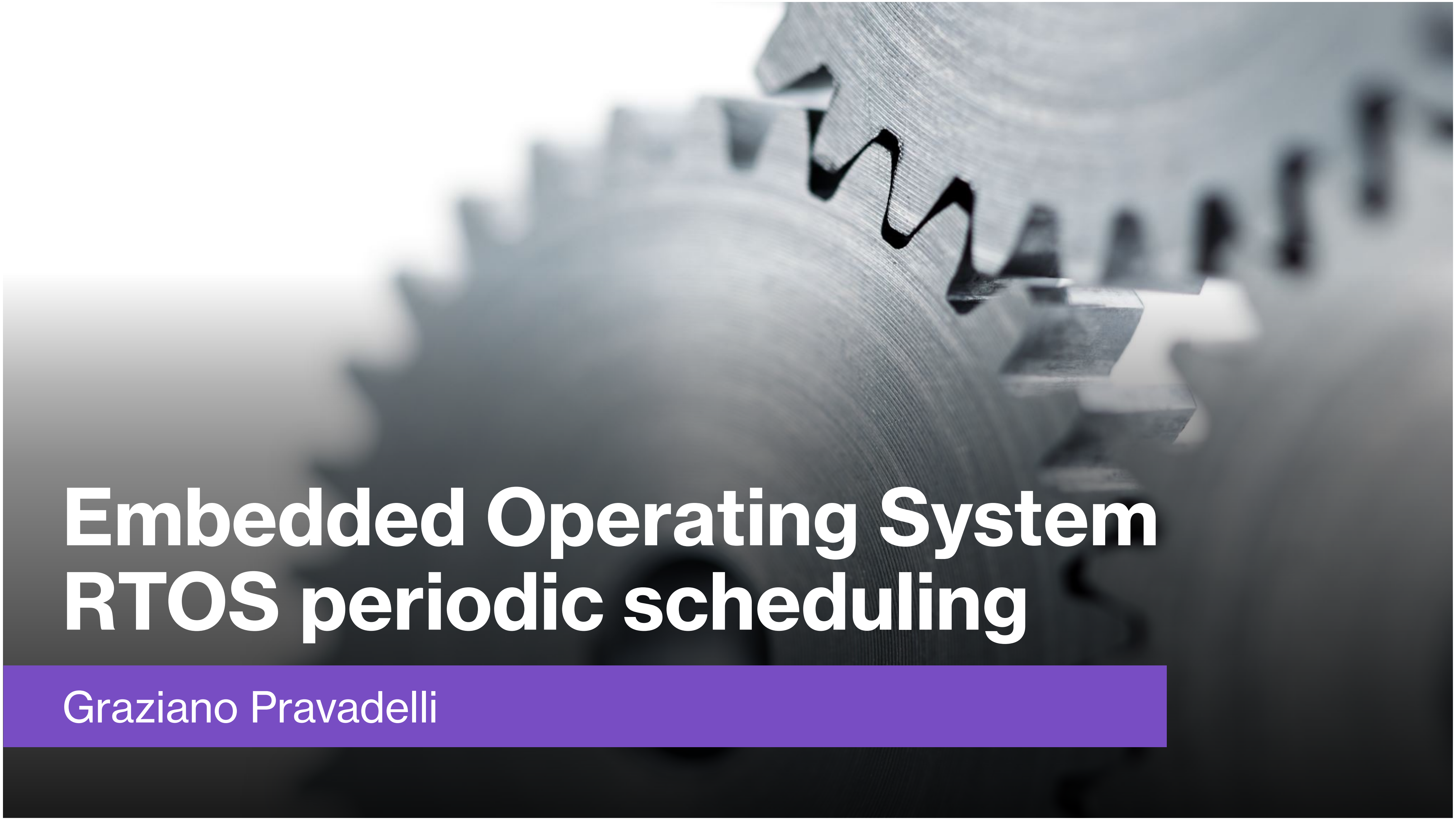


Embedded Operating System RTOS periodic scheduling

Graziano Pravadelli

Periodic activities

Major computational demand in many applications

Sensory data acquisition	Control loops	System monitoring

Usually, several periodic tasks running concurrently

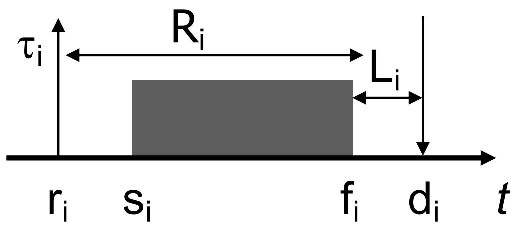
Assumptions

1. Instances of a task are periodically activated at constant rate.
2. All instances of a task have the same worst case execution time C_i
3. All instances of a task have the same deadline D_i, and $D_i = T_i$
4. All periodic tasks are independent
   (no precedence relations, no resource constraints)
5. No task can suspend itself (e.g., for I/O)
6. All tasks are released as soon as they arrive
7. All overheads due to the RTOS are assumed to be zero

