Lecture 9 – Real-time Scheduling

CSE 456: Embedded Systems

Real-Time Scheduling of Periodic Tasks

Overview

Table of some known *preemptive scheduling algorithms for periodic tasks*:

	Deadline equals period	Deadline smaller than period
static priority	RM (rate-monotonic)	DM (deadline-monotonic)
dynamic priority	EDF	EDF*

Model of Periodic Tasks

- *Examples:* sensory data acquisition, low-level servoing, control loops, action planning and system monitoring.
- When a *control application* consists of several concurrent periodic tasks with individual timing constraints, the OS has to guarantee that each periodic instance is regularly activated at its proper rate and is completed within its deadline.

- Definitions:

Γ : denotes a set of periodic tasks
τ_i : denotes a generic periodic task
$\tau_{i,j}$: denotes the jth instance of task i
$r_{i,j}, s_{i,j}, f_{i,j}, d_{i,j}$: denote the release time, start time, finishing time, absolute deadline of the jth instance of task i
Φ_i : denotes the phase of task i (release time of its first instance)
D_i : denotes the relative deadline of task i

Model of Periodic Tasks

- *The following hypotheses are assumed on the tasks:*
  - The instances of a periodic task are *regularly activated at a constant rate*. The interval T_i between two consecutive activations is called period. The release times satisfy

$$r_{i,j} = \Phi_i + (j-1)T_i$$

  - All instances have the *same worst case execution time* C_i
  - All instances of a periodic task have the *same relative deadline* D_i. Therefore, the absolute deadlines satisfy

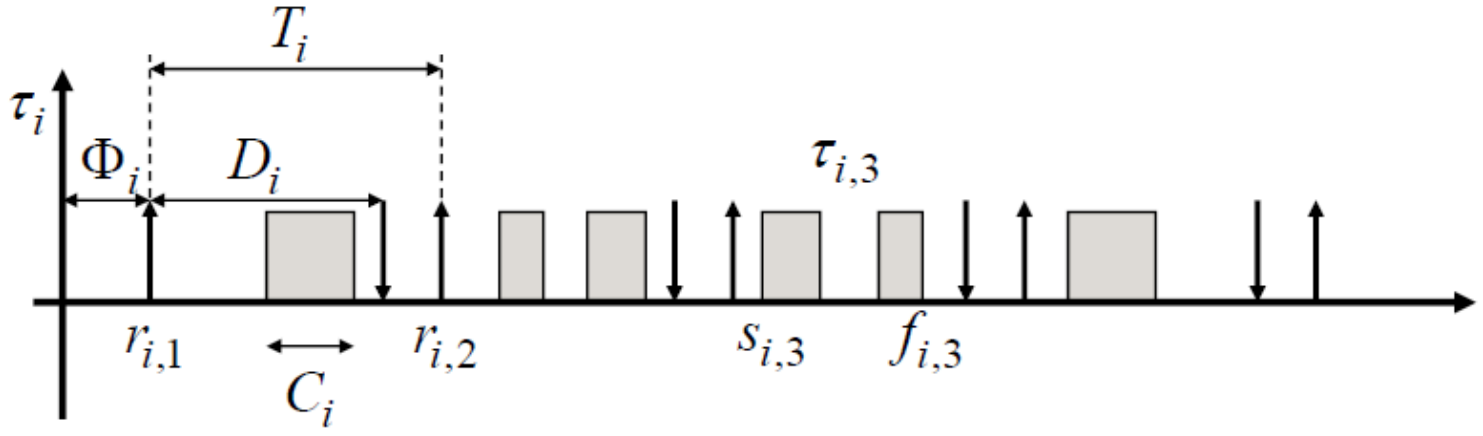
$$d_{i,j} = \Phi_i + (j-1)T_i + D_i$$

  - Often, the relative deadline equals the period $D_i = T_i$ (*implicit deadline*), and therefore

$$d_{i,j} = \Phi_i + jT_i$$

Model of Periodic Tasks

- *The following hypotheses are assumed on the tasks (continued):*
  - All periodic tasks are *independent*; that is, there are no precedence relations and no resource constraints.
  - *No task can suspend itself*, for example on I/O operations.
  - All tasks are *released as soon as they arrive*.
  - All *overheads* in the OS kernel are assumed to be *zero*.
  - *Example:*



Rate Monotonic Scheduling (RM)

- *Assumptions:*
  - Task priorities are assigned to tasks before execution and do not change over time (static priority assignment).
  - RM is intrinsically preemptive: the currently executing task is preempted by a task with higher priority.
  - Deadlines equal the periods $D_i = T_i$.

