrival time)
- Ex:
  - processing network packets

Periodic Tasks - Main concepts

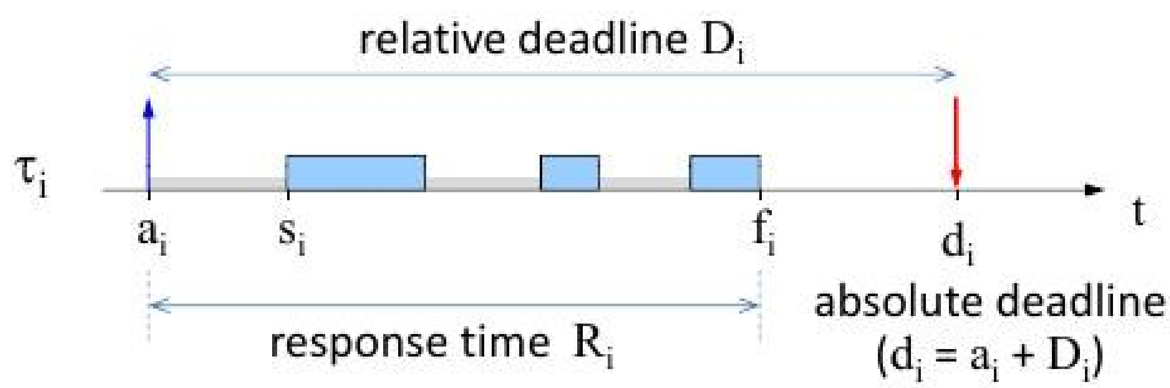
- Task τ_i (N tasks in the system, τ_1 to τ_N)
  - Execution time C_i (sometimes e_i)
  - Relative deadline D_i
  - Period T_i (sometimes p_i)

- Each job τ_{ij} of τ_i (first job: τ_{i0})
  - Activation time $a_{ij} = a_{ij-1} + T_i$ (usually with $a_{i0} = 0$)
  - Sporadic: $a_{ij} >= a_{ij-1} + T_i$
  - Absolute deadline $d_{ij} = a_{ij} + D_i$

Periodic Task Model

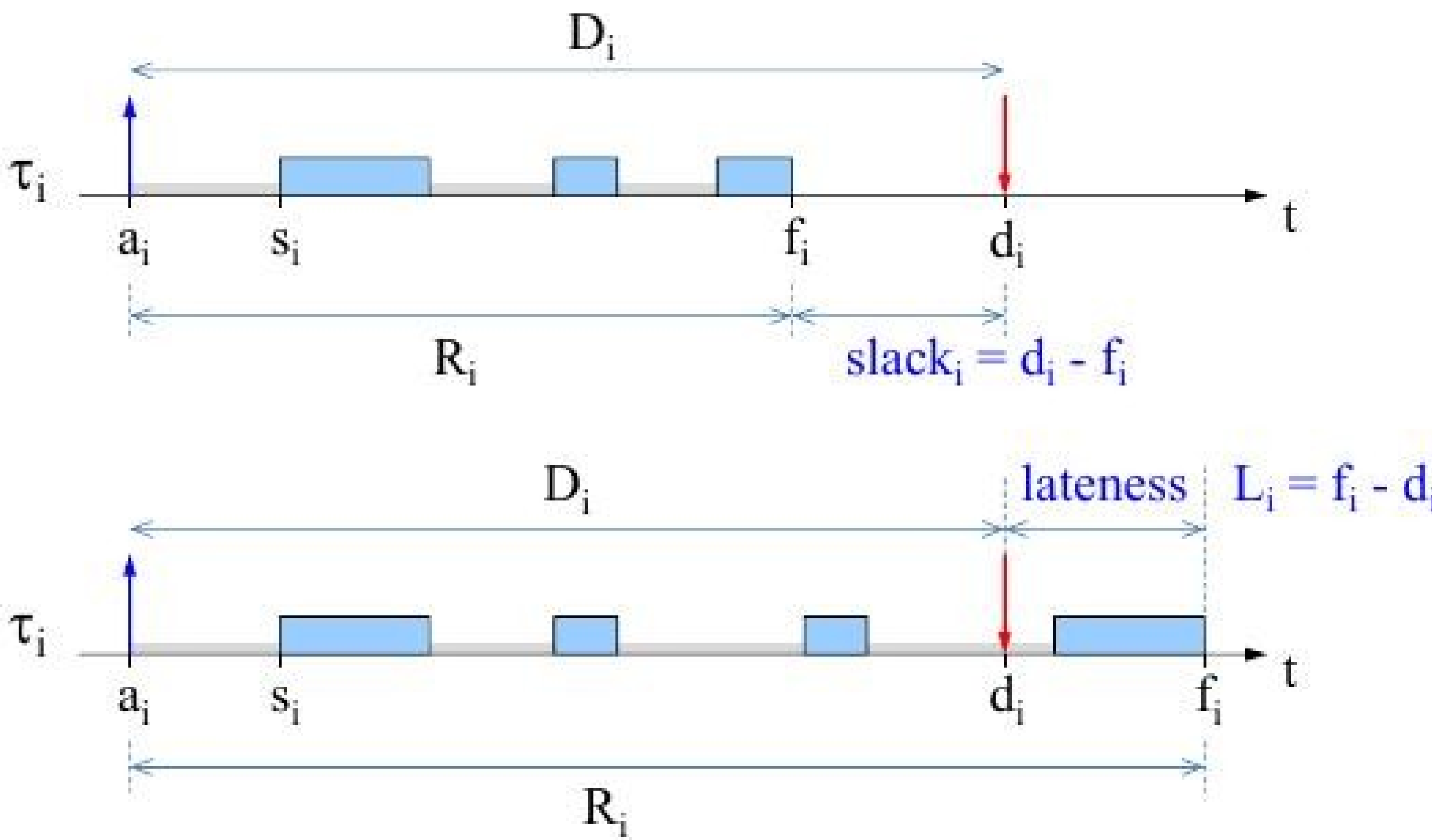


Periodic Task Model



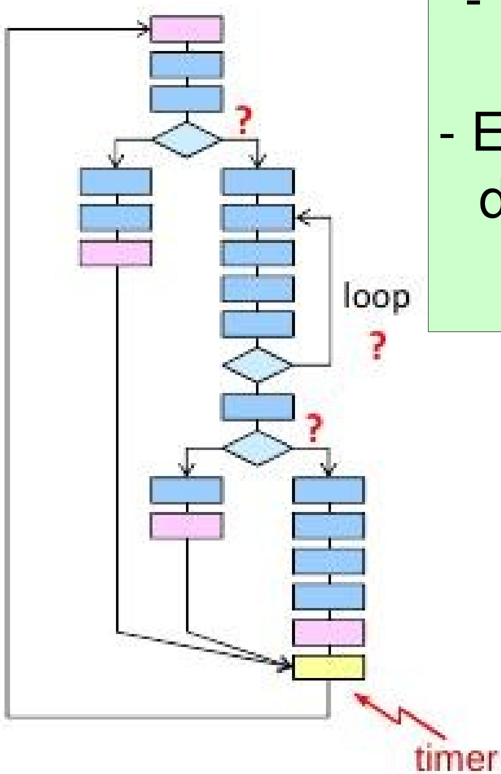
- start time (s_i), finish time (f_i), responde time R_i
- A real-time task τ_i is said to be feasible it it guaranteed to complete within its deadline, that is, if $f_i <= d_i$, or equivalently, if $R_i <= D_i$

