Scuola Superiore
Sant'Anna

Exploiting Uni-Processor Schedulability Analysis for Partitioned Task Allocation on Multi-Processors with Precedence Constraints

M. Bambagini, G. Buttazzo, S. Hendseth

3rd International Real-Time Scheduling Open Problems Seminar (RTSOPS 12)
Pisa, Italy, July 10 2012

TeCIP Institute, Scuola Superiore Sant'Anna
Area della Ricerca CNR, Via Moruzzi, 1
56127 Pisa, ITALY

Retis
Real-Time Systems Laboratory

Outline

Introduction

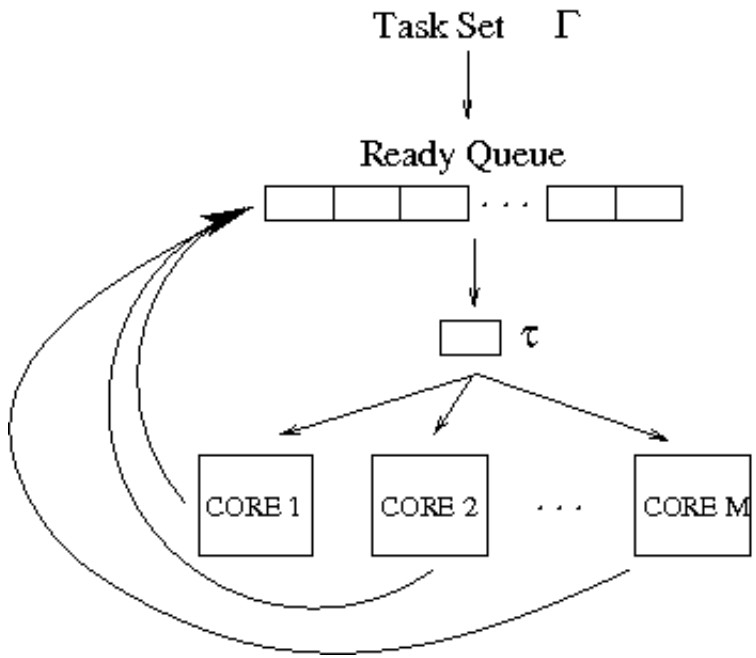
Scheduling a task set on distributed systems under RT and precedence constraints is an actual problem

Introduction

Scheduling a task set on distributed systems under RT and precedence constraints is an actual problem

Main approaches [1]:

- **Global scheduling**
  - Pro: Better expectations
  - Con: Task migration costs
- Partitioned scheduling
  - Pro: Divide and conquer
  - Con: Worst-case driven, NP-hard

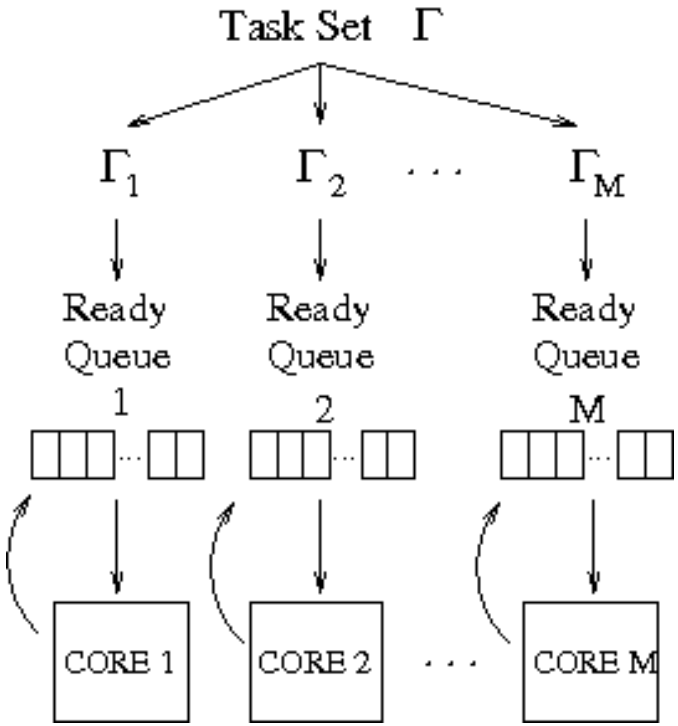


Introduction

Scheduling a task set on distributed systems under RT and precedence constraints is an actual problem

Main approaches [1]:

- Global scheduling
  - Pro: Better expectations
  - Con: Task migration costs
- **Partitioned scheduling**
  - Pro: Divide and conquer
  - Con: Worst-case driven, NP-hard