3,4: can be too tight for practical application

Characterization of periodic tasks

- A periodic task τ_i can be characterized (see assumptions 1-4) by:
  - phase ϕ_i
  - period T_i
  - worst case computation time C_i
- Additional parameters:
  - Response time $R_i = f_i - r_i$
  - Critical instant (of a task)
    - Release time of the instance resulting in the largest response time



Scheduling alternatives

Static scheduling

Dynamic (process-based) scheduling

Static scheduling – Cyclic executive

Often used in military and traffic systems

Fixed set of purely periodic tasks	
Layout a schedule such that the repeated execution of this schedule will cause all processes to run at their correct rate	Essentially a table of procedure calls, where each procedure represents part of a code for a "process"

Cyclic executive - Principles

- Tasks are mapped in a set of **minor cycles**
- The set of minor cycles constitute a **major cycle** (the complete schedule)
- Cycle durations:
  - Minor cycle m = min (T_i)
  - Major cycle M = LCM (T_i) ➡ M = k·m
- Example:
  - T = (7, 10, 21, 35)
    - m = 7
    - M = 2x3x5x7 = 210

LCM = Least Common Multiple

Cyclic executive – Example

- Task set

Process	period,T	Computation Time, C
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2

- Schedule

Draw the schedule

Cyclic executive – Example

```
loop
   wait_for_interrupt
   Procedure_For_A
   Procedure_For_B
   Procedure_For_C
   wait_for_interrupt
   Procedure_For_A
   Procedure_For_B
   Procedure_For_D
   Procedure_For_E
   wait_for_interrupt
   Procedure_For_A
   Procedure_For_B
   Procedure_For_C
   wait_for_interrupt
   Procedure_For_A
   Procedure_For_B
   Procedure_For_D
end loop
```

Cyclic executive - Advantages

- No actual process exists at run-time
- Each minor cycle is a sequence of procedure calls
- The procedures share common address space and can pass data between themselves
- No need for data protection, no concurrency

Cyclic executive - Disadvantages

It only handles periodic tasks

- Difficult to incorporate aperiodic processes without changing task sequence

Fragile in case of overload conditions

- What happens if a task exceeds the minor?

Task periods must be multiple of minor cycle time

- To make this manageable

Difficult to construct cyclic executive

- Equivalent to bin packing problem, NP-hard

Dynamic (process-based) scheduling

- Fixed-priority scheduling
  - Rate-monotonic scheduling
  - Deadline-monotonic scheduling
- Dynamic-priority scheduling
  - EDF

Rate Monotonic (RM)

Static priority scheduling

Rate monotonic ➡ priorities assigned according to request rates

Each task is assigned a (unique) priority based on its period

The shorter the period, the higher the priority

Given tasks t_i and t_j, $T_i < T_j$ ➡ $P_i > P_j$

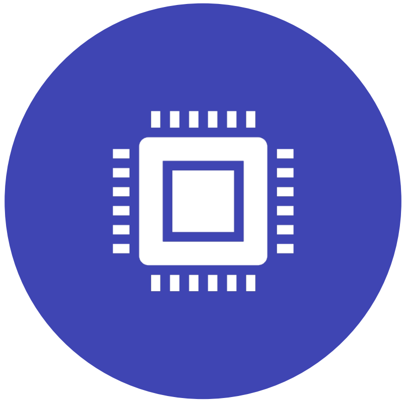
Intrinsically preemptive

Currently executing task is preempted by a newly released task with shorter period

RM optimality

If a set of processes can be scheduled (using preemptive priority-based scheduling) with a fixed priority-based assignment scheme, then RM can also schedule the set of processes

RM – What about schedulability?



PROCESSOR
UTILIZATION FACTOR

OR

WORST CASE RESPONSE
TIME ANALYSIS

Processor utilization factor (1)

- Given a set Γ of periodic tasks, the utilization factor U
  - is the fraction of processor time spent in the execution of the task set
  - determines the load of the CPU

$$U = \Sigma_{i=1 \ldots n} C_i/T_i$$

C_i/T_i = the fraction of processor time spent in executing τ_i

Processor utilization factor (2)

U can be improved by:

increasing computation times of the tasks, or

decreasing the periods of the tasks

Maximum value for U:

below it, G is schedulable

above it, G is not schedulable

Limit depends on:

task set (relations among task's periods)