Rate-Monotonic Scheduling Algorithm: Each task is assigned a priority. Tasks with higher request rates (that is with shorter periods) will have higher priorities. Tasks with higher priority interrupt tasks with lower priority.

Periodic Tasks

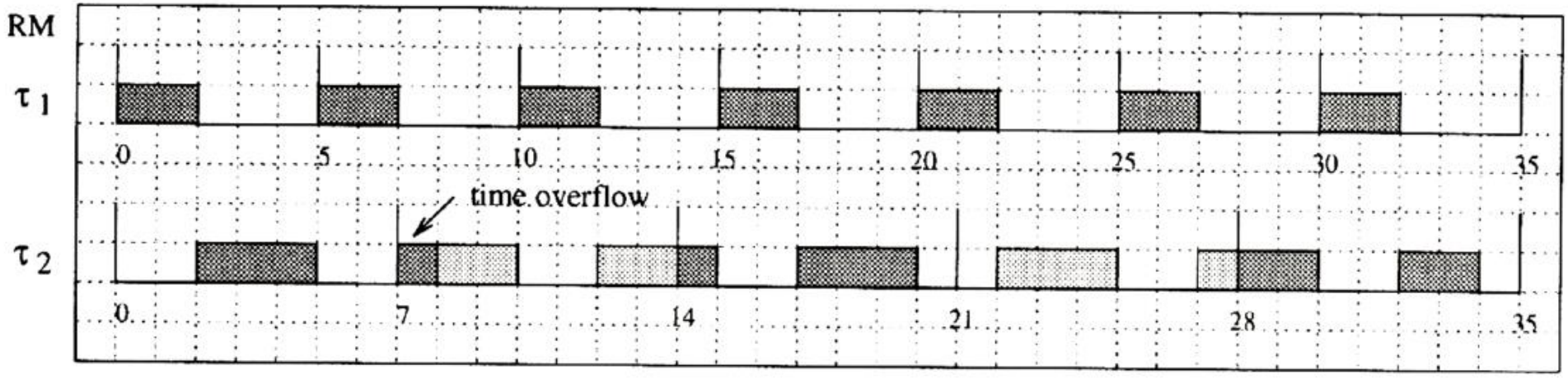
Example: 2 tasks, deadlines = periods

	Ci	Di
Task 1	2	5
Task 2	4	7

Periodic Tasks

Example: 2 tasks, deadlines = periods

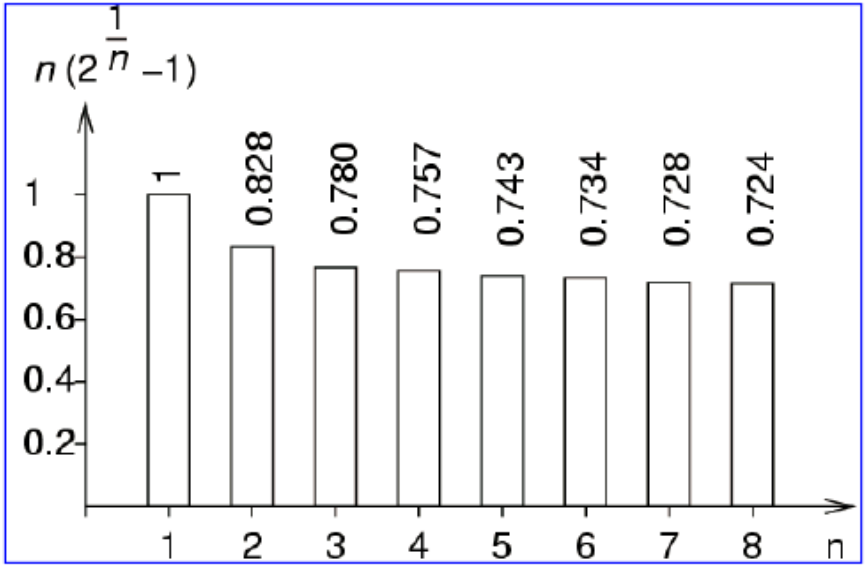
	Ci	Di
Task 1	2	5
Task 2	4	7



Rate Monotonic Scheduling (RM)

