ECE 3140 / CS 3420
EMBEDDED SYSTEMS

LECTURE 18

Prof. José F. Martínez

TR 1:25-2:40pm in 150 Olin

Cornell University
Computer Systems Laboratory

Non-Preemptive Protocol

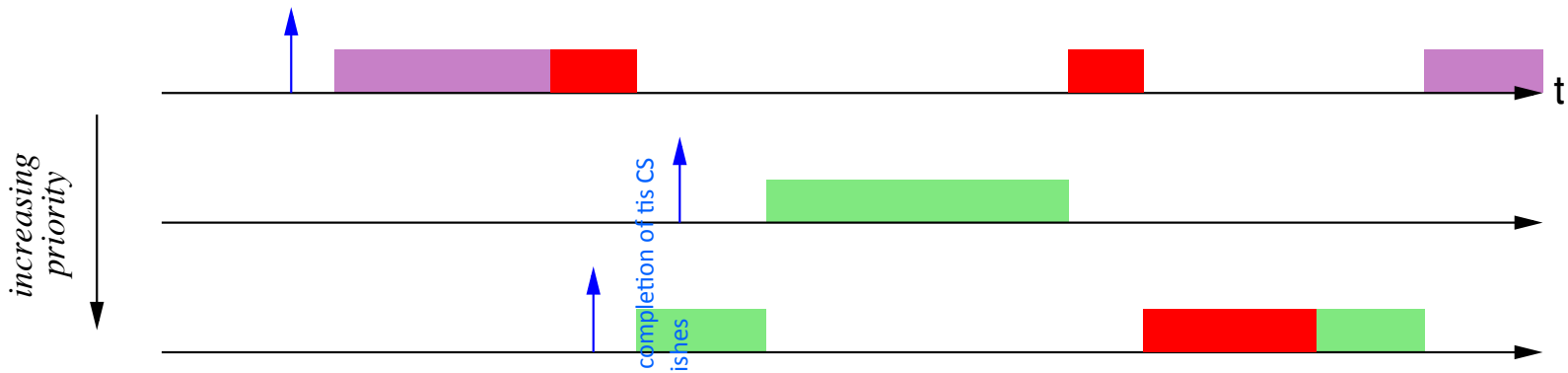
Simple modification:

- Preemption is forbidden in critical sections
- To implement: when a task enters a critical section, increase its priority to the maximum value.
- $p_{CS} = \max_i \{p_1, \ldots, p_n\}$

Drawbacks:

- High priority tasks that do not interfere with the critical section will be blocked