Slack and Lateness

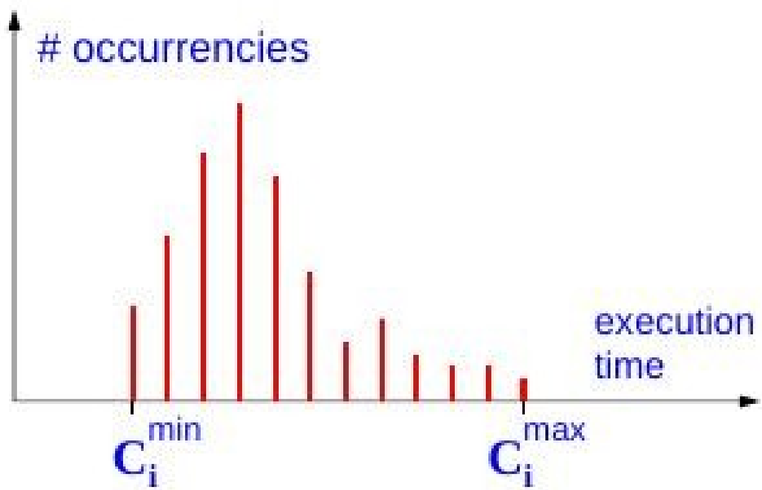


Federal University of Santa Catarina
Real-Time Operating Systems

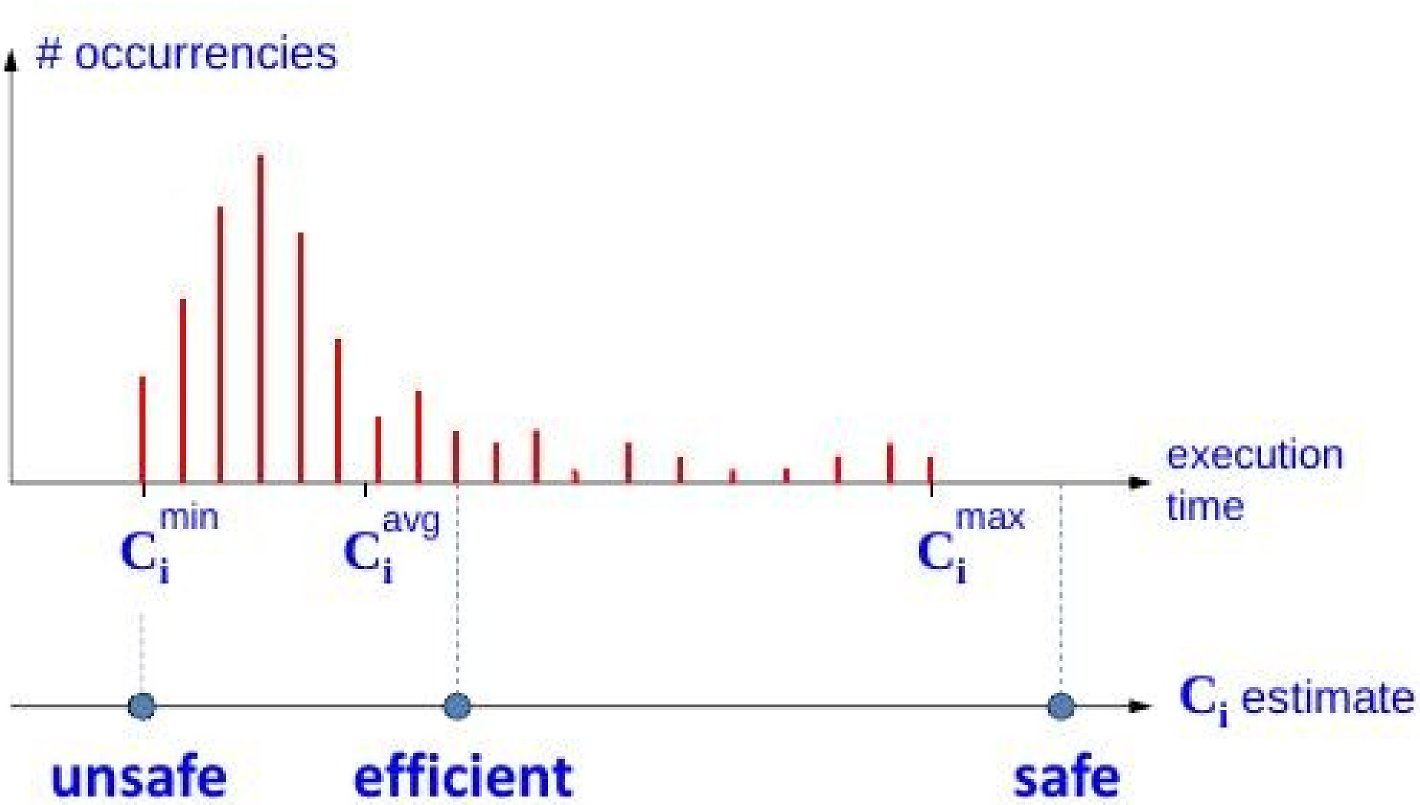
Estimating C_i is not easy



- Each job operates on different data and can take different paths
- Even for the same data, computation time depends on the processor state (cache, prefetch queue, preemptions)

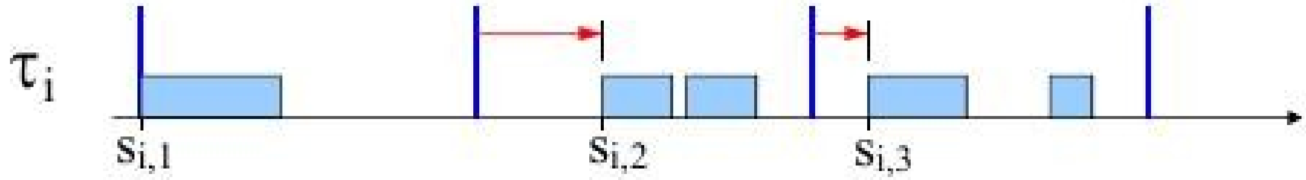


Predictability x Efficiency

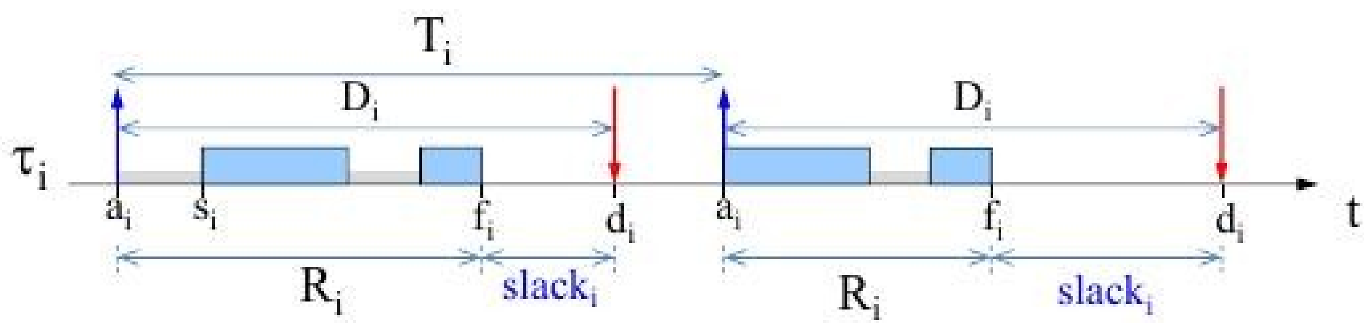


Jitter

- It is a measure of the time variation of a periodic event
- Start time jitter
  - Time to handle time interrupt and to insert the job into the running queue



Parameters summary



- Computation time (C_i)
- Period (T_i)
- Relative deadline (D_i)

These parameters are specified by the programmer and are known off-line

- Arrival time (a_i)
- Start time (s_i)
- Finishing time (f_i)
- Response time (R_i)
- Slack and Lateness
- Jitter

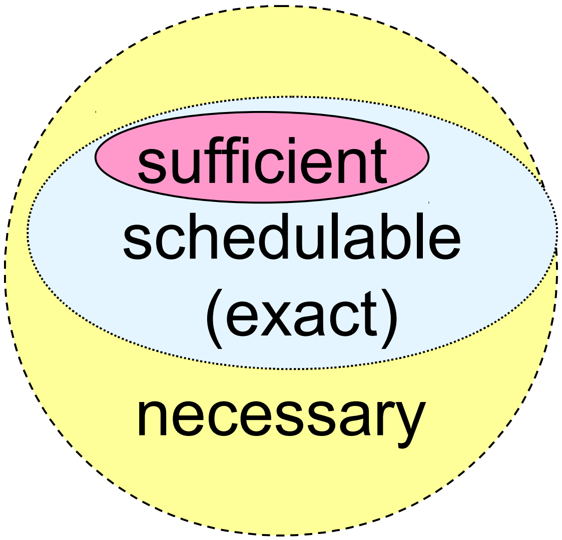
These parameters depend on the scheduler and on the actual execution, and are known at run time.

Utilization & Schedulability analysis

- Task Utilization for a periodic/sporadic task: $U_i = C_i / T_i$
  - Percentage of processor time required by the task
- System Utilization: $U = U_1 + U_2 + \ldots + U_N$
  - Percentage of processor time required by all tasks
- Base uniprocessor scheduling result: task set is clearly not schedulable if: $U > 1$
- For many scheduling algorithms, we can define a utilization bound U_b such that the task set if schedulable if: $U <= U_b$

Schedulability

- Set of tasks is schedulable under a set of constraints, if a schedule exists for that set of tasks & constraints
- Exact tests are NP-hard in many situations
- Sufficient tests: sufficient conditions for schedule checked. (Hopefully) small probability of not guaranteeing a schedule even though one exists
- Necessary tests: checking necessary conditions. Used to show no schedule exists. There may be cases in which no schedule exists & we cannot prove it



Task Constraints

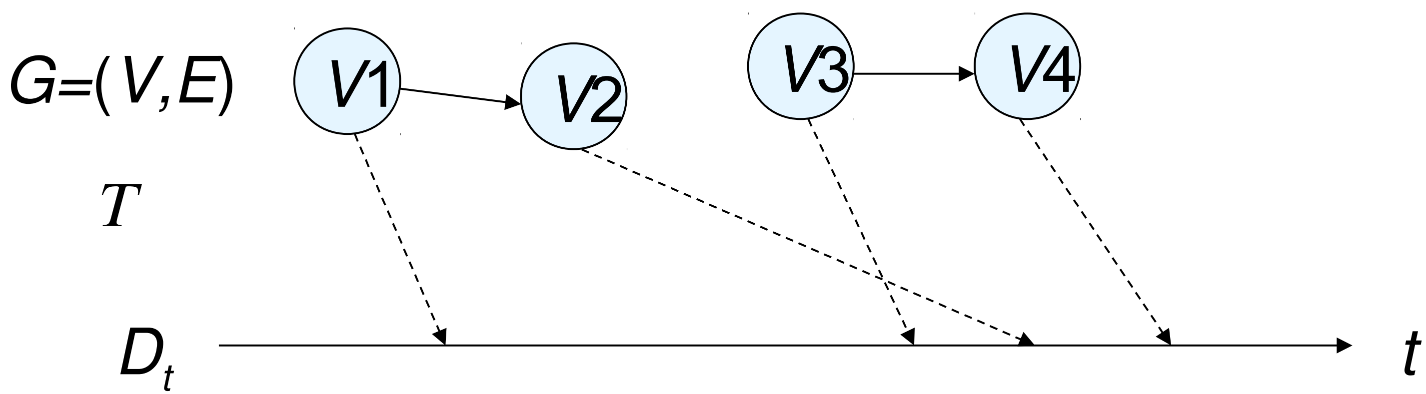
- Timing constraints
  - activation, completion, jitter
- Precedence constraints
  - they impose an ordering in the execution
- Resource constraints
  - they enforce a synchronization in the access of mutually exclusive resources

