UNIVERSIDAD DE CANTABRIA

A POSIX-Ada Interface for Application-Defined Scheduling

By: Mario Aldea Rivas (aldeam@unican.es)
Michael González Harbour (mgh@unican.es)

Ada-Europe'2002
Vienna, Austria, June 17-21, 2002

Table of Contents

UNIVERSIDAD DE CANTABRIA

1. Overview

Interface for Application-Defined Scheduling

Integrated into the POSIX-Ada bindings

- **POSIX: Portable Operating System Interface standard**

Not only an interface

- **Requires support in the operating system kernel**
- **Implemented in our operating system MaRTE OS**

Main points

- **Application-defined scheduling policies implemented by a special kind of tasks (*scheduler tasks*)**
- **Scheduler tasks can activate or suspend other tasks**

2. Motivation

UNIVERSIDAD DE CANTABRIA

The scheduling policies defined in Ada and POSIX are not sufficient for all application environments

- **Dynamic priorities allow better use of resources**
- **Dynamic systems require flexible scheduling schemes (multimedia)**

There are many policies based on dynamic priorities

- **It is not possible to standardize them all**

Our proposal:

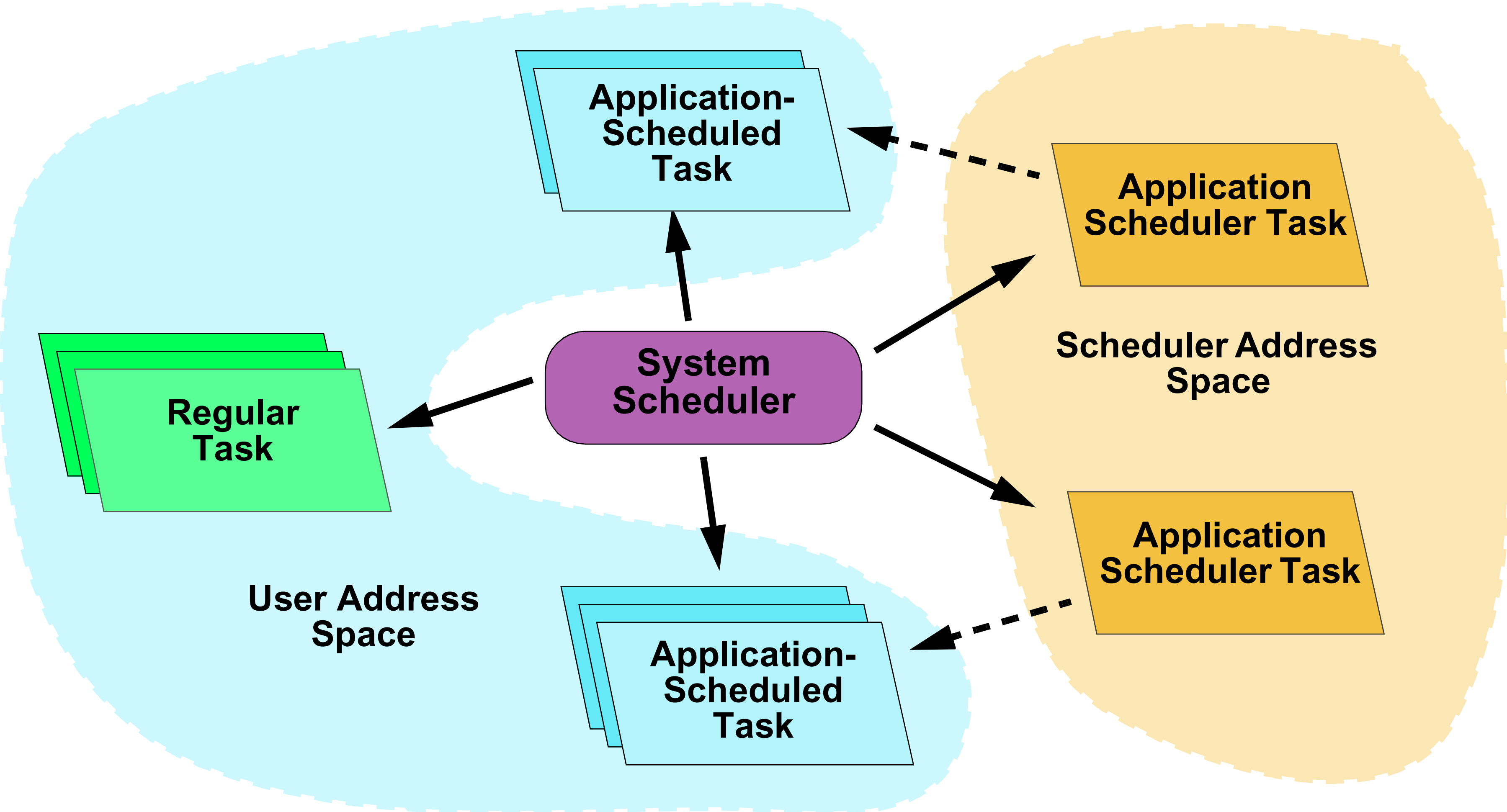
- **Allow applications to define their own scheduling policies and synchronization protocols**

