嵌入式系統

單元五：排程方法

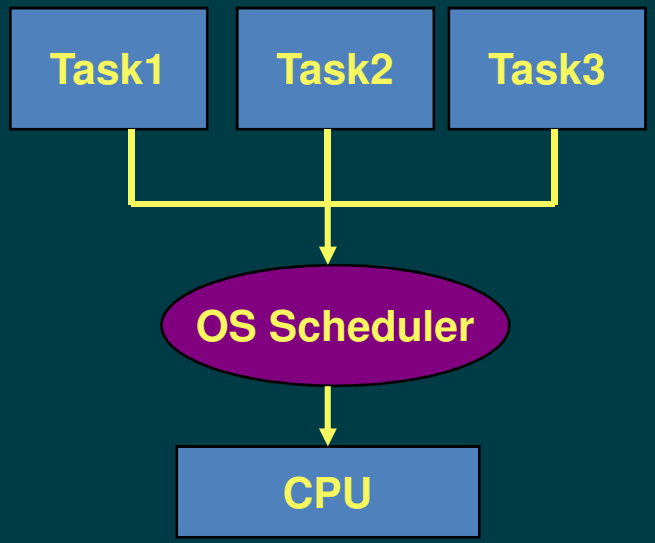
授課教師：雲林科技大學 張慶龍 老師

單元學習目標與大綱

- Non-Real-Time Scheduling
- Real-Time Scheduling
- Priority inversion problem

排程器 Scheduler

- 為RTOS之核心元件
- 依一定策略，決定Task之執行順序
- 排程策略要滿足
  - 所有Hard real-time tasks deadline
  - 最少的miss-deadline (for soft real-time tasks)
  - 最小的延遲反應



排程器 Scheduler

- Non-real-time scheduling
  - 以最高效能為考量
  - 滿足公平性(fairness)原則
  - 如：First-come-First-Served排程與round-robin排程
- Real-time scheduling
  - 以miss dead-line為最主要考量
  - 如：priority-based non-preemptive排程(又稱cooperative排程)
  - priority-based preemptive排程