Real-Time Scheduling

- Assume that we are given a task graph G=(V,E)
- **Def.**: A schedule T of G is a mapping

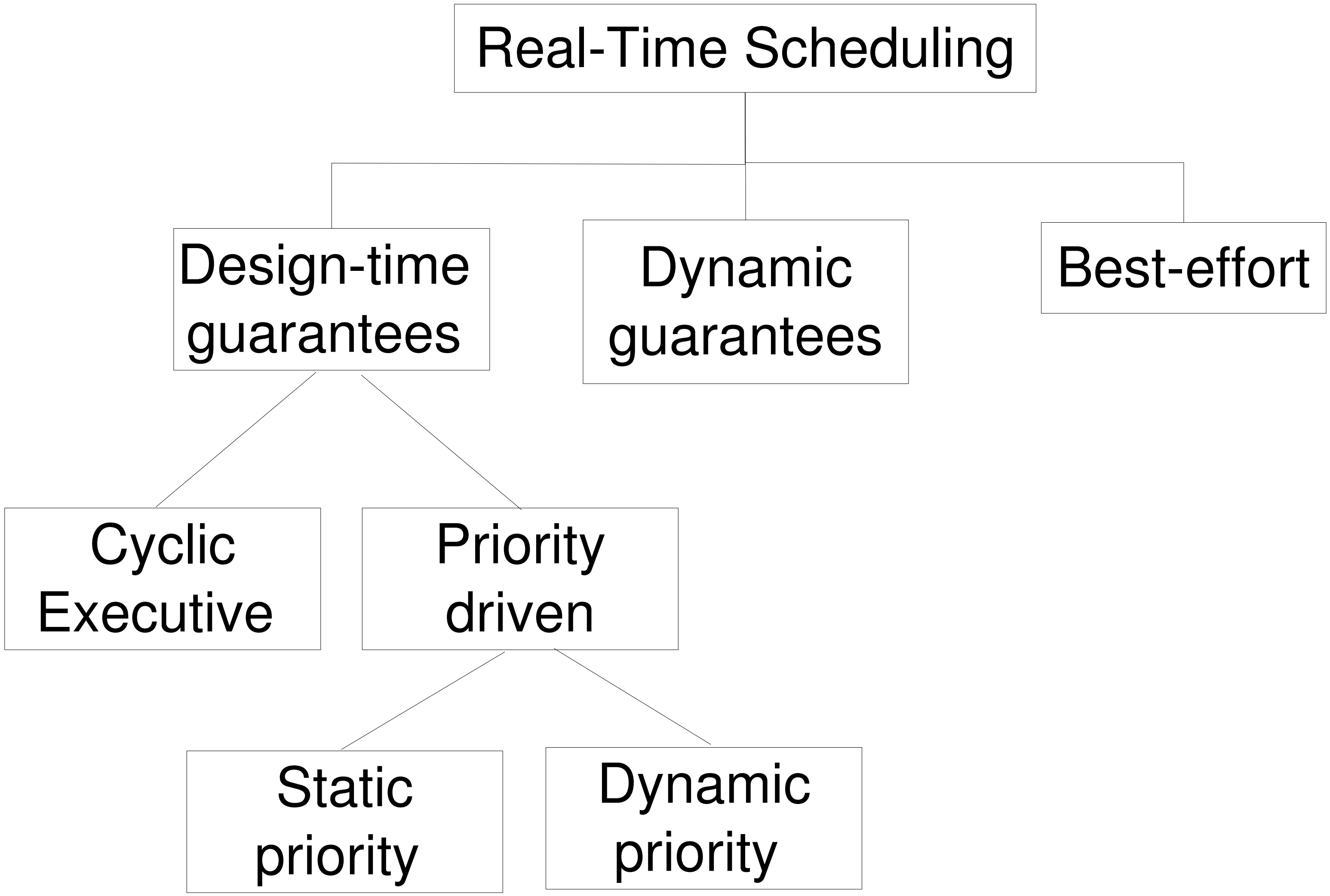
$$V \rightarrow D_t$$

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



- Typically, schedules have to respect a number of constraints (resource, dependency, timing)
- Scheduling = finding such a mapping

RT Scheduling Classification



The simplest model

- We first look at the simplest possible model, where:
  - All tasks are periodic
  - Single processor
  - Tasks do not share any resource
- Not very realistic, but instructive (we have simple results)!
- We will then (briefly) look at:
  - What happens when you start sharing resources
  - What happens if you schedule a mix of periodic/aperiodic tasks
  - What happens if you use a multiprocessor
  - More complex task models

Uniprocessor real-time scheduling

- Three main categories

1) Table-driven scheduling
   - build a table that dictates when each task executes

2) Fixed-priority scheduling
   - Each task is assigned a fixed priority

3) Dynamic-priority scheduling
   - Task priority varies at run-time (each job of the task has a different priority)

- Scheduler is typically preemptive - better system utilization

Cyclic Executive Scheduling

- Also known as cyclic scheduling
- It has been used for 30 years in military systems, navigation, and monitoring systems
- Only works for periodic tasks and off-line analysis
- Examples
  - Air traffic control systems
  - Space shuttle
  - Boeing 777
  - Airbus navigation system

Cyclic Executive Scheduling

- The time axis is divided in intervals of equal length (time slots)
- Each task is statically allocated in a slot in order to meet the desired request rate
- The execution in each slot is activated by a timer

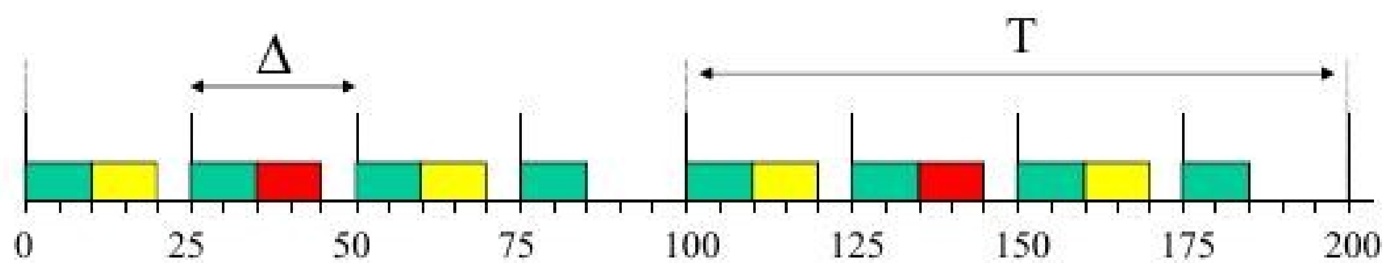
Cyclic Executive Scheduling

Example:

task	C_i	T_i
A	10 ms	25 ms
B	10 ms	50 ms
C	10 ms	100 ms

Δ = GCD (minor cycle)

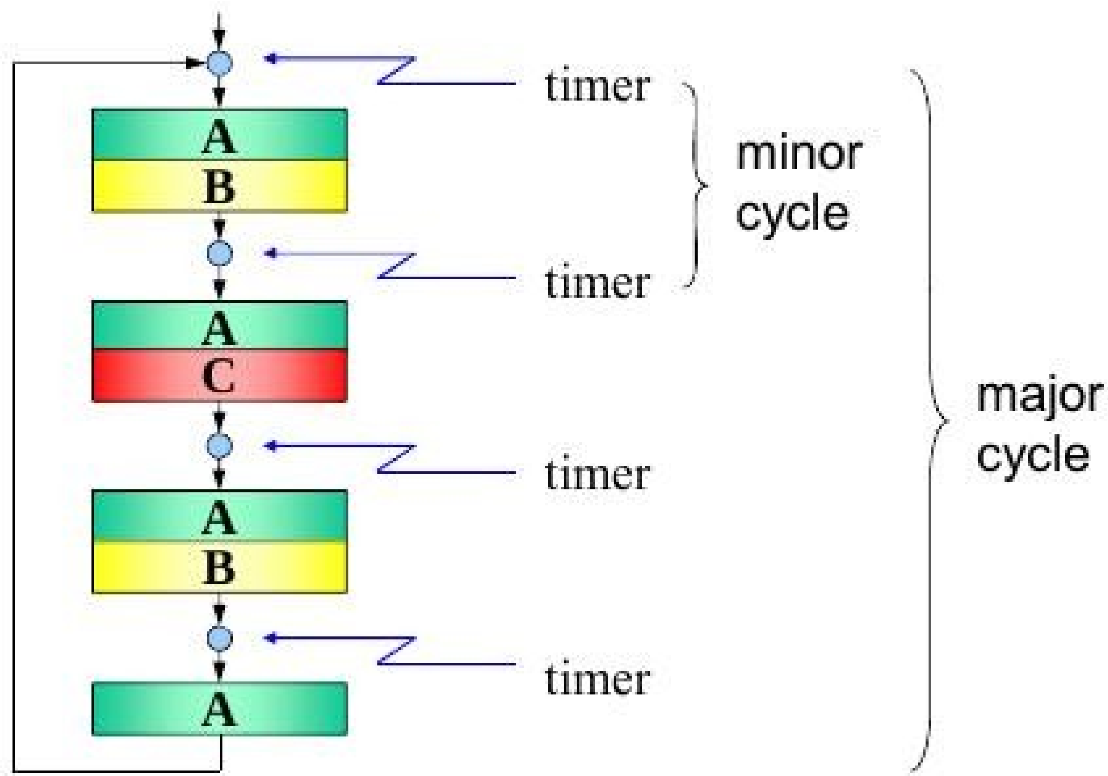
T = lcm (major cycle)