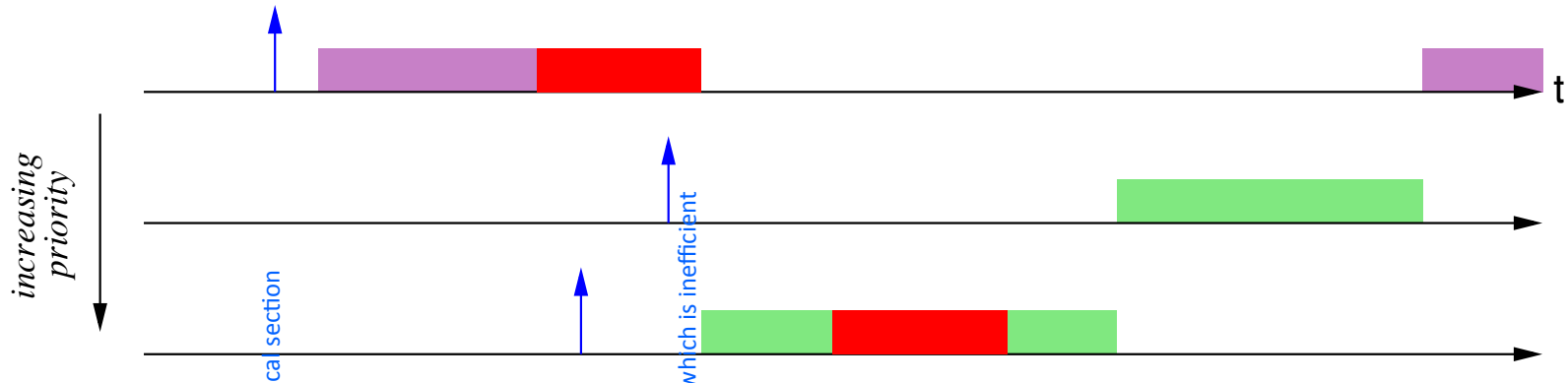
Non-Preemptive Protocol



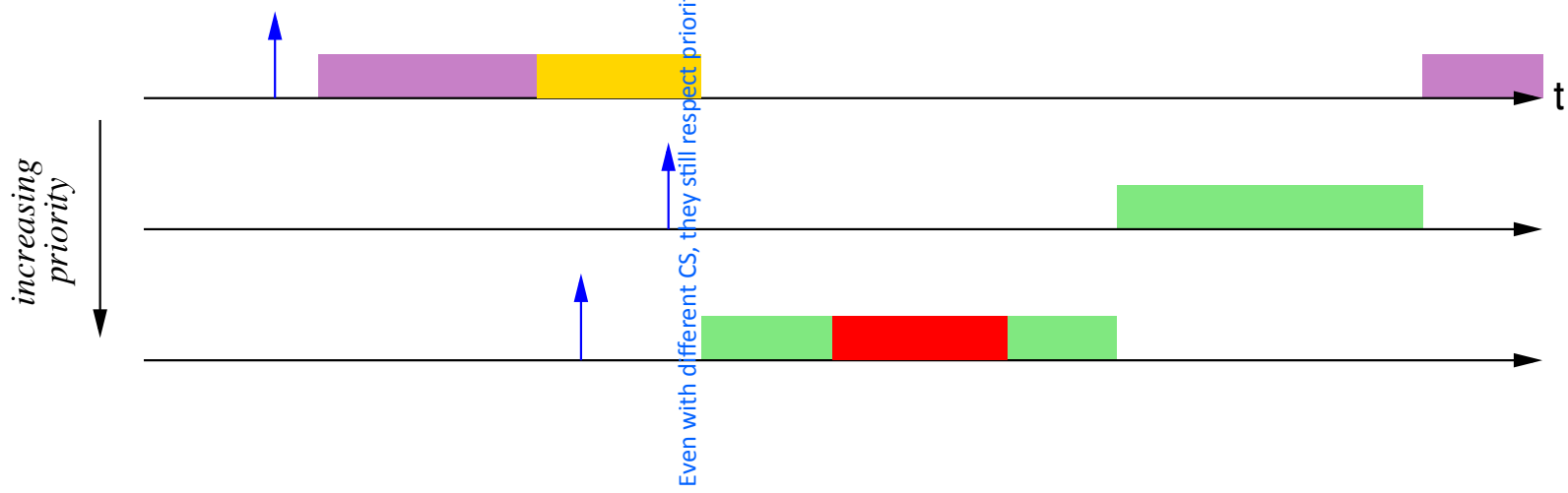
With NPP:

NON-PREEMPTIVE PROTOCOL

NPP schedule:



. . . even for critical sections that don't matter



Bump priority above everyone else that uses the same CME critical

Highest Locker Priority

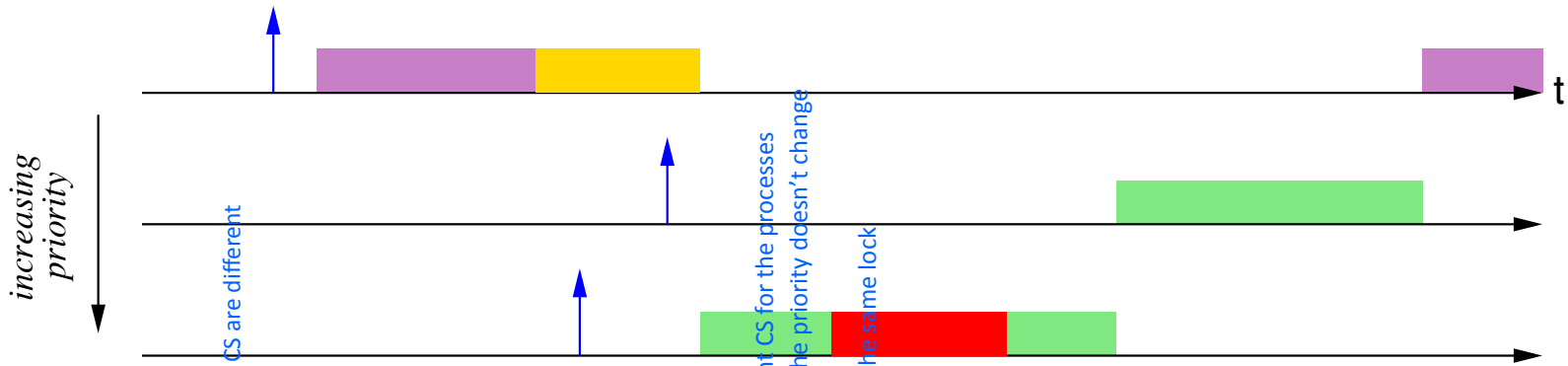
- A task in the critical section gets the highest priority among the tasks that use the critical section.
- To implement: when a task enters a critical section, increase its priority to the maximum value of the tasks that may access the critical section.
- $p_{CS} = \max_i \{p_i \mid \tau_i \text{ uses CS}\}$

