ECE3140 / CS3420
Embedded Systems

Resource Access Protocols

Prof. José F. Martínez

Cornell University

Outline: Resource Access Protocol

Scheduling algorithms with restrictions on shared resources (mutual exclusion)

- Problem: Priority inversion
- Algorithms
  - Non-Preemptive Protocol (NPP)
  - Highest Locker Priority (HLP)
  - Priority Inheritance Protocol (PIP)
  - [Priority Ceiling Protocol (PCP)]
- Reference
  - Chapter 7, “Hard Real-Time Computing Systems Predictable Scheduling Algorithms and Applications” by Giorgio C. Buttazzo (Free electronic copy through Cornell library)

Problem Example: Mars Pathfinder

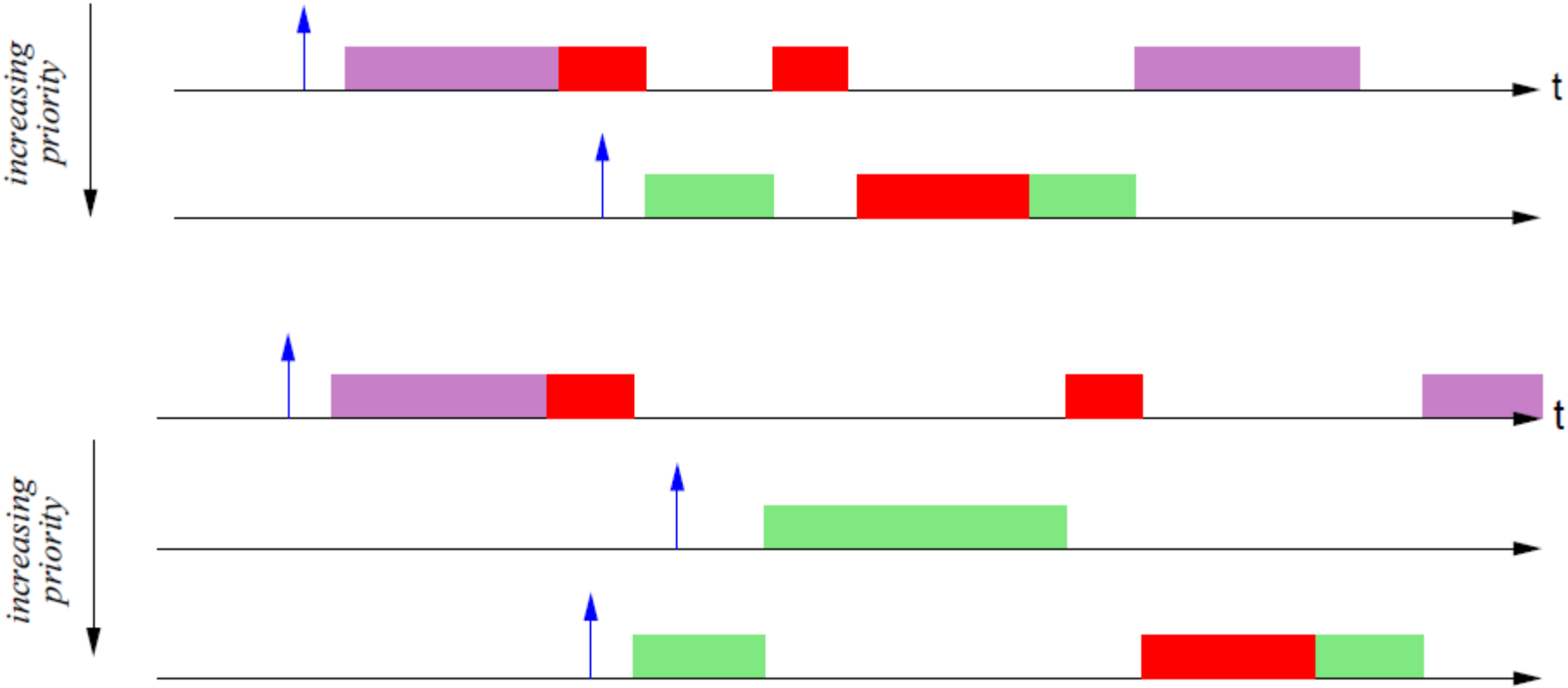
- Failure in a mars mission
  - July 4, 1997: landing on Mars
  - July 15, 1997: the system self-resets on a timeout; the robot stops working
  - July 17, 1997: the error is identified