Guarantee: $\begin{cases} C_A + C_B \leq \Delta \\ C_A + C_C \leq \Delta \end{cases}$

Cyclic Executive Scheduling

Implementation:



Cyclic Executive Scheduling

Coding:

```
#define    MINOR      25          // minor cycle = 25 ms

      initialize_timer(MINOR);    // interrupt every 25 ms

      while (1) {
            sync();               // block until interrupt
            function_A();
            function_B();
            sync();               // block until interrupt
            function_A();
            function_C();
            sync();               // block until interrupt
            function_A();
            function_B();
            sync();               // block until interrupt
            function_A();
      }
```

Cyclic Executive Scheduling

- Advantages
  - Simple implementation (no RTOS is required)
  - Low run-time overhead
  - Very low jitter

- Disadvantages
  - Very fragile during overload conditions => domino effect or task abort
  - It is difficult to expand the schedule (sensitivity to application changes)
  - It is not easy to handle aperiodic activities

Priority (static or dynamic) Scheduling

- Method
  - Assign priorities to each task according to the scheduling policy (static or dynamic)
  - Verify the feasibility of the schedule using analytical techniques (schedulability test of the scheduling algorithm)
  - Execute tasks on a priority-based RTOS

Rate Monotonic (RM)

- Each task is assigned a fixed priority proportional to its rate [Liu and Layland 73]
  - shorter period = higher priority
- Assumes periodic or sporadic tasks with $D_i = T_i$ on a uniprocessor
- Assumes that the context switch time is zero

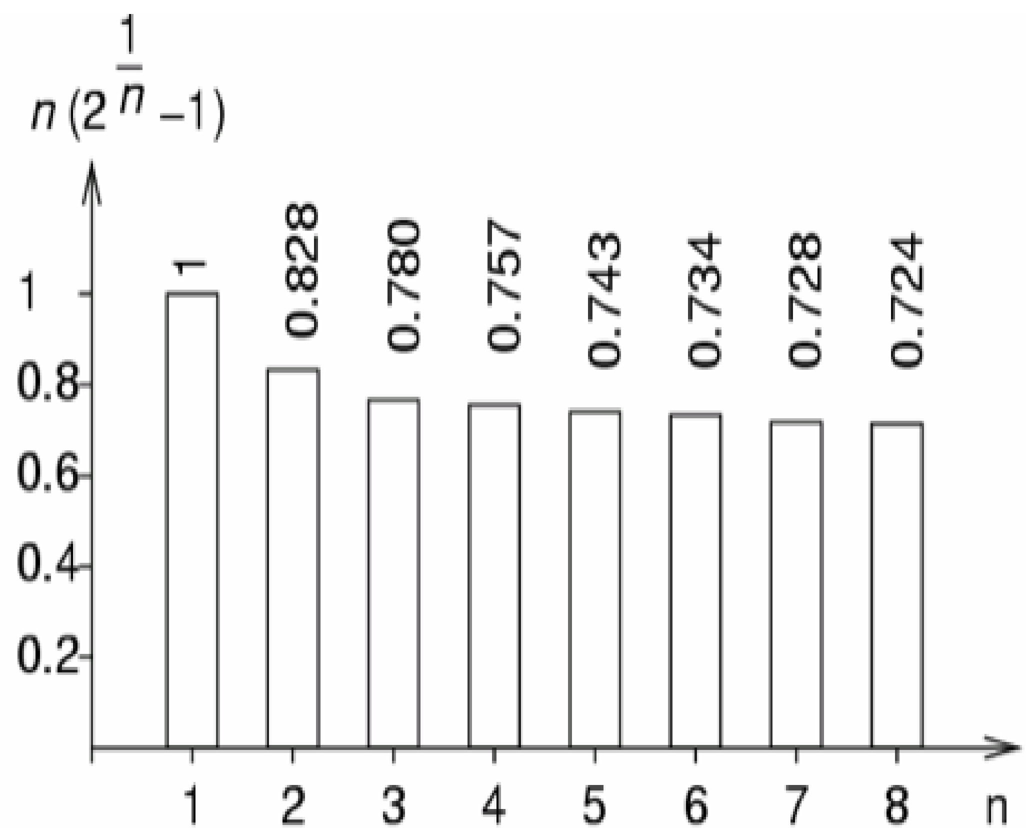
Rate Monotonic (RM)

- Sufficient schedulability test
  - Proposed by Liu and Layland in 1973
  - Task set schedulable if
  - $U <= U_b(N) = N(2^{1/N} - 1)$
  - When all tasks have harmonic periods $U <= U_b$
- Note that $\lim_{N->+inf} N(2^{1/N} - 1) = \log 2 \sim= 0.693$
- What happens if $U_b(N) < U <= 1$?
  - Nothing can be said according to the analysis
  - Task set might or might not be schedulable

Rate Monotonic (RM)

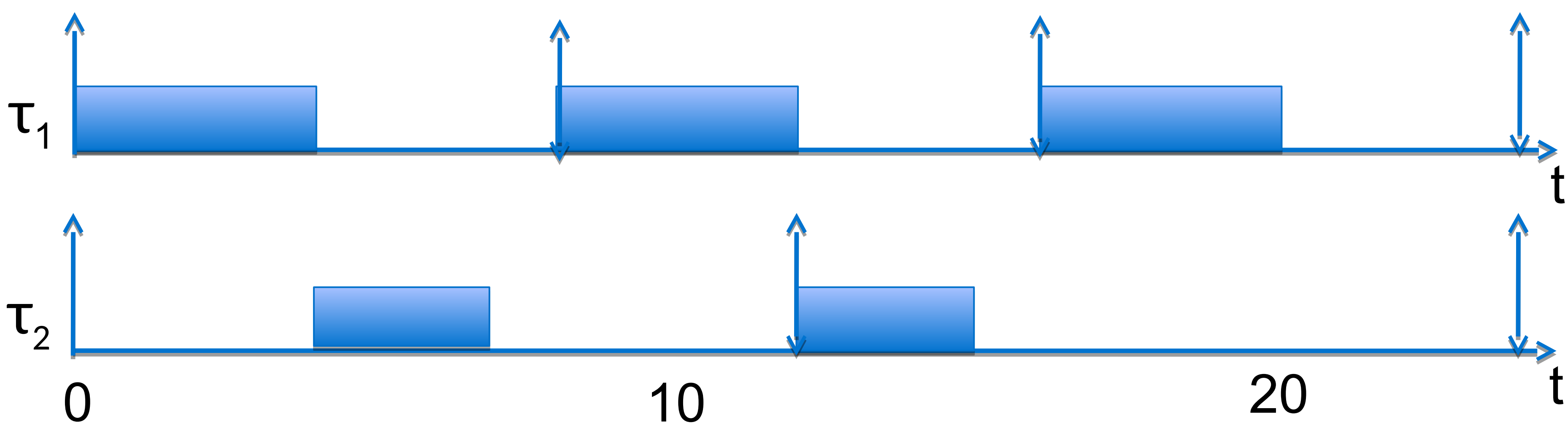
$$\mu = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{1/n} - 1)$$

$$\lim_{n \to \infty} (n(2^{1/n} - 1) = \ln(2)$$



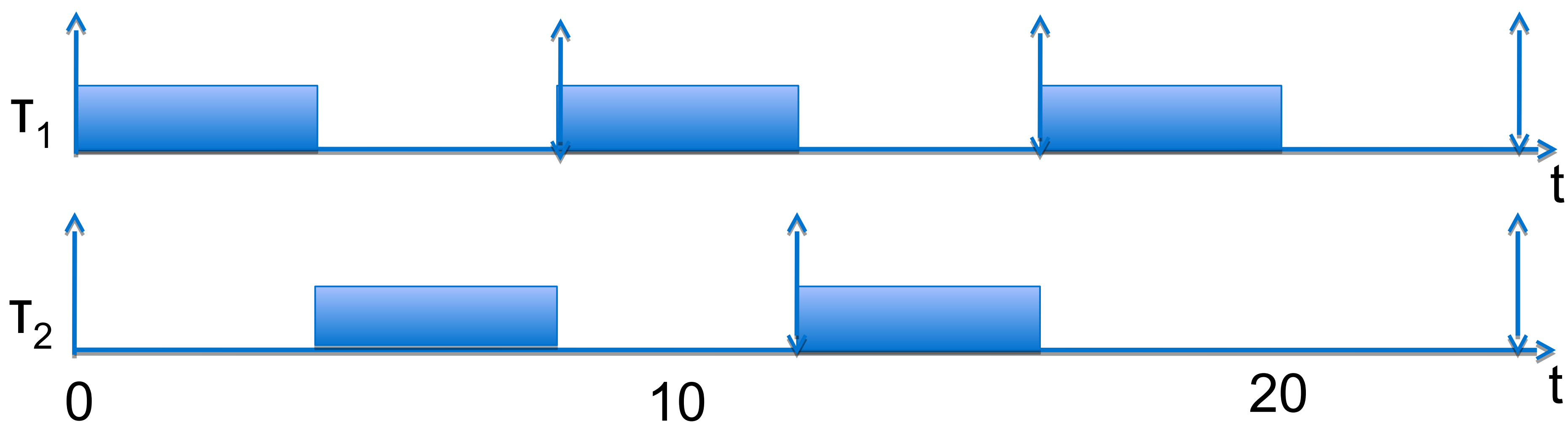
Rate Monotonic (RM)

