Multiprocessor resource sharing Protocol

Implementation and evaluation

Sebastiano Catellani

University of Padua
Supervisor: Prof. Tullio Vardanega

Overview

MrsP
Platform

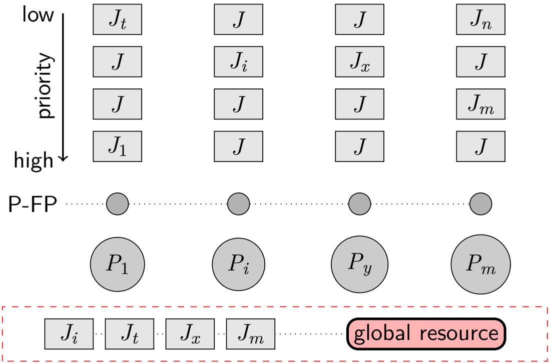


Figure: Partitioned Fixed-Priority scheduler on a platform with m processors (P_1, ..., P_m) and a global resource

Burns and Wellings design a multiprocessor extension of PCP/SRP with the aim of adapt a schedulability analysis to the protocol

Response Time Analysis incorporating PCP/SRP

The parameter e_j reflects the **contention** for the resource (r):

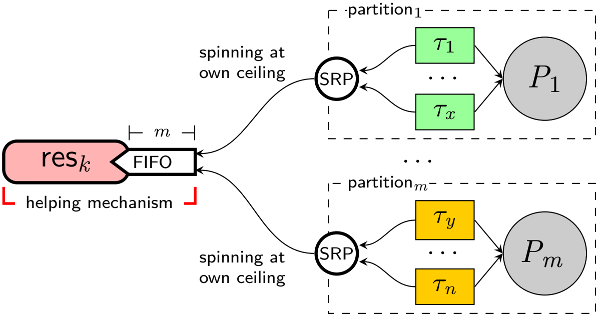
$$e_j = |map(G(r))| \times c_j$$

$$R_i = C_i + max\{e_j, \hat{b}\} + \sum_{\tau_j \in hp(i)} \lceil \frac{R_i}{T_j} \rceil C_j$$

$$C_i = WCET_i + \sum_{r^j \in F(\tau_i)} n_i e_j$$

MrsP

Multiprocessor resource sharing Protocol - 2



Protocol's properties

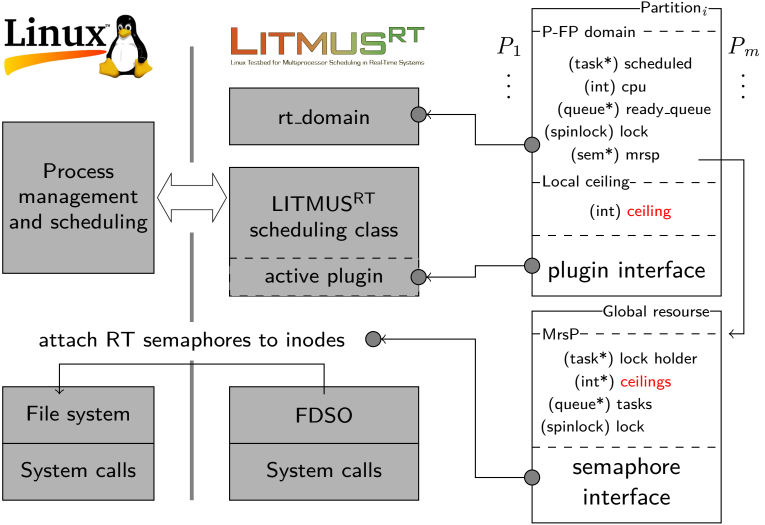
- It inherits the properties of PCP/SRP
- At most one job per processor requires the resource
- The length of the requests queue is at most $|map(G(r_j))|$
- At most e_j to gain the resource and to execute the critical section

Proposed solution
Algorithm

1) Each resource has a set of ceilings, one for each processor
2) An access request causes the rise of the job's priority and activates a local ceiling
3) The requests are queued and served in arrival order
4) A job executes, until resource's release, at the inherited priority
5) If preempted, the lock holder migrates to the first processor available

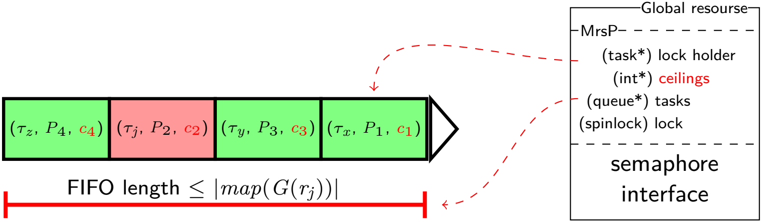
Key features

- Points 2 and 4 make MrsP independence-preserving
- Point 5 guarantees a limited waiting and blocking time