Problem Example: Continue

- What happened?
  - https://www.cs.unc.edu/~anderson/teach/comp790/papers/mars_pathfinder_long_version.html
- Problem: a high-priority task was blocked by a lower priority task for an unbounded interval of time. Phenomenon is referred to as priority inversion
  - Semaphore to control access to the list of file descriptors
  - High priority task: 1553 bus control software (‘bc_dist’)
  - Low priority task: AST/MET tasks, which collects meteorological science data
- Resource Access Protocols: modifications to limit delays due to resource contention
  - In fixed priority scheduling, adjust the priority of a task when accessing a shared resource

Priority Inversion

- Blocking on a critical section can increase the delay...
  - In the example, red sections indicate critical sections



- Ideas for solutions?

Non-Preemptive Protocol (NPP)

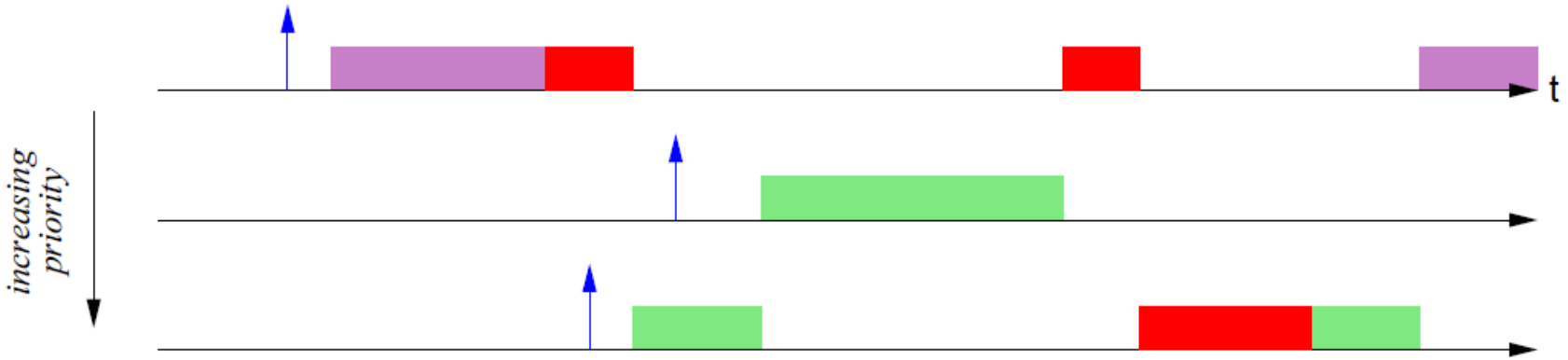
- Preemption is forbidden in critical sections
- To implement:
  - When a task enters a critical section, increase its priority to the maximum value

