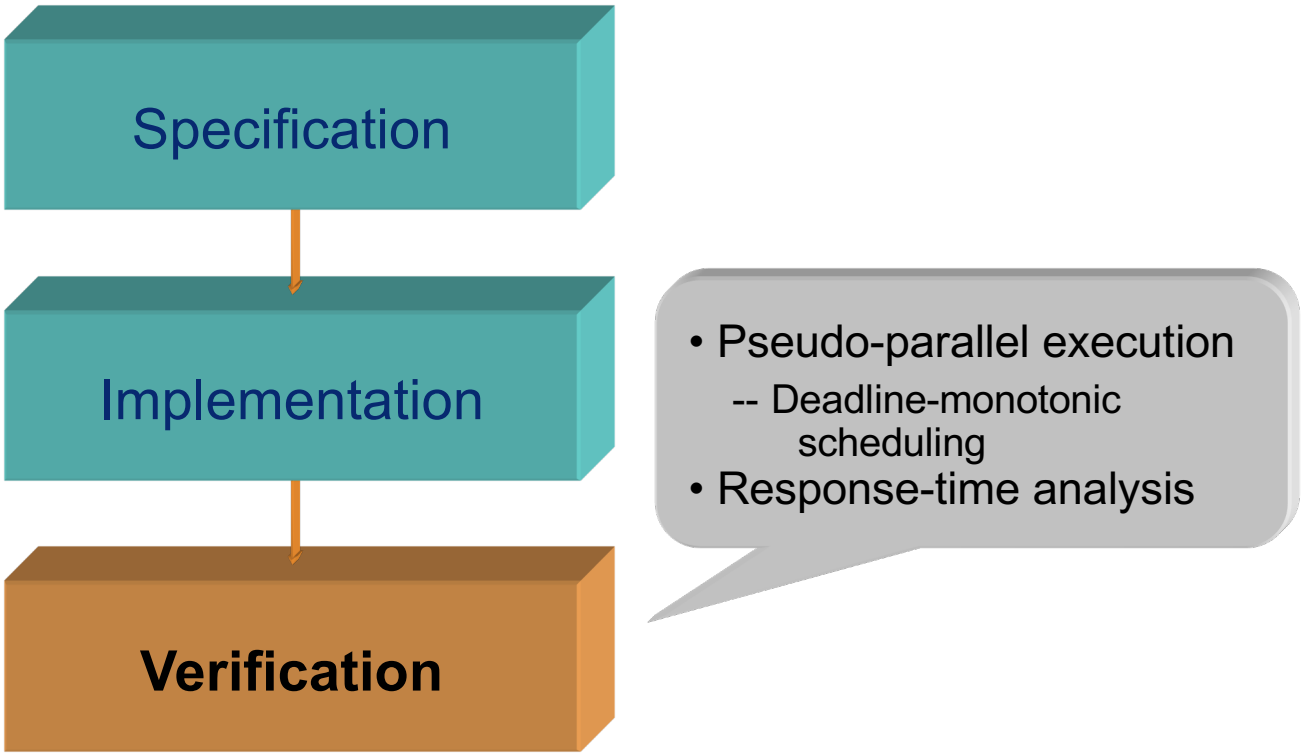
Real-Time Systems

Lecture #12

Professor Jan Jonsson

Department of Computer Science and Engineering
Chalmers University of Technology

Real-Time Systems



Example: scheduling using RM

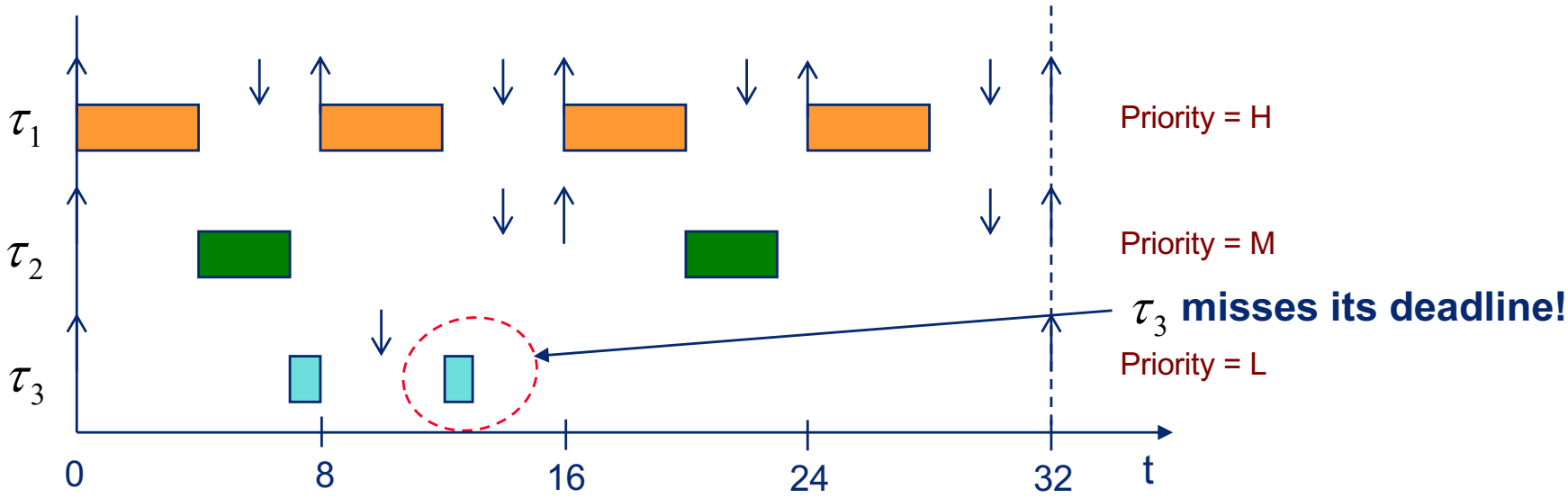
Problem: Assume a system with tasks according to the figure below. The timing properties of the tasks are given in the table. All tasks arrive the first time at time 0.