Schedulability analysis: A set of periodic tasks is schedulable with RM if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(2^{1/n} - 1\right)$$



This condition is sufficient but not necessary.

The term $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ denotes the *processor utilization factor U* which is the fraction of processor time spent in the execution of the task set.

Deadline Monotonic Scheduling (DM)

- Assumptions are as in rate monotonic scheduling, but *deadlines may be smaller than the period*, i.e.

$$C_i \leq D_i \leq T_i$$

Algorithm: Each task is assigned a priority. Tasks with smaller relative deadlines will have higher priorities. Tasks with higher priority interrupt tasks with lower priority.

- *Schedulability Analysis:* A set of periodic tasks is schedulable with DM if

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n\left(2^{1/n} - 1\right)$$

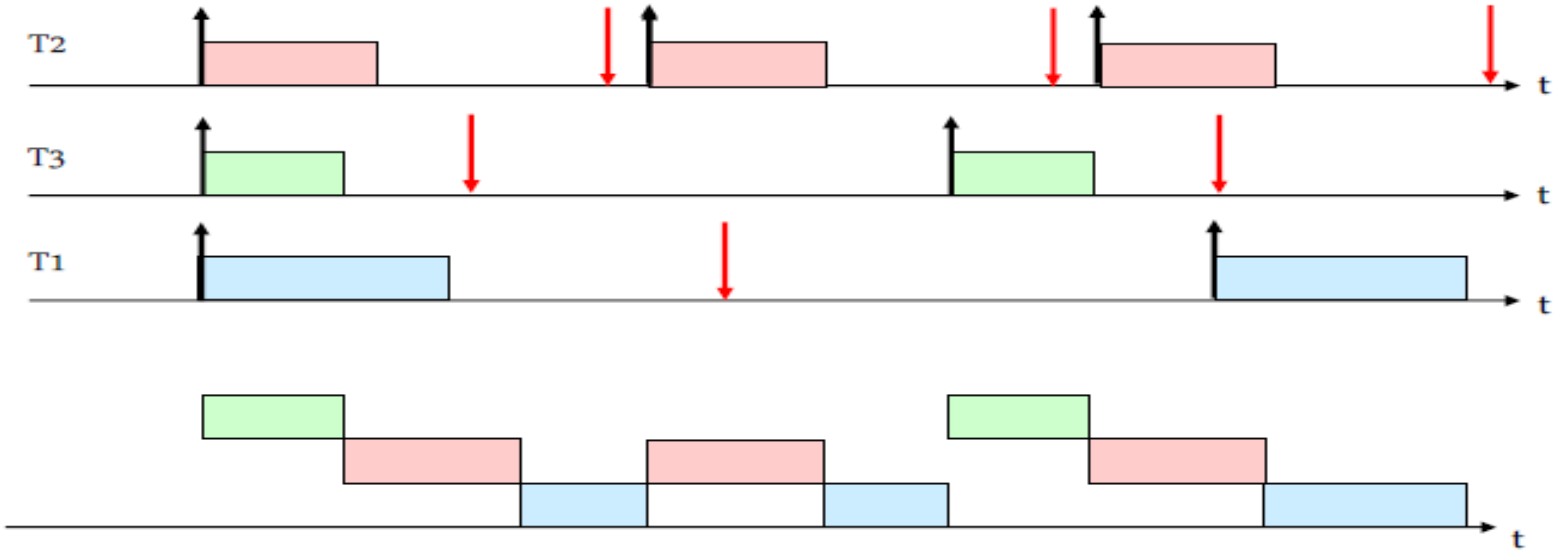
This condition is sufficient but not necessary (in general).