3. Some Requirements

UNIVERSIDAD DE CANTABRIA

- Allow a large variety of scheduling policies
- Compatibility with current scheduling policies in POSIX and Ada
- Coexistence of several scheduling algorithms
- Isolation of critical tasks from behavior of application schedulers
- Integration of synchronization protocols to avoid priority inversion or similar effects

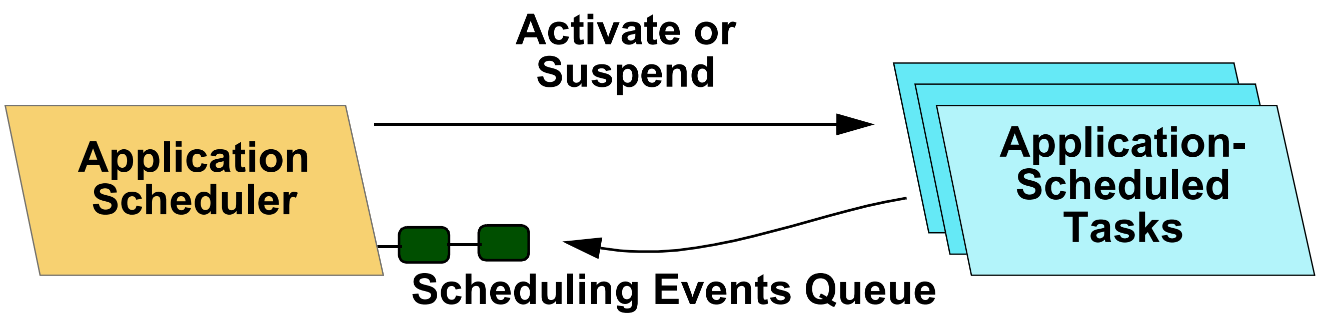
4. Model Description



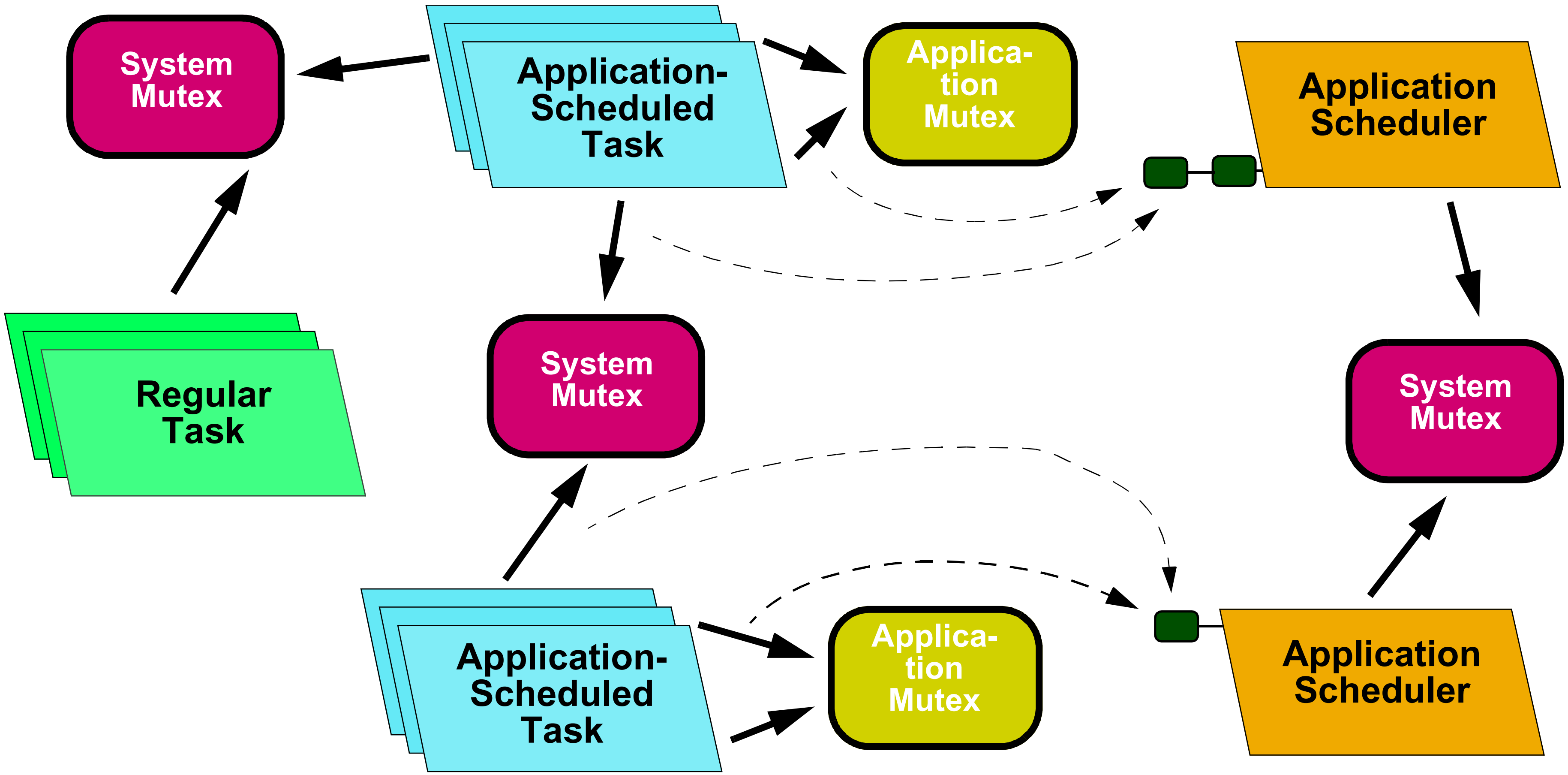
Model Description (Cont'd)

UNIVERSIDAD DE CANTABRIA

- Scheduler tasks are always invoked before their scheduled tasks
  - They inherit all the priorities inherited by them
  - Schedulers have precedence if same priority
- Schedulers can activate or suspend any of their tasks, and must accept (or reject) them at creation
- They are notified about events related to their scheduled tasks



Model Description (Cont'd)



5. Scheduling Events

UNIVERSIDAD DE CANTABRIA

They represent situations relevant to the application scheduler

They contain the following information

- Type of event
- Task that caused the event
- Event-dependent information:
  - Inherited priority
  - Involved application mutex
  - Signal received
  - Application-specific information

Events may be filtered by type

Scheduling Events (Cont'd)