Investigate the schedulability of the tasks when RM is used. (Note that $D_i < T_i$ for all tasks)

Task	C_i	D_i	T_i
τ_1	4	6	8
τ_2	3	14	16
τ_3	2	10	32

Example: scheduling using RM

Simulate an execution of the tasks using RM:



The tasks are not schedulable even though

$$U = \frac{4}{8} + \frac{3}{16} + \frac{2}{32} = \frac{24}{32} = 0.75 < U_{RM} = 3\left(2^{1/3} - 1\right) \approx 0.780$$

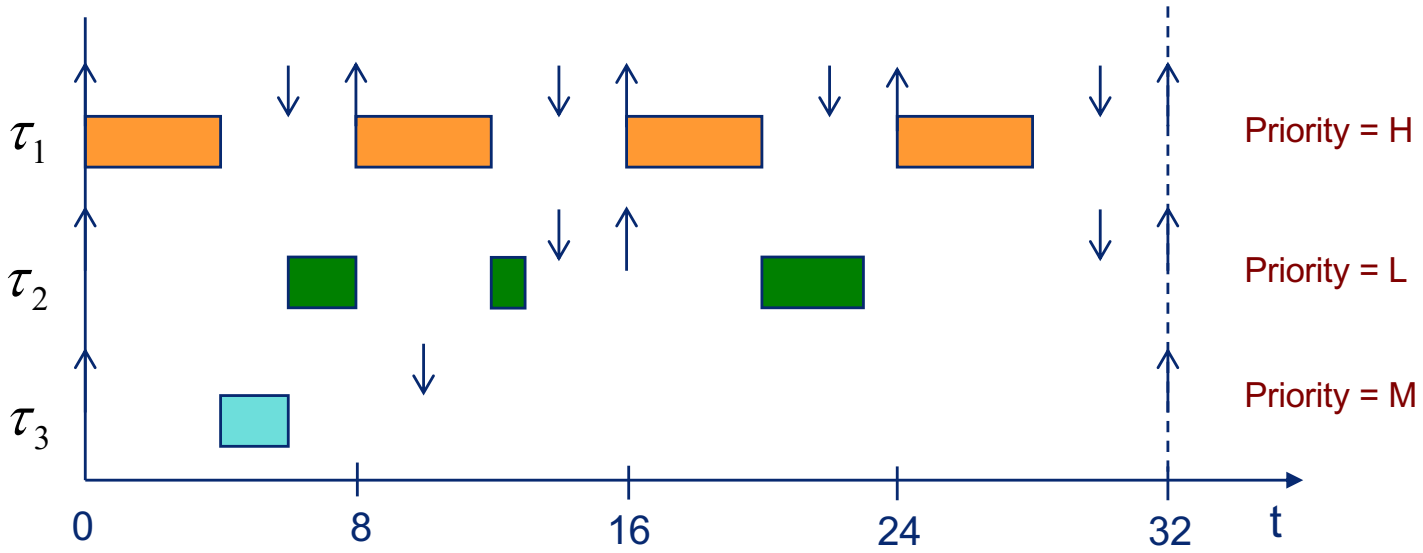
Deadline-monotonic scheduling

Properties:

- Uses static priorities
  - Priority is determined by urgency: the task with the shortest relative deadline receives highest priority
  - Proposed as a generalization of rate-monotonic scheduling (RM is a special case of DM, with $D_i = T_i$)
- Theoretically well-established
  - Exact feasibility test is an NP-complete problem (pseudo-polynomial time with response-time analysis)
  - DM is optimal among all scheduling algorithms that use static task priorities for constrained-deadline tasks (with $D_i \leq T_i$) (shown by J. Leung and J. W. Whitehead in 1982)

Example: scheduling using DM

Simulate an execution of the task set given earlier using DM:



All tasks now meet their deadlines!

Feasibility tests

What types of feasibility tests exist?

- Hyper period analysis (for any type of scheduler)
  - In an existing schedule no task execution may miss its deadline
- Processor utilization analysis (static/dynamic priority scheduling)
  - The fraction of processor time that is used for executing the task set must not exceed a given bound
- Response time analysis (static priority scheduling)
  - The worst-case response time for each task must not exceed the deadline of the task
- Processor demand analysis (dynamic priority scheduling)
  - The accumulated computation demand for the task set under a given time interval must not exceed the length of the interval

Response-time analysis

The response time R_i for a task τ_i represents the worst-case completion time of the task when execution interference from other tasks are accounted for.

The response time for a task τ_i consists of:

- C_i The task's uninterrupted execution time (WCET)
- I_i Interference from higher-priority tasks

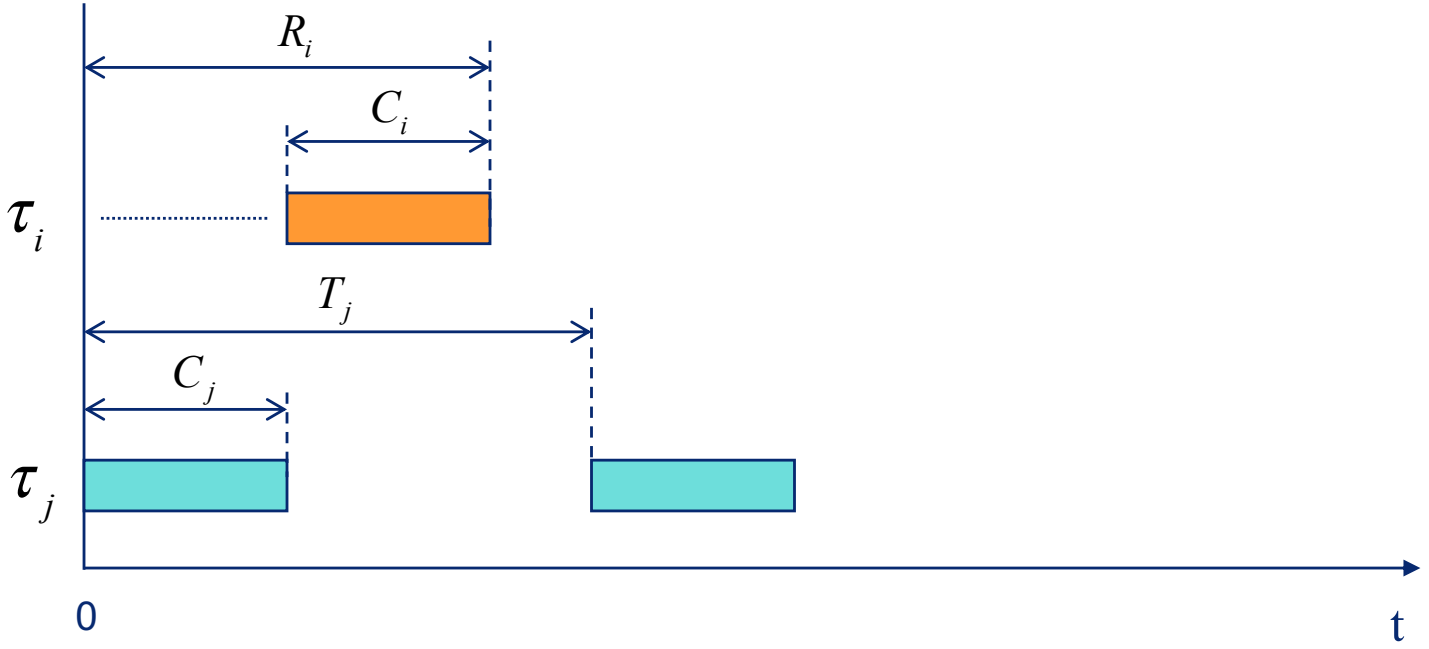
$$R_i = C_i + I_i$$

Response-time analysis

Interference:

Consider two tasks, τ_i and τ_j, where τ_j has higher priority

Case 1: $0 < R_i \leq T_j \Rightarrow R_i = C_i + C_j$

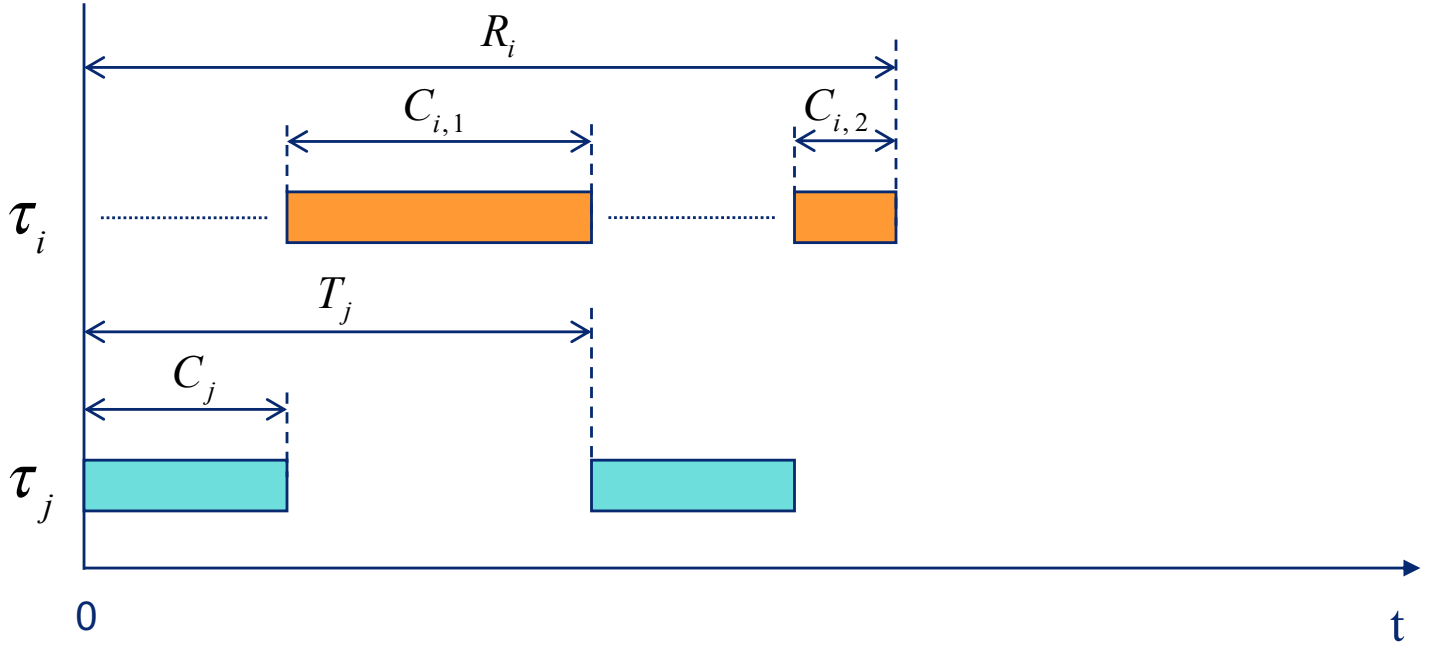


Response-time analysis

Interference:

Consider two tasks, τ_i and τ_j, where τ_j has higher priority

Case 2: $T_j < R_i \leq 2T_j \Rightarrow R_i = C_i + 2C_j$



Response-time analysis

Interference:

When task τ_i is preempted by higher-priority task τ_j:

The response time for τ_i is at most R_i time units.

If $0 < R_i \leq T_j$, task τ_i can be preempted at most one time by τ_j

If $T_j < R_i \leq 2T_j$, task τ_i can be preempted at most two times by τ_j

If $2T_j < R_i \leq 3T_j$, task τ_i can be preempted at most three times by τ_j

...

