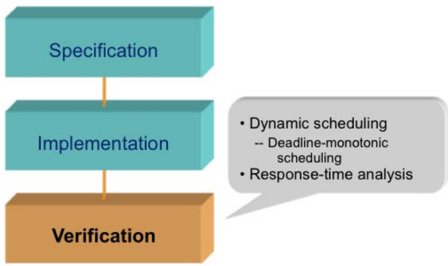
CHALMERS

Real-Time Systems



CHALMERS

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 (J. Leung and J. W. Whitehead, 1982)
    Note that RM is a special case of DM, with $D_i = T_i$
- Theoretically well-established
  - Exact feasibility test exists (an NP-complete problem)
  - DM is optimal among all scheduling algorithms that use static task priorities for which $D_i \le T_i$
    (shown by J. Leung and J. W. Whitehead in 1982)

CHALMERS

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

$$R_i = C_i + I_i$$

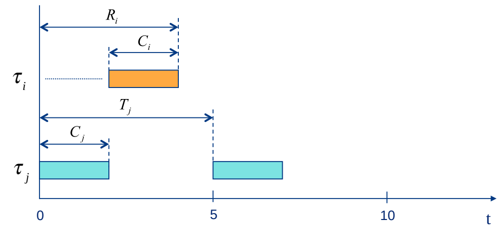
CHALMERS

Response-time analysis

Interference:

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

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



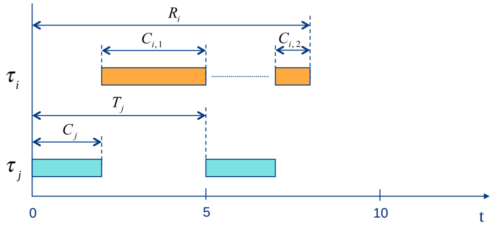
CHALMERS

Response-time analysis

Interference:

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

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



CHALMERS

Response-time analysis

Interference:

Task τ_i can be 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 limited by: $\left\lceil \frac{R_i}{T_j} \right\rceil$

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

CHALMERS

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:

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

CHALMERS

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)

CHALMERS

Exact feasibility test for DM

(Sufficient and necessary condition)

A sufficient and necessary condition for deadline-monotonic scheduling, for which $D_i \leq T_i$, is

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

where R_i is the response time for task τ_i

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

CHALMERS

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.
3. Task deadline does not exceed the period ($D_i \leq T_i$).
4. Task preemptions are allowed.

CHALMERS

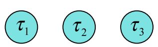
Example: scheduling using DM

Problem: Assume a system with tasks according to the figure below. The timing properties of the tasks are given in the table.

a) Calculate the task response times.

b) Show that the tasks are schedulable using DM

c) What is the outcome of Liu & Layland's feasibility test for RM?



Task	C_i	D_i	T_i
τ_1	12	52	52
τ_2	10	40	40
τ_3	10	30	30

We solve this on the blackboard!

CHALMERS

Extended response-time analysis

The test can be extended to handle:

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

In this course, we only study blocking.

CHALMERS

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 processes with lower priority than τ_i
- Blocking caused by non-preemptive scheduling
  - Blocking factor B_i represents largest WCET (not counting τ_i)

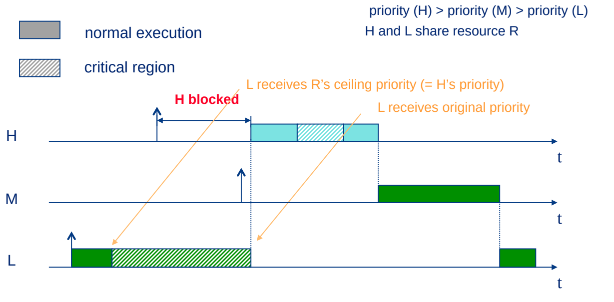
$$R_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Note that the feasibility test is now only sufficient since the worst-case blocking will not always occur at run-time.

CHALMERS

Extended response-time analysis

Blocking using ceiling priority protocol ICPP:



CHALMERS

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.

CHALMERS

Extended response-time analysis

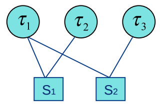
Determining the blocking factor for task τ_i:

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 .

CHALMERS

Example: scheduling using DM

Problem: Assume a system with tasks according to the figure below.
The timing properties of the tasks are given in the table.
Two semaphores S_1 and S_2 are used for synchronizing the tasks.
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

CHALMERS

Example: 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) Show whether the tasks are schedulable or not.

We solve this on the blackboard!