Dynamic Task Scheduling

Giorgio Buttazzo

Department of Computer Science
University of Pavia

E-mail: buttazzo@unipv.it

Handling shared resources

Problems caused by mutual exclusion

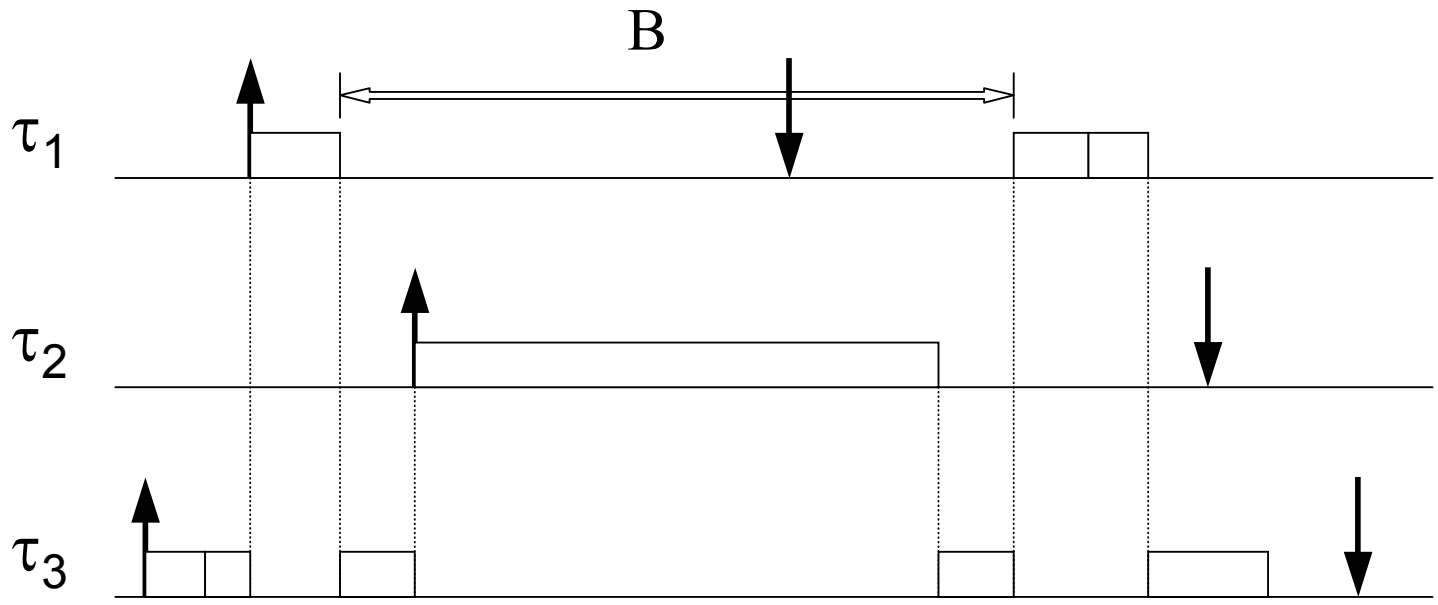
Priority Inversion

A high priority task is blocked by a lower-priority task a for an unbounded interval of time.

Deadline Inversion

A task with short deadline is blocked by a task with longer deadline a for an unbounded interval of time.

Conflict on a critical section



Solution

Introduce a concurrency control protocol for accessing critical sections.

Fixed Priority Protocols

- Non Preemptive Protocol (NPP)
- Highest Locker Priority (HLP)
- Priority Inheritance Protocol (PIP)
- Priority Ceiling Protocol (PCP)
- Immediate Priority Ceiling (IPC)

Dynamic Priority Protocols

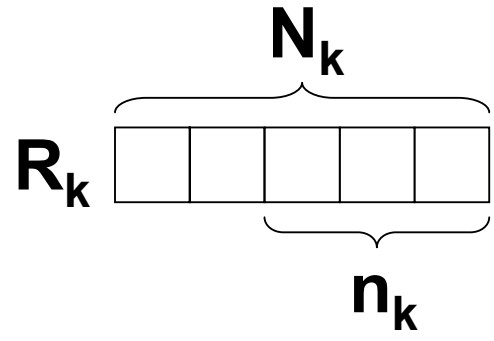
- Dynamic Priority Inheritance (DPI)
- Dynamic Priority Ceiling (DPC)
- Dynamic Deadine Modification (DDM)
- Stack Resource Policy (SRP)

Stack Resource Policy [Baker 1990]

- It works both with fixed and dynamic priority
- It limits blocking to 1 critical section
- It prevents deadlock
- It supports multi-unit resources
- It allows stack sharing
- It is easy to implement

Stack Resource Policy [Baker 90]

- For each resource $\mathbf{R_k}$:
  - $\Rightarrow$ Maximum units: $\mathbf{N_k}$
  - $\Rightarrow$ Available units: $\mathbf{n_k}$



- For each task τ_i the system keeps:
  - $\Rightarrow$ its resource requirements: $\mu_i(R_k)$
  - $\Rightarrow$ a priority $\mathbf{p_i}$: RM $p_i \propto 1/T_i$ EDF $p_i \propto 1/d_i$
  - $\Rightarrow$ a static preemption level: $\pi_i \propto 1/D_i$

Stack Resource Policy [Baker 90]

Resource ceiling

$$C_k(n_k) = \max_j \left\{ \pi_j : n_k < \mu_j(R_k) \right\}$$

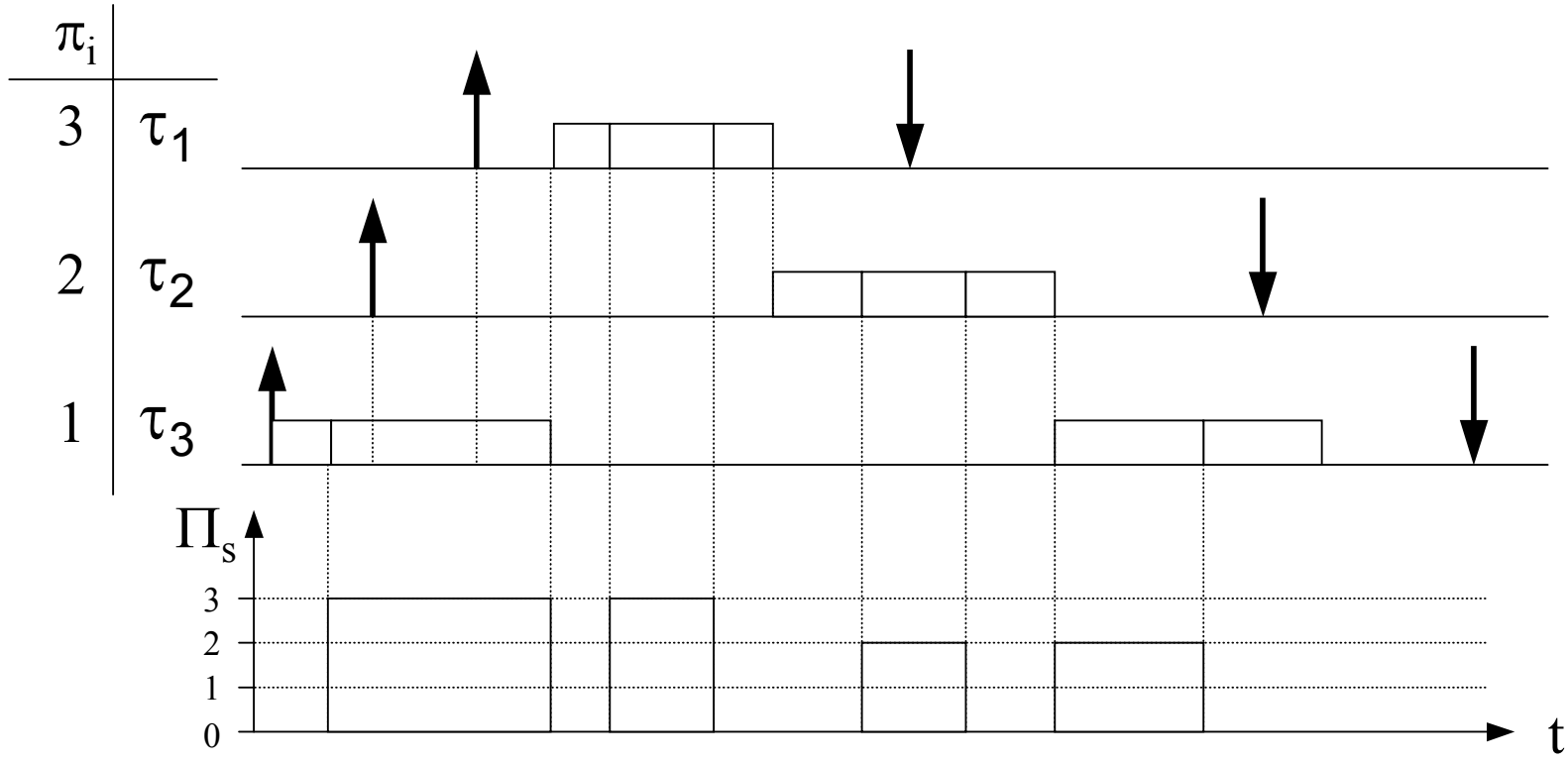
System ceiling

$$\Pi_s = \max_k \{ C_k(n_k) \}$$

SRP Rule

A job cannot preempt until
p_i is the highest and $\pi_i > \Pi_s$

Example

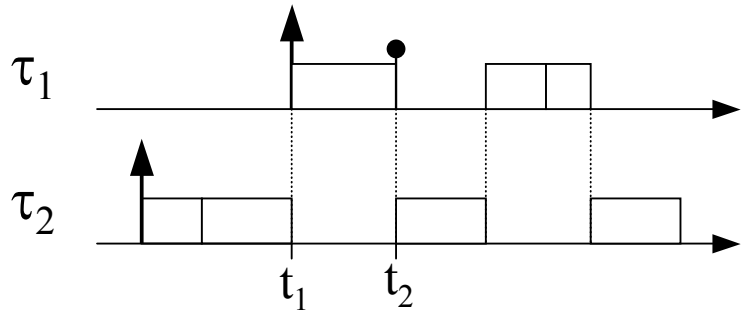


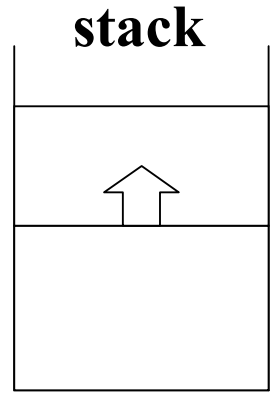
SRP: Notes

- Blocking always occurs at preemption time
- A task never blocks on a wait primitive (semaphore queuee are not needed)
- Semaphores are still needed to update the system ceiling
- Early blocking allows stack sharing

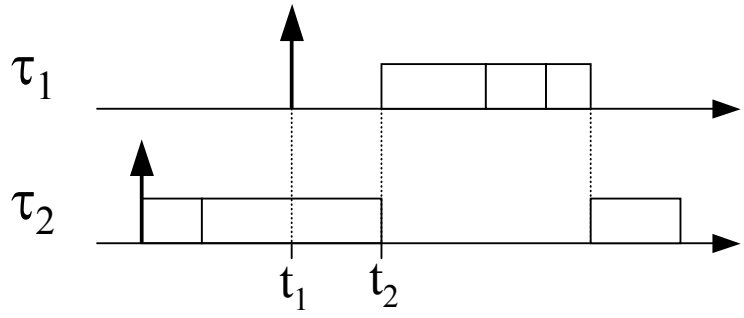
SRP: Stack sharing

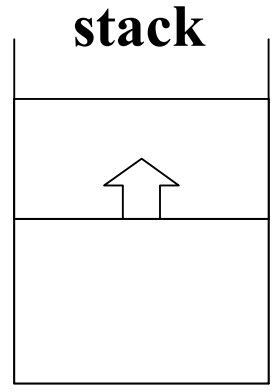
Classical blocking





Early blocking





SRP: Stack sharing

- If tasks can be grouped in **M** subsets with the same preemption level, then tasks within a group cannot preempt each other.
- Then the stack size is the sum of the stack memory needed by **M** tasks.
- If we have 100 tasks with 10 preemption levels, and each task requires 10 Kb of stack, then

$$\textbf{Stack size} = \begin{cases} \textbf{1 Mb} & \text{without SRP} \\ \textbf{100 Kb} & \text{under SRP (90\% less)} \end{cases}$$