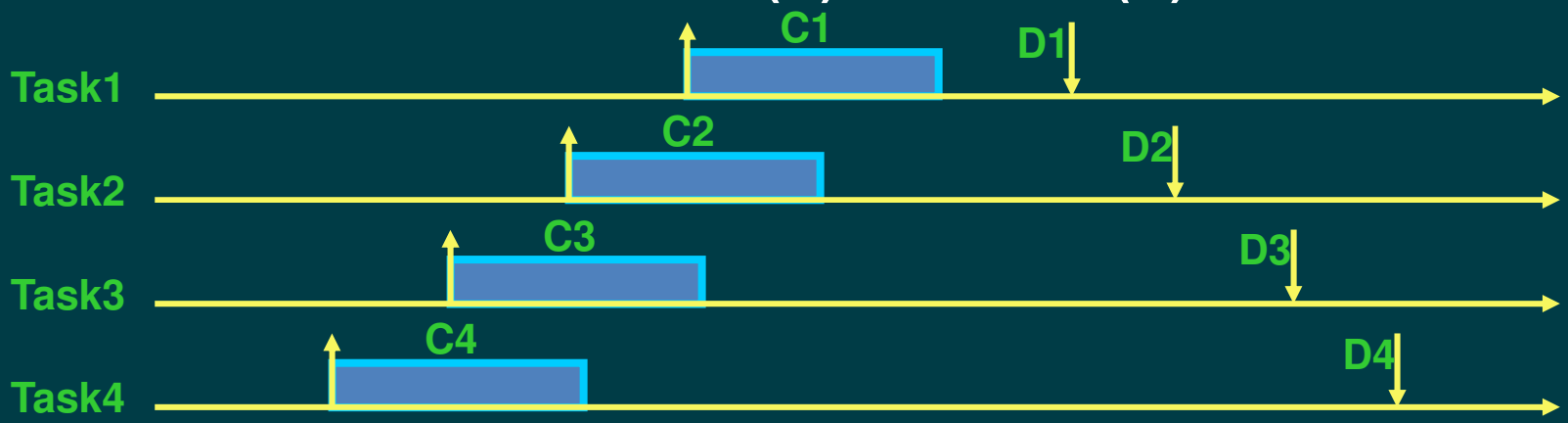
Part 1
Non-Real-Time Scheduling

Non-Real-Time Scheduling

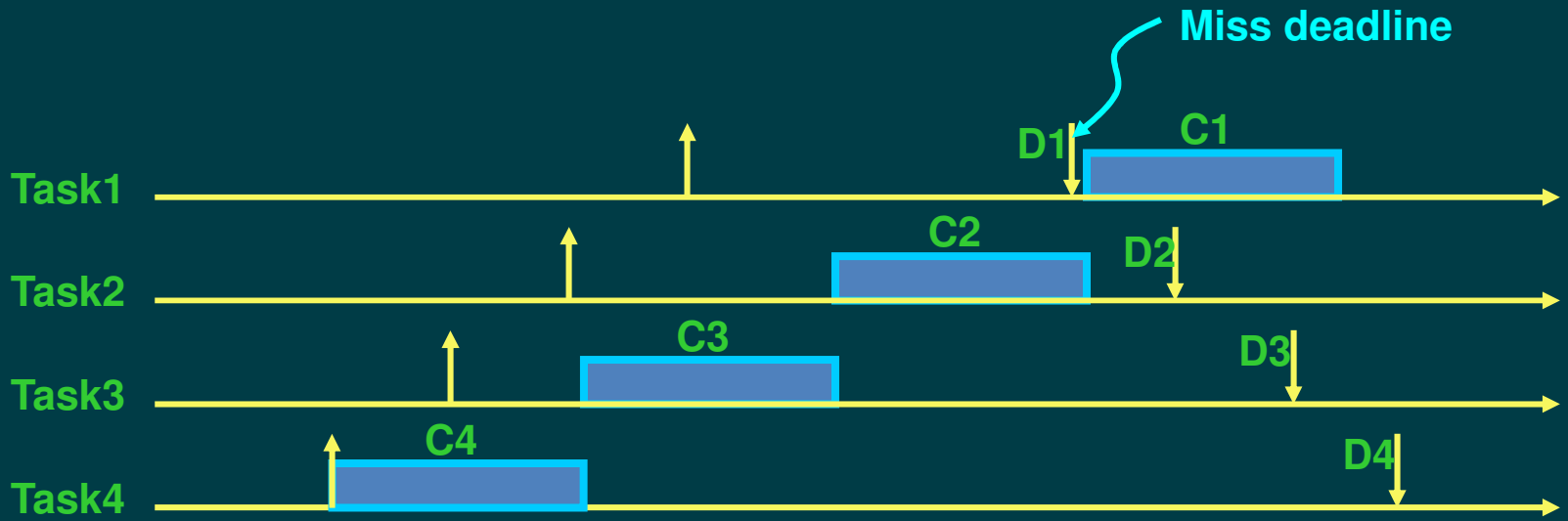
- FCFS (First-Come-First-Served)排程
  - 先到達系統的(ready task)先執行
  - 執行過程，不會被插斷執行(non-preemptive)
- RR (Round-Robin)排程
  - 所有的ready tasks每個輪流執行一小段時間