$$p_{CS} = \max_i\{p_1, \ldots, p_n\}$$

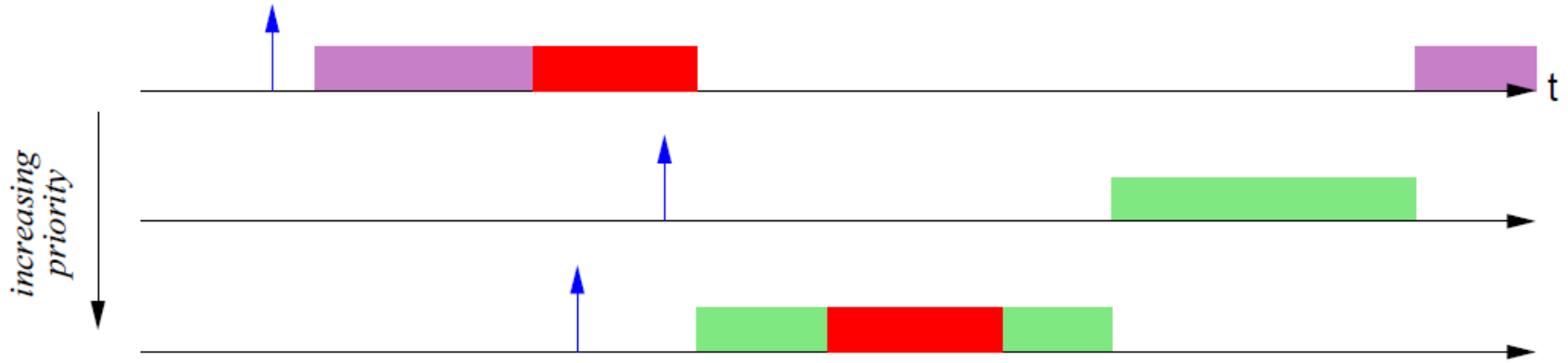
- Drawback:
  - High priority tasks that do not interfere with the critical section will be blocked

NPP Example

- Priority inversion without NPP

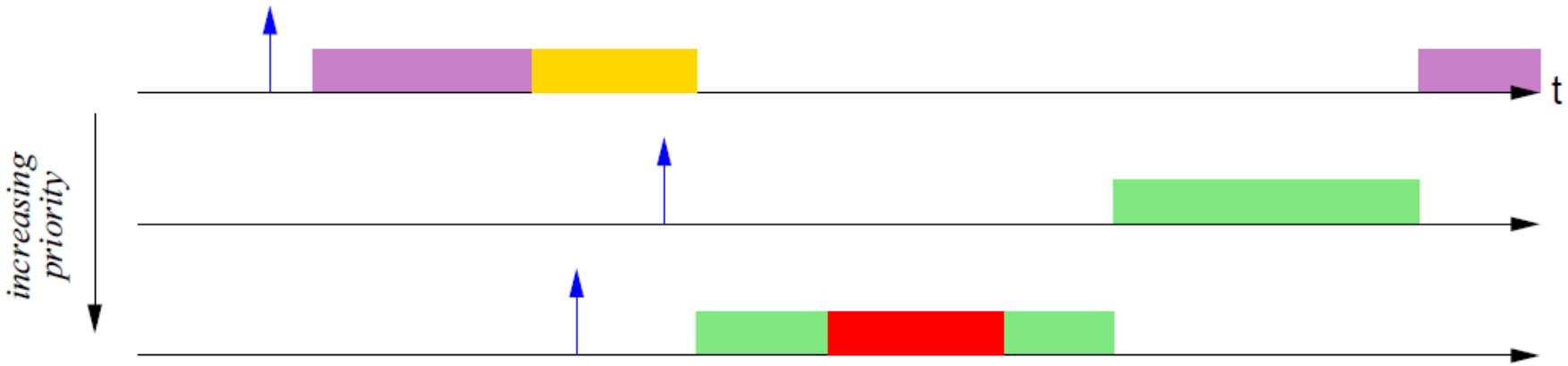


- With NPP



Inefficiency in NPP

- No preemption even for critical sections that don't matter
  - Yellow and Red represent critical sections protected with two different locks



