Ada-Spain'2013

Incorporación del DFP al lenguaje Ada

Mario Aldea[1], Alan Burns[2], Marina Gutiérrez[1] and Michael González[1]

[1]Universidad de Cantabria
{aldeam, gutierrezlm, mgh}@unican.es

[2]University of York
alan.burns@york.ac.uk

Abril, 2013
Madrid

Introduction

Earliest Deadline First (EDF) introduced in Ada 2005

- **Stack Resource Policy (SRP) was chosen as the protocol for resource sharing**

✗ **SRP is complex**

- ✗ **Mistake in the original definition of SRP in the Ada RM**
- ✗ **Erroneous initial implementation in MaRTE OS**

Recently a new protocol has been proposed for EDF

- **Deadline Floor inheritance Protocol (DFP)**

✓ **Simpler and more efficient than SRP**

We propose to include DFP in the Ada standard

SRP: Overview

Generalization of the "Immediate Ceiling Priority Protocol" (ICPP) used with `FIFO_Within_Priorities`

- **each task has a "preemption level" (PL)**
- **each resource has a "ceiling preemption level"**
  - **maximum PL of any task that calls the PO**
- **priority is used in the role of preemption levels**

The optimal way to assign PLs is deadline monotonic

- **the shorter the relative deadline, the higher the PL**

SRP has the same good properties than ICPP:

- ✓ **Minimizes priority inversion. Ensures the mutual exclusion (no lock required). A task can only be blocked once and at the very beginning of its execution. No Deadlocks.**

DFP: Overview

In an EDF-scheduled system, DFP is structurally equivalent to ICPP in a system scheduled under fixed priorities

Each resource has a "deadline floor"

- **The shortest relative deadline of any task that uses it**

Key rule: the absolute deadline of a task could be temporarily shortened while accessing a resource:

$$d = \min\{d, t+D\}$$

d: absolute deadline of the task
D: "deadline floor" of the resource
t: access time to the resource

DFP has all the key properties of SRP

- **the same worst-case blocking in both protocols**

DFP does not add any new rule to the EDF scheduling

- **much simpler to implement**

EDF and SRP in Ada

```
package Ada.Dispatching.EDF is
   subtype Deadline is Ada.Real_Time.Time;

   Default_Deadline : constant Deadline :=
                                        Ada.Real_Time.Time_Last;

   procedure Set_Deadline (
        D : in Deadline;
        T : in Ada.Task_Identification.Task_Id :=
                        Ada.Task_Identification.Current_Task);

   procedure Delay_Until_And_Set_Deadline (
        Delay_Until_Time : in Ada.Real_Time.Time;
        Deadline_Offset : in Ada.Real_Time.Time_Span);

   function Get_Deadline (
        T : Ada.Task_Identification.Task_Id :=
                           Ada.Task_Identification.Current_Task)
        return Deadline;

end Ada.Dispatching.EDF;
```

Integration in the fixed priorities Ada model

EDF works in a range of priority levels

```
pragma Priority_Specific_Dispatching
                    (EDF_Across_Priorities, 10, 30);
```

- or may cover the hole range of system priorities

```
pragma Task_Dispatching_Policy (EDF_Across_Priorities);
```

Preemption levels of tasks and protected objects

- are mapped to priorities in the EDF priority range

```
task T with Priority => 20;
-- if priority 20 is in an EDF range it represents
-- the preemption level of the task

protected Object with Priority => 24 is ...
-- if priority 24 is in an EDF range it represents
-- the preemption level of the protected object
```