Deadline Monotonic Scheduling (DM)

	Ci	Di	Ti
Task 1	5	10	20
Task 2	3	8	9
Task 3	2	5	15

Deadline Monotonic Scheduling (DM)

	Ci	Di	Ti
Task 1	5	10	20
Task 2	3	8	9
Task 3	2	5	15



Deadline Monotonic Scheduling (DM)

There is also a *necessary and sufficient schedulability test* which is computationally more involved. It is based on the following observations:

- The *worst-case processor demand* occurs when all tasks are released simultaneously; that is, at their critical instances.
- For each task *i*, the sum of its processing time and the *interference* imposed by higher priority tasks must be less than or equal to D_i .
- A measure of the *worst case interference* for task i can be computed as the sum of the processing times of all higher priority tasks released before some time t where tasks are ordered according to $m < n \Leftrightarrow D_m < D_n$:

$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

Deadline Monotonic Scheduling (DM)

- The *longest response time* R_i of a periodic task i is computed, at the critical instant, as the sum of its computation time and the interference due to preemption by higher priority tasks:

$$R_i = C_i + I_i$$

- Hence, the schedulability test needs to compute the smallest R_i that satisfies

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

  for all tasks i. Then, $R_i \leq D_i$ must hold for all tasks i.

- It can be shown that this *condition is necessary and sufficient*.

Deadline Monotonic Scheduling (DM)

The longest response times R_i of the periodic tasks *i* can be computed iteratively by the following algorithm:

```
Algorithm: DM_guarantee (Γ)
{       for (each τ_i∈Γ){
                I = 0;
                do {
                        R = I + C_i;
                        if (R > D_i) return(UNSCHEDULABLE);
                        I = ∑^{j=1,…,(i-1)} ⌈R/T_j⌉ C_j;
                } while (I + C_i > R);
        }
        return(SCHEDULABLE);
}
```