Linux
LITMUS^RT
Linux Testbed for Multiprocessor Scheduling in Real-Time Systems
Process management and scheduling
rt_domain
LITMUS^RT scheduling class
active plugin
attach RT semaphores to inodes
File system
System calls
FDSO
System calls
P1
Pm
Partition_i
P-FP domain
(task*) scheduled
(int) cpu
(queue*) ready_queue
(spinlock) lock
(sem*) mrsp
Local ceiling
(int) ceiling
plugin interface
Global resourse
MrsP
(task*) lock holder
(int*) ceilings
(queue*) tasks
(spinlock) lock
semaphore interface

Focused on managing the access requests queue

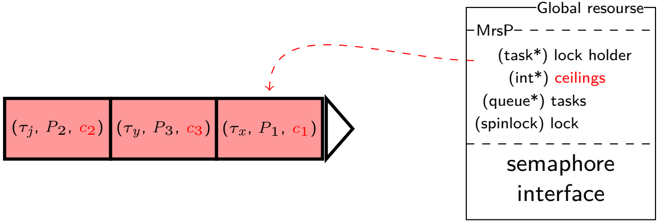


If preempted, the lock holder (J_x)

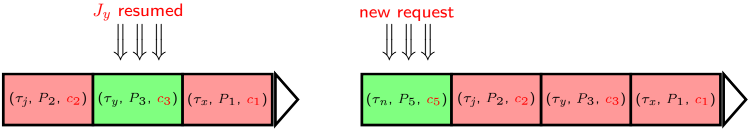
1. inherits the ceiling $(c_3 + 1)$
2. migrates to P_3
3. preempts J_y

Implementation
Queue management - 2

The job will be re-queued in the *ready_queue*



The algorithm catches the operations that

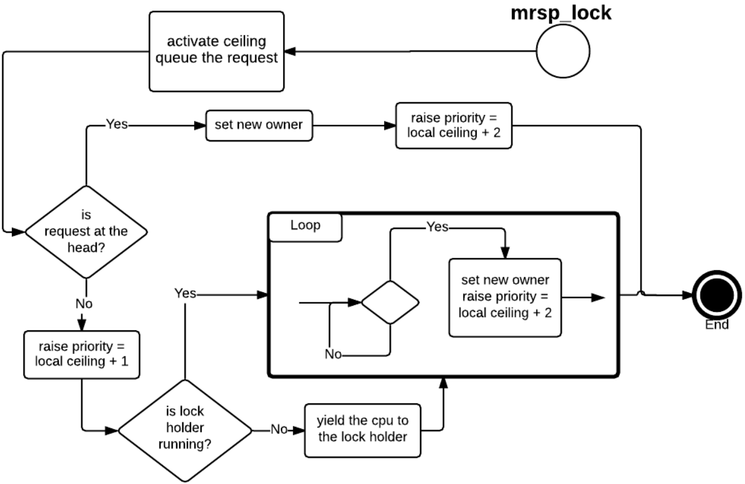


(a) a processor becomes available

(b) add a new request to the queue

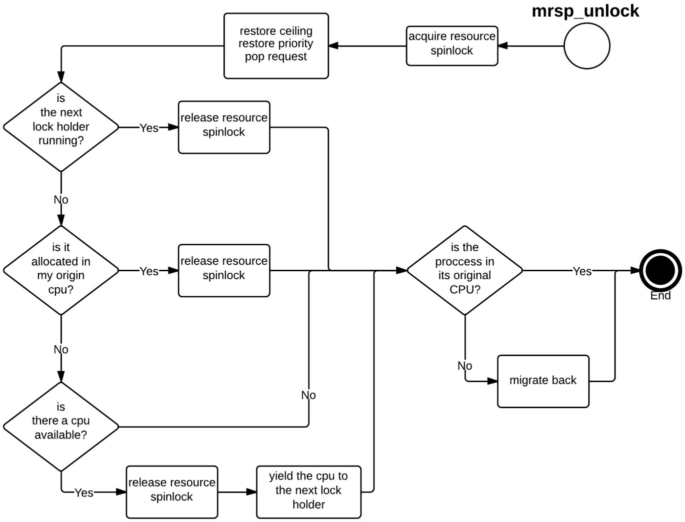
Implementation

Primitive: mrsp_lock



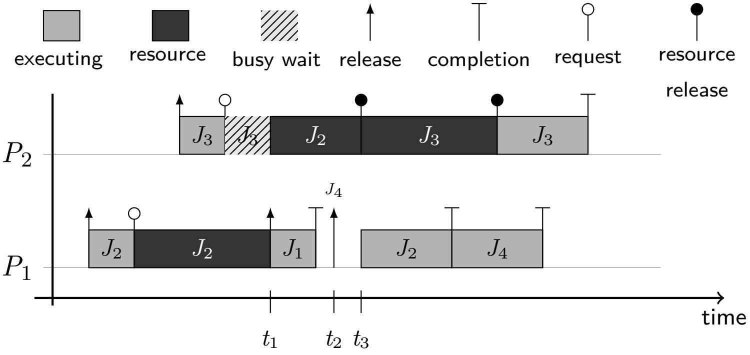
Implementation

Primitive: mrsp_unlock

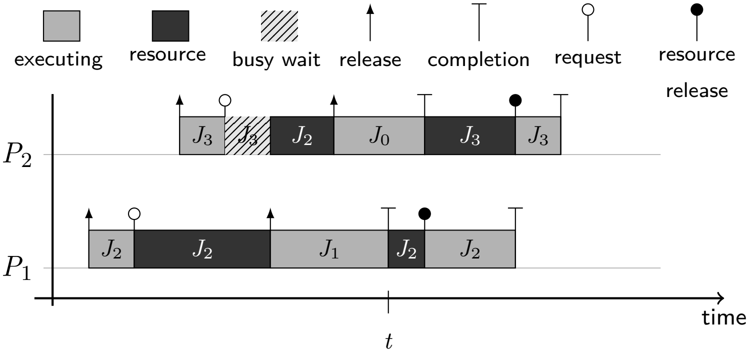


Implementation

Primitive: pfp_schedule and finish_switch - 1



- t_1: J_2 is marked for migration
- t_2: J_4's priority is lower than the local ceiling
- t_3: default migration mechanism



- t: J_1 completes and P_1 returns available

Experiments

Overview

Experiment #1: Comparison among protocols

MrsP outperfmors protocols based on simple ceiling or non preemption

Experiment #2: Sampling of the overheads

MrsP brings benefits at reasonable costs

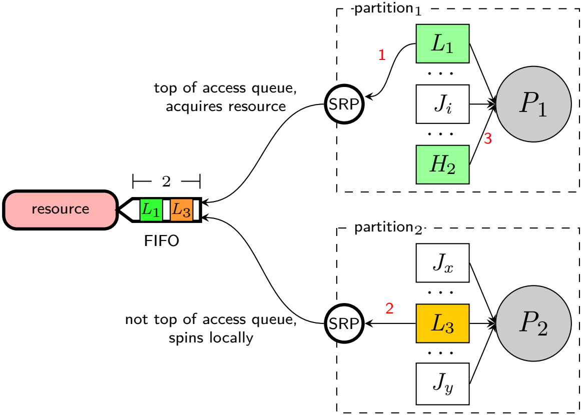
Experiment #3: Absence of global resources

The protocol doesn't interfere with the scheduler

Experiment #1

Comparison among protocols - 1

The experiment observes the response times of L_1, H_2 and L_3 while varying the critical section length and the WCET of H_2



Experiment #1

Comparison among protocols - 2

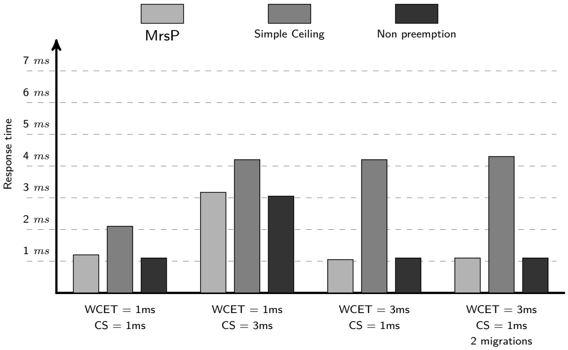


Figure: Response time of L_1

Experiment #1

Comparison among protocols - 3

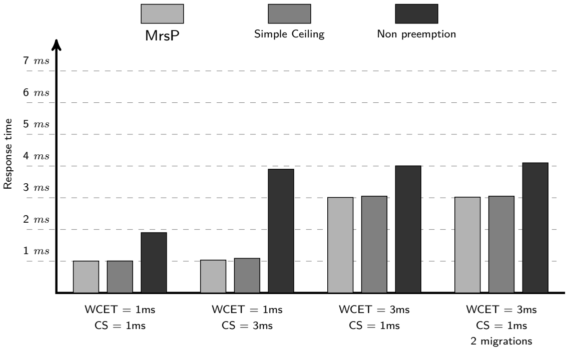


Figure: Response time of H_2

Experiment #1
Comparison among protocols - 4

MrsP

Sebastiano Catellani

MrsP

Proposed solution

Implementation

Experiments

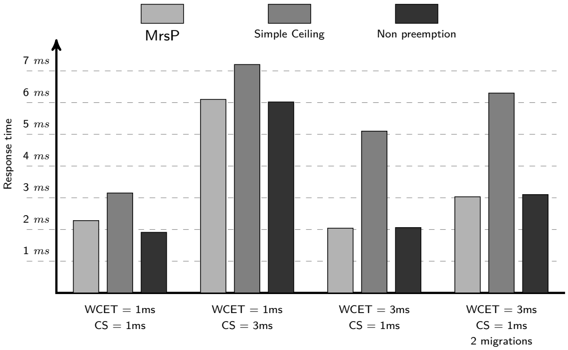
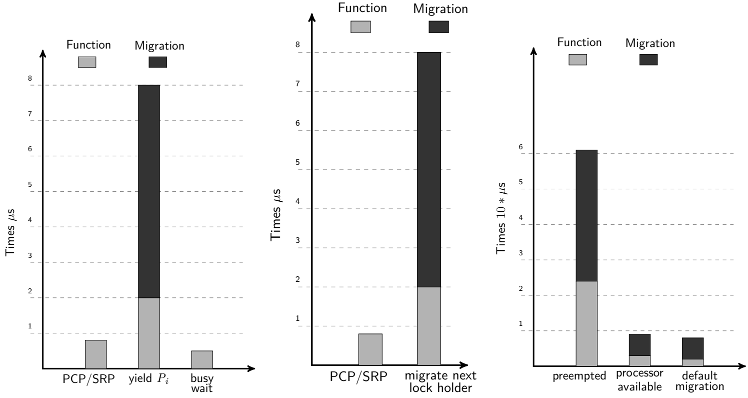


Figure: Response time of L_3

Experiment #2

Sampling of the overheads



(a) mrsp_lock (b) mrsp_unlock (c) finish_switch

Experiment #3

MrsP without global resources

The collected data show the same number of deadline miss

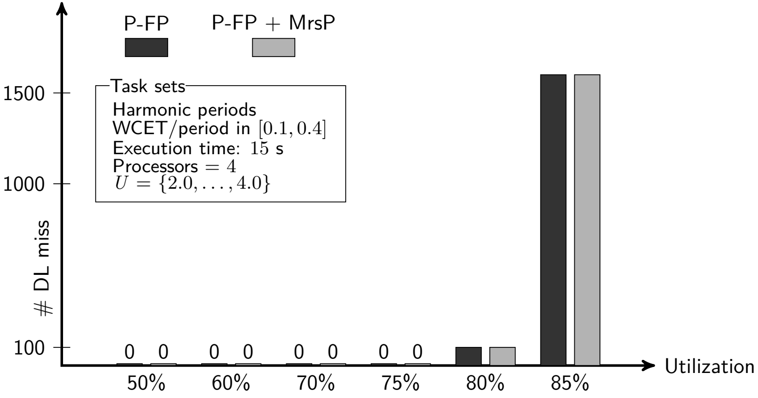
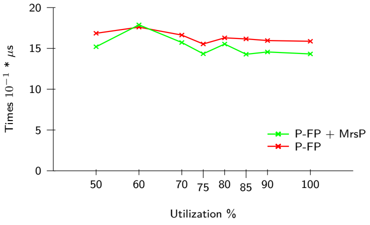


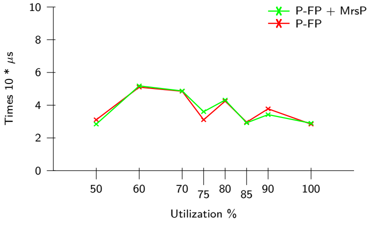
Figure: Number of *deadline miss*

Experiment #3

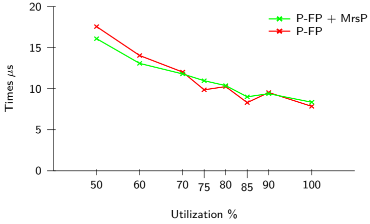
MrsP without global resources - pfp_schedule performance



(a) pfp_schedule: Min



(b) pfp_schedule: Max



(c) pfp_schedule: Average