- Case 1
  - τ_1 ($C_1 = 4$, $T_1 = 8$), high prio, τ_2 ($C_2 = 3$, $T_2 = 12$), low prio
  - Utilization: $U = 4/8 + 3/12 = 0.75 < U_b(2)$ ~=0.828
  - Schedulability analysis: schedulable



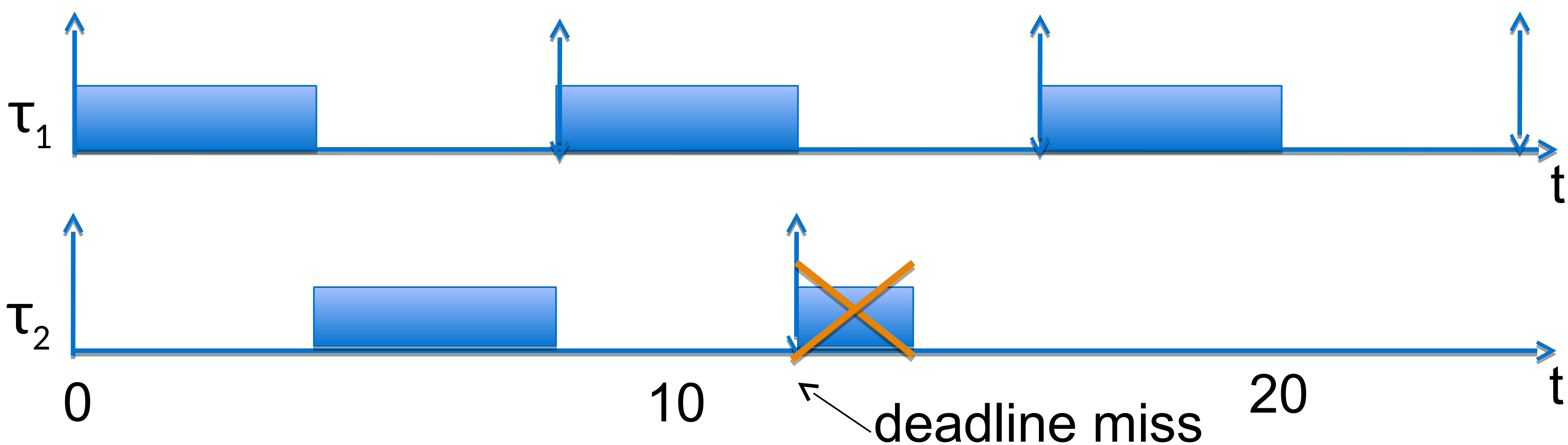
Rate Monotonic (RM)

- Case 2
  - τ_1 (C_1 = 4, T_1 = 8), high prio, τ_2 (C_2 = 4, T_2 = 12), low prio
  - Utilization: U = 4/8 + 4/12 ~= 0.833 > U_b(2) ~=0.828
  - Schedulability analysis: we do not know
  - In reality: it is schedulable



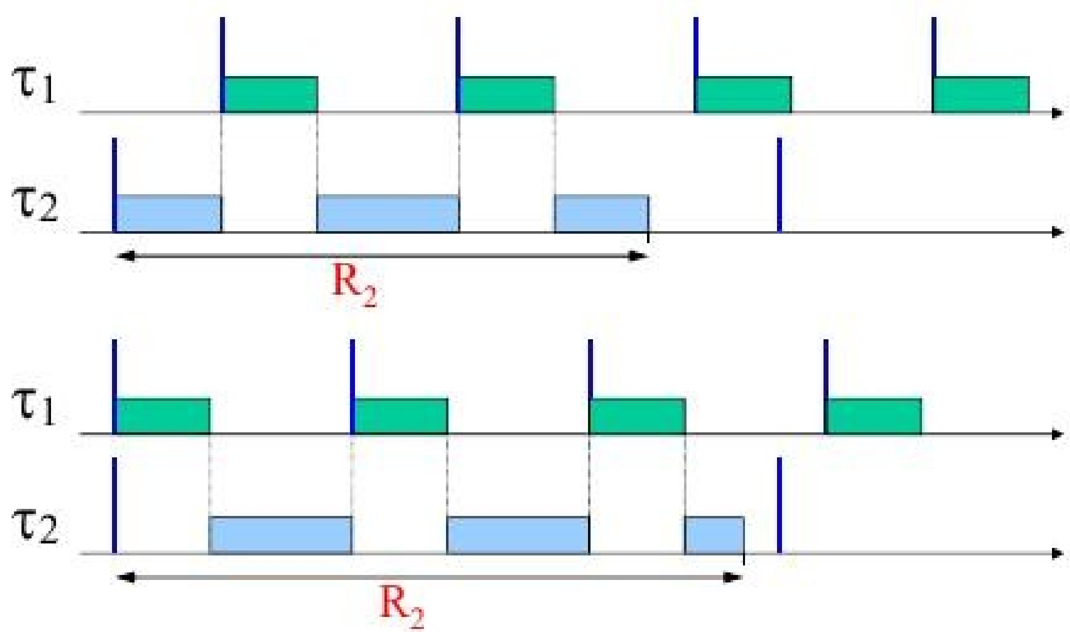
Rate Monotonic (RM)

- Case 3
  - τ_1 ($C_1 = 4$, $T_1 = 8$), high prio, τ_2 ($C_2 = 6$, $T_2 = 12$), low prio
  - Utilization: $U = 4/8 + 6/12 = 1 > U_b(2) \sim= 0.828$
  - Schedulability analysis: we do not know
  - In reality: it is not schedulable



Critical Instant

- For any task τ_i the longest response time occurs when it arrives together with all higher priority tasks



Rate Monotonic is optimal

- RM is optimal among all fixed priority algorithms when $D_i = T_i$

if there exists a fixed priority assignment which leads to a feasible schedule, then the RM schedule is feasible

$\Updownarrow$

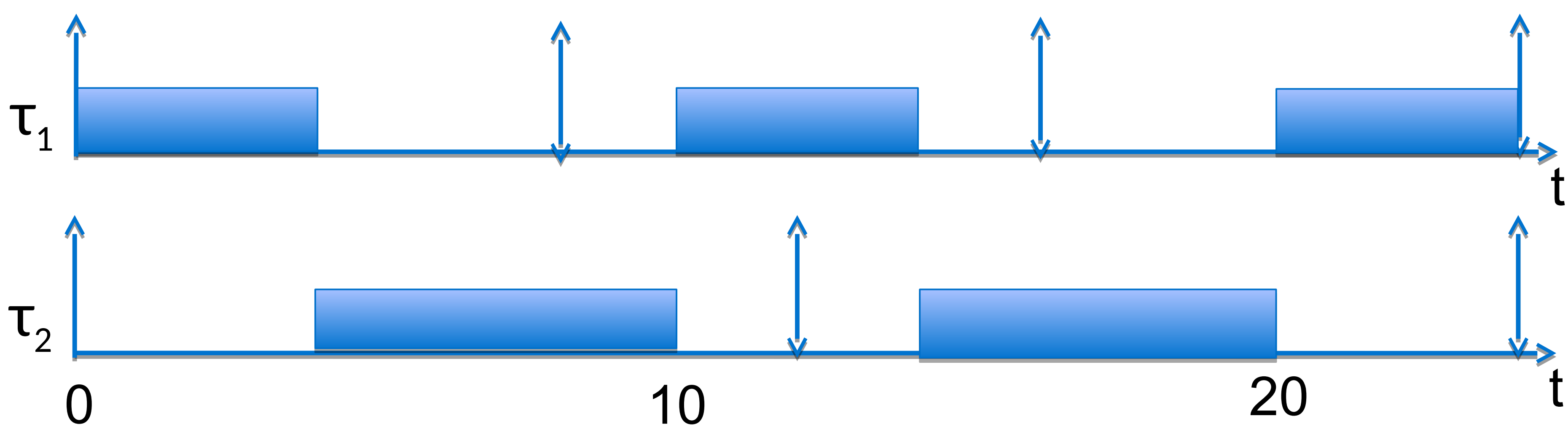
if a task set is not schedulable by RM, then it cannot be schedule by any fixed priority assignment