The number of interferences from τ_j is thus limited by: $\left\lceil \frac{R_i}{T_j} \right\rceil$

The total time for these interferences are: $\left\lceil \frac{R_i}{T_j} \right\rceil C_j$

Response-time analysis

Interference:

- For static-priority scheduling, the interference term is

$$I_i = \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where $hp(i)$ is the set of tasks with higher priority than τ_i.

- The response time for a task τ_i is thus:

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Response-time analysis

Interference:

- The equation does not have a simple analytic solution.
- However, an iterative procedure can be used:

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

- The iteration starts with a value that is guaranteed to be less than or equal to the final value of R_i (e.g. $R_i^0 = C_i$)
- The iteration completes at convergence ($R_i^{n+1} = R_i^n$) or if the response time exceeds some threshold (e.g. D_i)

Exact feasibility test for DM

(Sufficient and necessary condition)

A sufficient and necessary condition for DM scheduling of synchronous task sets, for which $D_i \leq T_i$, is

$$\forall i: R_i \leq D_i$$

where R_i is the worst-case response time for task τ_i

In other words: *for the task set to be schedulable with DM there must not exist an instance of a task execution in the schedule where the worst-case response time of the task exceeds its deadline.*

The response-time analysis and associated feasibility test was presented by M. Joseph and P. Pandya in 1986.

Exact feasibility test for DM

(Sufficient and necessary condition)

The test is valid under the following assumptions:

1. All tasks are independent
   - There must not exist dependencies due to precedence or mutual exclusion
2. All tasks are periodic or sporadic
3. All tasks have identical offsets (= synchronous task set)
4. Task deadline does not exceed the period
   (= constrained-deadline tasks)
5. Task preemptions are allowed

Example 1: scheduling using DM

Problem: We once again assume the system with tasks given in the beginning of this lecture.

Show, by using response-time analysis, that the tasks are schedulable using DM.

Task	C_i	D_i	T_i
τ_1	4	6	8
τ_2	3	14	16
τ_3	2	10	32

Extended response-time analysis

The test can be extended to handle:

- Blocking
- Start-time variations (”release jitter”)
- Time offsets (asynchronous task sets)
- Deadlines exceeding the period
- Overhead due to context switches, timers, interrupts, …

In this course, we only show how blocking is handled.

Extended response-time analysis

Blocking can be accounted for in the following cases:

- Blocking caused by critical regions
  - Blocking factor B_i represents the length of critical region(s) that are executed by tasks with lower priority than τ_i
- Blocking caused by non-preemptive scheduling
  - Blocking factor B_i represents largest WCET (not counting τ_i)

$$R_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Recollection from an earlier lecture

Priority Ceiling Protocol:

- Basic idea:

  Each resource is assigned a priority ceiling equal to the priority of the highest-priority task that can lock it.

  Then, a task τ_i is allowed to enter a critical region only if its priority is higher than all priority ceilings of the resources currently locked by tasks other than τ_i.

  When a task τ_i blocks one or more higher-priority tasks, it temporarily inherits the highest priority of the blocked tasks.

