

Linux Plumbers 2010

November 3rd, Boston



Scuola Superiore
Sant'Anna

di Studi Universitari e di Perfezionamento

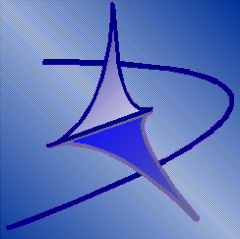
Real-Time API

Tommaso Cucinotta, Dhaval Giani, Dario Faggioli, Fabio Checconi

Real-Time Systems Lab (RETIS)

Center for Excellence in Information, Communication and Perception Engineering
(CEIICP)

Scuola Superiore Sant'Anna, Pisa (Italy)



Recently Proposed Real-Time Scheduler(s)



Features

- **Temporal isolation** among processes
- Applications have to provide reservation parameters (sporadic real-time task model)
 - **runtime** every **period**
- **Deadline-based scheduling**
- **Hierarchical scheduling**
 - Attach **more tasks** as a whole to a **single reservation**

Problems

- I) Suitable **kernel-space / user-space** interface
- II) Suitable **application-level** interface



Recently proposed schedulers and their APIs



EDF RT Throttling (a.k.a., The IRMOS Scheduler)

- Parameters: **runtime, period, cpu mask, tasks**
 - **RT priorities** of real-time tasks
- **cgroup**-based interface
 - Problem of **atomic changes** to scheduling parameters



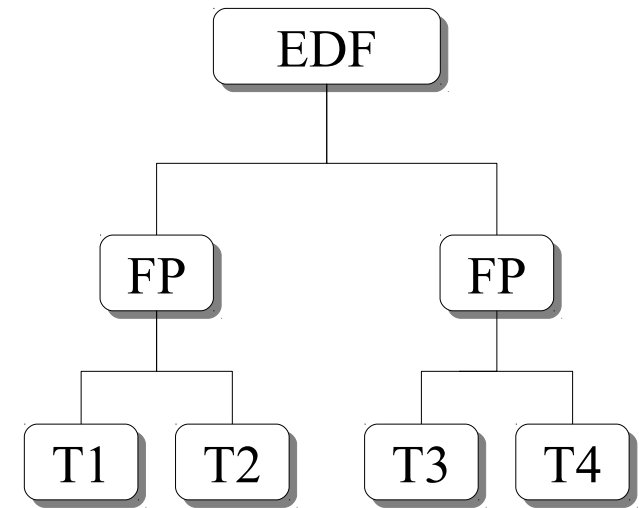
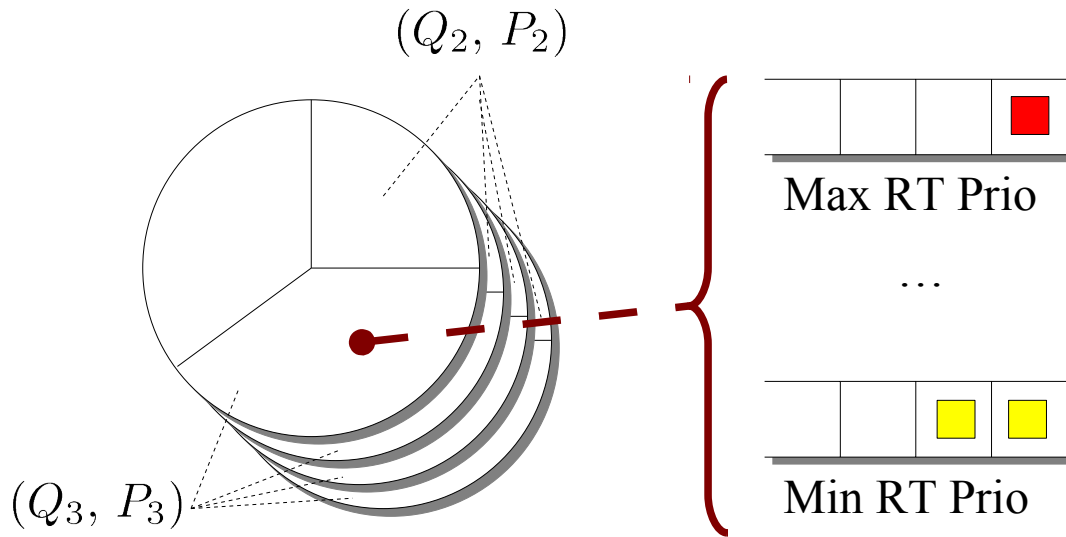
SCHED_SPORADIC

- Parameters: **runtime, period, low-priority**
- POSIX standard system call: `sched_setscheduler()`
 - **Breaks binary interface** & compatibility
- Alternative system call: **`sched_setscheduler_ex()`**

SCHED_DEADLINE

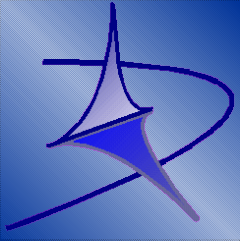
- Parameters: **runtime, period, flags**
- system call: **`sched_setscheduler_ex()`**

Hierarchical Scheduling



Needed operations

- **create** & **destroy** reservations
- **attach** & **detach** tasks ↔ reservations
- **list tasks** attached to reservations (and **list reservations**)
- Standard operations: **get** & **set parameters**