Earliest-Deadline First (EDF)

- Dynamic-Priority Scheduling Algorithm
- Task priority is inversely proportional to its current absolute deadline
  - Earlier deadline = higher priority
  - Each job of a task has a different deadline, hence a different priority
- Assumes periodic or sporadic tasks with $D_i = T_i$ on a uniprocessor, then…
- Schedulability analysis: the task set is schedulable if: $U <= U_b = 1$
  - Since task sets with $U > 1$ can not be scheduled by any algorithm, EDF is optimal

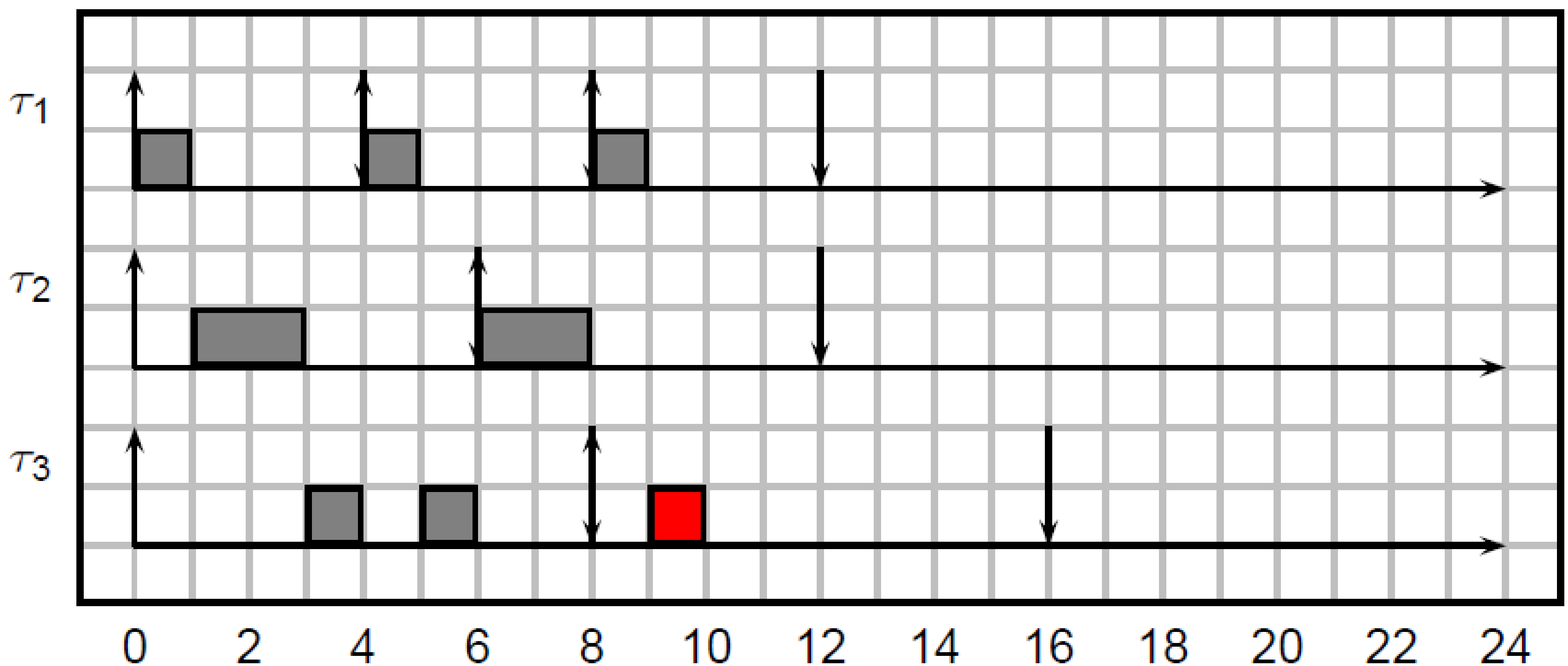
EDF example

- Case 3
  - τ_1 ($C_1 = 4$, $T_1 = 8$), high prio, τ_2 ($C_2 = 6$, $T_2 = 12$), low prio
  - Utilization: $U = 4/8 + 6/12 = 1$
  - Schedulability analysis: schedulable



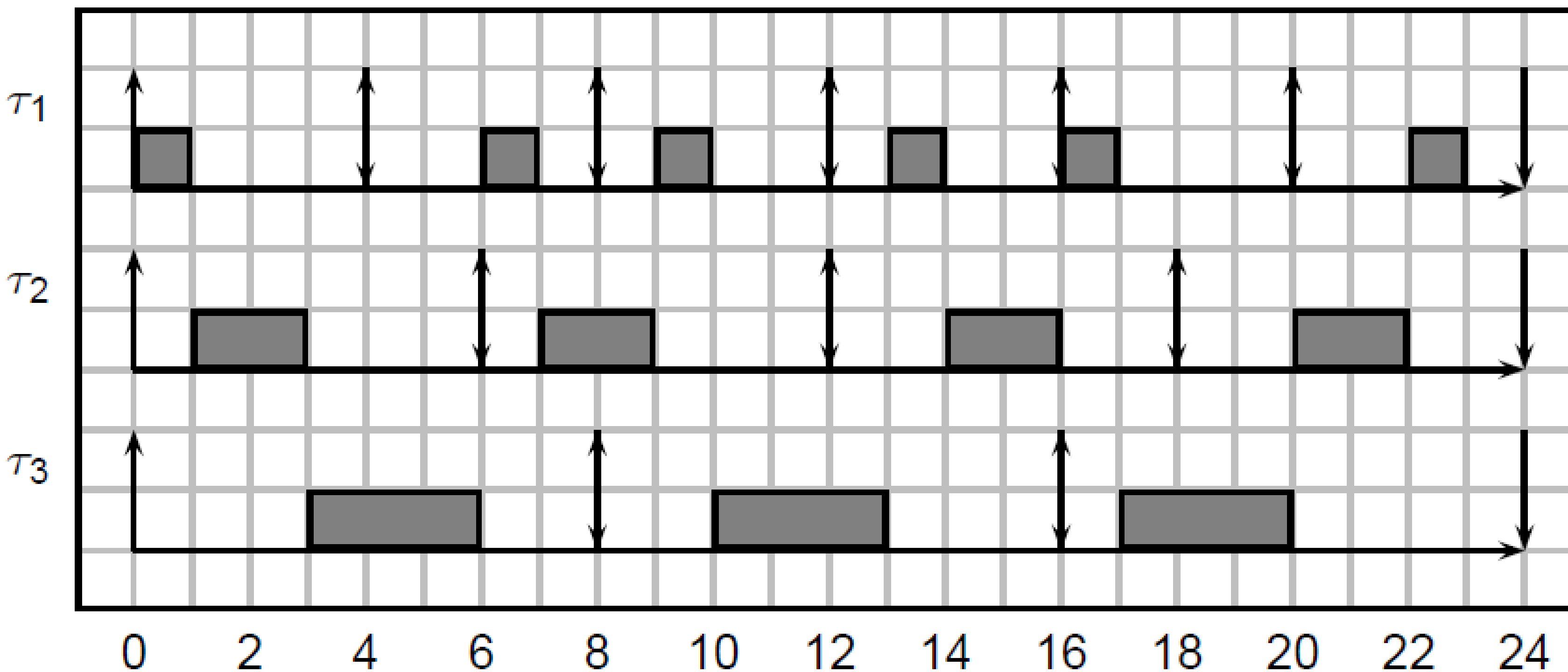
Another example

- τ_1 ($C_1 = 1$, $T_1 = 4$), τ_2 ($C_2 = 2$, $T_2 = 6$), τ_3 ($C_3 = 3$, $T_3 = 8$)
- Utilization: $U = 1/4 + 2/6 + 3/8 = 23/24$
- RM: don't know (utilization bound), in reality: not schedulable



Another example

- τ_1 ($C_1 = 1$, $T_1 = 4$), τ_2 ($C_2 = 2$, $T_2 = 6$), τ_3 ($C_3 = 3$, $T_3 = 8$)
- Utilization: U = 1/4 + 2/6 + 3/8 = 23/24
- EDF: schedulable



EDF is optimal

- EDF is optimal among all algorithms

if there exists a feasible schedule for a task set, then EDF will generate a feasible schedule

⇕

if a task set is not schedulable by EDF, then it cannot be schedule by any algorithm

EDF x RM

- In practice, industrial systems heavily favor RM over EDF. Why?

- For most task sets, RM has better utilization bound than log 2
  - There are more complex, necessary analysis
  - If task periods are harmonic (every period is an integer divisor of any larger period), then $U_b = 1$. This happens often in practice

EDF x RM

- RM is easier to implement in systems with limited number of priority levels

- RM is more transparent – easier to understand what is going on if something goes wrong (ex: overload)
  - I.e. if a task executes for longer than its prescribed worst-case time, higher priority tasks will be left untouched.