Drawbacks:

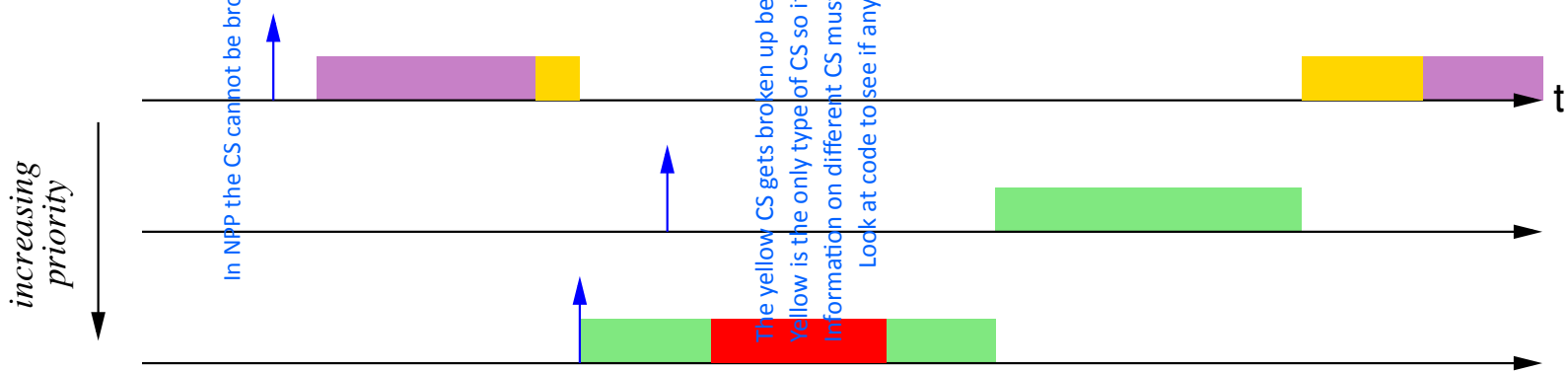
- A task could be blocked because it *might* enter the critical section, not because it is in the critical section.

HIGHEST LOCKER PRIORITY

NPP:



HLP:

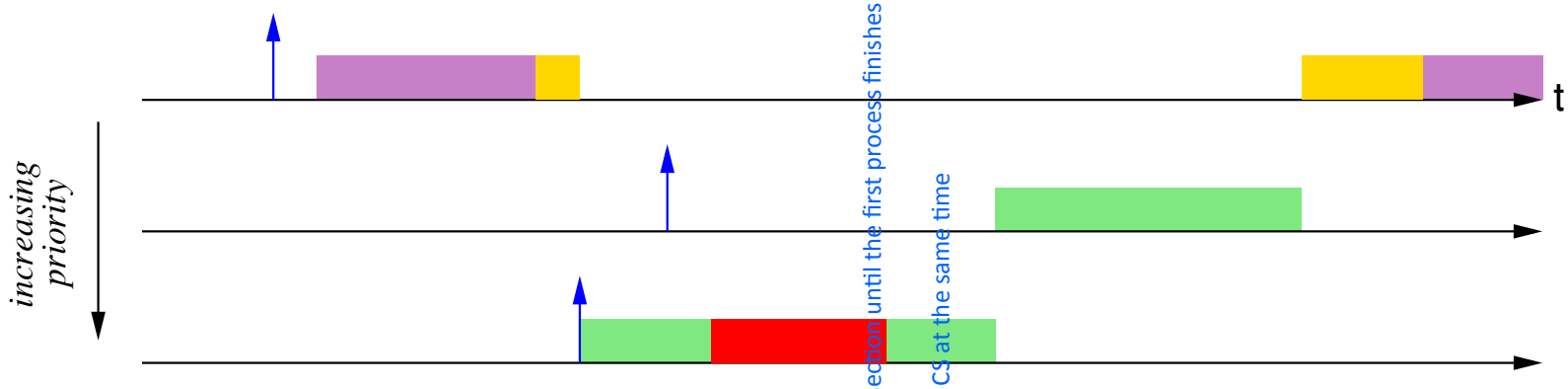


In NPP the CS cannot be broken up even though the CS are different

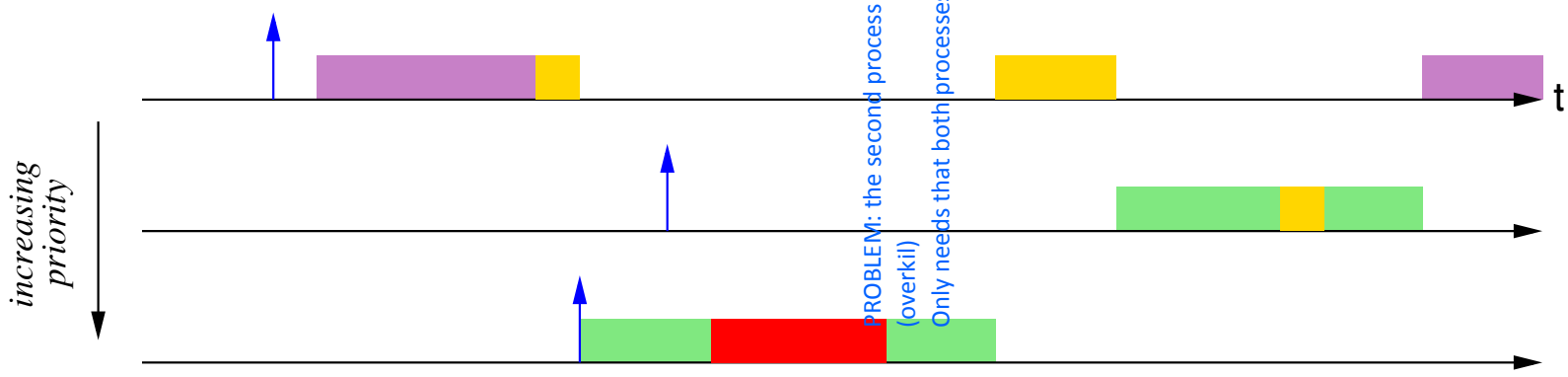
The yellow CS gets broken up because there are different CS for the processes
Yellow is the only type of CS so it can be broken up; so the priority doesn't change
Information on different CS must be known offline
Lock at code to see if anything would every use the same lock

HIGHEST LOCKER PRIORITY

HLP:



If the middle task might use the yellow lock:



Priority Inheritance Protocol

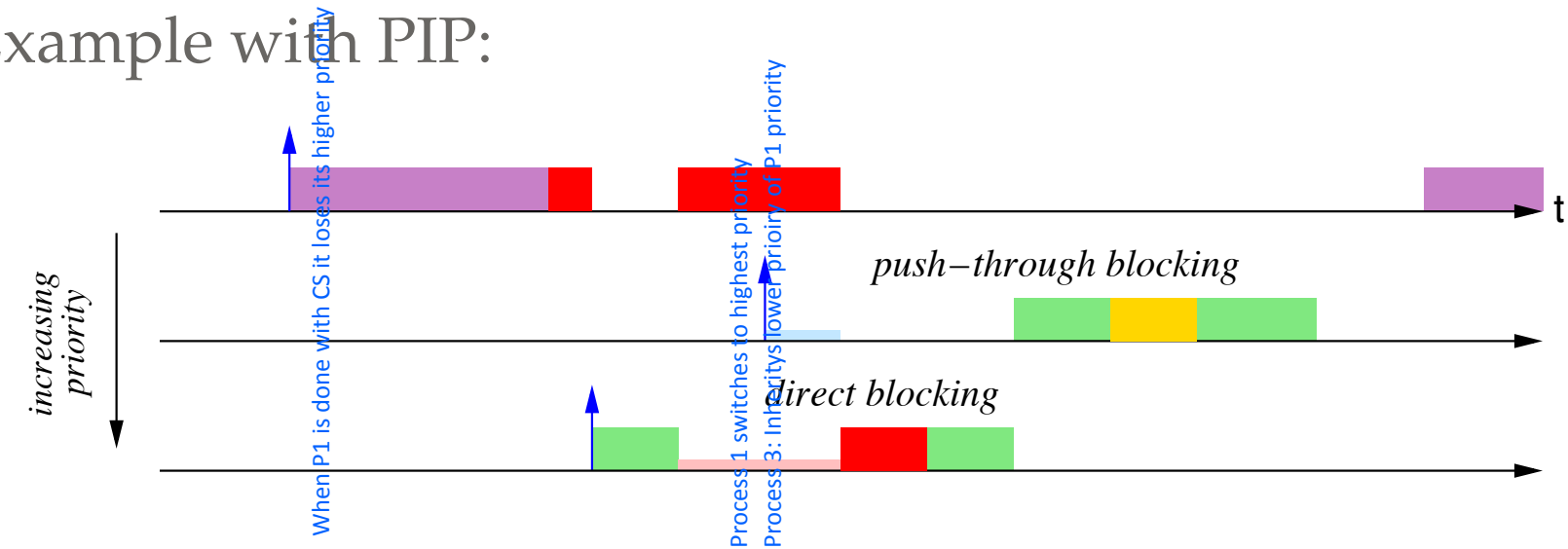
- A task in a critical section increases its priority only if it blocks other tasks.
- A task in a critical section inherits the highest priority among those tasks that it blocks.
- $p_{CS} = \max_i \{p_i \mid \tau_i \text{ blocked on CS}\}$

