Implementation of the Ada 2005 Task Dispatching Model in MaRTE OS and GNAT

Mario Aldea Rivas (`aldeam@unican.es`)

Michael González Harbour (`mgh@unican.es`)

José F. Ruiz (`ruiz@adacore.com`)

XII Jornadas de Tiempo Real
Madrid (Universidad Carlos III)
5 y 6 de febrero de 2009

Contents

- MaRTE OS
  - Current status of MaRTE OS
  - On-going work in MaRTE OS
- Implementation of the Ada 2005 Task Dispatching Model in MaRTE OS and GNAT
  - Ada 2005 dispatching model
  - Implementation of task dispatching
  - Round robin dispatching policy
  - EDF dispatching policy
  - Performance metrics
  - Conclusions

Current status of MaRTE OS

- We go on adding functionality to MaRTE OS...
  - EDF scheduling policy (`SCHED_EDF`)
  - Support for C++ (exceptions, uSTL)
  - “x86”, “Linux” and “Linux Lib” arch. in one installation
  - TLSF 2.3.2
- Communication protocols and middlewares
  - PolyORB (DSA and CORBA) adapted for RT-EP
  - RT-WMP wireless protocol (UNIZAR)
  - RT-EP protocol with a bandwidth reservation layer, distributed mutexes, broadcast services
  - FRESCAN network protocol for CAN bus with bandwidth reservation and sporadic servers
  - DTM, the Distributed Transaction Manager of FRESCOR
  - MyCCM, implements the OMG CCM (Thales)

Current status of MaRTE OS

- Drivers
  - CAN
  - Wireless Ralink RT61
  - Ethernet drivers (SiS900, eepro100 and RTL8139)
  - IDE disk driver (HD and CompactFlash) and FAT 16 filesystem
  - Yenta CardBus
  - Advantech data acquisition and digital IO cards (pcm_3718...)
  - laser-sick-lms200 (UNIZAR)
  - GPS Novatel ProPak (UNIZAR)
  - P2OS (compass, odometer, sonar, motors..) (UNIZAR)
  - I^2C and Compass CMPS03
  - SVGA, BTTV, Soundblaster 16
  - Mouse, Joystick, keyboard, serial port

Current status of MaRTE OS (cont'd)

- Adapted to GNAT GPL 2008
- Will NOT be included in next version of GNAT compiler but...
  - it will be offered at AdaCore's *Libre Site* as an extra runtime
- Implementation of the Annex D of Ada 2005:
  - ✓ Timing Events
  - ✓ Execution Time Clocks and Execution Time Timers
  - ✓ Dynamic Priorities for Protected Objects
  - ✓ Immediate Priority Change
  - ✓ Group Execution Time Budgets
  - ✓ EDF dispatching policy
  - ✓ Round Robin dispatching policy
  - ✓ Priority Specific Dispatching
  - ✗ Non-Preemptive dispatching policy

On-going work in MaRTE OS

- Support for multicore architectures (Daniel Medina)
- Integration of the Distributed Transaction Manager in the PolyORB+FRSH middleware (Hector Pérez)
- Improve integration with GNAT compiler
- Implementation of the SPI protocol (Mónica Puig-Pey)
- Industrial projects:
  - controller for a welding robot
  - GPS based orientation system

Implementation of the Ada 2005 Task Dispatching Model in MaRTE OS and GNAT

Ada 2005 dispatching model

Ada 2005 defines a powerful scheduling model

- policies: FIFO, EDF, Round robin and Non-preemptive
- uses a hierarchical approach (FIFO, EDF and Round robin)
  - Non-preemptive only applicable to the whole partition

Two means to define dispatching policies:

- Global: applied to the whole partition for all priorities

```
pragma Task_Dispatching_Policy (policy);
```