Highest Locker Priority

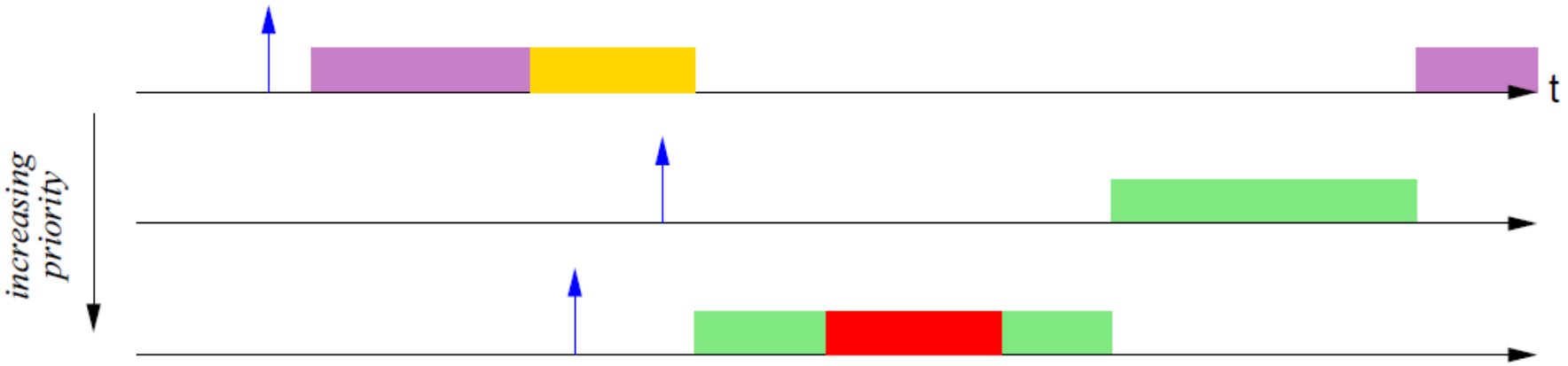
- A task in the critical section gets the highest priority among the tasks that use the critical section
- To implement:
  - when a task enters a critical section, increase its priority to the maximum value of the tasks that may access the critical section

$$p_{CS} = \max_i \{p_i | \tau_i \; uses \; the \; lock \; for \; CS\}$$

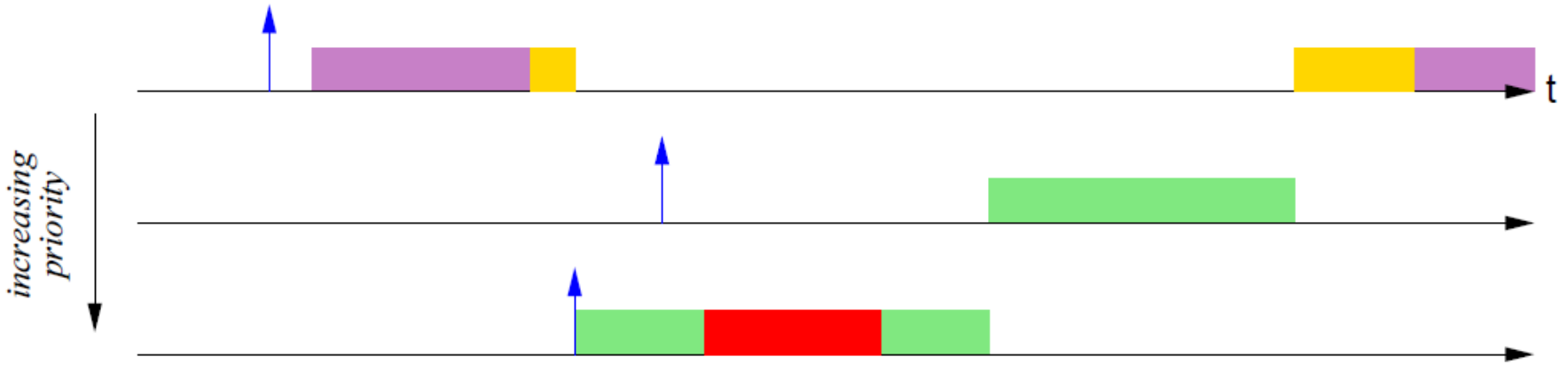
- Drawback:
  - A task could be blocked because it might enter the critical section, not because it is in the critical section.