Non-Real-Time 排程範例

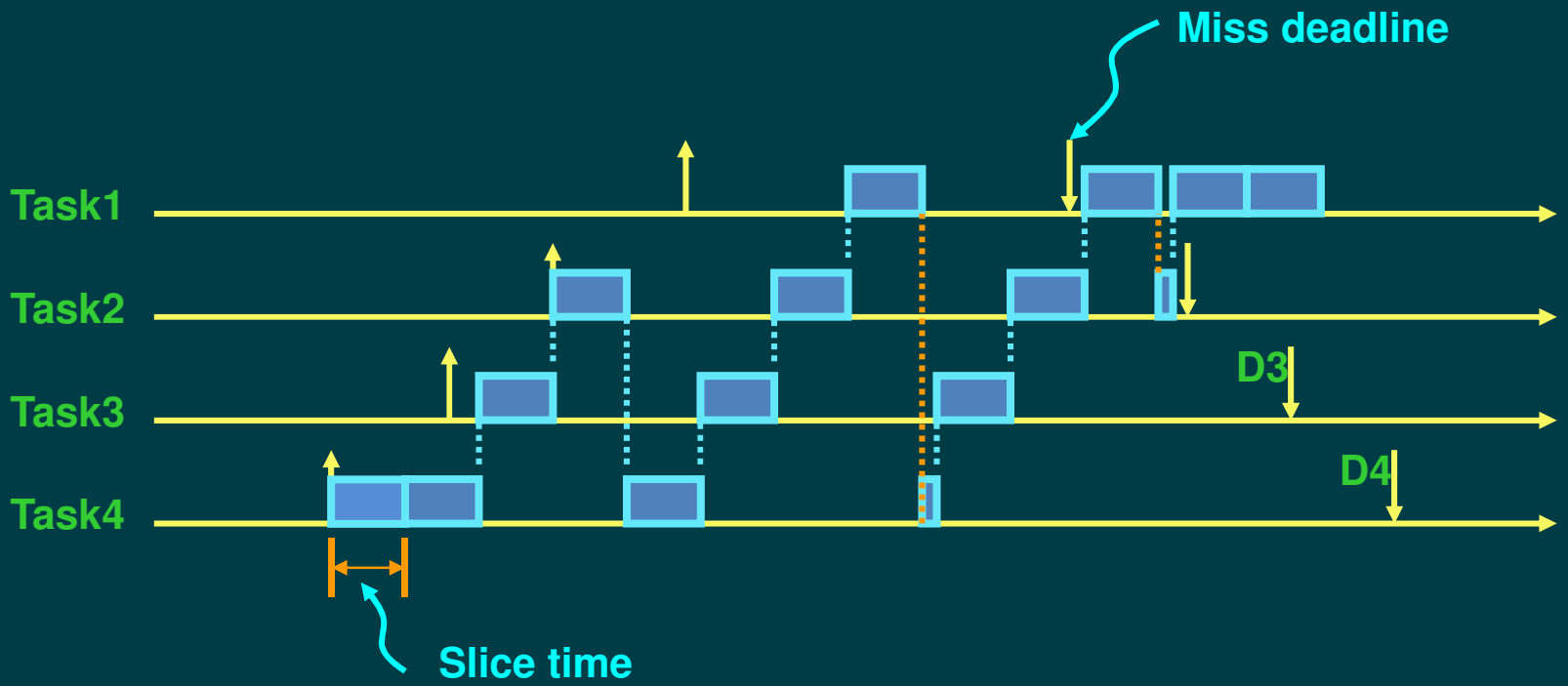
- 假設環境
  - 系統上只有一顆CPU
  - Task ready的時間不固定
  - 每個Task有其deadline (D)與執行時間(C)