Different deadline assignments

- What happens if D_i != T_i ?
- EDF still optimal
- Instead of RM, use DM – deadline monotonic (best among fixed priority algorithms)
- If $D_i < T_i$, utilization bound still works by changing periods to deadlines in the formula – however this is pessimistic

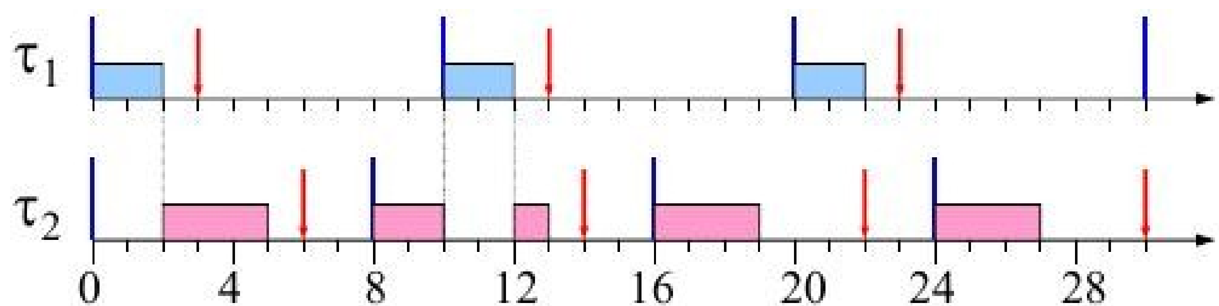
Different deadline assignments

- There exist exact schedulability analyses for both EDF and fixed priority that can be applied whatever the deadline – but they are pseudo-polynomial

- There are also analyses for asynchronous task sets – but exact analysis is exponential

Deadline Monotonic (DM)

- Assign priorities according to the deadline
  - shorter deadline = higher priority
- Optimal when $D_i < T_i$ for fixed priority scheduling

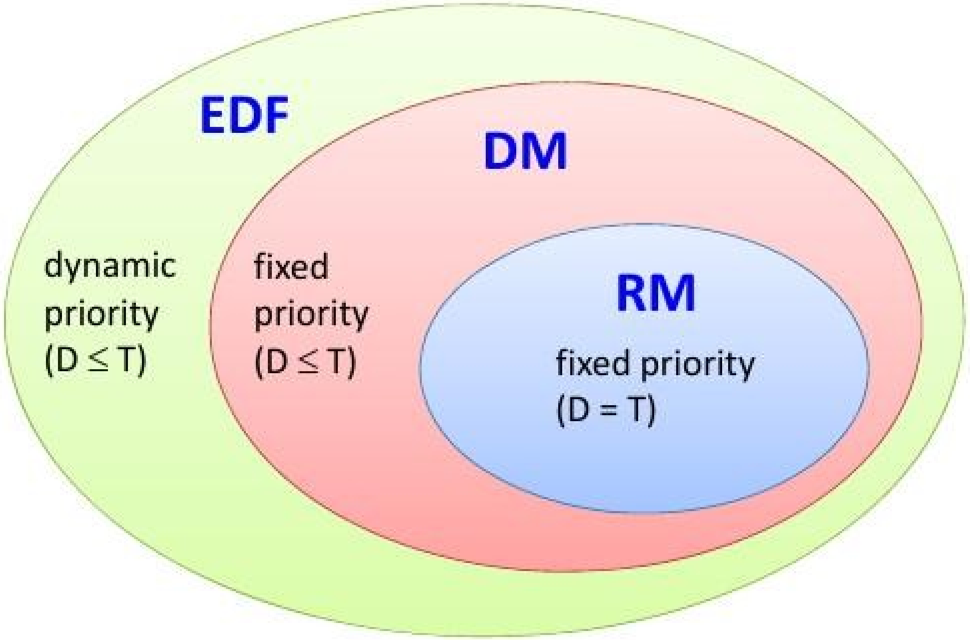


- Problem with the utilization bound

$$U_p = \sum_{i=1}^{n} \frac{C_i}{D_i} = \frac{2}{3} + \frac{3}{6} = 1.16 > 1$$

but the task set is schedulable

Optimality


EDF
DM
RM
dynamic
priority
(D ≤ T)
fixed
priority
(D ≤ T)
fixed priority
(D = T)

Response Time Analysis (RTA)

- Proposed by Audsley 90
- Iterative solution for fixed priority systems
- Exact schedulability test
- Tasks ordered by decreasing priority
- $R_i^{(k)}$: worst-case response time for τ_i at step k

$$R_i^{(0)} = C_i + \sum_{j=1}^{i-1} C_j$$

$$R_i^{(k)} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i^{(k-1)}}{T_j} \right\rceil C_j$$

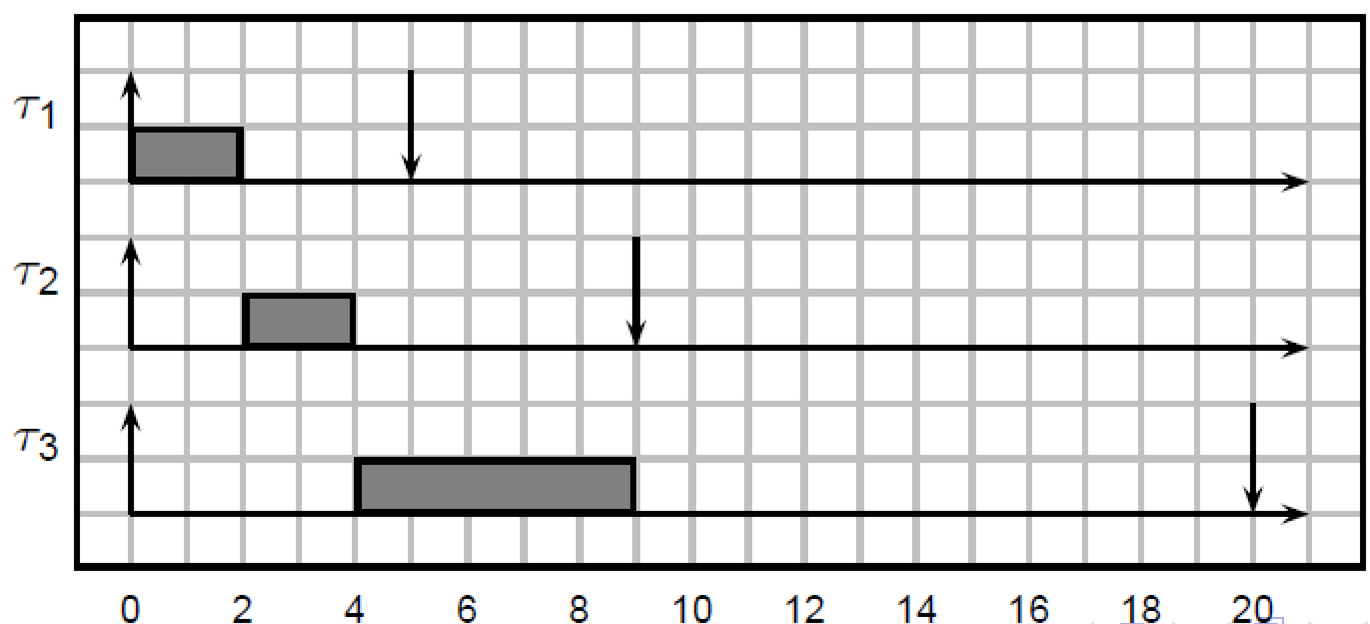
Response Time Analysis (RTA)

- If the iteration converges -> fixed point represents the worst-case response time for the task
- key idea: $\left\lceil \frac{R_i^{(k-1)}}{T_j} \right\rceil$ is the max # of jobs of τ_j that interfere with τ_i

RTA: example

- τ_1 (C_1 = 2, T_1 = 5), τ_2 (C_2 = 2, T_2 = 9), τ_3 (C_2 = 5, T_2 = 20); U = 0.872 > $U_b(3)$ = 0.780

