Announcement

- Homework assignment 3
  - Real-time scheduling
  - Due on 11/4 before class

ECE 1175
Embedded System Design

Real-Time Scheduling - II

Wei Gao

Comparison

- Schedulability
  - RMS may not guarantee schedulability even when CPU is not fully utilized
  - EDF can guarantee schedulability as long as CPU is not fully utilized
- Overhead
  - RMS: low overhead, priorities are never changed
  - EDF: higher overhead, task priorities may need to be changed online
- Optimality
  - RMS: optimal static priority scheduling algorithm
  - EDF: optimal dynamic priority scheduling algorithm

Relaxing Assumptions

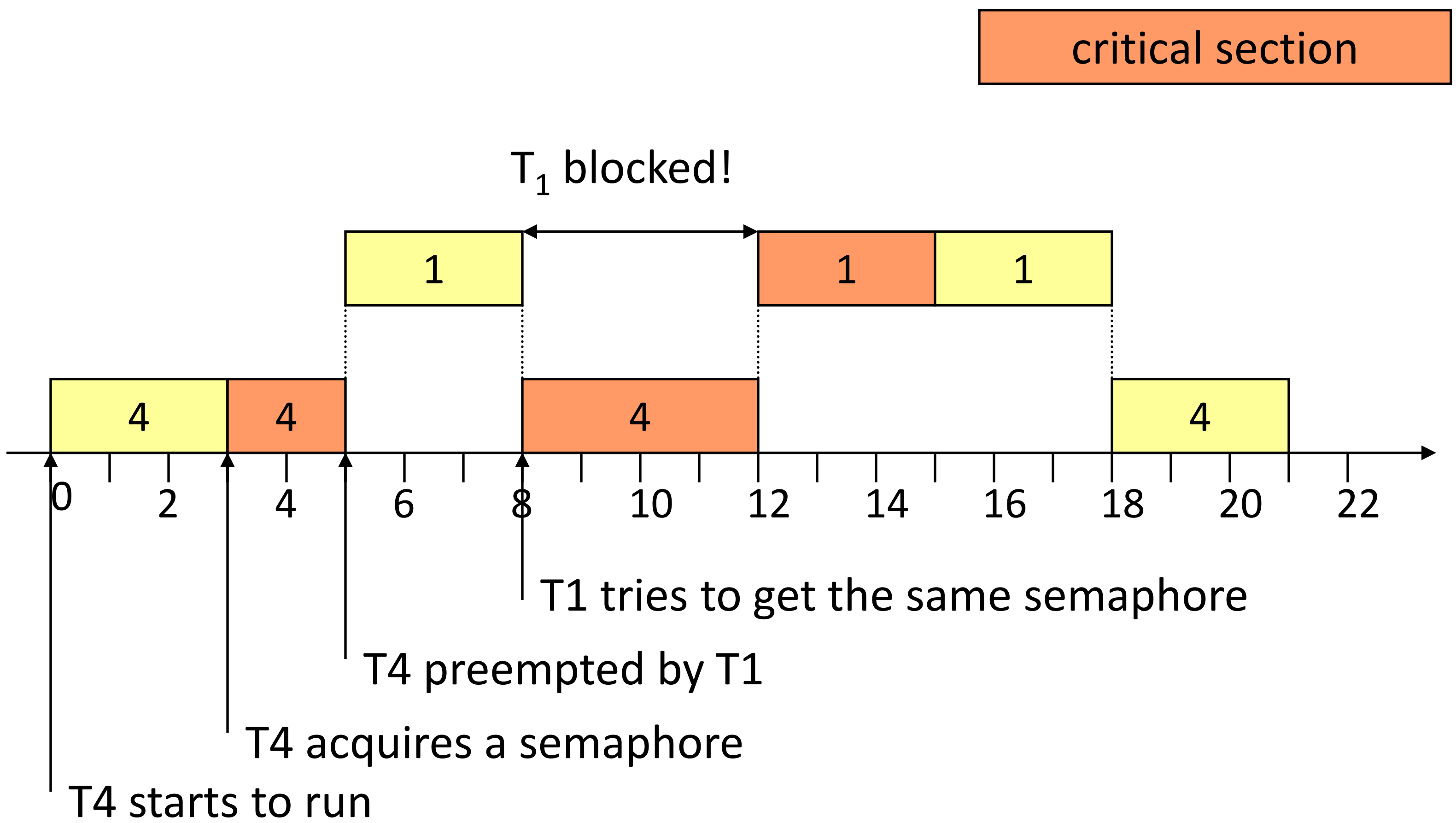
- Single processor.
- All tasks are periodic.
- Zero context switch time.
- Relative deadline = period.
- No priority inversion.