Two types of blocking:

- Direct: task blocked on a lock
- Push-through: task blocked because a lower priority task inherited a higher priority

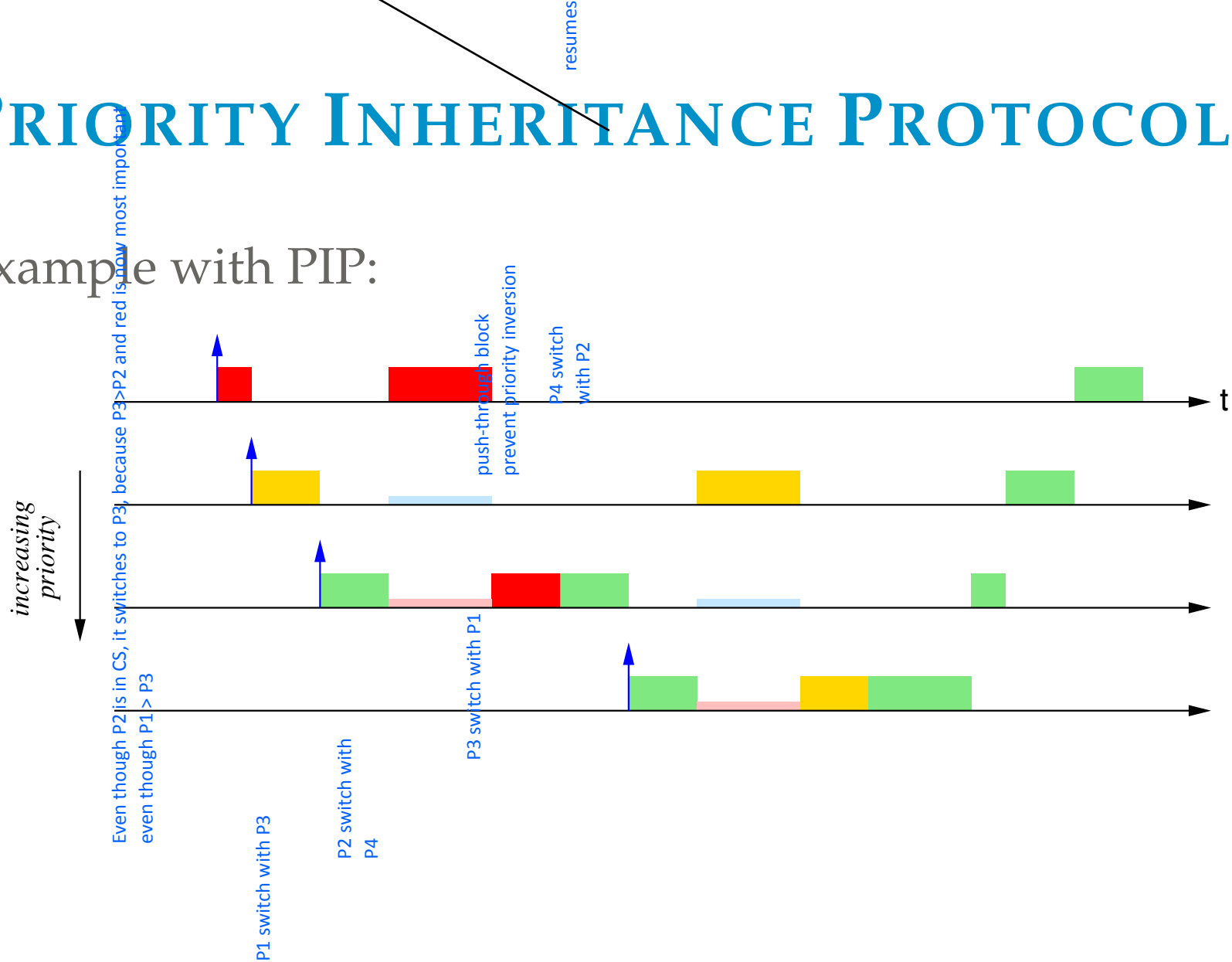
Priority Inheritance Protocol

Example with PIP:

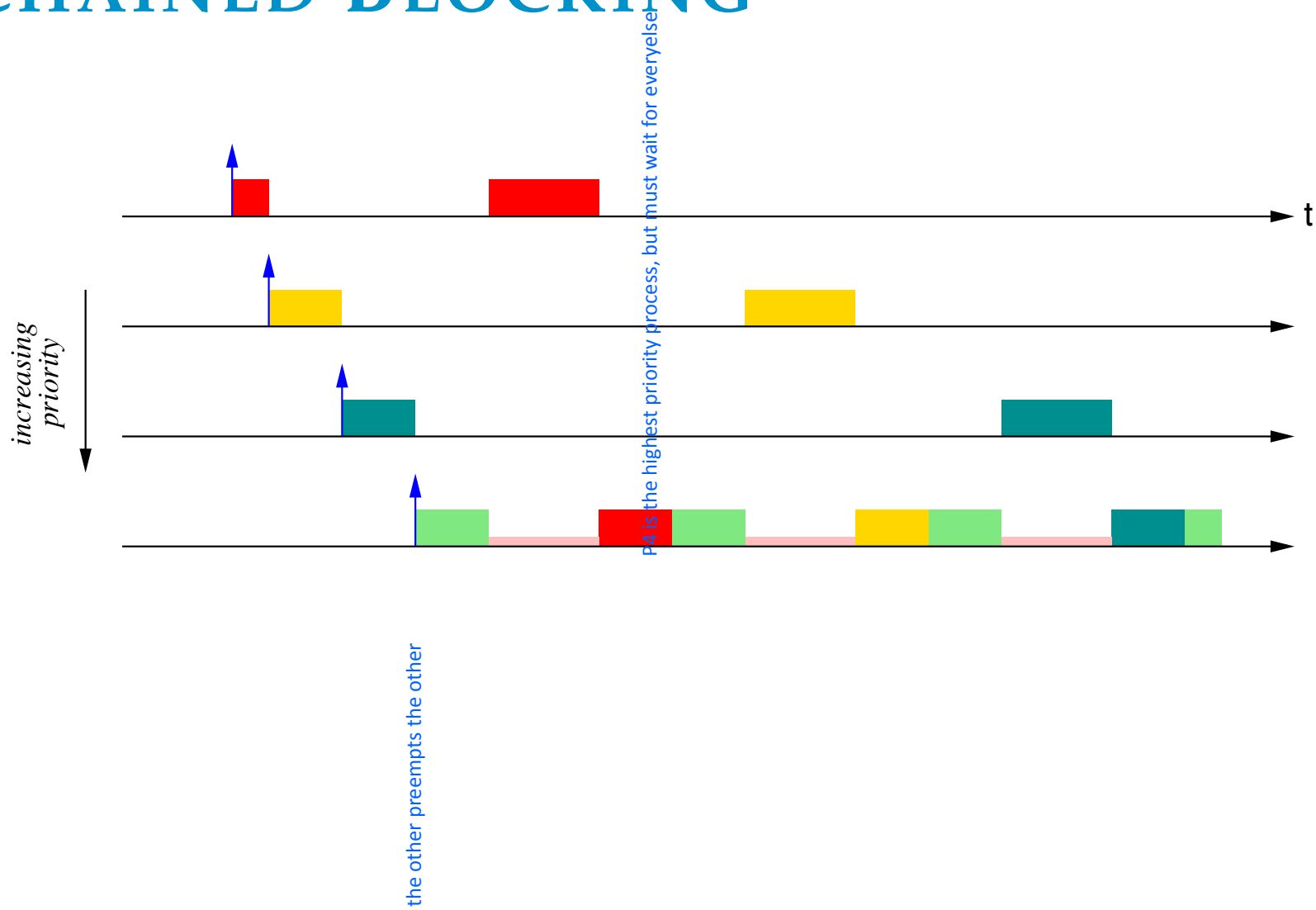


Priority Inheritance Protocol

Example with PIP:



Chained Blocking



Priority Ceiling Protocol

Attempts to reduce chained blocking

- A modification of the PIP protocol
- Each lock is assigned a *ceiling*
  - For a lock l_k,

$$C(l_k) = \max_i \{p_i \mid \tau_i \text{ uses } l_k\}$$

  - A task τ_i can enter the critical section only if

$$p_i > \max_k \{C(l_k) \mid l_k \text{ is locked by tasks } \neq \tau_i\}$$