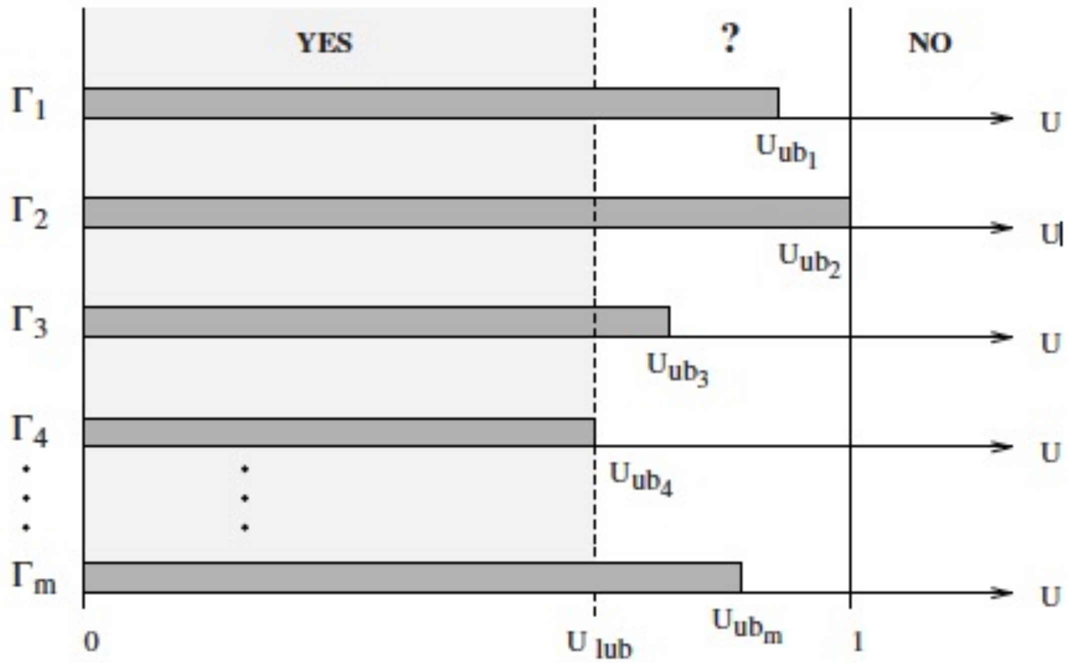
algorithm used to schedule the tasks

Processor utilization factor (3)

- **Upper bound of U, U_{ub} (Γ, A)**
  - The processor is fully utilized, given a task set and a scheduling algorithm
  - Task set Γ is schedulable using A, but any increase of computation time in one of the tasks may make the set infeasible
- **Least upper bound of U, U_{lub}**
  - Minimum of U_{ub} over Γ that fully utilizes the processor, for a given algorithm
  - U_{lub} allows to easily test for schedulability

$$U_{\mathrm{lub}}(A) = \min(U_{ub}(\Gamma, A)), \forall \Gamma.$$

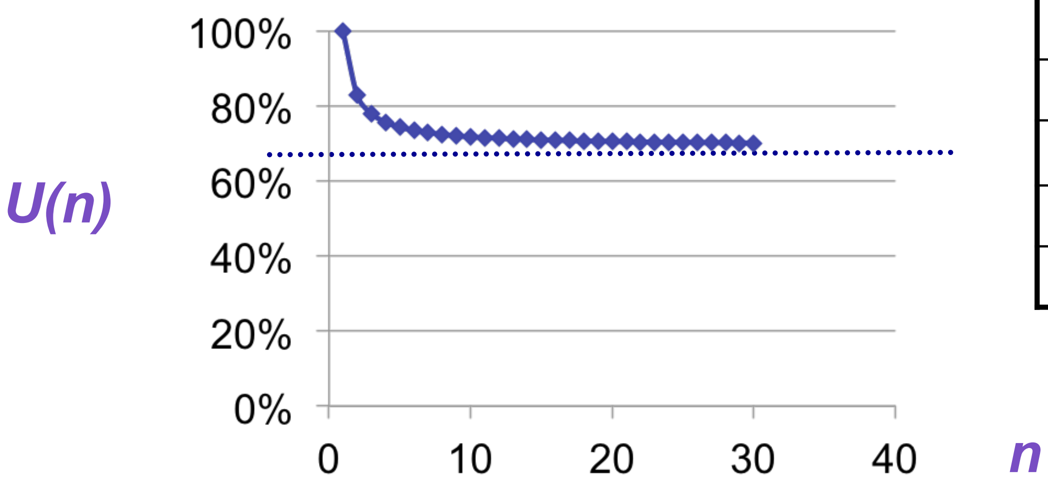
U-based schedulability test



- $U_{\Gamma i}$ <= $U_{lub}(A)$ ➡ Γ_i schedulable
- $U_{\Gamma i}$ > $U_{lub}(A)$ ➡ Γ_i may be schedulable, if the periods of the tasks are suitable related
- $U_{\Gamma i}$ > 1 ➡ Γ_i not schedulable