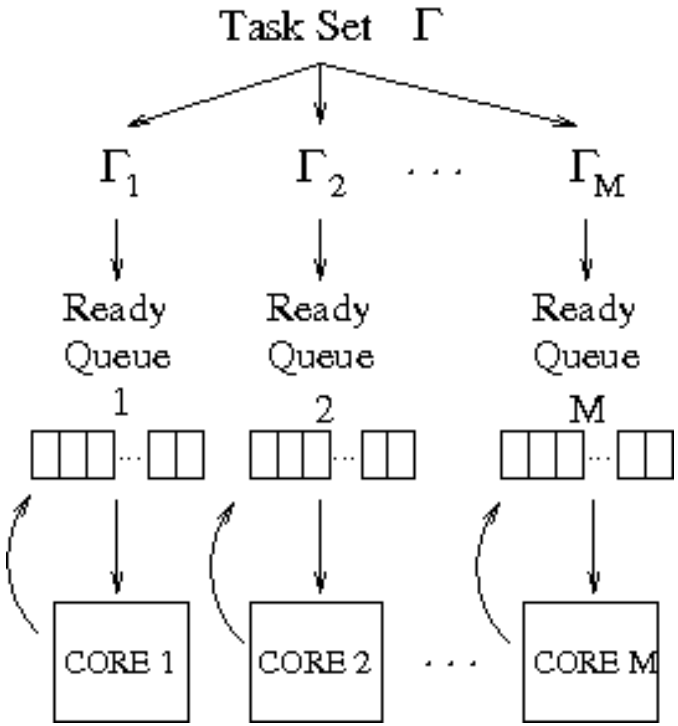


Introduction

Scheduling a task set on distributed systems under RT and precedence constraints is an actual problem

Main approaches [1]:

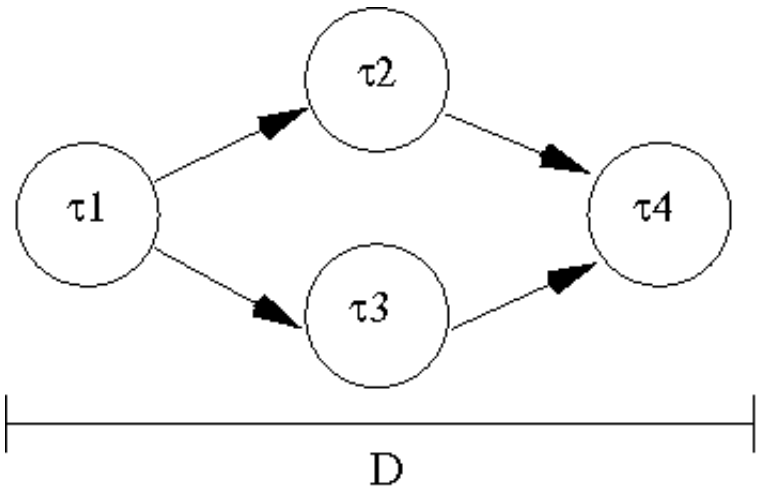
- Global scheduling
  - Pro: Better expectations
  - Con: Task migration costs
- **Partitioned scheduling** ⇐
  - Pro: Divide and conquer
  - Con: Worst-case driven, NP-hard

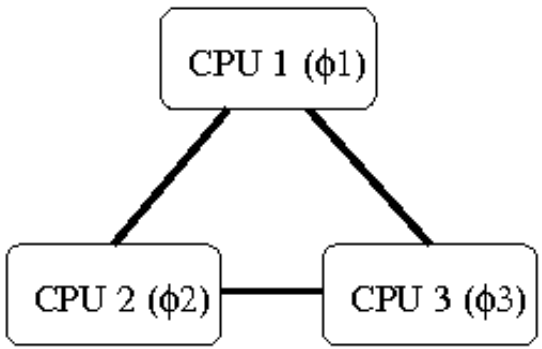


Introduction

Up to date partitioned problem: Allocate n preemptive real-time tasks on m heterogeneous processors, to minimize f, considering:

- non-negligible communication costs
- precedence constraints among tasks
- application deadline and period