UNIVERSIDAD DE CANTABRIA

Scheduling event type	Additional information
NEW_TASK	none
TERMINATE_TASK	none
READY	none
BLOCK	none
YIELD	none
SIGNAL	POSIX signal information
CHANGE_SCHED_PARAM	none
EXPLICIT_CALL	Application message
TIMEOUT	none

Scheduling Events (Cont'd)

UNIVERSIDAD DE CANTABRIA

Scheduling event type	Additional information
PRIORITY_INHERIT	Inherited system priority
PRIORITY_UNINHERIT	Uninherited system priority
INIT_MUTEX	Pointer to the app. mutex
DESTROY_MUTEX	Pointer to the app. mutex
LOCK_MUTEX	Pointer to the app. mutex
TRYLOCK_MUTEX	Pointer to the app. mutex
UNLOCK_MUTEX	Pointer to the app. mutex
BLOCK_AT_MUTEX	Pointer to the app. mutex
CHANGE_MUTEX_SCHED_PARAM	Pointer to the app. mutex

6. Scheduling Actions

Schedulers have an operation (Execute_Actions) to request execution of multiple actions, including:

- accept or reject a task that has requested attachment
- activate or suspend a task
- accept or reject initialization of an application mutex
- grant the lock on an application mutex

As part of Execute_Actions, the scheduler suspends until:

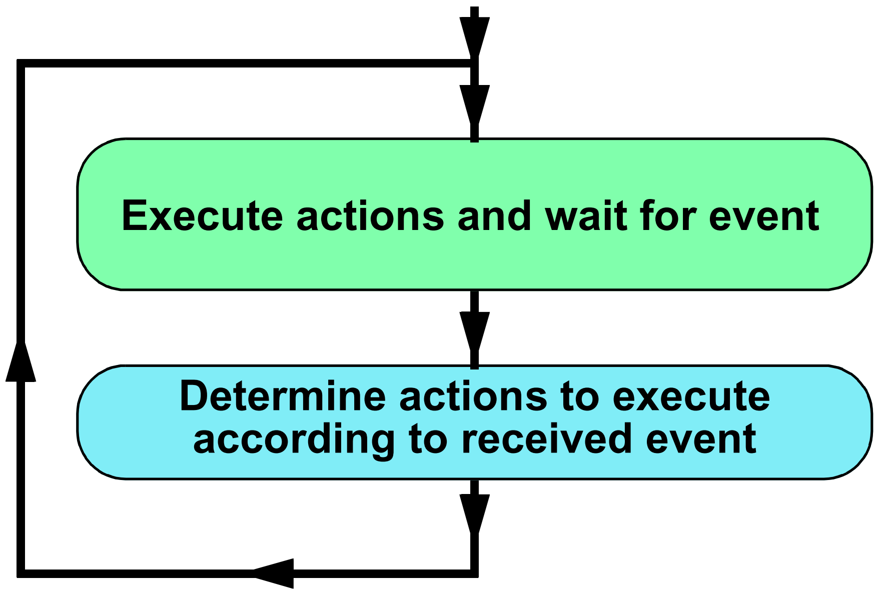
- the next scheduling event,
- arrival of a POSIX signal (usually indicating expiration of regular or CPU-time timers)
- or the expiration of a timeout

Execute Actions

UNIVERSIDAD DE CANTABRIA

```
procedure Execute_Actions
   (Sched_Actions : in  Scheduling_Actions;
    Set           : in  POSIX_Signals.Signal_Set;
    Event         : out Scheduling_Event;
    Timeout       : in  POSIX.Timespec;
    Current_Time  : out POSIX.Timespec)
    -- other overloaded versions exist
```

Structure of a scheduler:



7. Example of an Application-Defined Policy: EDF

UNIVERSIDAD DE CANTABRIA

Application scheduled task:

```
with POSIX_Application_Scheduling;
package AppSched renames POSIX_Application_Scheduling;
...
package EDF_Policy is new
   AppSched.Application_Defined_Policy (EDF_Parameters);
...

task body Periodic_EDF_Task is
   Param : EDF_Parameters := (Deadline, Period);
begin
   EDF_Policy.Change_Task_Policy (EDF_Scheduler_Task_Id, Param);
   loop
      -- do useful work
      ...;
      -- task will wait until the next period
      AppSched.Invoke_Scheduler;
   end loop;
end Periodic_EDF_Task;
```

Example of an Application-Defined Policy: EDF (Cont'd)

UNIVERSIDAD DE CANTABRIA

Application Scheduler task:

```
task body EDF_Scheduler is
   ...;
   Most_Urgent_Task     : Task_Id;
   Now, Earliest_Start : POSIX.Timespec;
   Event                : AppSched.Scheduling_Event;
   Sched_Actions        : AppSched.Scheduling_Actions;

begin
   AppSched.Become_An_Application_Scheduler;

   AppSched.Set_Clock(Clock_Realtime);
   AppSched.Set_Flags(Absolute_Timeout);
   Now := POSIX_Timers.Get_Time(Clock_Realtime);

   -- Scheduling events processing loop
   loop
      Schedule_Next(Most_Urgent_Task, Earliest_Start);
```

Example of an Application-Defined policy: EDF (Cont'd)

```
if Most_Urgent_Task /= Null_Task_Id then
   AppSched.Add_Activate (Sched_Actions, Most_Urgent_Task);
end case;
AppSched.Execute_Actions_with_Timeout
           (Sched_Actions, Event, Earliest_Start);

-- process scheduling events
case AppSched.Get_Event_Code(Event) is

 when New_Task =>
   if Schedulability_Test (AppSched.Get_Task(Event)) then
     AppSched.Add_Accept (Sched_Actions,
                          AppSched.Get_Task(Event));
     Add_To_List_of_Tasks (AppSched.Get_Task(Event), Now);
   else
     AppSched.Add_Reject (Sched_Actions,
                          AppSched.Get_Task(Event));
   end if;

 when Terminate_Task =>
   Eliminate_From_List_of_Tasks (AppSched.Get_Task(Event));
```

Example of an Application-Defined policy: EDF (Cont'd)

UNIVERSIDAD DE CANTABRIA

```
      when Ready =>
        Make_Active (AppSched.Get_Task (Event));

      when Block =>
        Make_Blocked (AppSched.Get_Task (Event));

      when Explicit_Call =>
        AppSched.Add_Suspend (Sched_Actions,
                              AppSched.Get_Task (Event));
        Make_Timed (AppSched.Get_Task (Event));

      when Timeout =>
        null;

      when Others =>  null;
     end case;

   end loop; -- Scheduling events processing loop

end EDF_Scheduler;
```

MaRTE OS

UNIVERSIDAD DE CANTABRIA

***M**inim**a**l **R**eal-**T**ime OS for **E**mbedded Applications*

- Follows the POSIX.13 Minimum Realtime System Profile
  - Single process
  - No file system
- Adds new POSIX interfaces: CPU-time Timers
- Written in Ada, with few parts in C or assembly language
- Gnat run-time system runs on top of it
- POSIX threads interface for C threads
- Usable both for C and Ada (and mixed) applications
- Free software

http://marte.unican.es

Performance on a 1.1GHz Pentium III

UNIVERSIDAD DE CANTABRIA

Context switches in Ada

Scheduling Algorithm	Context switch (μs)	Overhead for a 1KHz task
Regular Ada delay until statement	5	1%
Round robin within priorities	4.5	0.9%
Constant Bandwidth Server	6	1.2%

Conclusions and Further Work

UNIVERSIDAD DE CANTABRIA

APIs for application scheduling in Ada and C

- Performance is acceptable
- Great flexibility
- Compatible with previous fixed priority scheduling
- Implemented in MaRTE OS

Future work

- POSIX standardization
- Integration into the Ada language
  - will require some language extensions (pragmas, ...), specially for protected objects