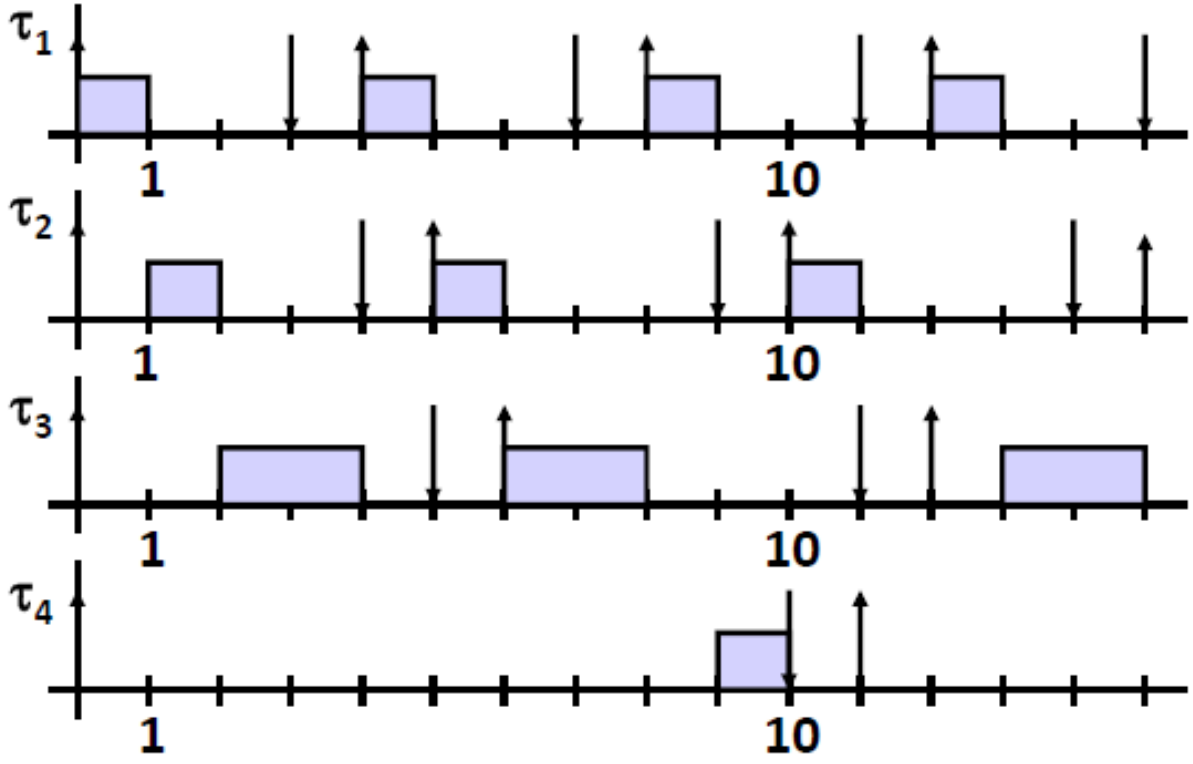
DM Example

Example:

- Task 1: $C_1 = 1; T_1 = 4; D_1 = 3$
- Task 2: $C_2 = 1; T_2 = 5; D_2 = 4$
- Task 3: $C_3 = 2; T_3 = 6; D_3 = 5$
- Task 4: $C_4 = 1; T_4 = 11; D_4 = 10$
- Algorithm for the schedulability test for task 4:
  - Step 0: $R_4 = 1$
  - Step 1: $R_4 = 5$
  - Step 2: $R_4 = 6$
  - Step 3: $R_4 = 7$
  - Step 4: $R_4 = 9$
  - Step 5: $R_4 = 10$

DM Example

$U = 0.874$ $\quad \sum_{i=1}^{n} \frac{C_i}{D_i} = 1.08 > n\left(2^{1/n} - 1\right) = 0.757$



EDF Scheduling (earliest deadline first)

- *Assumptions:*
  - dynamic priority assignment
  - intrinsically preemptive
  - $D_i \leq T_i$
- *Algorithm:* The currently executing task is preempted whenever another periodic instance with earlier deadline becomes active.

$$d_{i,j} = \Phi_i + (j-1)T_i + D_i$$

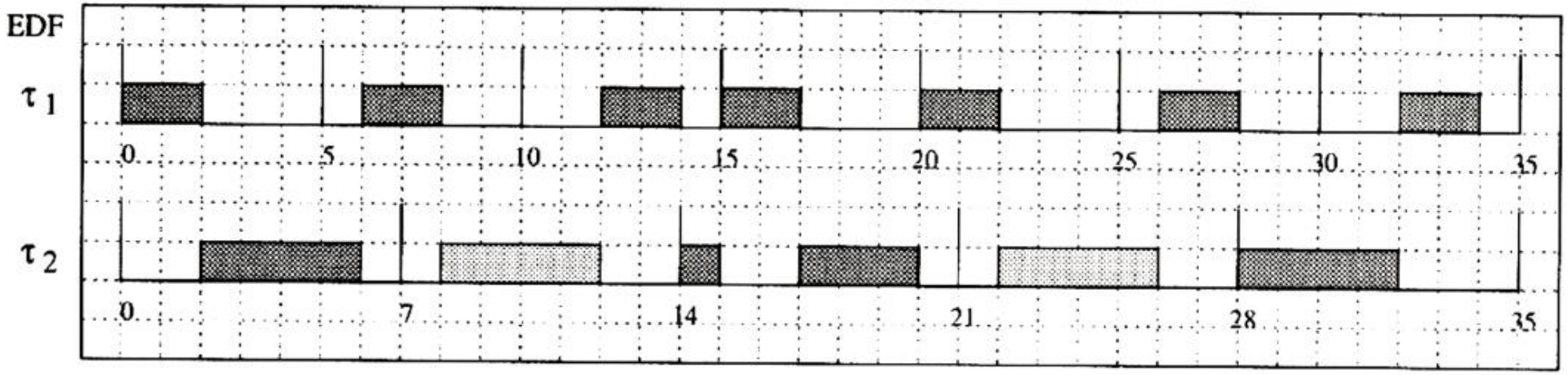
- *Optimality:* No other algorithm can schedule a set of periodic tasks if the set that can not be scheduled by EDF.
- The proof is simple and follows that of the aperiodic case.