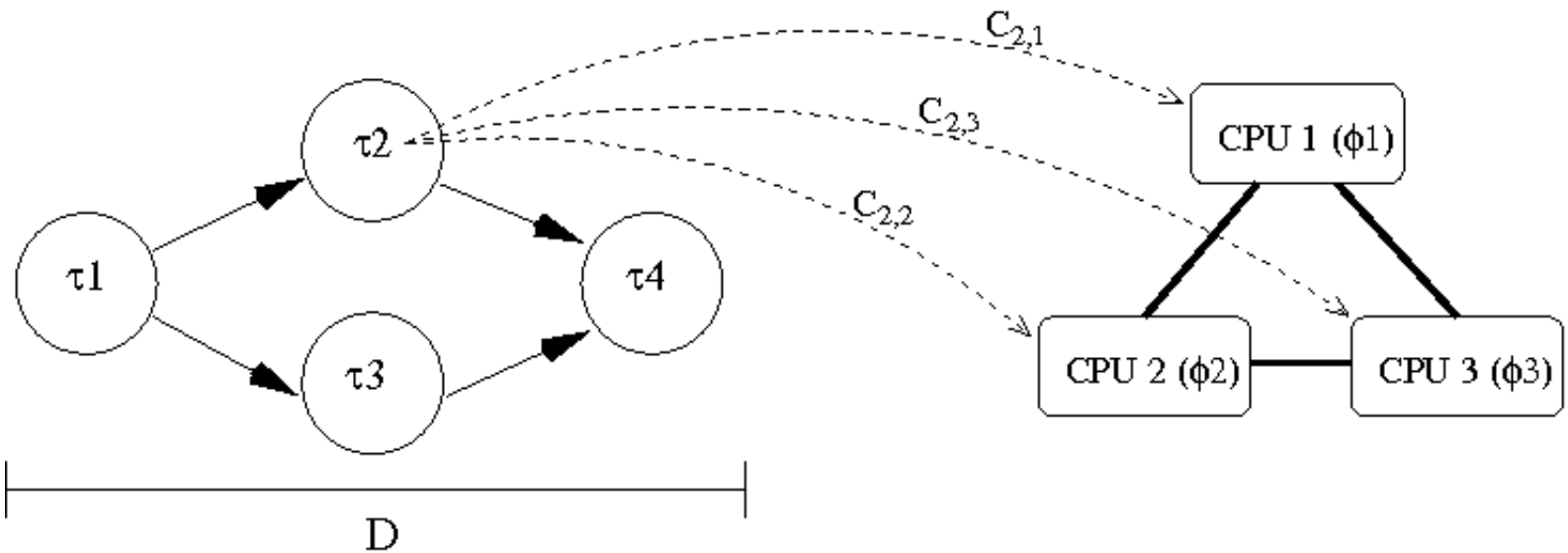




Introduction

Up to date partitioned problem: Allocate n preemptive real-time tasks on m heterogeneous processors, to minimize f, considering:

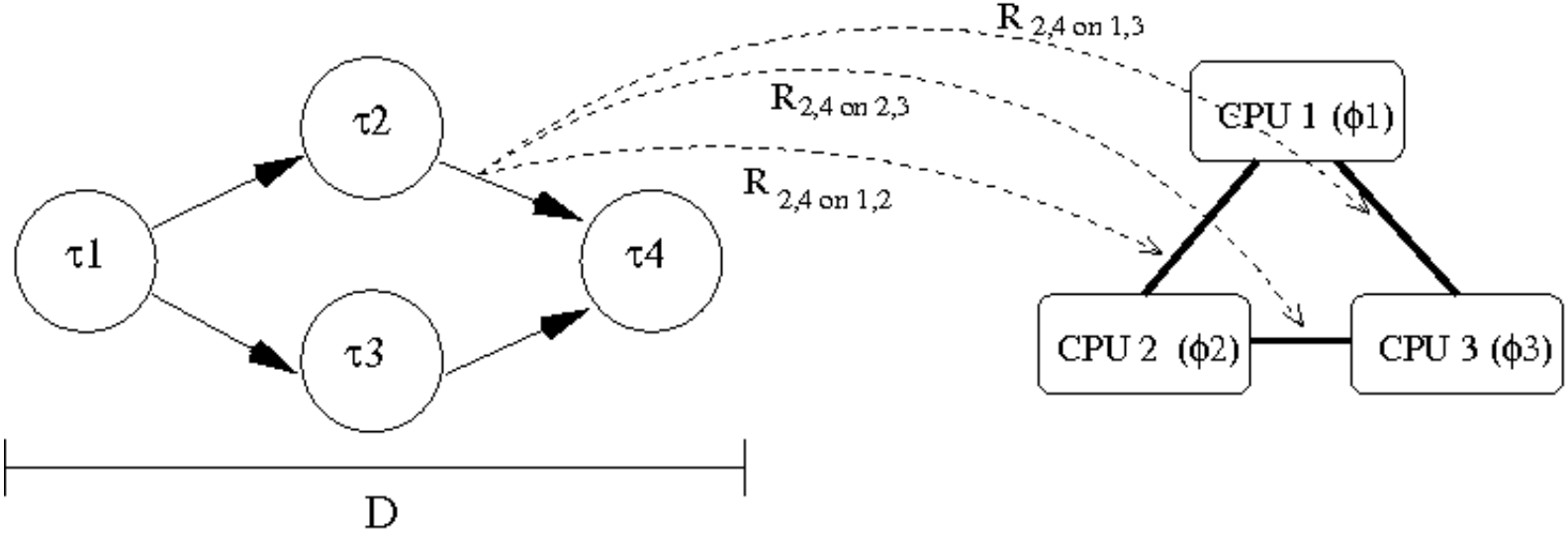
- non-negligible communication costs
- precedence constraints among tasks
- application deadline and period



Introduction

Up to date partitioned problem: Allocate n preemptive real-time tasks on m heterogeneous processors, to minimize f, considering:

- non-negligible communication costs
- precedence constraints among tasks
- application deadline and period



Introduction

Classical solving approach of a partitioned algorithm:

1. Task allocation
2. Static scheduling: $\forall \tau_i \in \Gamma \Rightarrow s_i, f_i$
   - 2.1 Precedence: $\forall \tau_i, \tau_j \in \Gamma | \tau_i \rightarrow \tau_j : s_j \geq f_i + comm_{\tau_i \rightarrow \tau_j}$
   - 2.2 Real-time: $\max_{\forall \tau_i \in \Gamma} f_i \leq D$

Introduction

Classical solving approach of a partitioned algorithm:

1. Task allocation
2. Static scheduling: $\forall \tau_i \in \Gamma \Rightarrow s_i, f_i$
   - 2.1 Precedence: $\forall \tau_i, \tau_j \in \Gamma | \tau_i \rightarrow \tau_j : s_j \geq f_i + comm_{\tau_i \rightarrow \tau_j}$
   - 2.2 Real-time: $\max_{\forall \tau_i \in \Gamma} f_i \leq D$