- What if priority inversion exists?

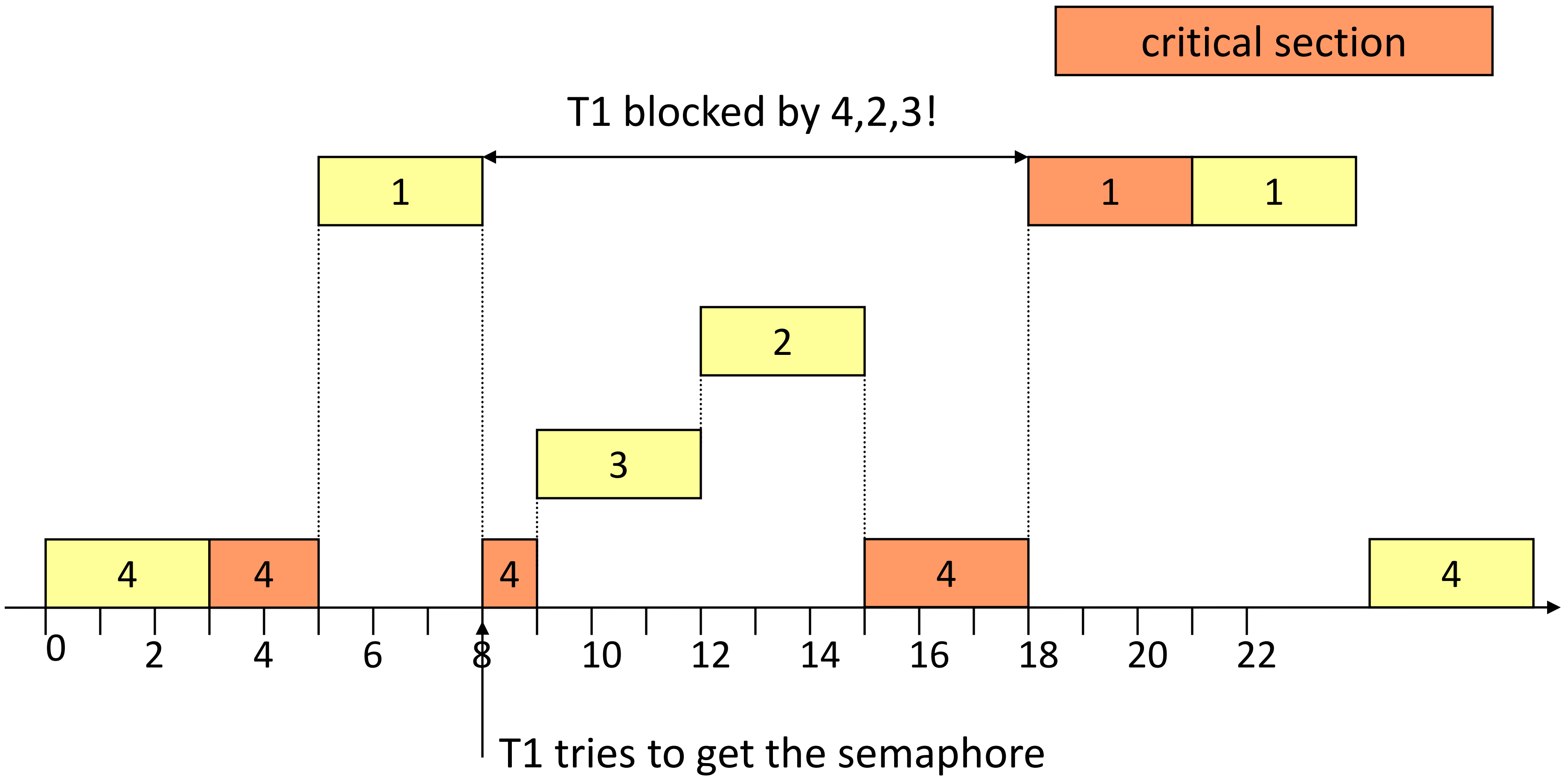
Priority Inversion

- A lower-priority task blocks a higher-priority task from running.
- Sources of priority inversion
  - Access shared resources guarded by semaphores
    - Lower-priority task gets the resource first
  - Access non-preemptive subsystems
    - Communication subsystems
    - Storage