FCFS (First-Come-First-Served)排程



RR (Round-Robin)排程

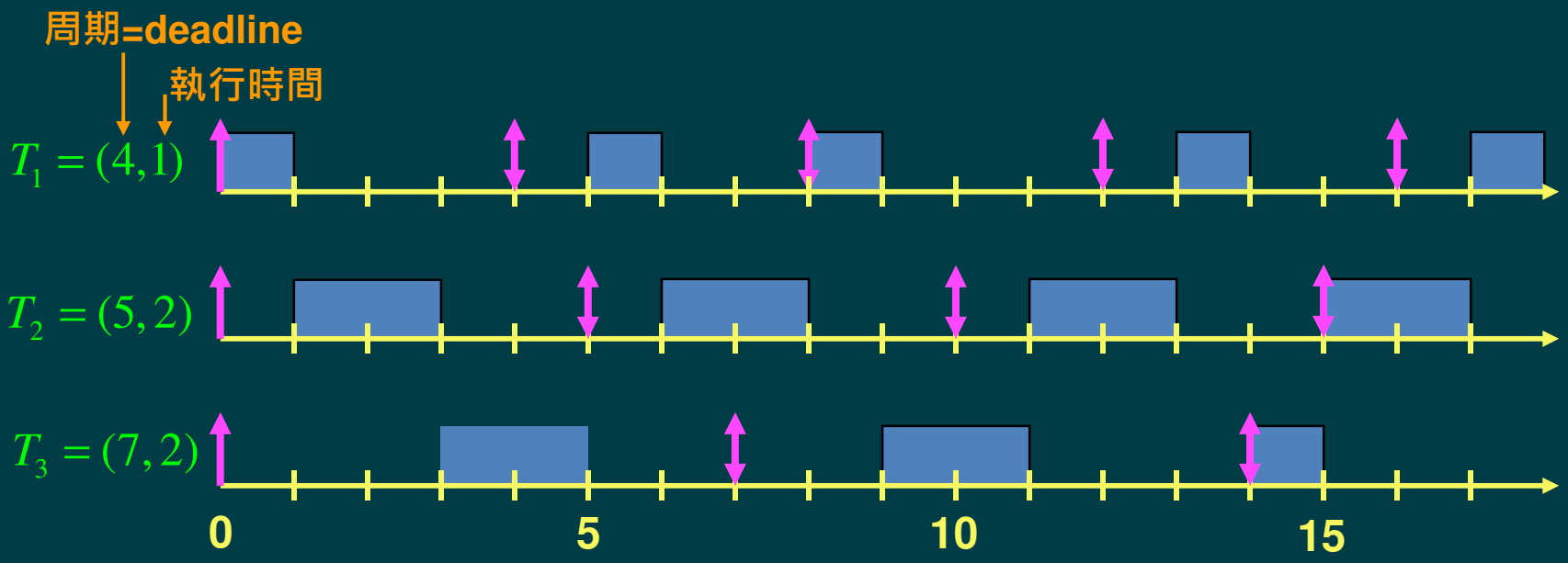


Part 2
Real-Time Scheduling

Task優先權決定

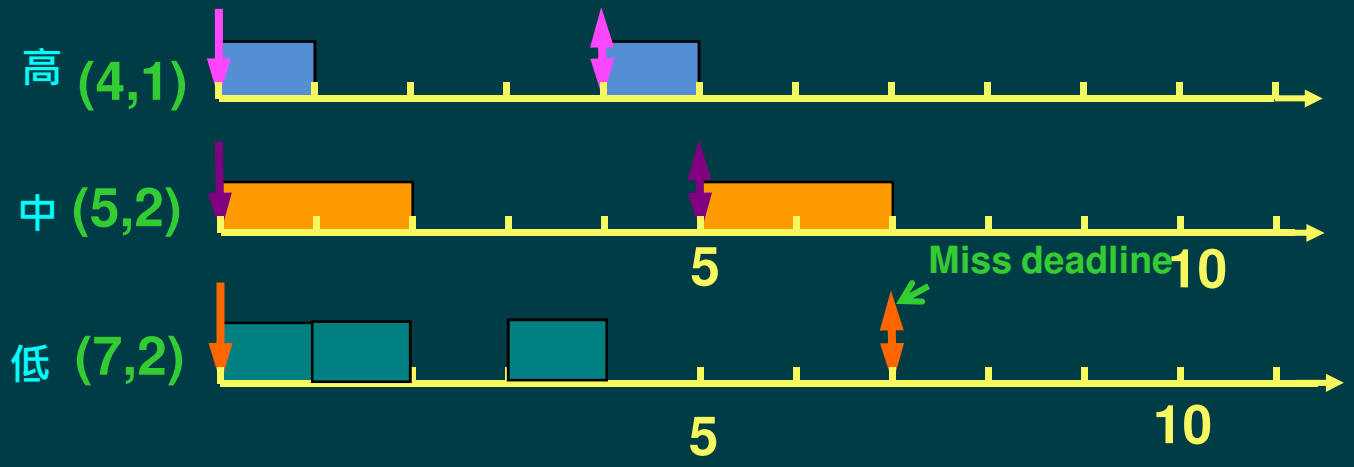
- **Dynamic-priority Algorithm**
  - Task在執行過程，依其演算法，動態調整Task優先權，以滿足系統之要求
  - 如：Earliest Deadline First (EDF)
    - 挑選離deadline時間最短的ready task進系統執行
- **Fixed-priority Algorithm**
  - 在設計過程即已決定各個Task之優先權
  - 如：Rate Monotonic (RM)

Earliest Deadline First (EDF)