Highest Locker Priority (HLP) Example

- NPP

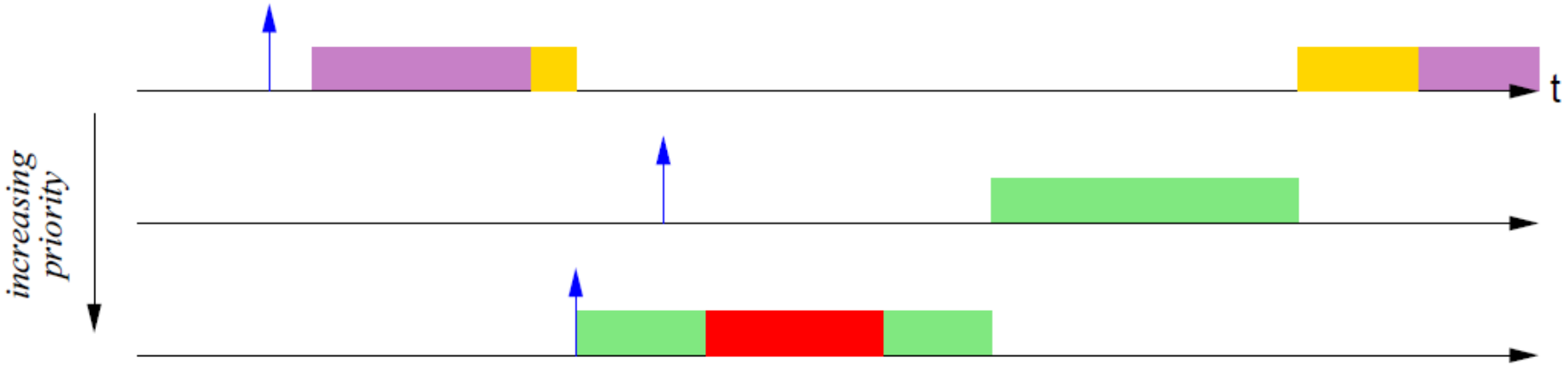


- HLP
  - Yellow lock: Priority 1, Red lock: Priority 4

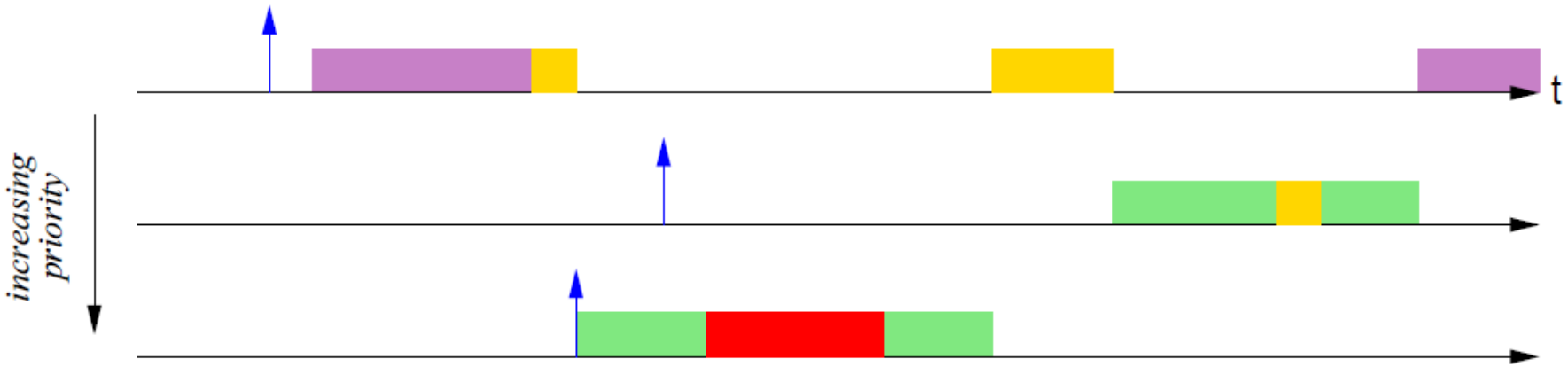


HLP Example with Different Priorities

- HLP



- If the middle task might use the yellow lock:
  - Yellow lock: Priority 2, Red lock: Priority 4



