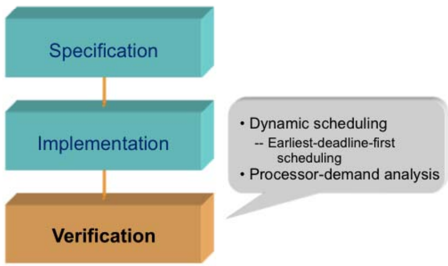
CHALMERS

# Real-Time Systems




CHALMERS

## Example: scheduling using EDF

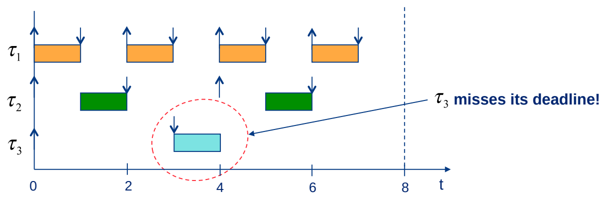
Problem: Assume a system with tasks according to the figure below. The timing properties of the tasks are given in the table. Investigate the schedulability of the tasks when EDF is used. (Note that $D_i < T_i$ for all tasks)


| Task | $C_i$ | $D_i$ | $T_i$ |
|---|---|---|---|
| $\tau_1$ | 1 | 1 | 2 |
| $\tau_2$ | 1 | 2 | 4 |
| $\tau_3$ | 1 | 3 | 8 |

CHALMERS

## Example: scheduling using EDF

Simulate an execution of the tasks:




The tasks are not schedulable even though

$$U = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8} = 0.875 < 1$$

CHALMERS

## Feasibility analysis for EDF

What analysis methods are suitable for general EDF:

- Utilization-based analysis?
  Not suitable! Not general enough or exact enough
  - Does not work well for the case of $D_i < T_i$
- Response-time analysis?
  Not suitable! Analysis much more complex than for DM
  - Critical instant does not necessarily occur when all tasks arrive at the same time for the first time.
  - Instead, response time of a task is maximized at <u>some</u> scenario where all other tasks arrive at the same time; the worst such scenario has to be identified <u>for each task</u> before the response time of that task can be calculated.

CHALMERS

# Processor-demand analysis

Processor demand:

- The processor demand for a task $\tau_i$ in a given time interval $[0, L]$ is the amount of processor time that the task needs in the interval in order to meet the deadlines that fall within the interval.
- Let $N_i^L$ represent the number of instances of $\tau_i$ that must complete execution before $L$.
- The total processor demand up to $L$ is

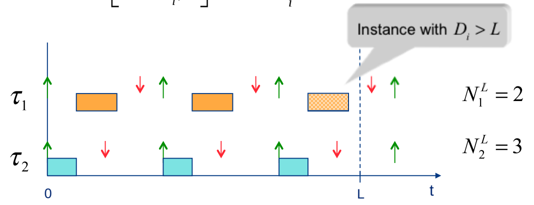
$$C_P(0,L) = \sum_{i=1}^{n} N_i^L C_i$$

CHALMERS

# Processor-demand analysis

Processor demand:

- We can calculate $N_i^L$ by counting how many times task $\tau_i$ has arrived during the interval $[0, L - D_i]$.
- We can ignore instances of the task that arrived during the interval $[L - D_i, L]$ since $D_i > L$ for these instances.




CHALMERS

# Processor-demand analysis

Processor-demand analysis:

- We can express $N_i^L$ as

$$N_i^L = \left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1$$

- The total processor demand is thus

$$C_P(0,L) = \sum_{i=1}^{n} \left( \left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i$$

CHALMERS

# Exact feasibility test for EDF

**(Sufficient and necessary condition)**

A sufficient and necessary condition for earliest-deadline-first scheduling, for which $D_i \le T_i$, is

$$\forall L : C_P(0,L) \le L$$

where $C_P(0,L)$ is the total processor demand in $[0, L]$.

The processor-demand analysis and associated feasibility test was presented by S. Baruah, L. Rosier and R. Howell in 1990.

CHALMERS

## Exact feasibility test for EDF

**(Sufficient and necessary condition)**

How large interval must be examined?

- Largest interval is LCM of the task's periods
- This interval can be further shortened (see course book)

How many time points must be examined?

- Only absolute deadlines need to be examined
- The set of deadlines can be further reduced (see course book)

The test can consequently be improved as follows:

$$\forall L \in K : C_P(0, L) \leq L$$

$$K = \left\{ D_i^k \mid D_i^k = kT_i + D_i,\ D_i^k \leq \mathrm{LCM}\left\{ T_1, \ldots, T_n \right\}, 1 \leq i \leq n, k \geq 0 \right\}$$

CHALMERS

## Example: scheduling using EDF

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

a) Determine, by analyzing the processor demand, whether the tasks are schedulable or not using EDF.

b) Determine, by using simulation, whether the tasks are schedulable or not using EDF.

$\tau_1$ $\tau_2$ $\tau_3$

| Task | $C_i$ | $D_i$ | $T_i$ |
|---|---|---|---|
| $\tau_1$ | 2 | 3 | 4 |
| $\tau_2$ | 2 | 7 | 8 |
| $\tau_3$ | 3 | 12 | 16 |

**We solve this on the blackboard!**

CHALMERS

## Extended processor-demand analysis

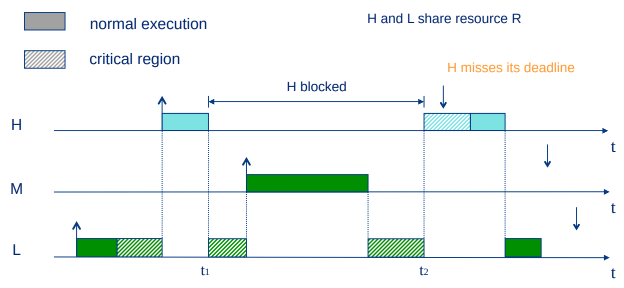
The test can be extended to handle:

- Blocking
- Start-time variations ("release jitter")
- Time offsets
- …

In this course, we only study blocking.

CHALMERS

## Deadline inversion



CHALMERS

# Stack Resource Policy (SRP)

The tasks are assigned <u>preemption levels</u>, the properties of which are:

- The preemption level of task $\tau_i$ is denoted $\pi_i$
- Task $\tau_i$ is not allowed to preempt another task $\tau_j$ unless $\pi_i > \pi_j$
- If $\tau_i$ has higher priority than $\tau_j$ and arrives later, then $\tau_i$ must have a higher preemption level than $\tau_j$.

Note:

- The preemption levels are static values, even though the tasks priorities may be dynamic.
- For EDF scheduling, suitable levels can be derived if tasks with shorter relative deadlines get higher preemption levels, that is:

$$\pi_i > \pi_j \quad \Leftrightarrow \quad D_i < D_j$$

CHALMERS

# Stack Resource Policy (SRP)

Deadline inversion can be reduced with <u>resource ceilings</u>:

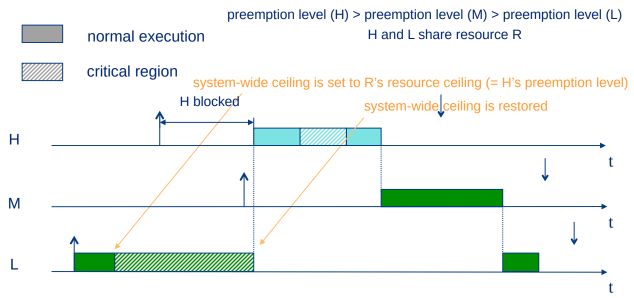
1. Each shared resource is assigned a ceiling that is always equal to the maximum preemption level among all tasks that may be blocked when requesting the resource.
2. The protocol keeps a <u>system-wide ceiling</u> that is equal to the maximum of the current ceilings of all resources.
3. A task with the earliest deadline is allowed to preempt only if its preemption level is higher than the system-wide ceiling.

Note:

- The original priority of the task is not changed at run-time.
- The resource ceiling is a <u>dynamic</u> value calculated at run-time as a function of current resource availability.
- Otherwise, the behavior of the SRP protocol is very similar to the ICPP, and SRP also exhibits identical properties regarding maximum blocking time and freedom from deadlock.

CHALMERS

# Stack Resource Policy (SRP)




Again: note the similarity to the behavior of ICPP

CHALMERS

# Extended processor-demand analysis

Blocking can be accounted for in the following way:

- Blocking factor $B_i$ represents the length of critical / non-preemptive regions that are executed by tasks with <u>lower preemption levels</u> than $\tau_i$
- Tasks are indexed in the order of increasing preemption levels, that is: $\pi_i > \pi_j \quad \Leftrightarrow \quad i < j$

$$\forall L \in K, \forall i \in [1,n]: C_P^i(0,L) \leq L$$

$$C_P^i = \sum_{k=1}^{i} \left( \left\lfloor \frac{L - D_k}{T_k} \right\rfloor + 1 \right) C_k + \left( \left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) B_i$$

CHALMERS

# Extended processor-demand analysis

Determining the blocking factor for task $\tau_i$:

1. Determine the worst-case resource ceiling for each critical region, that is, assume the run-time situation where the corresponding resource is unavailable.
2. Identify the tasks that have a preemption level lower than $\tau_i$ and that calls critical regions with a worst-case resource ceiling equal to or higher than the preemption level of $\tau_i$.
3. Consider the times that these tasks lock the actual critical regions. The longest of those times constitutes the blocking factor $B_i$.

CHALMERS

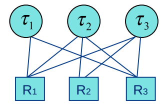
# Example: scheduling using EDF

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

Three resources R1, R2 and R3 have three, one, and three units available, respectively.

The parameters $H_{R1}$, $H_{R2}$ and $H_{R3}$ represent the longest time a task may use the corresponding resource.

The parameters $\mu_{R1}$, $\mu_{R2}$ and $\mu_{R3}$ represent the number of units a task requests from the corresponding resource.




| Task | $C_i$ | $D_i$ | $T_i$ | $H_{R1}$ | $H_{R2}$ | $H_{R3}$ | $\mu_{R1}$ | $\mu_{R2}$ | $\mu_{R3}$ |
|---|---|---|---|---|---|---|---|---|---|
| $\tau_1$ | 6 | 10 | 50 | 2 | - | 2 | 1 | - | 1 |
| $\tau_2$ | 7 | 17 | 50 | 1 | 2 | 2 | 2 | 1 | 3 |
| $\tau_3$ | 10 | 25 | 50 | 2 | 3 | 2 | 3 | 1 | 1 |

CHALMERS

# Example: scheduling using EDF

Problem: (cont'd)

Task $\tau_1$ first requests R3 and then, while using R3, requests R1

Task $\tau_2$ first requests R3 and then, while using R3, requests R2; then, after releasing the two resources, $\tau_2$ requests R1

Task $\tau_3$ first requests R2 and then, while using R2, requests R1; then, after releasing the two resources, $\tau_3$ requests R3

Examine the schedulability of the tasks when the SRP (Stack Resource Policy) protocol is used.

a) Derive the ceilings (dynamic and worst-case) of the resources.
b) Derive the blocking factors for the tasks.
c) Show whether the tasks are schedulable or not.

**We solve this on the blackboard!**

CHALMERS

# Feasibility tests

Summary

| | $D_i = T_i$ | $D_i \leq T_i$ |
|---|---|---|
| Static priority (RM/DM) | $U \leq n(2^{1/n} - 1)$ | $\forall i : R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \leq D_i$ |
| Dynamic priority (EDF) | $U \leq 1$ | $\forall L : \sum_{i=1}^{n} \left( \left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i \leq L$ |