EDF Scheduling

	Ci	Di=Ti
Task 1	2	5
Task 2	4	7

EDF Scheduling

	Ci	Di=Ti
Task 1	2	5
Task 2	4	7



EDF Scheduling

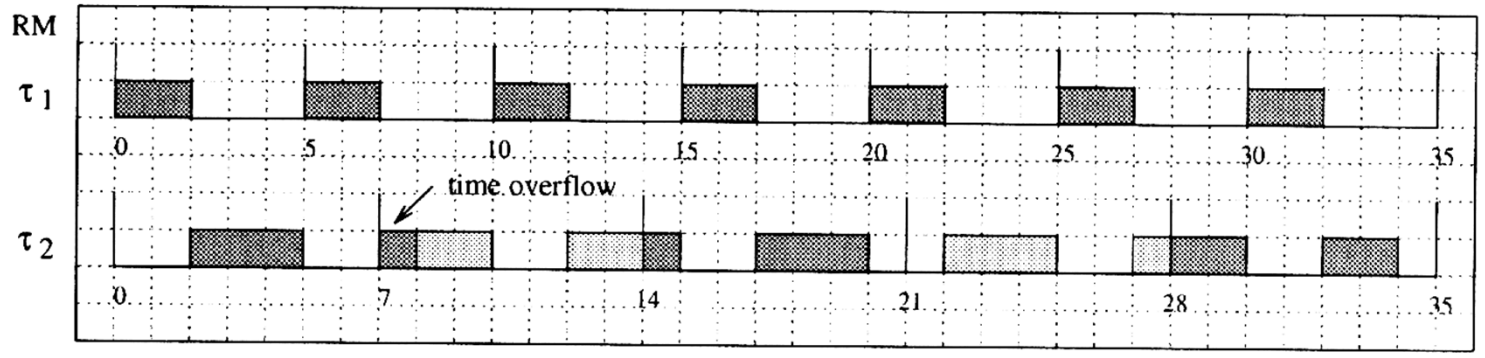
A *necessary and sufficient schedulability test* for $D_i = T_i$:

A set of periodic tasks is schedulable with EDF if and only if $\sum_{i=1}^{n} \frac{C_i}{T_i} = U \leq 1$

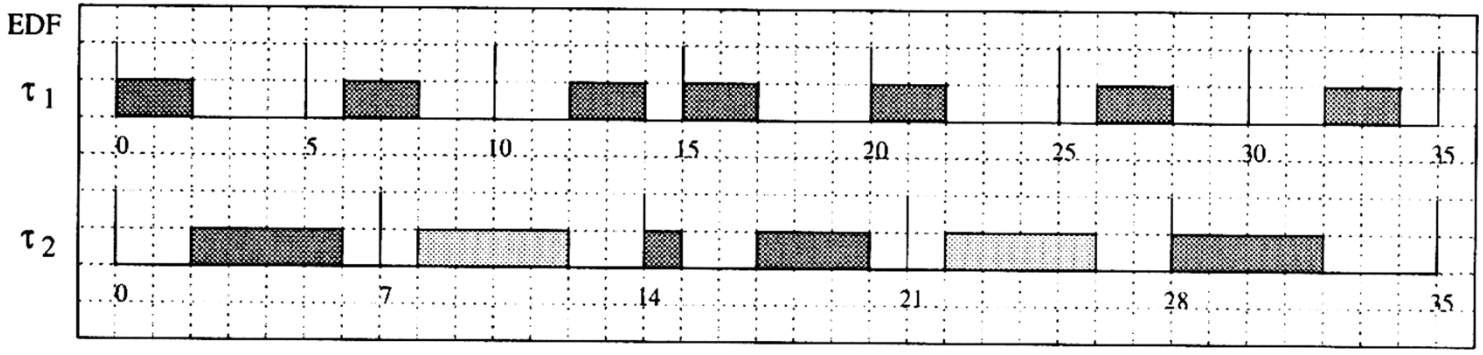
The term $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ denotes the *average processor utilization.*

EDF Scheduling

	Ci	Di=Ti
Task 1	2	5
Task 2	4	7



(a)



(b)