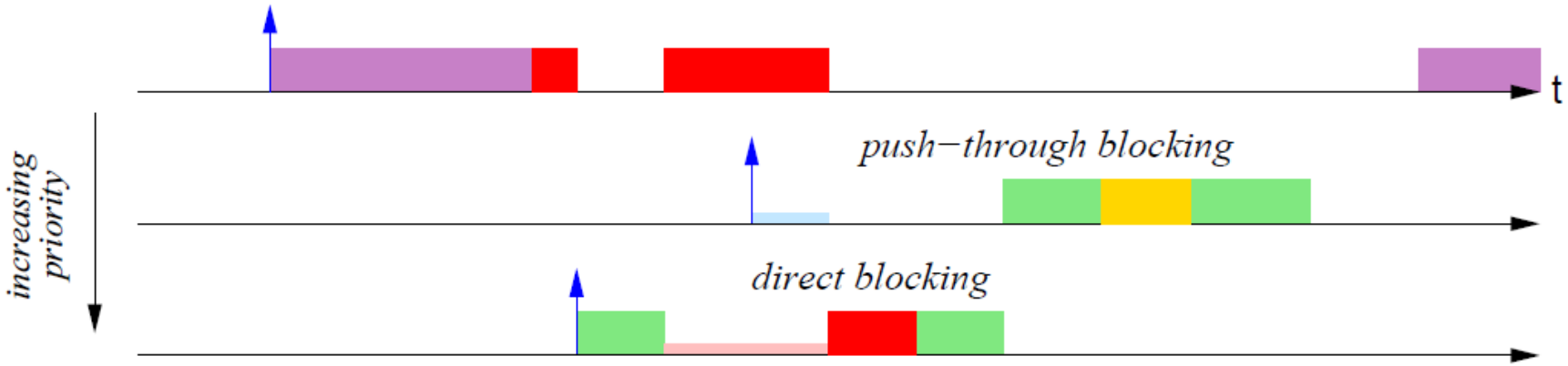
Priority Inheritance Protocol

- 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

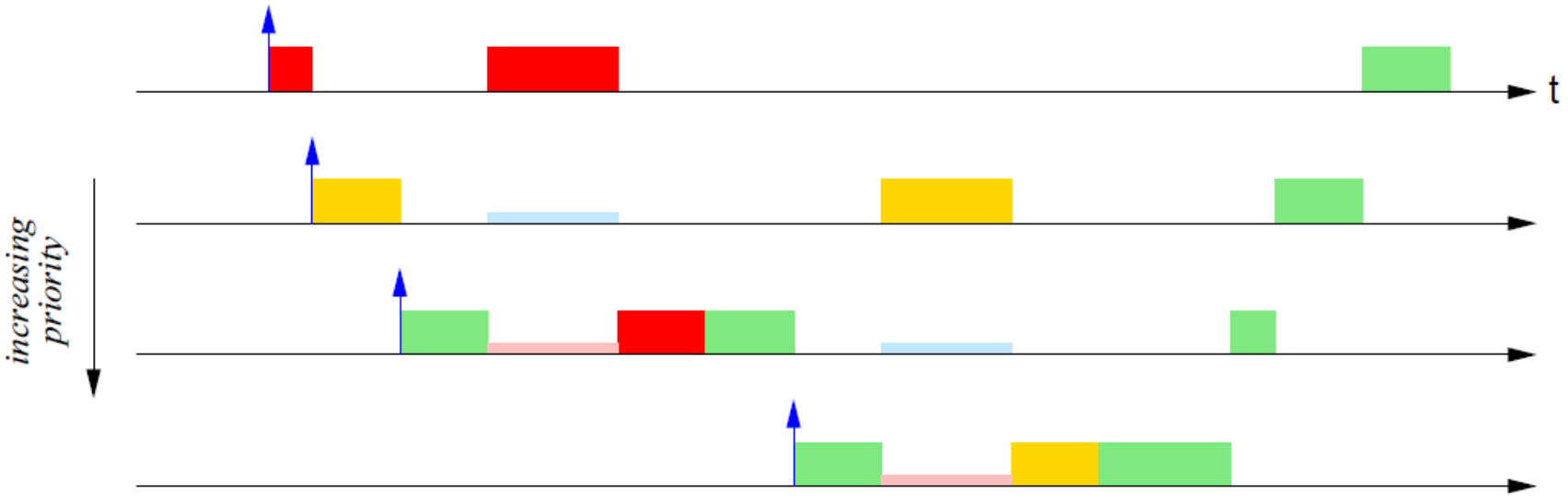
$$p_{CS} = \max_i \{p_i | \tau_i \; blocked \; on \; CS\}$$