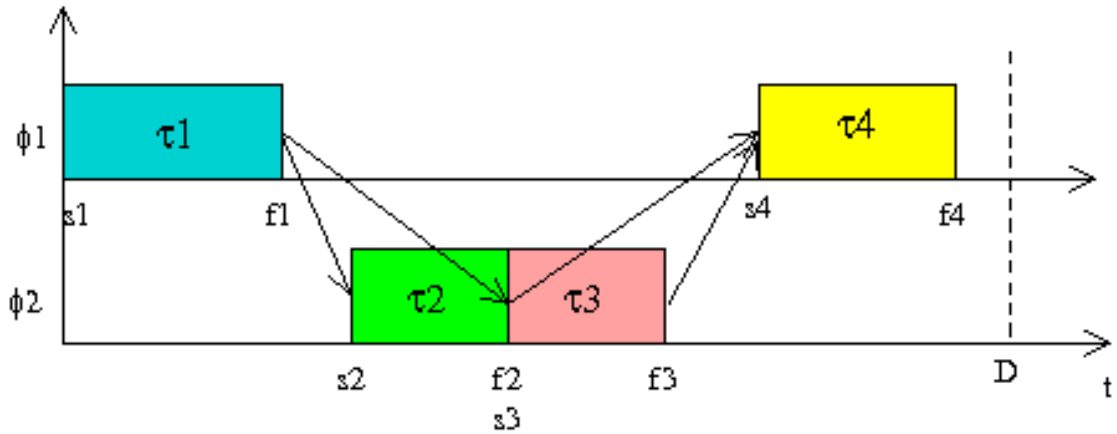
$$\phi_1 = \{\tau_1, \tau_4\},\ \phi_2 = \{\tau_2, \tau_3\},\ \phi_3 = \{\oslash\}$$

Introduction

Classical solving approach of a partitioned algorithm:

1. Task allocation
2. Static scheduling: $\forall \tau_i \in \Gamma \Rightarrow s_i, f_i$
   - 2.1 Precedence: $\forall \tau_i, \tau_j \in \Gamma | \tau_i \rightarrow \tau_j : s_j \geq f_i + comm_{\tau_i \rightarrow \tau_j}$
   - 2.2 Real-time: $\max_{\forall \tau_i \in \Gamma} f_i \leq D$

$$\phi_1 = \{\tau_1, \tau_4\},\ \phi_2 = \{\tau_2, \tau_3\},\ \phi_3 = \{\oslash\}$$

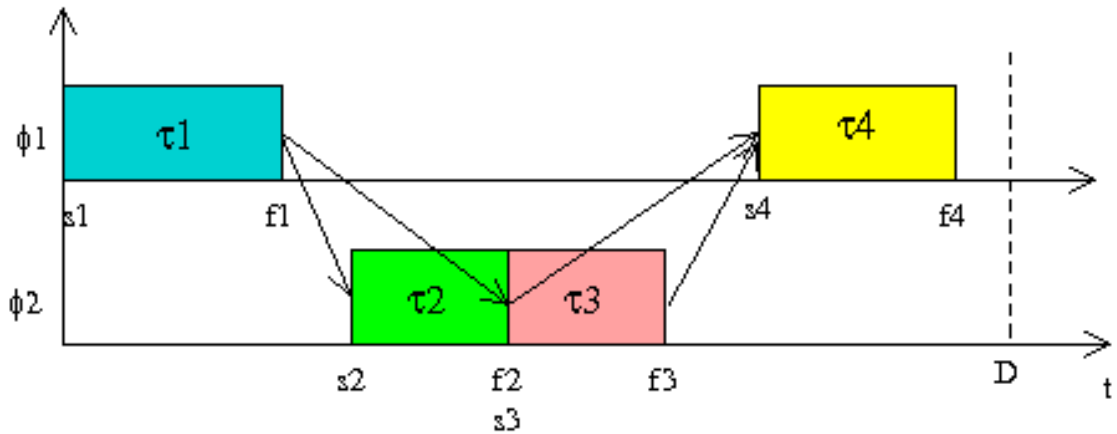


Introduction

Classical solving approach of a partitioned algorithm:

1. Task allocation
2. Static scheduling: $\forall \tau_i \in \Gamma \Rightarrow s_i, f_i$
   - 2.1 Precedence: $\forall \tau_i, \tau_j \in \Gamma | \tau_i \rightarrow \tau_j : s_j \geq f_i + comm_{\tau_i \rightarrow \tau_j}$
   - 2.2 Real-time: $\max_{\forall \tau_i \in \Gamma} f_i \leq D$

$$\phi_1 = \{\tau_1, \tau_4\},\ \phi_2 = \{\tau_2, \tau_3\},\ \phi_3 = \{\oslash\}$$



Approaches: complete search[2,3], meta-heuristics[4], heuristics[5]

Open problem

Instead of coping with static schedules, how is it possible to transform precedence constraints into real-time constraints to exploit well-known uni-processor analysis?

In other words: how to assign arrival times and deadlines so that the schedule automatically guarantees the precedence constraints?

Open problem

Instead of coping with static schedules, how is it possible to transform precedence constraints into real-time constraints to exploit well-known uni-processor analysis?

In other words: how to assign arrival times and deadlines so that the schedule automatically guarantees the precedence constraints?