$$R_3^{(0)} = C_3 + 1 \cdot C_1 + 1 \cdot C_2 = 9$$

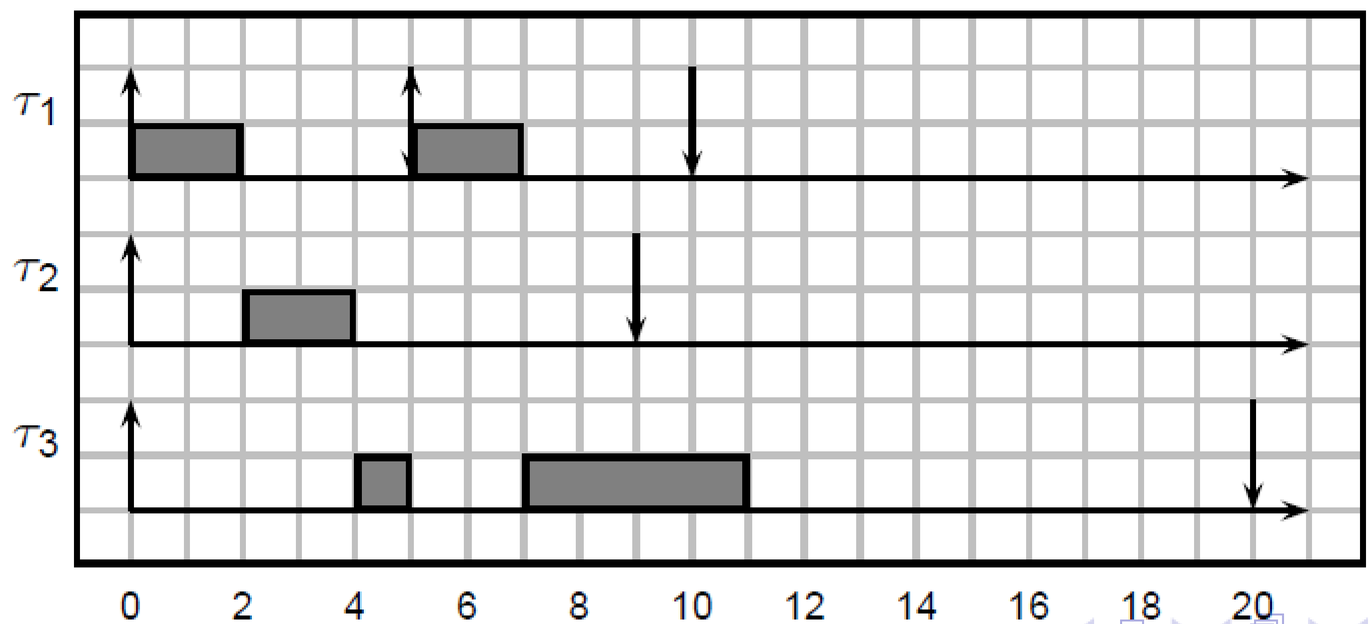


RTA: example

- τ_1 (C_1 = 2, T_1 = 5), τ_2 (C_2 = 2, T_1 = 9), τ_3 (C_2 = 5, T_1 = 20); U = 0.872 > $U_b(3)$ = 0.780

$$R_3^{(0)} = C_3 + 1 \cdot C_1 + 1 \cdot C_2 = 9$$

$$R_3^{(1)} = C_3 + 2 \cdot C_1 + 1 \cdot C_2 = 11$$



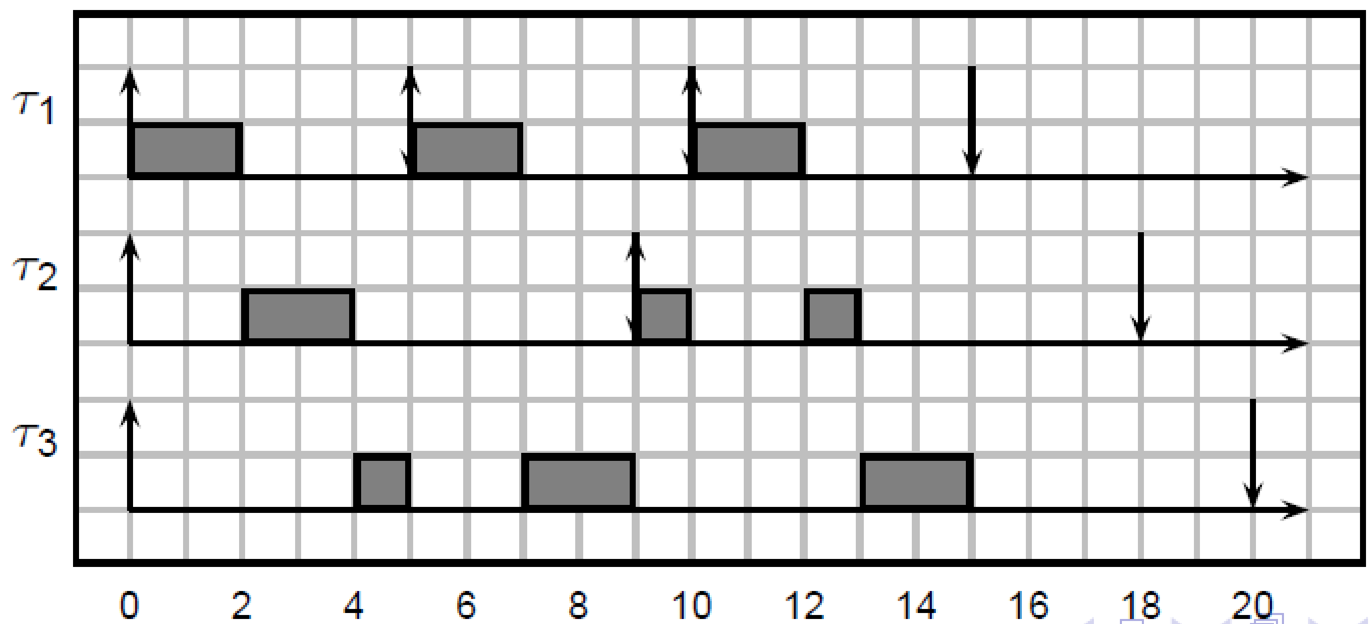
RTA: example

- τ_1 ($C_1 = 2$, $T_1 = 5$), τ_2 ($C_2 = 2$, $T_2 = 9$), τ_3 ($C_3 = 5$, $T_3 = 20$); $U = 0.872 > U_b(3) = 0.780$

$$R_3^{(0)} = C_3 + 1 \cdot C_1 + 1 \cdot C_2 = 9$$

$$R_3^{(1)} = C_3 + 2 \cdot C_1 + 1 \cdot C_2 = 11$$

$$R_3^{(2)} = C_3 + 3 \cdot C_1 + 2 \cdot C_2 = 15$$



RTA: example

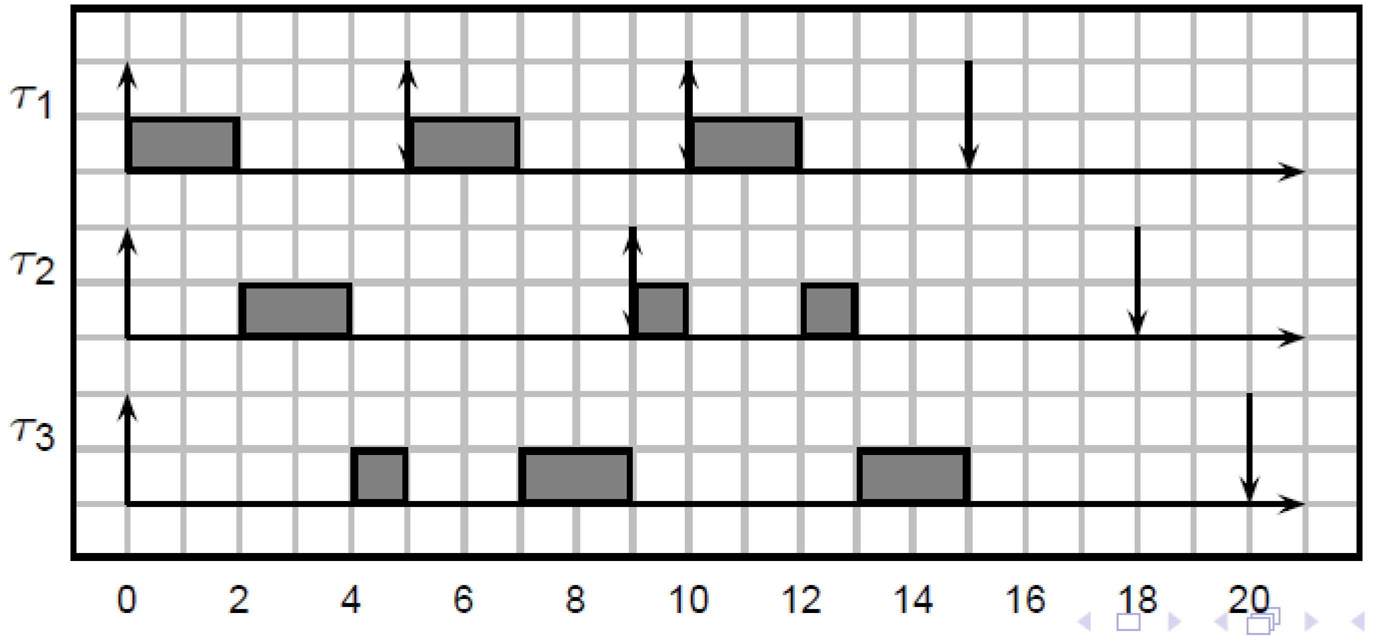
- τ_1 ($C_1 = 2$, $T_1 = 5$), τ_2 ($C_2 = 2$, $T_2 = 9$), τ_3 ($C_3 = 5$, $T_3 = 20$); $U = 0.872 > U_b(3) = 0.780$

$$R_3^{(0)} = C_3 + 1 \cdot C_1 + 1 \cdot C_2 = 9$$

$$R_3^{(1)} = C_3 + 2 \cdot C_1 + 1 \cdot C_2 = 11$$

$$R_3^{(2)} = C_3 + 3 \cdot C_1 + 2 \cdot C_2 = 15$$

$$R_3^{(3)} = C_3 + 3 \cdot C_1 + 2 \cdot C_2 = 15 = R_3^{(2)}$$



Least Laxity First (LLF)

- Assign a higher priority to a task with smaller laxity
  - As EDF, it is optimal

- There is the need to know the current time to compute the priority

Review

- Periodic task model
- Uniprocessor RT scheduling
  - Task are independent
  - Cyclic executive
  - RM
  - EDF
  - RM
  - LLF

References

- Giorgio Buttazo. Real-time systems course. Scuola Superiore Santanna.
- Rodolfo Pellizzoni. Cyber-physical systems course. University of Waterloo.
- Farines, Silva, Oliveira. Sistemas de Tempo Real. DAS/UFSC. Julho 2000.
- G.C. Buttazzo: Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and applications. Springer, 2004.