- Two types of blocking:
  - **Direct**: a higher-priority task blocked on a lock (resource) already held by a lower-priority task
  - **Push-through**: a medium-priority task blocked by a low-priority task because the low-priority task inherited a higher priority from a task it directly blocks

Example of PIP

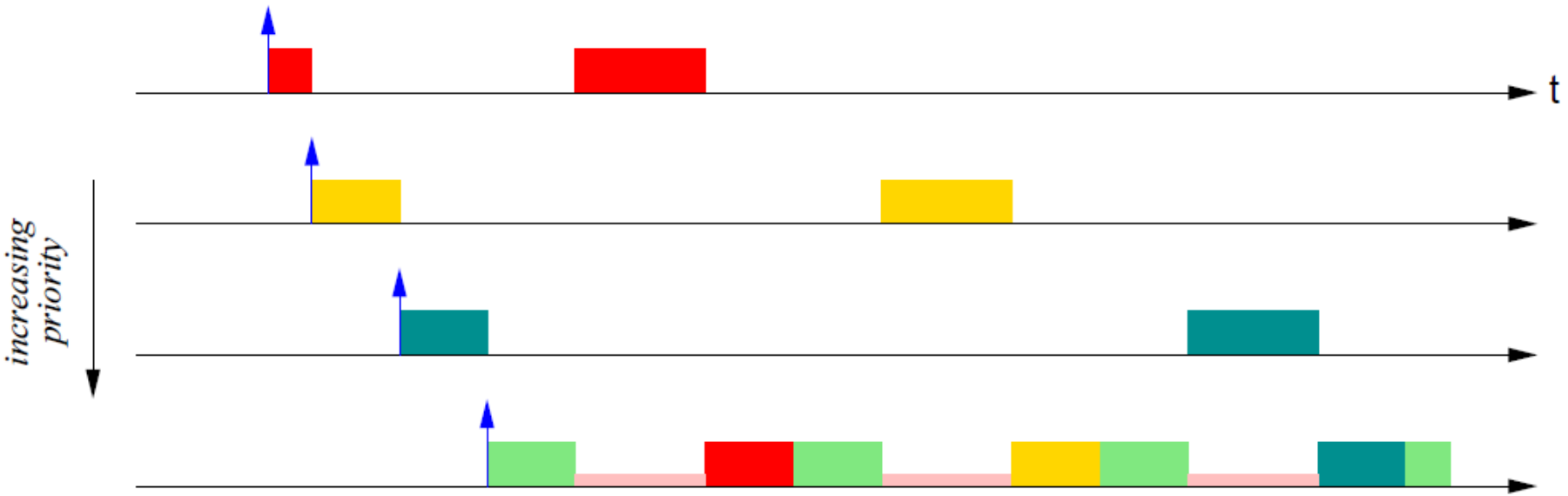


PIP Example with More Tasks



Chained Blocking

- PIP allows preemption inside a critical section
- A high-priority task may be blocked by multiple critical sections in lower-priority tasks



Priority Ceiling Protocol

- Attempts to reduce chained blocking
- A modification of the PIP protocol
- Each lock is assigned a *priority ceiling*
  - For a lock l_k,

$$C(l_k) = \max_i \{p_i | \tau_i \; uses \; l_k\}$$

  - A task τ_i can enter the critical section only if

$$\mathrm{p}_i > \max_k \{C(l_k) | l_k \; is \; locked \; by \; tasks \neq \tau_i\}$$

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