Rate Monotonic (RM)
Task Priority Assignment

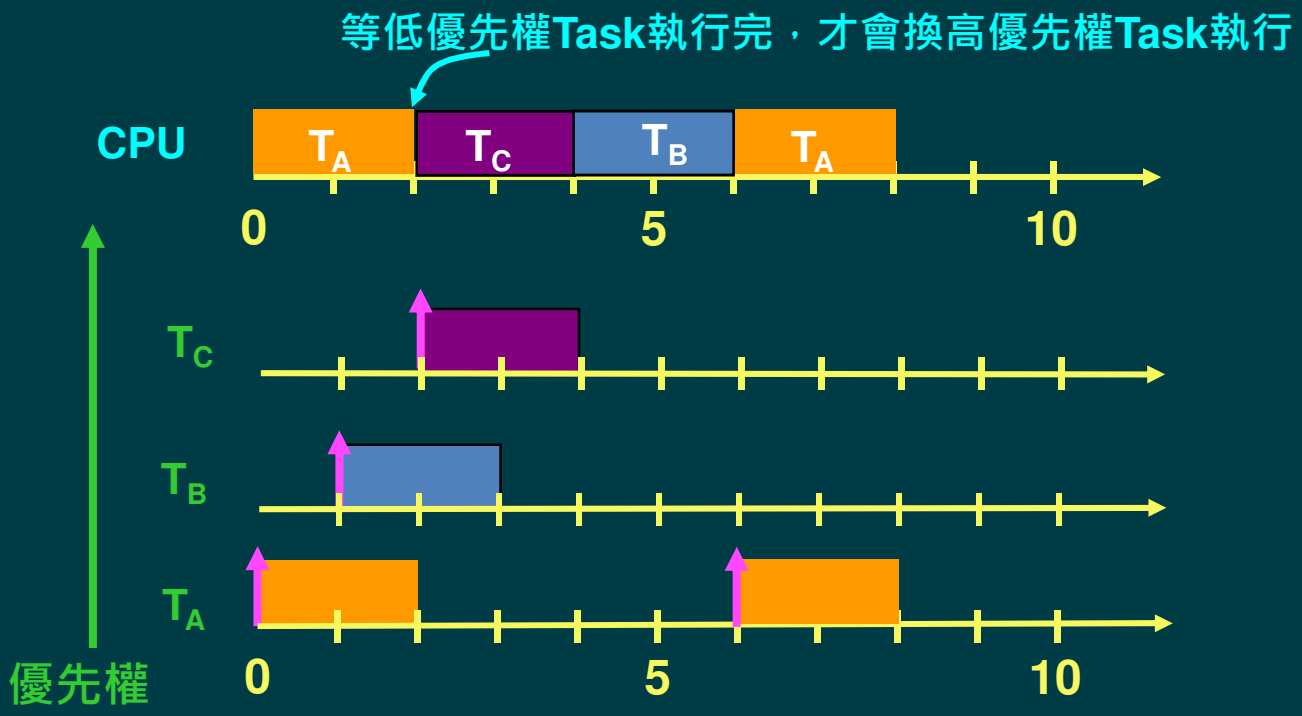
- 最佳之靜態優先權設定方法
- 依據Task之執行周期來設定優先權
- 周期愈短的Task優先權愈高



Priority-based non-preemptive 排程機制

- 又稱Cooperative scheduling
- 依優先權大小來挑選Task進系統執行
- 高優先權的Task具有較優先執行的權力
- 高優先權的Task需等目前在執行的Task放棄CPU使用權後，才可執行
  - 高優先權的Task無法插斷低優先權的Task