Priority Inversion

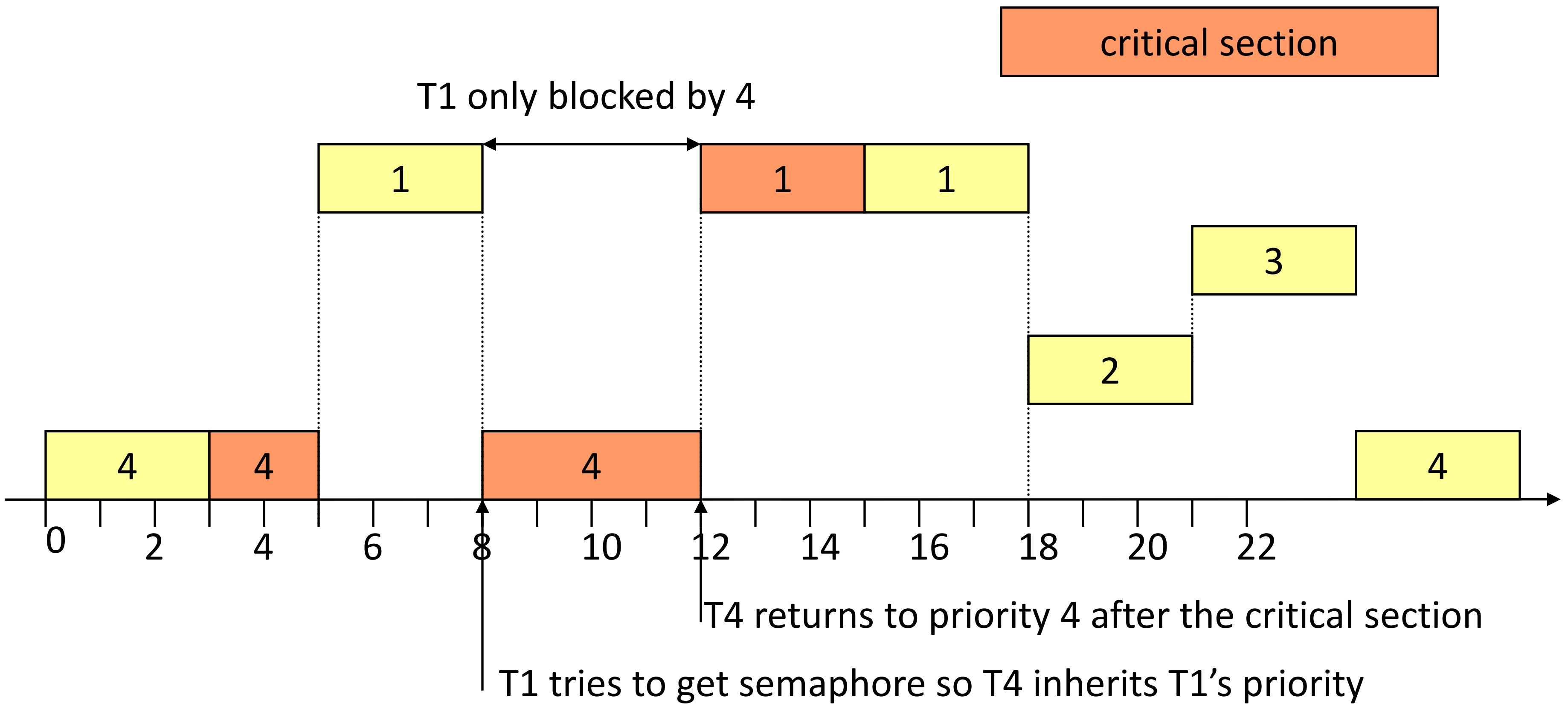


Unbounded Priority Inversion



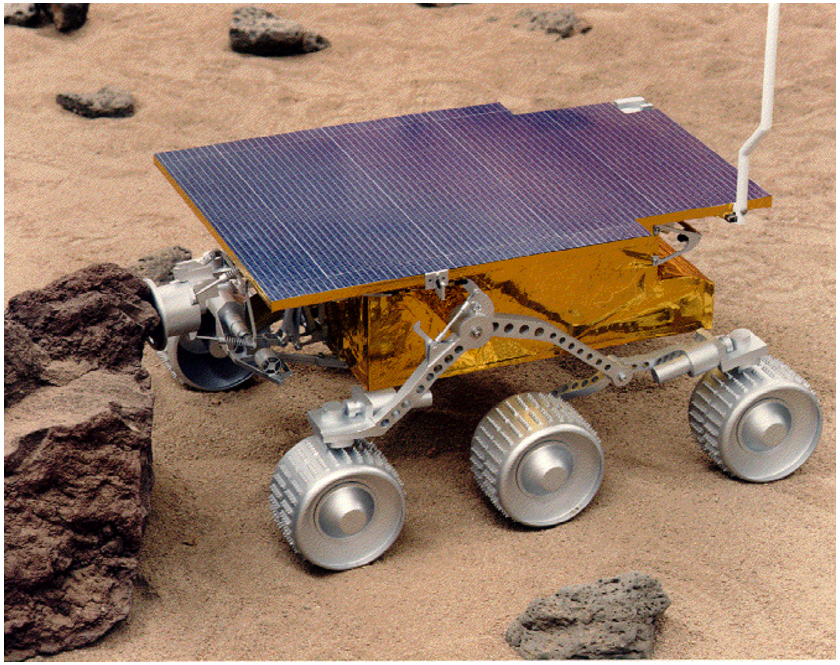
Solution: Priority Inheritance

- Let the low-priority task inherit the priority of the blocked high-priority task.



Real Incident

- Mars Pathfinder
- Priority inversion on Mars