- Local: applied to a priority band

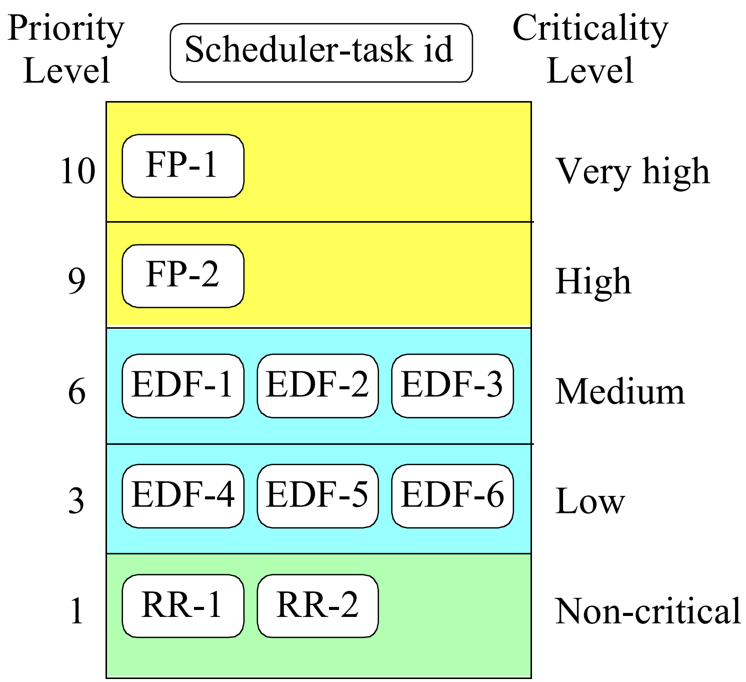
```
pragma Task_Dispatching_Policy
          (policy, first_prio, last_prio);
```

Policy of a task is determined by its base and active priorities

Ada 2005 dispatching model (cont'd)

Combine in the same application the good properties of the different policies

- **FIFO ⇨ predictability**
- **EDF ⇨ better usage of resources**
- **Round robin ⇨ fair distribution of resources**



Implementation of task dispatching

Dispatching pragmas can be placed in any compilation unit

- **The compiler:**
  - **checks pragmas consistency for each individual unit**
  - **adds this information to the *Ada Library Information* (ALI) file**
- **The binder:**
  - **checks partition-wise coherence**
  - **mades available to the run time a structure with pairs priority-policy**

Implementation of task dispatching(cont'd)

The run time has to take care of the policy changes consequence of priority changes

- `Ada.Dynamic_Priorities` → Change of base priority
- Task's activation, rendezvous and protected actions → Change of active priority

Change of base priority, activation and rendezvous are handled by the same procedure

- changes the policy of the OS thread when necessary

Priority inheritance during a protected action relays in the ceiling protocol of the mutex

- GNAT does NOT change the policy of the thread
- but, as we will see, Ada dispatching rules are observed

Round robin dispatching policy

Ada `Round_Robin_Within_Priorities` policy

- each task can execute at most during a “quantum”
- when the quantum is exhausted the task is moved to the tail of its priority queue
- typically used to share spare time among background activities

Our implementation is based on the POSIX `SCHED_RR` policy

- Round robin Ada tasks are directly mapped on `SCHED_RR` threads

Ada and POSIX round robin policies are very similar

- ... but there are some issues to solve

Round robin: Implementation in the run time and the compiler

The main difference between Ada and POSIX round robin policies is the Ada rule for quantum expiration:

- "*When a task has exhausted its budget and is without an inherited priority (and is not executing within a protected operation), it is moved to the tail of the ready queue for its priority level*" (RM D.2.5 14/2)

Quantum expiration during activation and rendezvous is avoided at run time level:

- invoking `marte_disable_rr_quantum()` / `marte_enable_rr_quantum()` at the appropriate points
- non POSIX solution

As we will see, quantum expiration during protected actions is avoided at Operating System level

Round robin: Support at the operating system level

GNAT uses POSIX mutexes to implement protected objects

- task holds a mutex while executing a protected operation

A task shouldn't be re-queued within a protected operation...

- but POSIX is silent about if a thread can be re-queued while holding a mutex

We have modified MaRTE:

- now a thread that has exhausted its quantum is not re-queued while holding a mutex

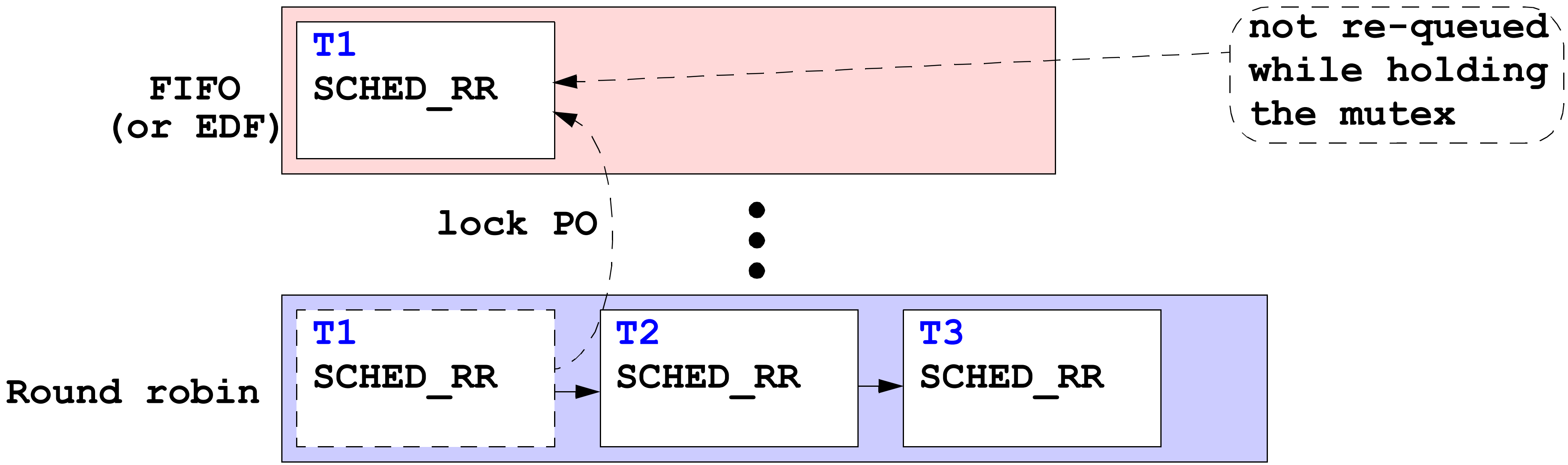
Other POSIX OSs (Linux) allow re-queuing in such situation

- difficult to implement Ada round robin in those OSs

Round robin: (cont'd) Support at the operating system level

Another issue to consider: a task can "jump" to a different policy band when using a PO

- in that situation policy is not changed by the run time
- but task is not re-queued ⇨ behaviour expected for FIFO and EDF task executing a protected operation



Round robin: (cont'd) Support at the operating system level

Final issue: Ada defines an operation that allows setting the quantum for a priority level or a range of levels...

- but in POSIX all the priority levels share the same quantum, chosen by the OS and that can not be changed

The possible contradiction is avoided by the RM rules:

- "*An implementation shall document the quantum values supported*" (RM D.2.5 16/2)
- "*Due to implementation constraints, the quantum value returned by Actual_Quantum might not be identical to that set with Set_Quantum*" (RM D.2.5 18/2)

EDF dispatching policy

New scheduling parameters: deadline and preemption level

Deadline can be set:

- At task creation ⇨ pragma `Relative_Deadline`
- Dynamically ⇨ operations from `Ada.Dispatching.EDF`

Preemption level

- To be used with Protected Objects and Baker's protocol
  - Baker's protocol for EDF is equivalent to `Ceiling_Locking` for `FIFO_Within_Priorities`
- Ada reuses priority as the preemption level for tasks and protected objects

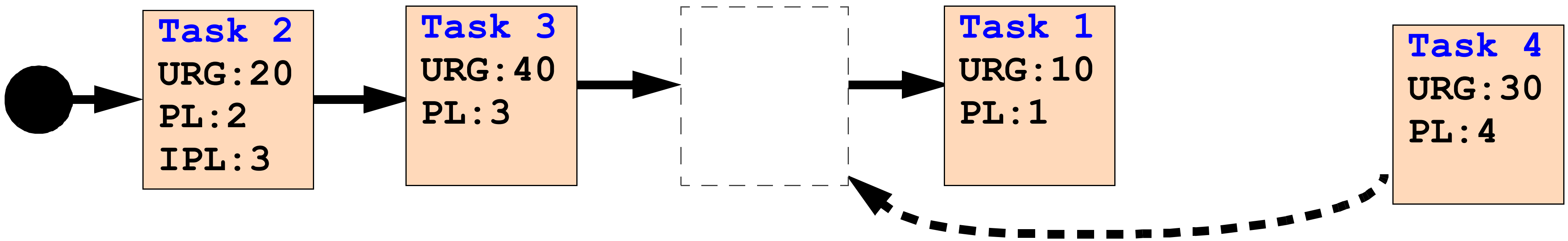
EDF is not included in the POSIX standard

EDF: Support at the operating system level

MaRTE implements EDF as an additional POSIX policy

- Relatively easy to implement because MaRTE already had:
  - Baker's protocol and abstract notion of "urgency"
- EDF threads has an urgency value inversely proportional to its absolute deadline (U α 1/D)
- Implementation in an pure fixed-priority OS can be tough

Urgency and Baker's protocol in MaRTE ready queue:

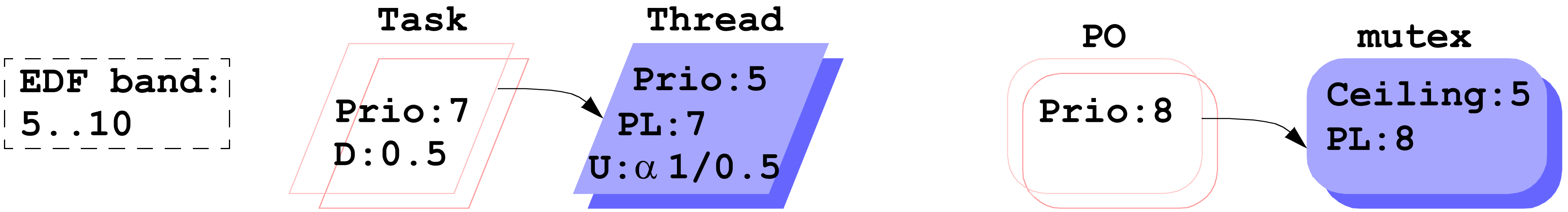


  - Don't suffers from the problem discovered in the Ada definition of EDF

EDF: Support at the operating system level

Mapping of `EDF_Across_Priorities` on `SCHED_EDF`:

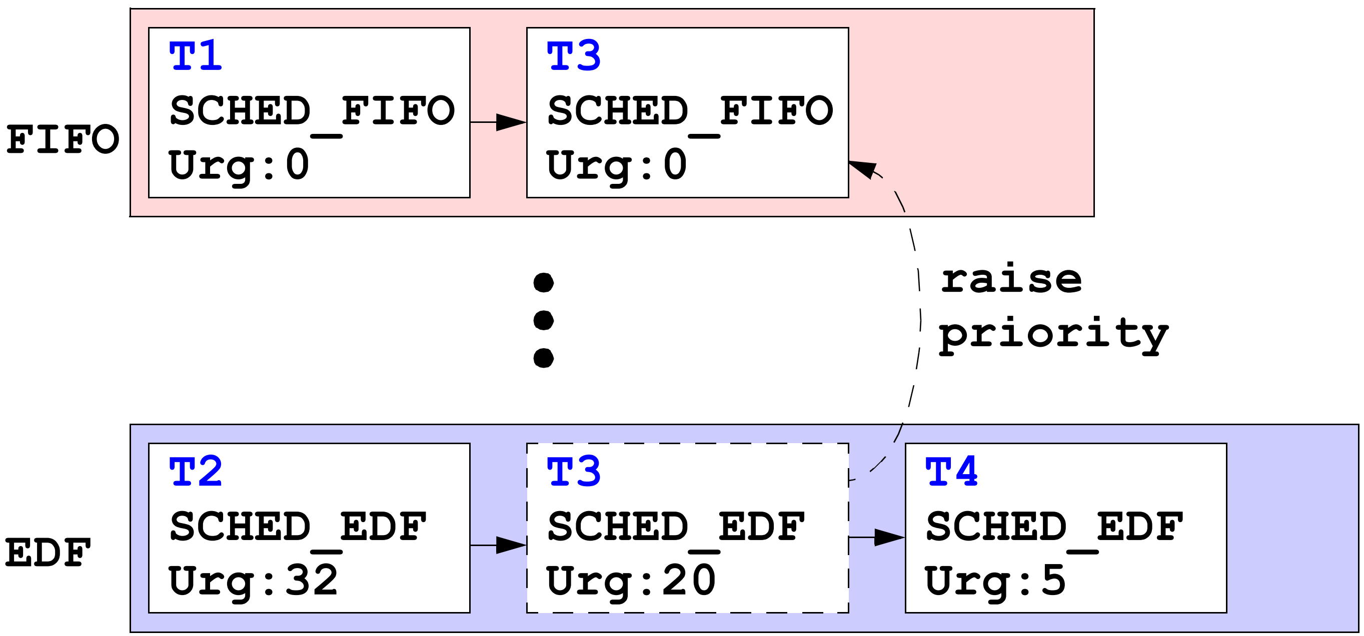
- Absolute deadline (`D`) → urgency (α `1/D`)
- Task and PO priority → preemption level
- Lowest priority in the EDF band → priority/ceiling



EDF: Support at the operating system level

Issue: policy change due to activation, rendezvous or dynamic priority change

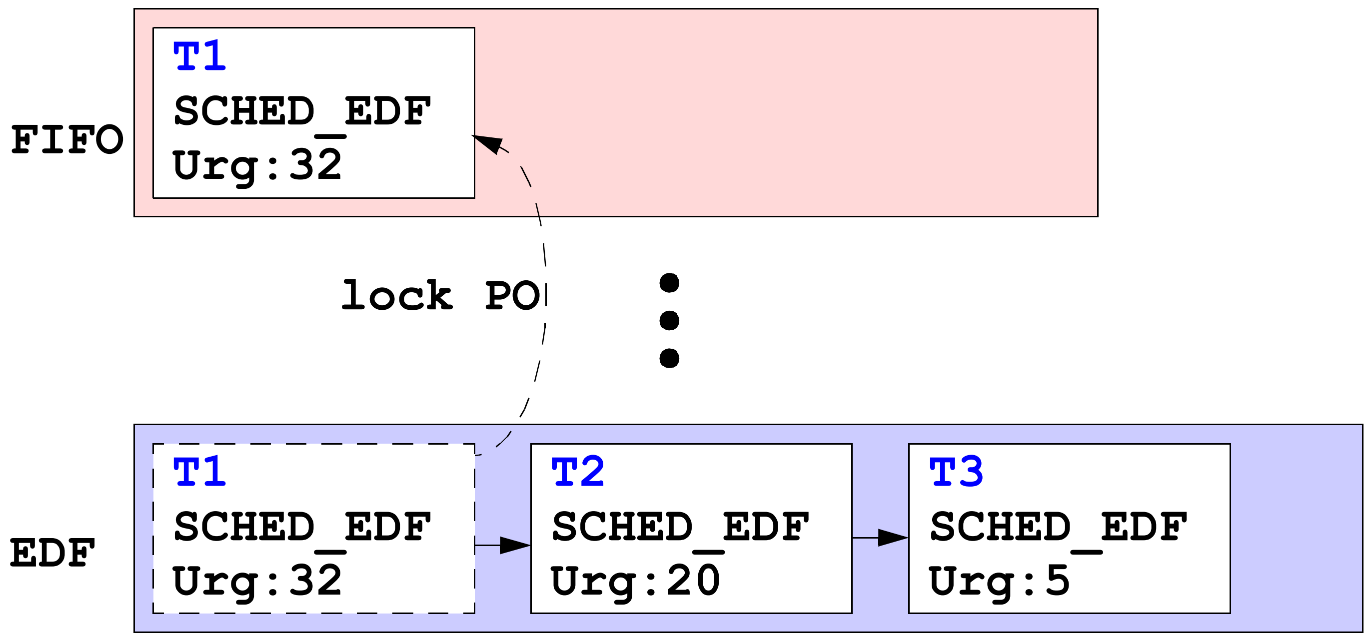
- the run time changes the policy of the thread
- MaRTE sets thread urgency to 0 after a policy change
  - FIFO ordering among non-EDF threads is observed



EDF: Support at the operating system level

Issue: policy change due to protected operation

- run time does NOT change the policy → the EDF thread keeps its urgency
- ... but the FIFO order is observed
  - Ada rules ensure the new ready queue will be empty



EDF: Support at the operating system level

Procedure `EDF.Delay_Until_And_Set_Deadline`

- atomically performs deadline change and suspension
- improve efficiency by avoiding spurious activations
- there is NOT an equivalent in POSIX

Solution adopted in MaRTE:

- function to set the deadline that the thread will have the next time it becomes runnable
- it is a general solution: can be used together with any suspending o timed-blocking function
- GNAT used `pthread_cond_timedwait()` to implement the `delay` and `delay until` instructions

Performance metrics

Description	Fixed priorities	EDF	Round Robin
1. Protected operation with ceiling above base priority	1.67 μs	1.67 μs	
2. Task entry call with mixed EDF and Fixed Priority policies	6.80 μs	3.40 μs	
3. Task entry call with only fixed priorities	9.00 μs		
4. Context switch time with delay until, from task that suspends itself to a lower priority (or longer deadline) task	8.00 μs	9.20 μs	
5. Context switch time with delay until, from running task task to higher priority (or shorter deadline) task that gets active	5.90 μs	6.80 μs	
6. Context switch time caused by end of time slice			10 μs

- **Measurements performed in a Pentium III at 800MHz**

Conclusions

Ada dispatching model is a major advance

- it allows getting the best benefits of different scheduling policies
- but there is a lack of available implementations

MaRTE OS is one of the first implementations

✓ Priority Specific Dispatching
✓ FIFO, EDF and Round robin
× Non-Preemptive dispatching policy

Available (GPL) at `http://marte.unican.es`

- for bare x86 and Linux