1 distributed problem $\longrightarrow$ m simpler real-time local problems

Open problem

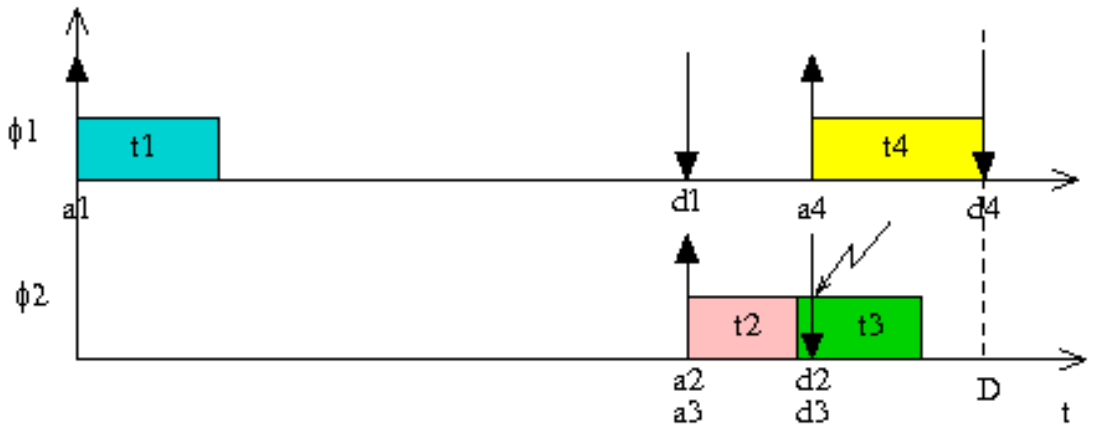
A similar assignment problem was proposed by Buttazzo et al. [9] who in turn extended the idea proposed by Chetto et al. [10]

Homogeneous systems and negligible communication costs are considered. Tasks are collected into flows (processors independent)

Open problem

A similar assignment problem was proposed by Buttazzo et al. [9] who in turn extended the idea proposed by Chetto et al. [10]

Homogeneous systems and negligible communication costs are considered. Tasks are collected into flows (processors independent)

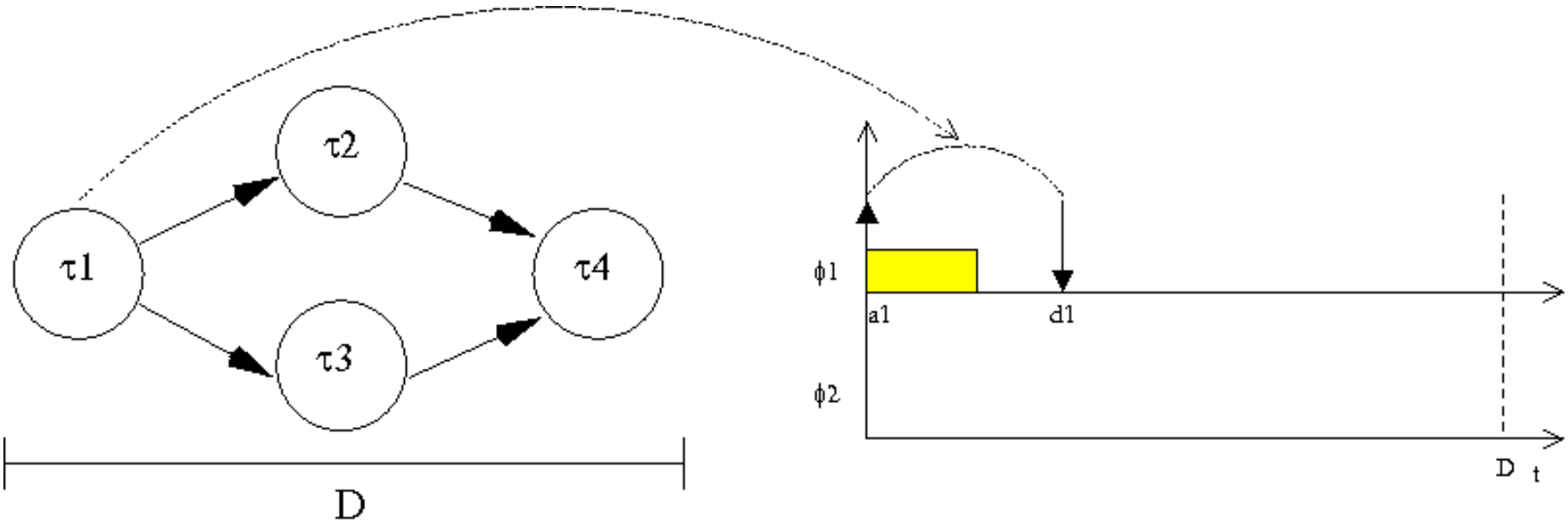


This problem is still an open question

Motivational example

Given the allocation: τ_1, τ_4 on ϕ_1, τ_2, τ_3 on ϕ_2 and ϕ_3 empty

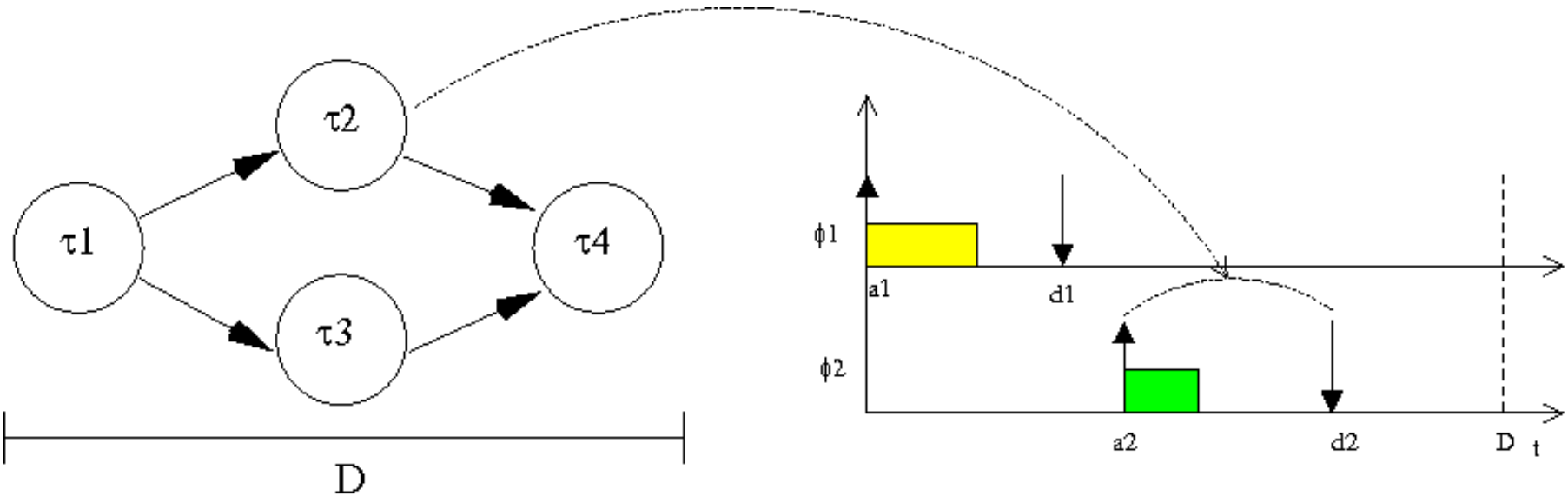
Deadline and activation assignment step (assuming EDF):



Motivational example

Given the allocation: τ_1, τ_4 on ϕ_1, τ_2, τ_3 on ϕ_2 and ϕ_3 empty

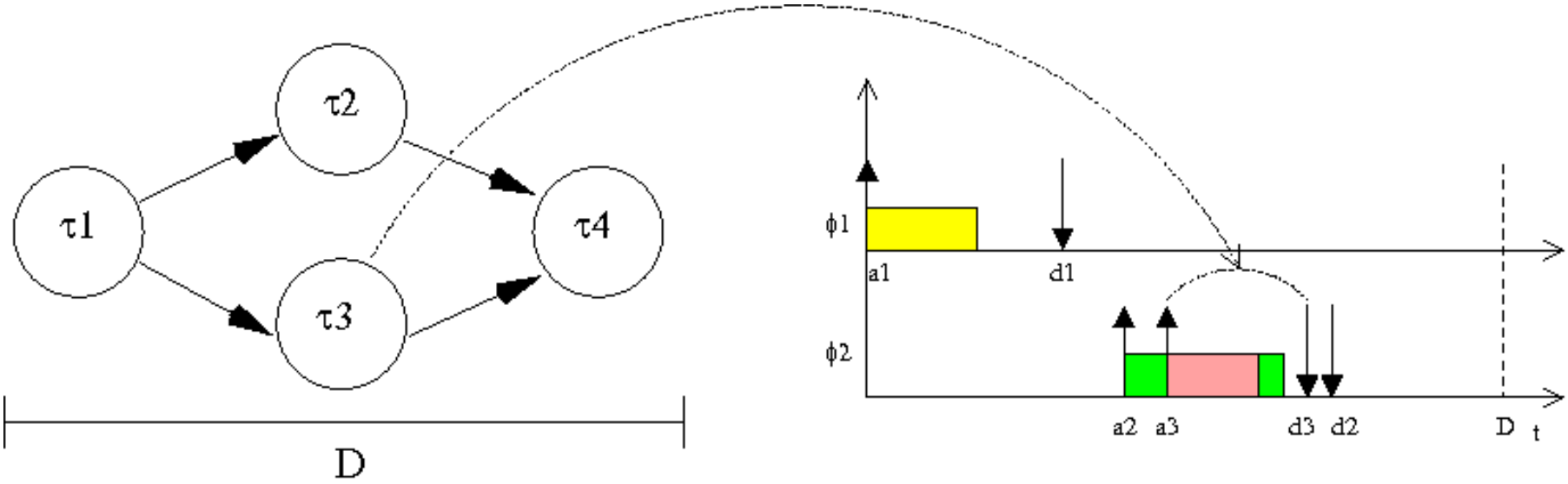
Deadline and activation assignment step (assuming EDF):



Motivational example

Given the allocation: τ_1, τ_4 on ϕ_1, τ_2, τ_3 on ϕ_2 and ϕ_3 empty

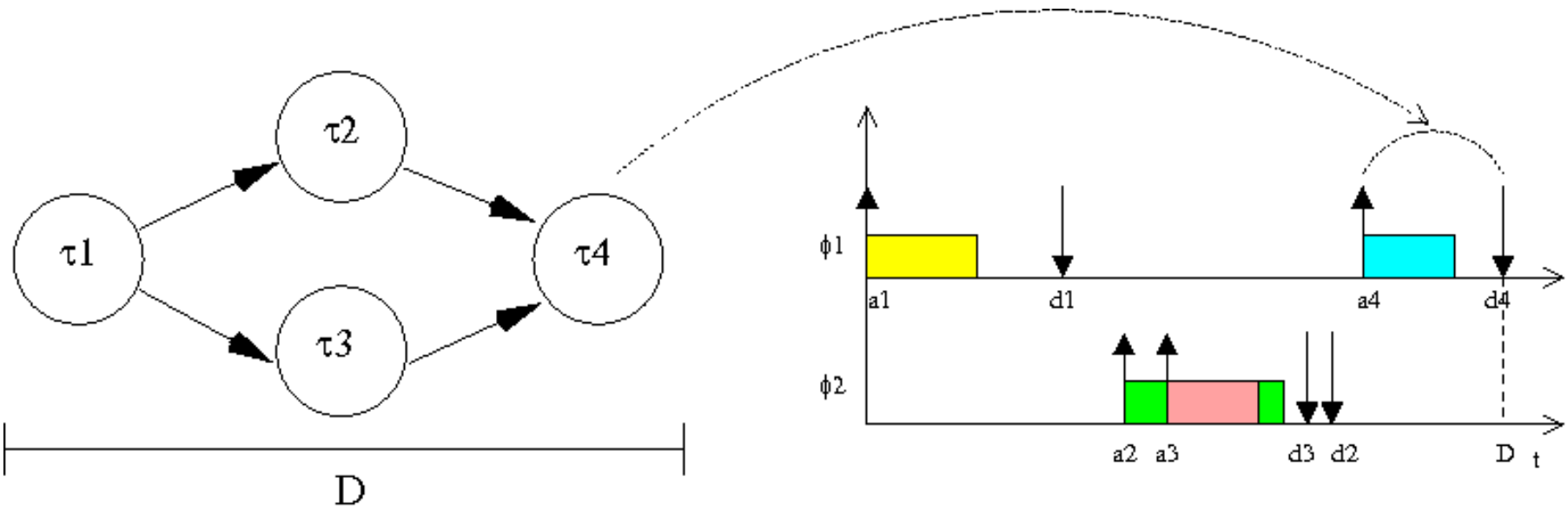
Deadline and activation assignment step (assuming EDF):



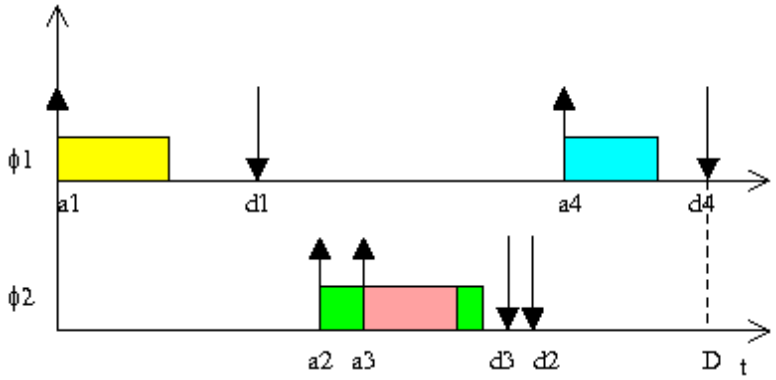
Motivational example

Given the allocation: τ_1, τ_4 on ϕ_1, τ_2, τ_3 on ϕ_2 and ϕ_3 empty

Deadline and activation assignment step (assuming EDF):