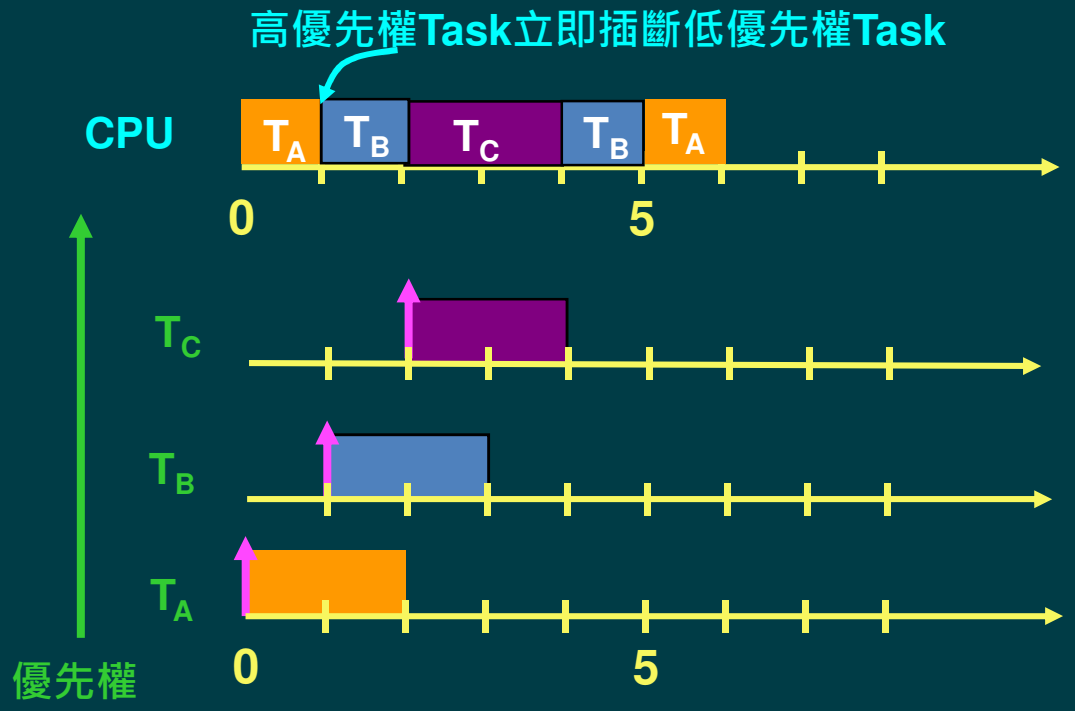
Priority-based non-preemptive 排程機制



Priority-based preemptive 排程機制

- 依優先權大小來挑選Task進系統執行
- 高優先權的Task具有較高優先執行
- 高優先權的Task可立即插斷低優先權的Task
- 系統永遠是執行Ready-list內最高優先權的Task