  - http://www.cse.chalmers.se/~risat/Report_MarsPathFinder.pdf
  - https://www.youtube.com/watch?v=lyx7kARrGeM



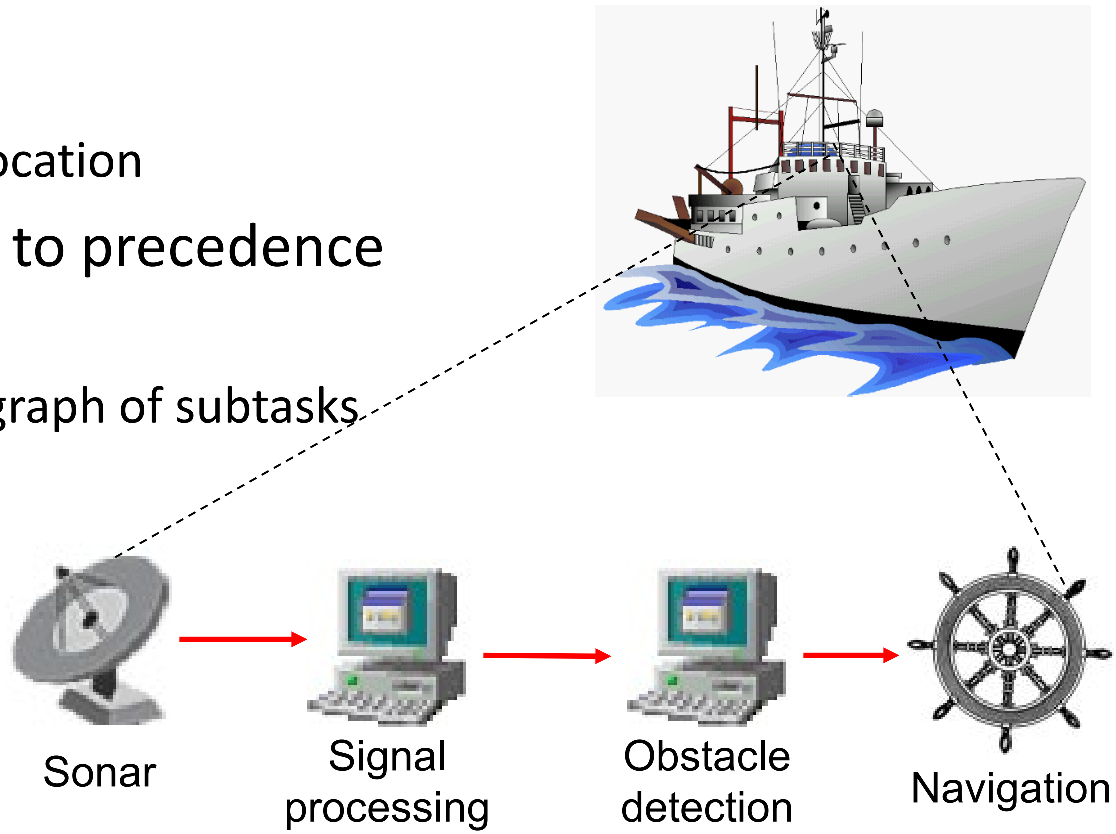
Relaxing Assumptions

- Single processor.
- All tasks are periodic.
- Zero context switch time.
- Relative deadline = period.
- No priority inversion. (relaxed)

- What if we have multiple processors?