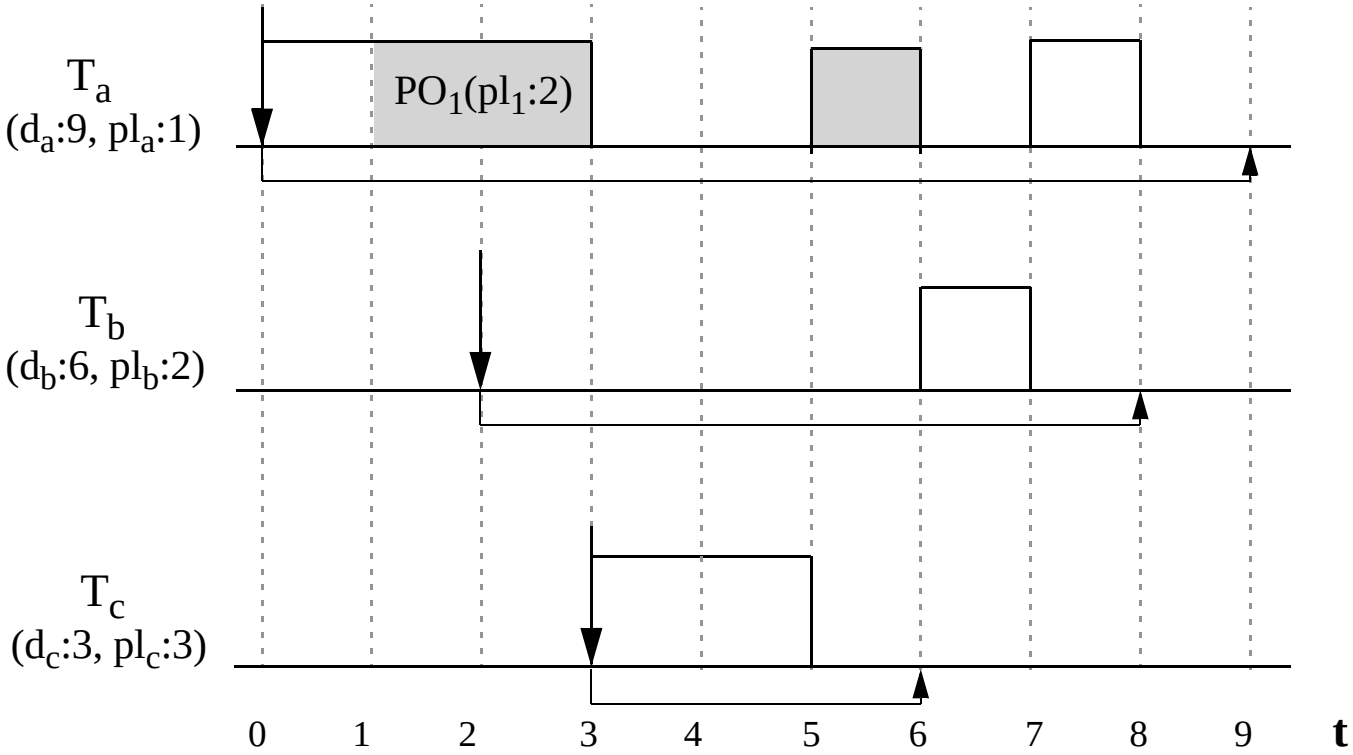
- ✗ Drawback: limited number of distinct preemption levels

Priority inheritance for EDF tasks

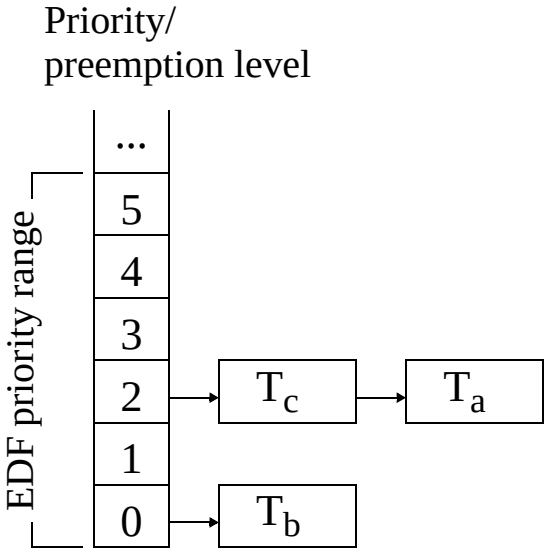
The sources of priority inheritance are redefined:

- **By default, the active priority is the lowest priority in the EDF priority range**
- **When an task executes a protected operation it will inherit the priority (preemption level) of the protected object**
- **A third source of priority inheritance is defined:**
  - **"*the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks; and furthermore T has an earlier deadline than all other tasks on ready queues with priorities in the given EDF_Across_Priorities range that are strictly less than P*"**

Example of EDF scheduling



(a) Task's schedule



(b) Ready queue at t=4

Algorithm to place a task in the ready queue

```
procedure Add_To_Ready_Queue (T : Task) is
begin
   Prio_Max := Lower_Prio_In_EDF_Range;

   for Prio in Lower_Prio_In_EDF_Range+1 ..
                                          T.Preempion_Level-1 loop
      if not Queue(Prio).Empty then
         if T.Deadline < Queue(Prio).Head.Deadline then
            Prio_Max := Prio;
         else
            exit;
         end if;
      end if;
   end loop;

   if Prio_Max = Lower_Prio_In_EDF_Range then
      Queue(Lower_Prio_In_EDF_Range).Add_In_Deadline_Order (T);
   else
      Queue(Prio_Max).Add_Head(T);
   end if;
end Add_To_Ready_Queue;
```

DFP implementation in MaRTE OS

Quite straightforward

- **DFP does not impose additional scheduling rules to EDF**

Only one DFP specific parameter

- **the deadline floor of the mutexes**

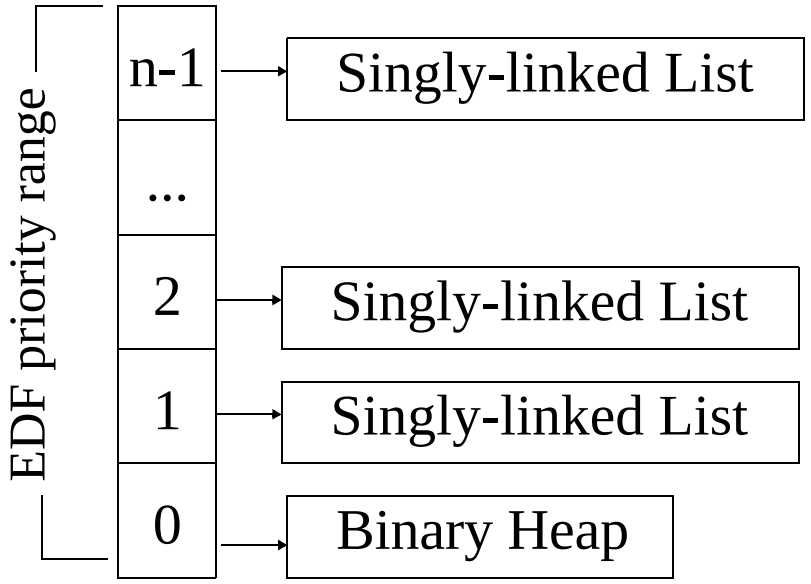
Implementation simplified: only nested critical sections are supported

- **according to Ada semantics for POs**
- **with this limitation a mutex only require to store the original deadline of the task that is holding it**

Comparative analysis

Data structure required for the ready queue

(a) Ready queue for EDF&DFP



(b) Ready queue for EDF&SRP

- **The data structure required by DFP is much simpler**

Lock mutex

DFP

```
procedure Task_Locks_Mutex (Task, Mutex) is
   ...
   Mutex.Owner_Deadline := Task.Deadline;
   Heir_Deadline := Clock + Mutex.Deadlinefloor;
   if Task.Deadline > Heir_Deadline then
      Task.Deadline := Heir_Deadline;
   end if;
   ...
end Running_Task_Locks_Mutex;
```

SRP

```
procedure Task_Locks_Mutex (Mutex) is
   ...
   Task.Num_Mutex_Owned ++;
   Mutex.Owner_Preemption_Level := Task.Preemption_Level;
   if Task.Preemption_Level < Mutex.Preemption_Level then
      Task.Preemption_Level := Mutex.Preemption_Level;
      Reorder (Task);
   end if;
   ...
end Task_Locks_Mutex;
```

Unlock mutex

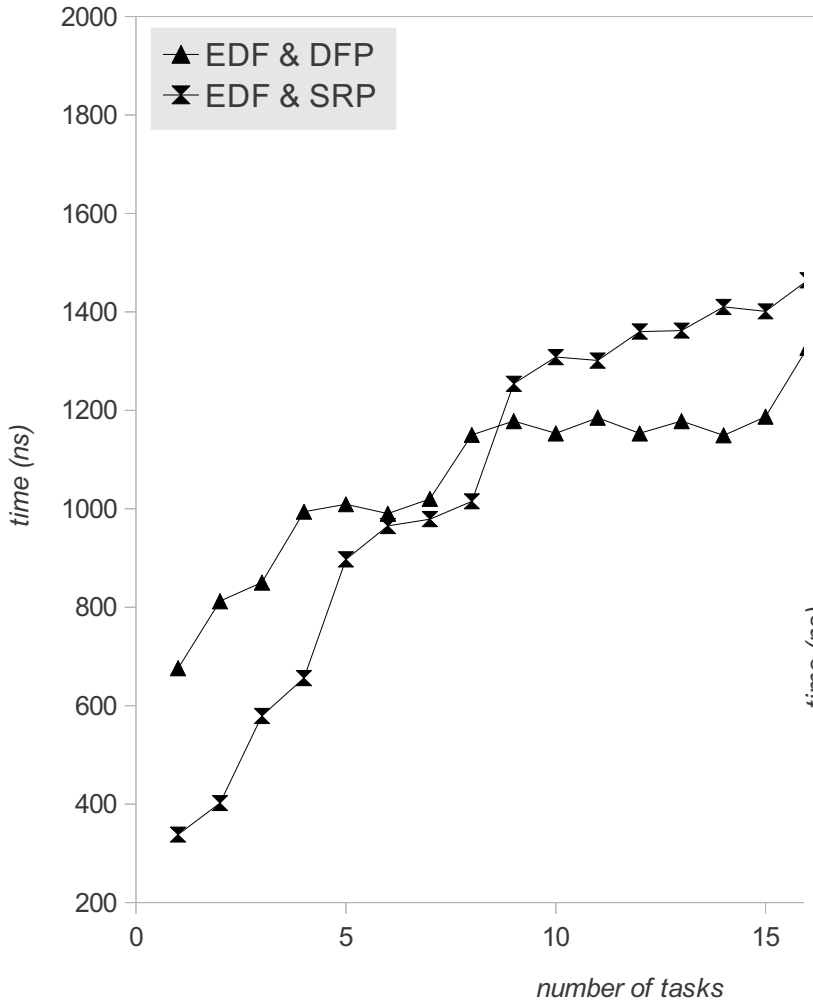
DFP

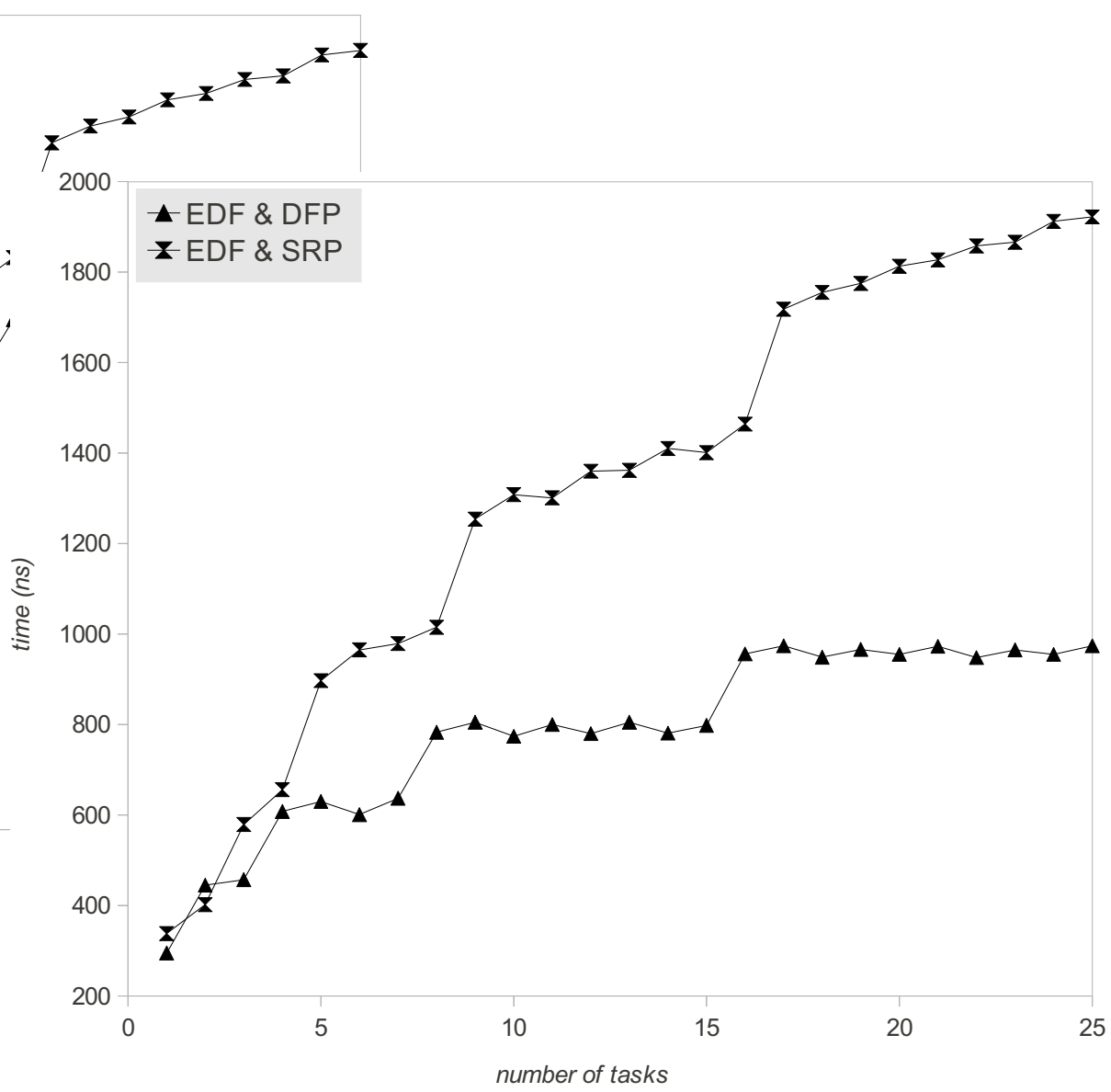
```
procedure Task_Unlocks_Mutex (Task, Mutex) is
   ...
   Task.Deadline := Mutex.Owner_Deadline;
   Reorder_and_Dispatch (Task);
   ...
end Task_Unlocks_Mutex;
```

SRP

```
procedure Task_Unlocks_Mutex (Task, Mutex) is
   ...
   Task.Num_Mutex_Owned --;
   Task.Preemption_Level :=
                Mutex.Owner_Preemption_Level;
   Reorder_and_Dispatch (Task);
   ...
end Task_Unlocks_Mutex;
```

Performance





Alternatives to include DFP in Ada RM

How to deal with the current definition of the EDF:

1. **Redefine `EDF_Across_Priorities` to use DFP instead of SRP**
2. **Add new policy `EDF_With_Deadline_Floor`**
   - **and keep `EDF_Across_Priorities` in the standard but declare it obsolescent**
3. **Add new policy `EDF_With_Deadline_Floor`**
   - **and keep `EDF_Across_Priorities` in the standard**
   - **An implementation could chose to implement both, one or none of these two dispatching policies**

Dispatching pragmas

```
pragma Task_Dispatching_Policy
                              (EDF_With_Deadline_Floor);


pragma Priority_Specific_Dispatching
    (EDF_With_Deadline_Floor, first_prio,last_prio);
```

It could be allowed ranges of any extension

- **for analogy with the other dispatching policies**
- **RM should state that the particular priority value of the tasks and protected objects inside such range will be ignored**

`Ceiling_Locking` policy

```
pragma Locking_Policy (Ceiling_Locking);
-- Ceiling_Locking is applied to the hole partition
```

Is the identifier name appropriate?

- **No: DFP is going to be used in `EDF_With_Deadline_Floor` priority ranges**
- **Yes: DFP is the application of the "Ceiling Locking" concept to the EDF scheduling**

Deep modifications required in the definition of the ceiling locking policy (RM D.3)

- **currently it is defined in terms of priorities**
- **it would be necessary to define it also in terms of deadlines**

Initial deadline floor of a PO

New aspect required:

```
protected Object with
  Deadline_Floor => Ada.Real_Time.Milliseconds (24)
is ...
```

Options for the name of the aspect:

- **`Deadline_Floor`: descriptive**
- **`Deadline`: in analogy to the priority ceiling of the POs named `Priority` instead of `Priority_Ceiling`**

Dynamic change of the deadline floor of a PO

New attribute `Deadline`

- **it would behave very much like the `Priority` attribute**

```
protected body PO is

   procedure Change_Relative_Deadline
                          (D: in Real_Time.Time_Span) is
   begin
      ... -- PO'Deadline has old value here
      PO'Deadline := D;
      ... -- PO'Deadline has new value here
   end Change_Relative_Deadline; -- relative deadline
                                 -- is changed here

   ...
end PO;
```

Dynamic change of the relative deadline of a task

Required for mode changes systems

```
with Ada.Real_Time;
with Ada.Task_Identification;

package Ada.Dispatching.EDF.Dynamic_Relative_Deadlines is

   procedure Set_Relative_Deadline
     (D : in Real_Time.Time_Span;
      T : in Ada.Task_Identification.Task_Id :=
                        Ada.Task_Identification.Current_Task);

   function Get_Relative_Deadline
     (T : Ada.Task_Identification.Task_Id :=
                       Ada.Task_Identification.Current_Task)
     return Real_Time.Time_Span;

end Ada.Dispatching.EDF.Dynamic_Relative_Deadlines;
```

Relative Deadline

```
task EDF_Task with Relative_Deadline => Time;
```

Currently is not considered a property of the task at the same importance level than the priority

- **the relative deadline is only used to assign the first absolute deadline of a task after its activation**

With DFP the relative deadline should be a property of the task

- **used to detect floor violations when accessing POs**

New relative deadline checks required?

When the absolute deadline changes

- **with `Set_Deadline/Delay_Until_And_Set_Deadline`**
- **the new absolute deadline must verify:**

```
abs_deadline > now + rel_deadline - clock_jitter
```

Better to assign atomically absolute and relative deadlines?

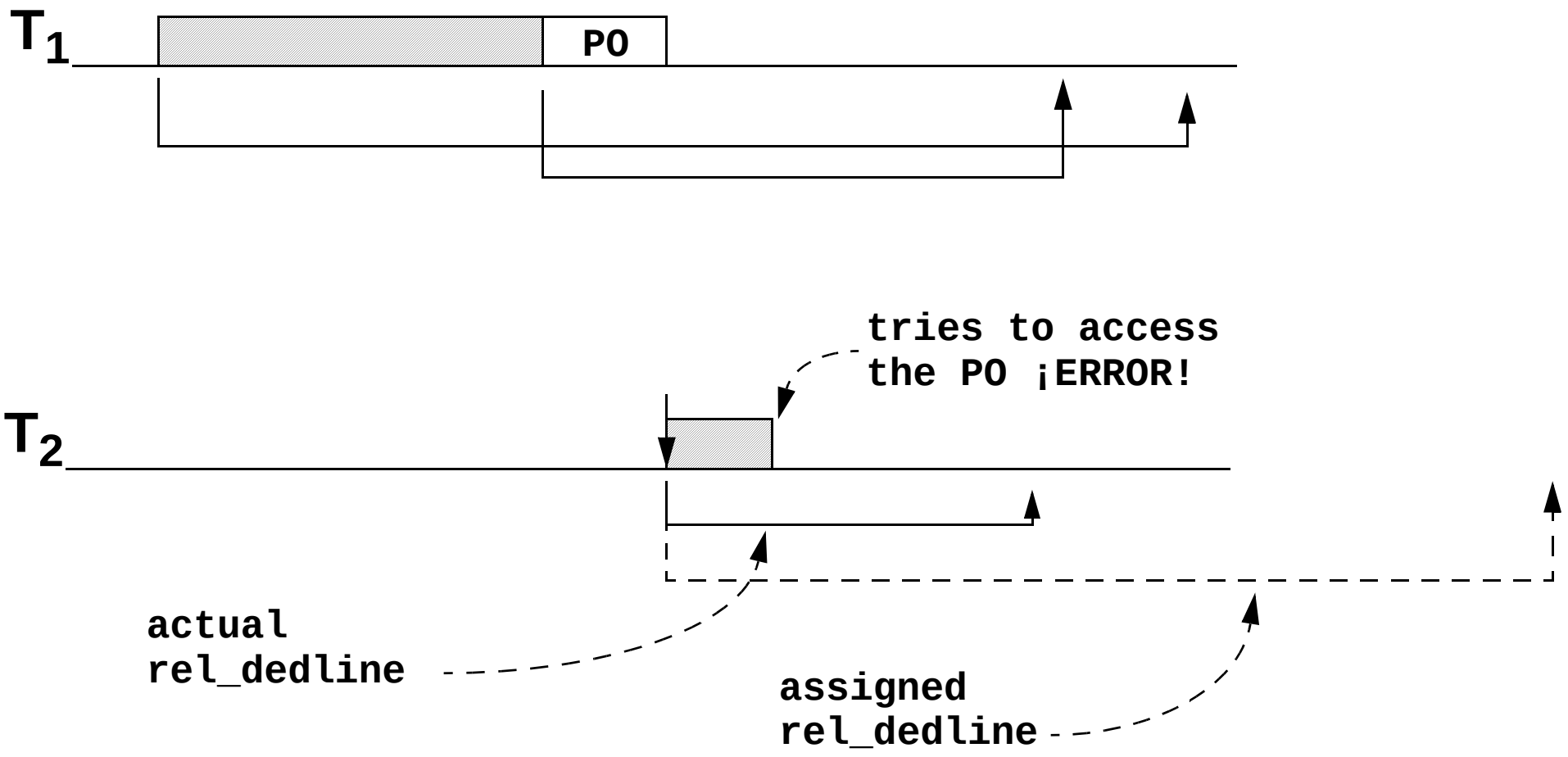
```
Delay_Until_And_Set_Deadline (Next, Rel_Deadline);
```

```
Set_Deadline (abs_deadline);
```

```
rel_deadline := abs_deadline - activation_time
                                        ?
```

Problem to avoid

Task activates with shorter relative deadline than the one it has assigned



Integration of DFP in the current Ada dispatching model

Two main issues:

- **Backwards compatibility: current EDF applications should need minor changes to run with EDF+DFP**
- **Interaction with the other dispatching policies: tasks with different policies interact using protected objects**
  - **Effects of this interaction should be well determined and not lead to priority inversions nor deadlocks**

Backwards compatibility

Current EDF applications can execute with no changes for the proposed EDF&DFP policy

- **only required to change the policy identifier in pragma `Task_Dispatching_Policy` or `Priority_Specific_Dispatching`**

Assigning deadline floors to the POs would be desirable but not required

- **default deadline for a PO would be `Time_Span_Zero`**
- **assigning deadline floors to POs will improve system schedulability**

Interaction with other dispatching policies

`EDF_With_Deadline_Floor` > `FIFO_Within_Priorities`

- **no problem if deadline of the FIFO tasks is infinite (`Ada.Real_Time.Time_Span_Last`)**

`FIFO_Within_Priorities` > `EDF_With_Deadline_Floor`

- **Standard `Ceiling_Locking` rules are applied**

`EDF_With_Deadline_Floor` > `EDF_With_Deadline_Floor`

- **Deadline floors of POs should consider the deadlines of all the tasks that access them in both priority ranges**

Conclusions

- The DFP is an alternative to the SRP
  - with the same key properties
- DFP is simpler to understand, describe, and implement
  - DFP has better performance than SRP
- DFP can be integrated in the current Ada dispatching model
  - Different alternatives for including the protocol in Ada RM has been presented
- DFP standardization should be addressed in a future version of the language (Ada 2020?)
  - IRTAW gave its preliminary approval