Priority-based preemptive 排程機制

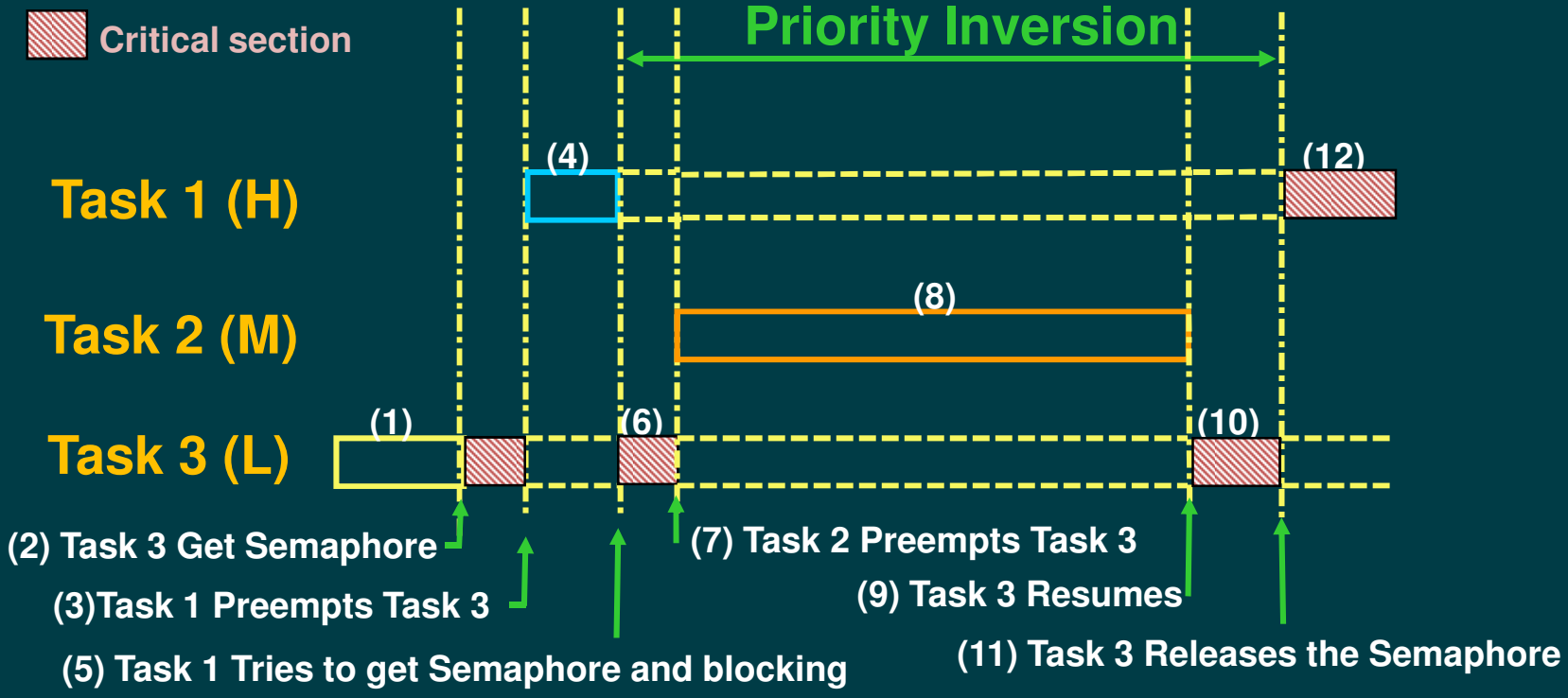


Part 3
Priority Inversion Problem

Priority Inversion Problem

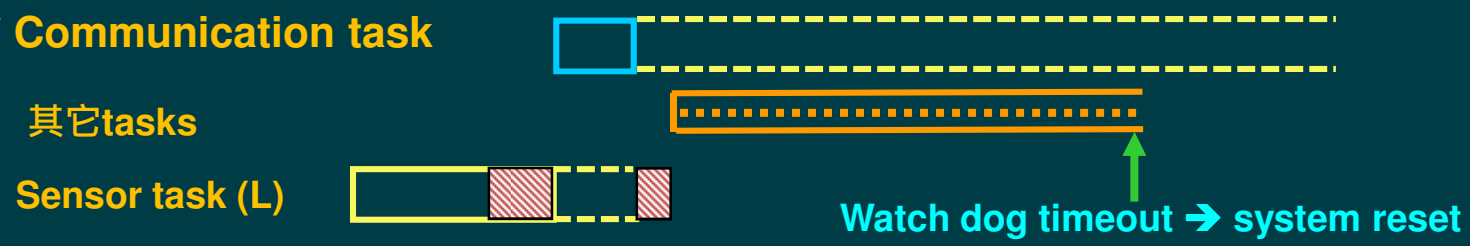
- 在Priority-based preemptive 即時作業環境，在ready tasks中，永遠是挑選最高優先權的task來執行
- 所謂priority inversion problem為低優先權的Task會block高優先權的Task，而且block的時間長度是不確定的
  - 即高優先權的Task可能因此無法即時執行，而 miss deadline，造成嚴重後果
  - 如: Mars Pathfinder (火星探路者號)因priority inversion問題，造成資訊無法回傳至地球

Priority Inversion Problem



Mars Pathfinder

- 低優先權的sensor task先進入critical section
- 較高優先權的communication task也要進入critical section➔被blocking
  - Sensor task被其它高優先權的Task插斷執行
- Communication task一直無法進入critical section
  - 造成watch dog(看問狗) Timeout➔對系統reset
- 造成Mars pathfinder在某些情況即會被reset，無法將資訊回傳至地球



Priority Inversion Problem Solution

- Priority inversion 主要是因mutually exclusive semaphore 保護 critical section 而引起
- Priority inheritance
  - Low priority task先佔用semaphore而進入critical section
  - 若higher priority task也要進入critical section而被blocking，此時low priority task繼承higher priority task的優先權
  - 當low priority task離開critical section時，即還原回原來的優先權

Priority Inheritance Example