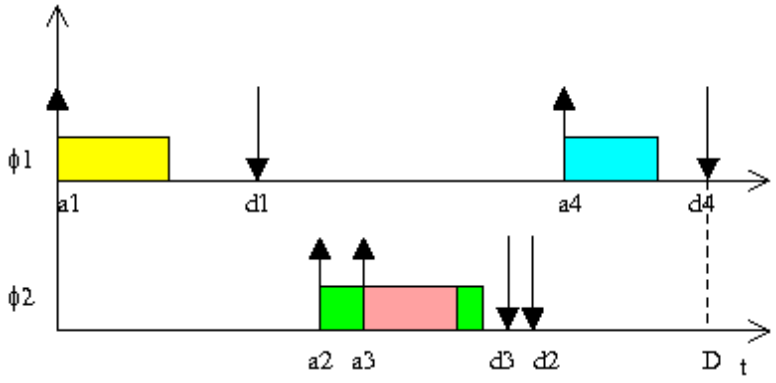


Motivational example

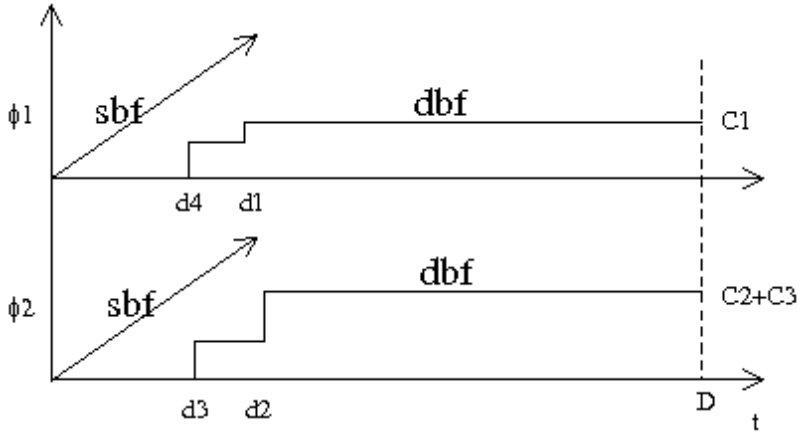


Given the arrivals and deadline assignment shown in the image

Motivational example

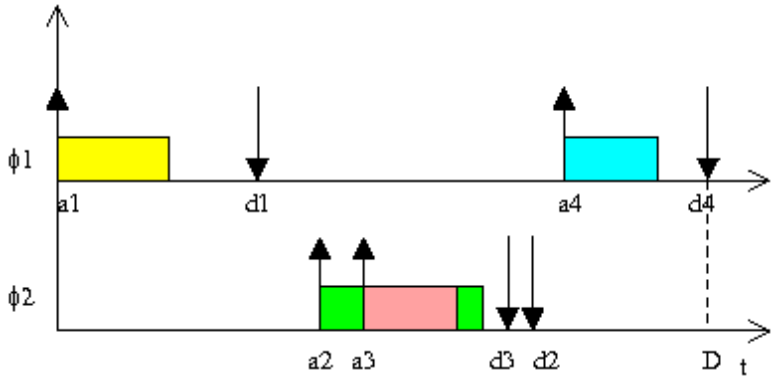


Given the arrivals and deadline assignment shown in the image

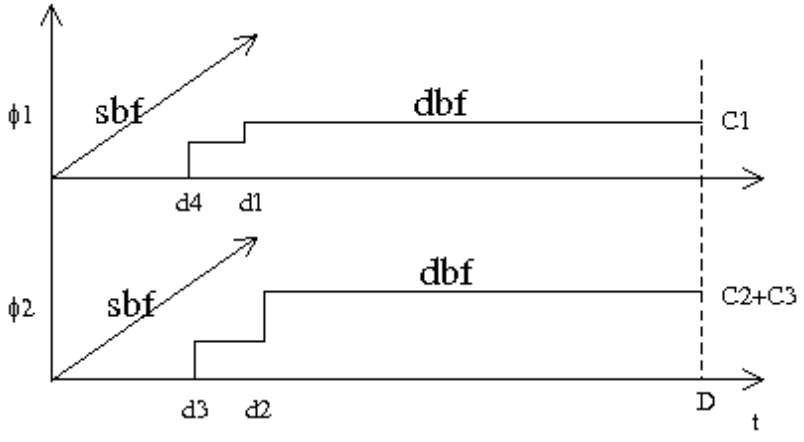


Task set feasibility can be computed by the Processor Demand Criterion [7] or the offset analysis [8]

Motivational example



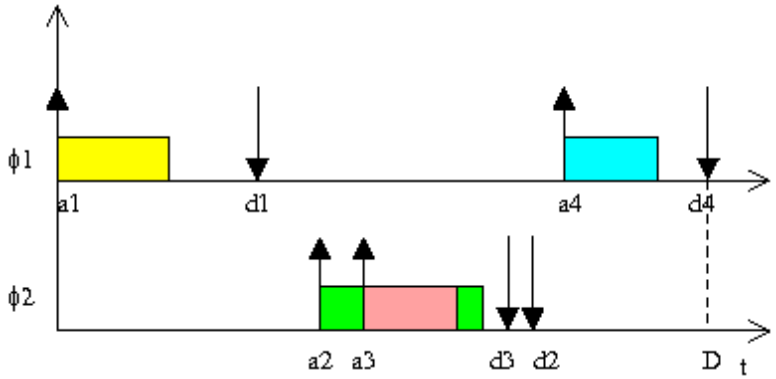
Given the arrivals and deadline assignment shown in the image



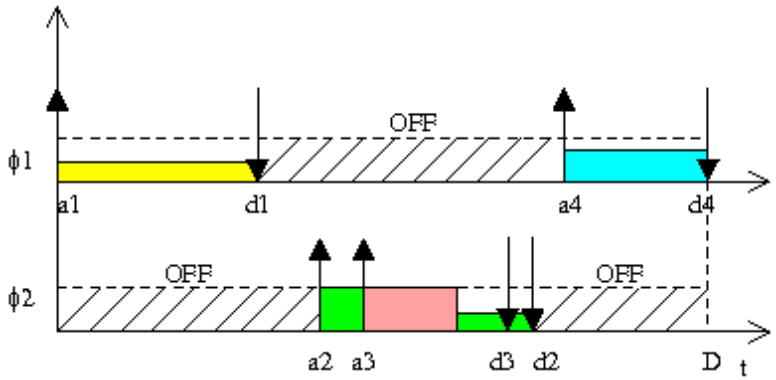
Task set feasibility can be computed by the Processor Demand Criterion [7] or the offset analysis [8]

Polynomial complexity: hyper period == the application period

Motivational example



Given the arrivals and deadline assignment shown in the image



Moreover, knowing tasks' arrival and deadline can help us to further minimize the cost function
Es.: energy consumption

Final remarks

Transforming the precedence constraints into real-time constraints

The scheduler automatically guarantee the precedence constraints.

All the analysis for uni-processor systems can be exploited

Final remarks

Transforming the precedence constraints into real-time constraints

The scheduler automatically guarantee the precedence constraints.

All the analysis for uni-processor systems can be exploited

How can we do the assignment? Would it be more convenient than a static schedule?

thank you

m.bambagini@sssup.it