Recollection from an earlier lecture

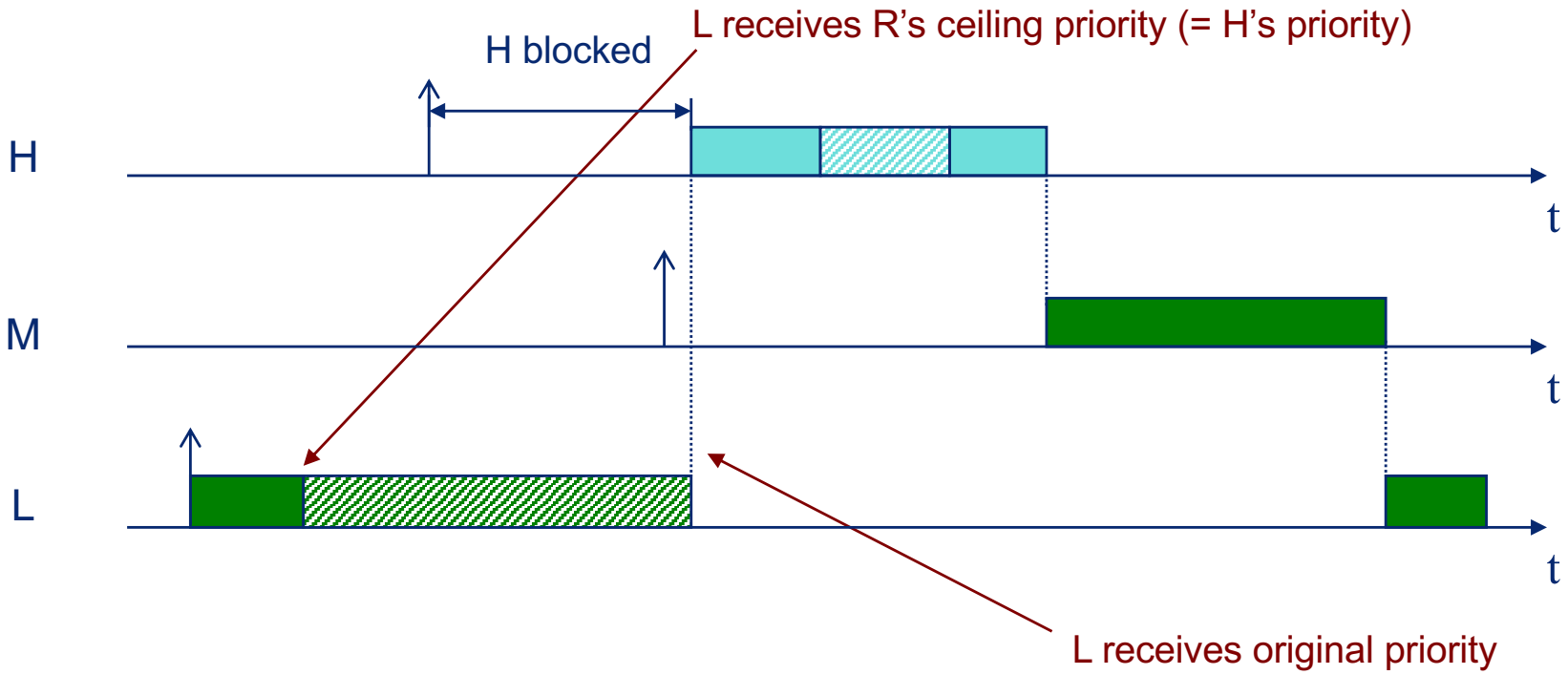
Blocking using ceiling priority protocol ICPP:

priority (H) > priority (M) > priority (L)

H and L share mutex resource R

normal execution

critical region



Extended response-time analysis

Blocking caused by lower-priority tasks:

- When using a priority ceiling protocol (such as ICPP), a task τ_i can only be blocked once by a task with lower priority than τ_i.
- This occurs if the lower-priority task is within a critical region when τ_i arrives, and the critical region's ceiling priority is higher than or equal to the priority of τ_i.
- Blocking now means that the start time of τ_i is delayed (= the blocking factor B_i)
- As soon as τ_i has started its execution, it cannot be blocked by a lower-priority task.

Extended response-time analysis

Determining the blocking factor for task τ_i:

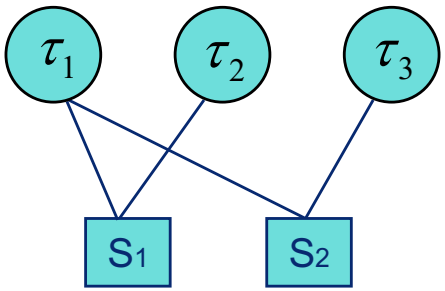
1. Determine the ceiling priorities for all critical regions.
2. Identify the tasks that have a priority lower than τ_i and that calls critical regions with a ceiling priority equal to or higher than the priority of τ_i.
3. Consider the times that these tasks lock the actual critical regions. The longest of those times constitutes the blocking factor B_i.

Example 2: scheduling using DM

Problem: Assume a system with three tasks using two resources, according to the figure below. The timing properties of the tasks are given in the table. Note that $D_i \leq T_i$.

Two semaphores, S_1 and S_2, are used for protecting the resources.

The parameters H_{S1} and H_{S2} represent the longest time a task may lock semaphore S_1 and S_2, respectively.



Task	C_i	D_i	T_i	H_{S1}	H_{S2}
τ_1	2	4	5	1	1
τ_2	3	12	12	1	-
τ_3	8	24	25	-	2

Example 2: scheduling using DM

Problem: (cont'd)

Examine the schedulability of the tasks when ICPP (Immediate Ceiling Priority Protocol) is used.

a) Derive the ceiling priorities of the semaphores.

b) Derive the blocking factors for the tasks.

c) Determine whether the tasks are schedulable or not using DM.