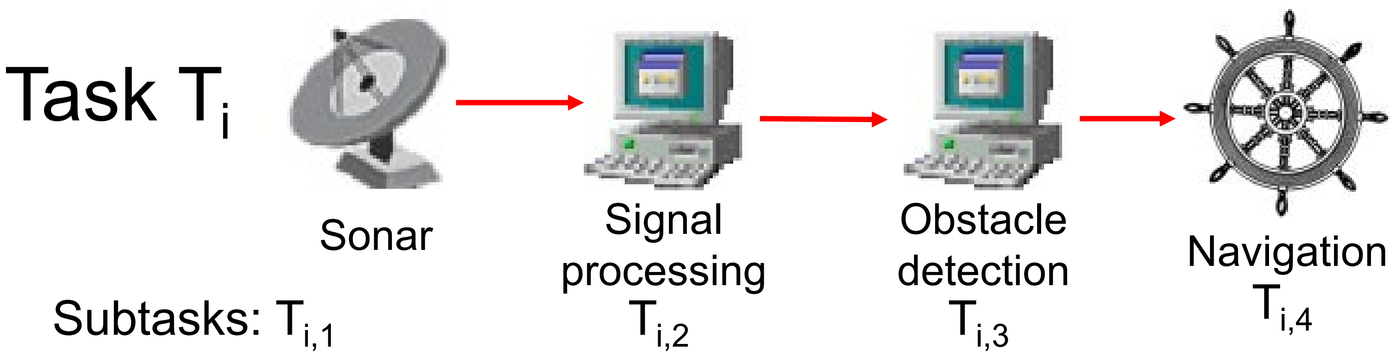
End-to-End Task Model

- An (end-to-end) task is composed of multiple subtasks running on multiple processors
  - Message/event
  - Remote method invocation
- Subtasks are subject to precedence constraints
  - Task = a chain/tree/graph of subtasks
  - E.g. ship navigation



Notation

- $T_i = \{T_{i,1}, T_{i,2}, \dots, T_{i,n(i)}\}$
  - $n(i)$: the number of subtasks of T_i
- Precedence constraint: Job $J_{i,j}$ cannot be released until $J_{i,j-1}$ is completed.



End-to-End Scheduling Framework

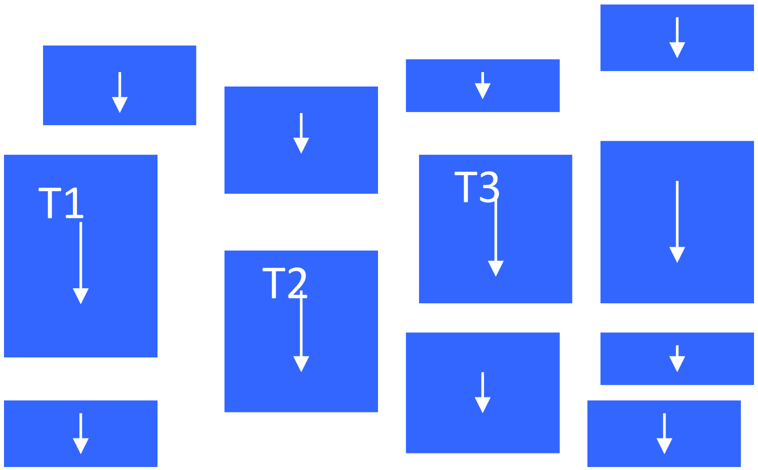
1. **Task allocation**
2. Synchronization protocol
3. Subdeadline assignment
4. Schedulability analysis

Task Allocation

- Load code (e.g., objects) to processors
- Strategies
  - Offline, static allocation subject to resource availability
  - Allocate a task when it arrives dynamically
  - Re-allocate (migrate) a task after it starts
- Optimal solutions for maximum schedulability
  - How to meet all deadlines in an optimal way?
    - E.g., minimize the number of needed processors
  - NP-hard: heuristics needed

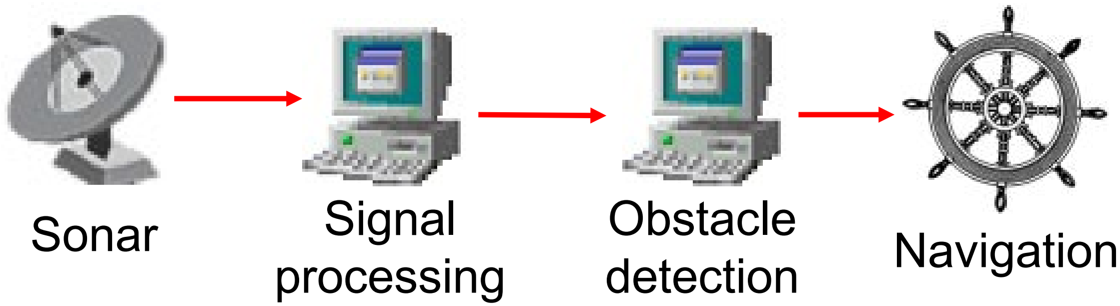
Bin Packing for Task Allocation

- Pack subtasks to bins (processors) with limited capacity
  - Size of a subtask
    - Utilization: $C_{i,j}/P_i$
  - Capacity of each bin is its utilization bound
  - Goal: minimize the number of bins subject to the capacity constraints



End-to-End Scheduling Framework

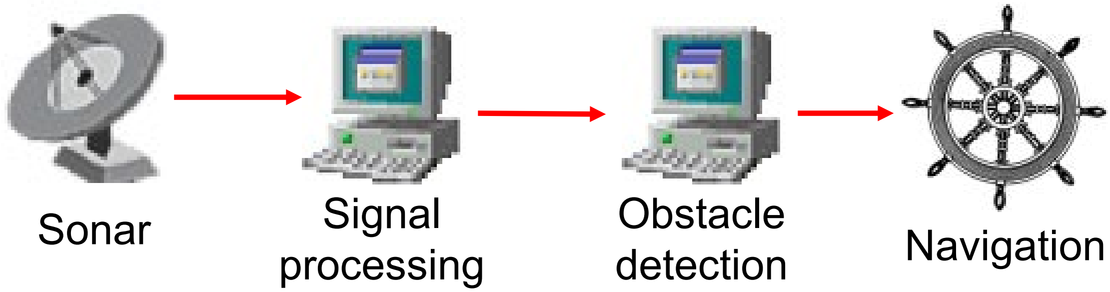
1. Task allocation
2. **Synchronization protocol**
3. Subdeadline assignment
4. Schedulability analysis



After a subtask is finished, should the next subtask start immediately or wait for a while?

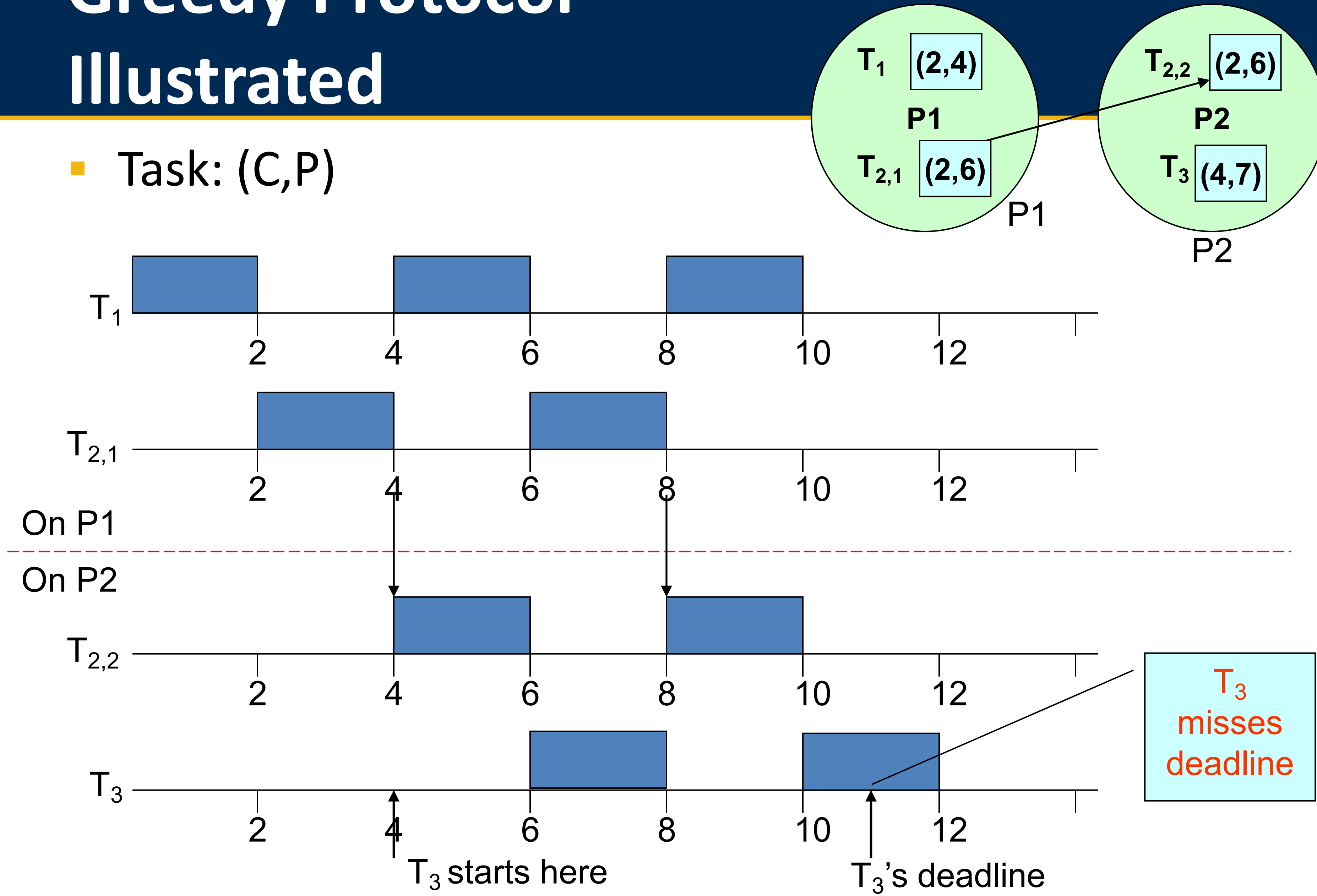
Greedy Protocol

- After a subtask is finished, the next subtask starts immediately
- Release job $J_{i,j;k}$ as soon as $J_{i,j-1;k}$ is completed
- Subsequent subtasks may not be periodic under a greedy protocol
  - Difficult for schedulability analysis
  - High-priority tasks arrive early → high worst-case response time for lower-priority tasks



Greedy Protocol Illustrated

- Task: (C,P)



Properties of Greedy Protocol

- Low overhead
- Low average response time
- Difficult schedulability analysis
  - Subsequent subtasks are no longer periodic
- High worst-case response time

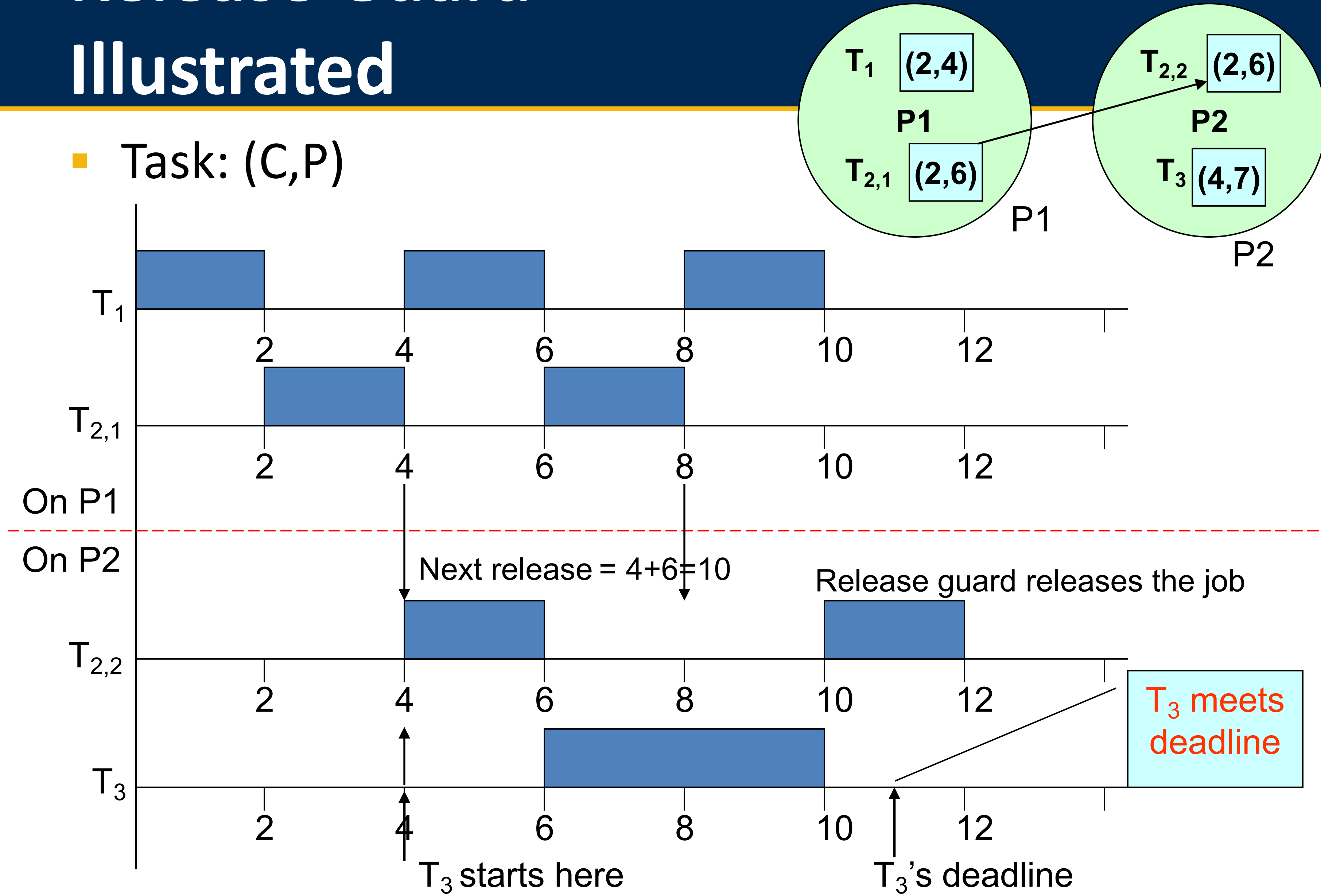
Release Guard

- After a subtask is finished, the next subtask may wait for a while before release
- Every subtask (if not a first subtask) has a release guard, which
  - waits for the preceding subtask for a result/event
  - then releases the job
    - at the point of exact one period from the last release time (Rule1)

    OR
    - whenever the processor becomes idle (Rule 2)
- Release guard strategy improves worst response time without affecting schedulability

Release Guard Illustrated

- Task: (C,P)



End-to-End Scheduling Framework

1. Task allocation
2. Synchronization protocol
3. **Subdeadline assignment**
4. Schedulability analysis

Subdeadline Assignment Algorithms

- Notation
  - (Relative) deadline D_i of task T_i
  - (Relative) subdeadline D_{ij} of subtask T_{ij} $(1 \leq j \leq n(i))$

- Ultimate Deadline (UD): $D_{ij} = D_i$
  - Example
    - An end-to-end task T1, with deadline as 12, has 4 subtasks: T11, T12, T13 and T14 with execution times as: 3, 1, 1, 1.
    - D11 = D12 = D13 = D14 = D1 = 12
  - But T11, T12, T13 must finish earlier than the end-to-end deadline such that T14 can have time to run.

Common Assignment Algorithms

- Proportional Deadline (PD):
  - Assign deadline proportionally to execution time

$$D_{ij} = D_i \frac{C_{ij}}{\sum_{k=1}^{n(i)} C_{ik}}$$

  - Example
    - An end-to-end Task T1, with deadline as 12, has 4 subtasks T11, T12, T13 and T14 with execution time as: 3, 1, 1, 1.
    - D11 = 12 * 3/(3+1+1+1) = 6
    - D12 = 12 * 1/(3+1+1+1) = 2
    - D13 = 12 * 1/(3+1+1+1) = 2
    - D13 = 12 * 1/(3+1+1+1) = 2

End-to-End Scheduling Framework

1. Task allocation
2. Synchronization protocol
3. Subdeadline assignment
4. **Schedulability analysis**
    - Decide an appropriate scheduling algorithm
        - RMS, EDF, etc
    - For each processor, conduct uniprocessor schedulability analysis
        - Then synchronize among multiple processors

Summary

- End-to-end scheduling framework
  - Task allocation
    - Bin packing
  - Synchronization protocol
    - Greedy protocol, release guard
  - Subdeadline assignment
    - Ultimate deadline, proportional deadline
  - Schedulability analysis

Reading

- Priority inversion on Mars
  - http://www.cse.chalmers.se/~risat/Report_MarsPathFinder.pdf
  - https://www.youtube.com/watch?v=lyx7kARrGeM