Other Features



Warning: features & parameters may easily grow

- Addition of parameters, such as
 - **deadline**
 - **desired** vs **guaranteed** runtime (for **adaptive reservations**)
- Set of **flags** for controlling variations on behaviour
 - **work conserving** vs **non-conserving** reservations
 - what happens at **fork()** time
 - what happens on tasks **death** (**automatic reclamation**)
 - **notifications** from kernel (e.g., **runtime exhaustion**)
- **Controlled access** to RT scheduling by **unprivileged applications** (e.g., per-user “quotas”)
- **Monitoring** (e.g., residual runtime, available bandwidth)
- Integration/interaction with **power management**



What US/KS mechanism(s) ?

cgroup-based interface ?

- **multi-valued** cgroup entries (for atomic changes)

system-call interface ?

- Only **sched_setscheduler[_ex]()**
- A **set of system calls** ?

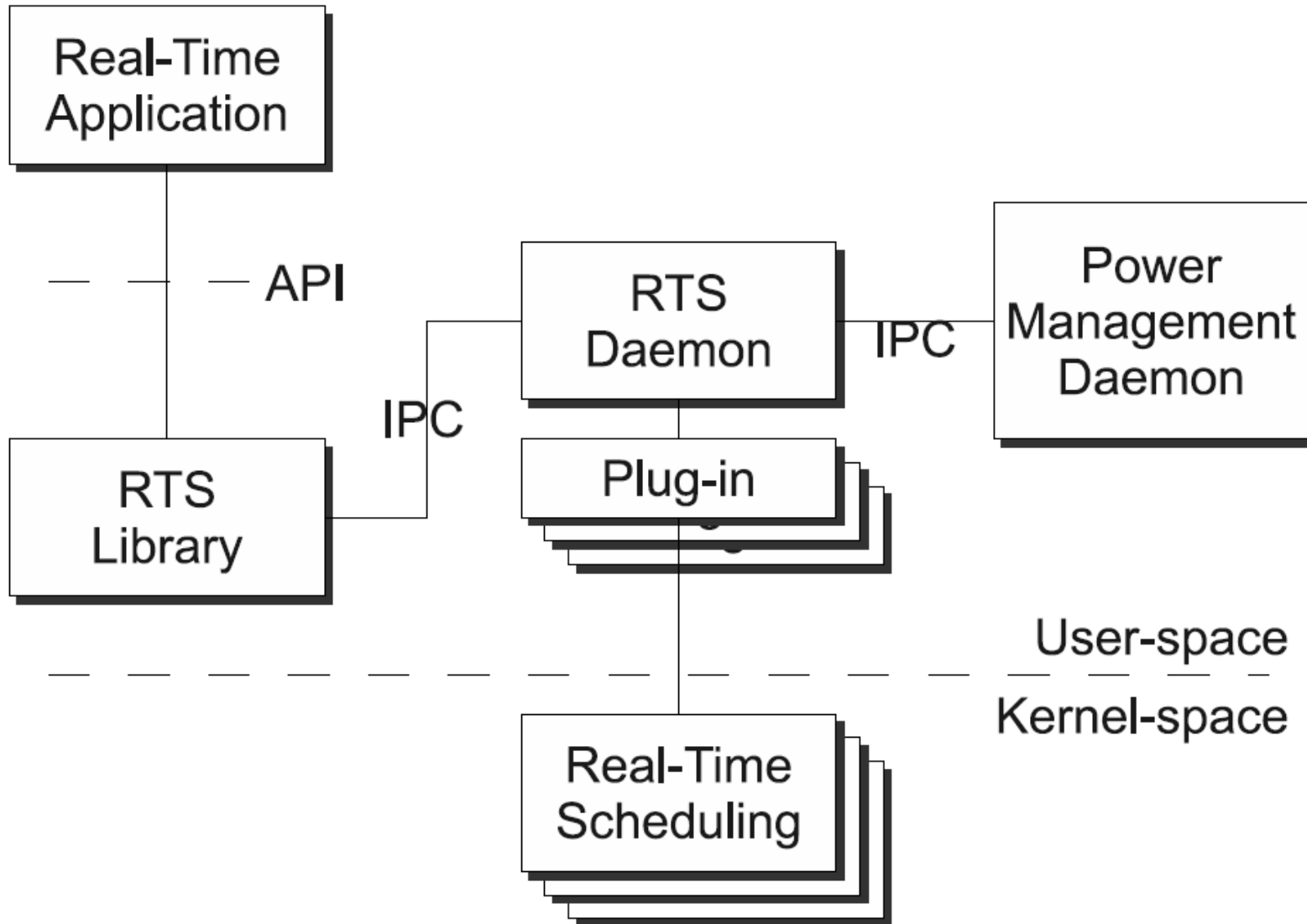
~~**Special device & ioctl() ?**~~

proc-based interface ? (e.g., for monitoring)

Integration with capabilities ?

- **setrlimit() / getrlimit()**

Proposed API for applications





Thanks for your attention



Help!!!

<http://retis.sssup.it/people/tommaso>