  - As in PIP, tasks inherit the (highest) priority of the task(s) they block

LAST ONE! The most fancy

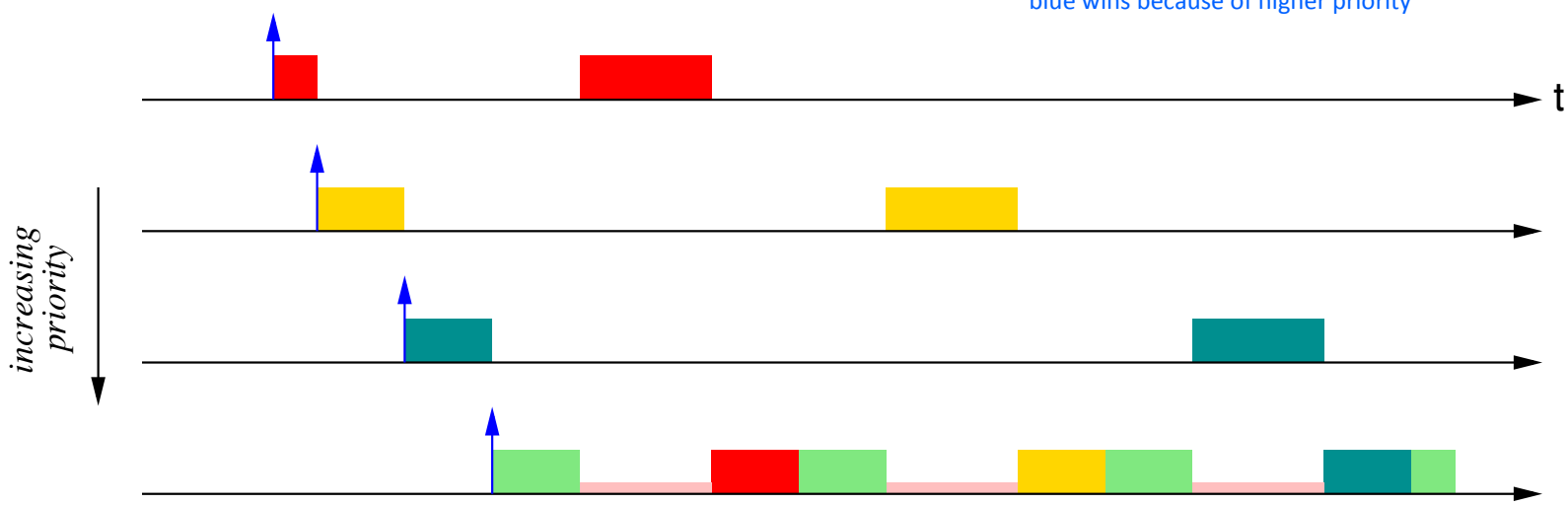
**IDEA: in order to avoid chain block, push the CS to either side of me

Priority Ceiling Protocol

All CS have a ceiling of priority of 3
i CAN ONLY GO INTO THE priority if my priority is higher then the active locks

PIP:

Once red CS finishes, then everyone wants to enter CS, blue wins because of higher priority



encumbered by blue because blue has already started: must use PIP

PCP: