CHALMERS

Schedulability analysis

Schedulability analysis:
The process of determining whether a task set can be scheduled by a given run-time scheduler in such a manner that all task instances will complete by their deadlines.

Schedulability analysis typically involves a **feasibility test** that is customized for the actual run-time scheduler used.

CHALMERS

Schedulability analysis

Static-priority schedulability analysis:

- The priority assignment problem
  - Given a set of tasks, *does there exist an assignment of static priorities to these tasks* satisfying the property that the system can be scheduled by a static-priority run-time scheduler such that all task instances will complete by their deadlines?
- The feasibility testing problem
  - Given a set of tasks, *and an assignment of priorities to these tasks*, can the system be scheduled by a static-priority run-time scheduler such that all task instances will complete by their deadlines?

CHALMERS

Schedulability analysis

Complexity of schedulability analysis: (Leung & Merrill, 1980)

> The problem of deciding if a task set can be scheduled in such a manner that all task instances will complete by their deadlines is NP-complete for each fixed m ≥ 1 processors.

Complexity of multiprocessor schedulability analysis: (Leung & Whitehead, 1982)

> The problem of deciding if a task set can be scheduled on m processors is NP-complete in the strong sense.

CHALMERS

Schedulability analysis

Complexity of feasibility testing: (Leung, 1989)

> The problem of deciding whether or not the schedule produced by a particular static or dynamic priority assignment is valid is NP-complete for m ≥ 1 processors.

Observation:

- If an optimal static priority assignment can be easily found, the priority-assignment problem reduces to the feasibility testing problem.

CHALMERS

Schedulability analysis

Complexity of uniprocessor schedulability analysis:
(Leung & Whitehead, 1982)

> There is a pseudo-polynomial time algorithm to decide if a task set can be scheduled on one processor in such a way that all task instances will complete by their deadlines.

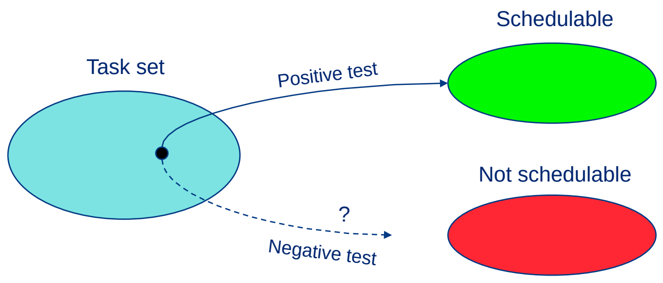
Proof:

- The deadline-monotonic priority assignment is optimal, and can be obtained in polynomial time
- The uniprocessor feasibility testing problem can be solved in pseudo-polynomial time (using critical instant analysis).

CHALMERS

Feasibility tests

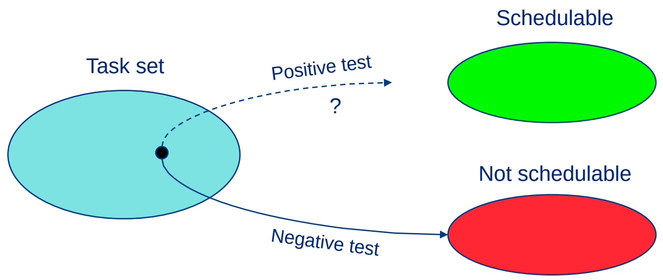
- A feasibility test is sufficient if it with a positive answer shows that a set of tasks is definitely schedulable.
  - A negative answer says nothing! A set of tasks can still be schedulable despite a negative answer.



CHALMERS

Feasibility tests

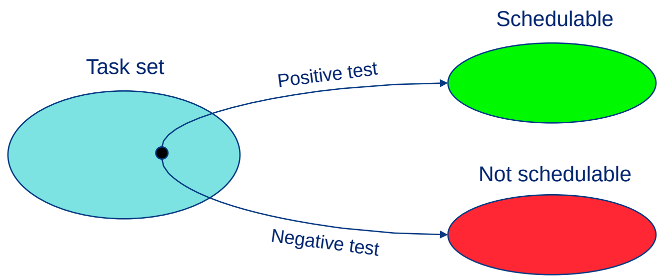
- A feasibility test is <u>necessary</u> if it with a <u>negative</u> answer shows that a set of tasks is definitely <u>not</u> schedulable.
  - A positive answer says nothing! A set of tasks can still be impossible to schedule despite a positive answer.



CHALMERS

Feasibility tests

- An <u>exact</u> feasibility test is both sufficient and necessary. If the answer is positive the task set is definitely schedulable, and if the answer is negative the task set is definitely not schedulable.



CHALMERS

Feasibility tests

What techniques for feasibility testing exist?

- Processor utilization analysis (for static/dynamic priorities)
  - The fraction of processor time that is used for executing the task set may not exceed a given bound
  - Mature for RM and EDF scheduling on a uniprocessor
- Response time analysis (for static priorities)
  - Worst-case response time for each task is calculated and compared against the deadline of the task
  - Mature for DM scheduling on a uniprocessor
- Processor demand analysis (for dynamic priorities)
  - The accumulated computation demand for the task set under a given time interval must not exceed the length of the interval
  - Mature for EDF scheduling on a uniprocessor

CHALMERS

Feasibility tests

Processor utilization analysis:

- The utilization U for a set of periodic tasks is the fraction of the processor's capacity that is used for executing the tasks.
- Since C_i / T_i is the fraction of processor time that is used for executing task τ_i the utilization for n tasks is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

CHALMERS

Feasibility tests

Processor utilization analysis for RM: (Liu & Layland, 1973)

- A sufficient condition for RM scheduling is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(2^{1/n} - 1\right)$$

- A conservative lower bound on the utilization can be derived by letting $n \rightarrow \infty$

$$\lim_{n \rightarrow \infty} n\left(2^{1/n} - 1\right) = \ln 2 \approx 0.693$$

CHALMERS

Feasibility tests

Processor utilization analysis for RM: (Liu & Layland, 1973)

- The sufficient schedulability condition is only valid if:
  1. All tasks are independent
  2. All tasks are periodic or sporadic
  3. Task deadline equals the period ($D_i = T_i$)

CHALMERS

Feasibility tests

Processor utilization analysis for RM: (Liu & Layland, 1973)

- The proof of the condition uses the fact that the worst-case response time for a task occurs at a critical instant (where all tasks arrive at the same time)
- The feasibility test is derived using an analysis of this special case
- The proof also shows that if the task set is schedulable for the critical instant case, it is also schedulable for any other case
- The proof is given in Krishna and Shin (Section 3.2.1) Highly recommended reading!

CHALMERS

Feasibility tests

Processor utilization analysis for EDF: (Liu & Layland, 1973)

- A sufficient and necessary condition for EDF scheduling is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

- The exact feasibility condition is only valid if:
  1. All tasks are independent
  2. All tasks are periodic
  3. Task deadline equals the period ($D_i = T_i$)

CHALMERS

Feasibility tests

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

$$R_i = C_i + I_i$$

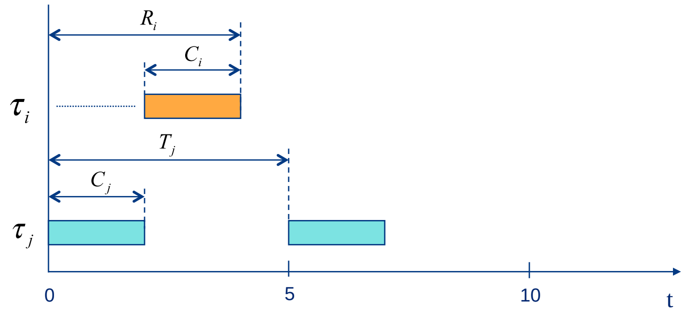
CHALMERS

Feasibility tests

Response-time analysis: (interference)

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

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



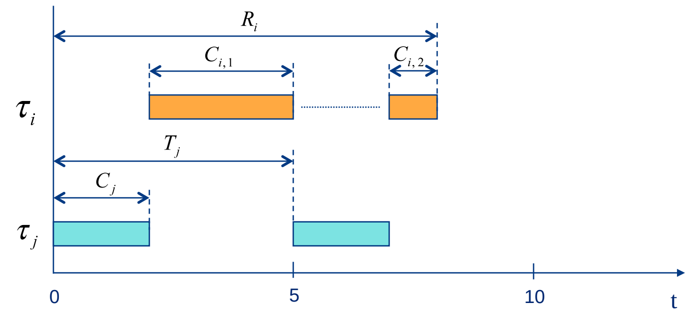
CHALMERS

Feasibility tests

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

Feasibility tests

Response-time analysis: (interference)

Task τ_i can be preempted by higher-priority task τ_j.

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

- If $0 < R_i \leq T_j$, task τ_i can be preempted at most <u>one time</u> by τ_j
- If $T_j < R_i \leq 2T_j$, task τ_i can be preempted at most <u>two times</u> by τ_j
- If $2T_j < R_i \leq 3T_j$, task τ_i can be preempted at most <u>three times</u> 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

Feasibility tests

Response-time analysis:

- 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

Feasibility tests

Response-time analysis:

- 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

Feasibility tests

Response-time analysis for DM: (Joseph & Pandya, 1986)

- A sufficient and necessary condition for DM scheduling is

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

- The exact feasibility condition is only valid if:
  1. All tasks are independent
  2. All tasks are periodic or sporadic
  3. Task deadline does not exceed the period ($D_i \leq T_i$)

CHALMERS

Feasibility tests

Processor-demand analysis:

- The processor demand for a task τ_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 τ_i that must complete execution before L.
- The total processor demand up to L is

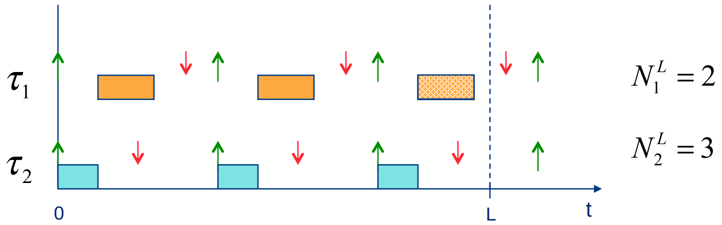
$$C_P(0, L) = \sum_{i=1}^{n} N_i^L C_i$$

CHALMERS

Feasibility tests

Processor-demand analysis:

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



CHALMERS

Feasibility tests

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

Feasibility tests

Processor-demand analysis for EDF: (Baruah et al., 1990)

- A sufficient and necessary condition for EDF scheduling with $D_i \leq T_i$ is

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

where the set of control points K is

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

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)	$U \leq 1$	$\forall L : \sum_{i=1}^{n} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i \leq L$

CHALMERS

Feasibility tests

Extended response-time analysis:

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

CHALMERS

Feasibility tests

Response-time analysis with blocking:

- 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

Feasibility tests

Response-time analysis with blocking:

- When using priority ceiling protocols (such as PCP or 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

Feasibility tests

Response-time analysis with blocking:

Determining the blocking factor for 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.