U-based schedulability test [Liu&Layland 73]

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



n	*U(n)*
1	1.000
2	0.828
3	0.780
4	0.757
5	0.743
6	0.735
7	0.729
8	0.724
9	0.721
10	0.718

For large values of *n* the bound asymptotically reaches 69.3% (ln2)

Any task set with U < 69.3% is schedulable under RM

U-based schedulability test - Notes

- This schedulability test is **sufficient, but not necessary**
- If a process set passes the test, it will meet all deadlines
- if it fails the test, it may or may not fail at run-time

- The utilization-based test only gives a yes/no answer
- **No indication of actual response times** of processes
- Overestimation of processor load

U-based schedulability test – Example 1

Process	Period, *T*	Computation time, *C*	Priority, *P*	Utilization, *U*
Task_1	50	12	1	0.24
Task_2	40	10	2	0.25
Task_3	30	10	3	0.33

Is the set schedulable?

$U = 12/50 + 10/40 + 10/30 = 0.82$

$U > U(3) = 3\,(2^{1/3} - 1) = 0.78$

Undecidable, and… no!

Draw the schedule

U-based schedulability test – Example 2

Process	Period, *T*	Computation time, *C*	Priority, *P*	Utilization, *U*
Task_1	50	25	1	0.5
Task_2	40	5	2	0.125
Task_3	30	4	3	0.133

Is the set schedulable?

$U = 25/50 + 5/40 + 4/30 = 0.758$

$U < U(3) = 3\,(2^{1/3} - 1) = 0.78$

Yes!

Draw the schedule

U-based schedulability test – Example 3

Process	Period, *T*	Computation time, *C*	Priority, *P*	Utilization, *U*
Task_1	80	40	1	0.500
Task_2	40	10	2	0.250
Task_3	20	5	3	0.250

Is the set schedulable?

$U = 40/80+10/40+ 5/20 = 1$

$U > U(3) = 3\,(2^{1/3} - 1) = 0.78$

Undecidable, and... yes!

Draw the schedule

Response time analysis

Sufficient and necessary conditions for schedulability

Two stages:

- The worst-case response time of each process is obtained analytically
- The response times are then individually compared with the process deadlines

Response time analysis

- For task i, the worst-case response time is given by $\mathbf{R_i = C_i + I_i}$
  - **I_i is the maximum interference**
    that process i can experience in any time during the interval $[t, t+R_i)$
  - **Interference = preemption**
    - For the highest priority process, its worst-case response time equals its computation time (i.e., R = C)
    - Other processes suffer interference from higher-priority processes

Response time analysis – What is I_i?

Let i, j be two tasks where priority(j) > priority(i)

During the interval $[0,R_i)$ we have

# of releases of j instances is $\lceil R_i / T_j \rceil$	Max interference of j is $\lceil R_i / T_j \rceil C_j$

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

hp(i) **= set of tasks with higher priority than i**

Response time analysis – Solution (1)

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

Solving by forming a recurrence equation
where the set $\{w_i^0, w_i^1, w_i^2, \ldots\ldots, w_i^n, \ldots.\}$ is monotonically non-decreasing

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

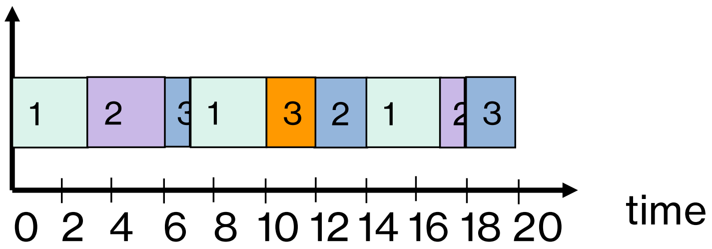
Response time analysis – Solution (2)

- The equation is solved when $w^{n+1} = w^n$
- If the equation does not have a solution, then the w values will continue to rise
  - Stop when w > D → not schedulable
- Value of w^0 ?
  - The smallest possible value for R_i is C_i

Response time analysis – Example (1)

- Task 1 => $R_1 = C_1 = 3 <= 7$ OK
- Task 2 =>
  - $w_2^0 = C_2 = 3$
  - $w_2^1 = 3 + \lceil 3/7 \rceil 3 = 6$
  - $w_2^2 = 3 + \lceil 6/7 \rceil 3 = 6 = w_2^1$ => $R_2 = 6 <= 12$ OK
- Task 3 =>
  - $w_3^0 = C_3 = 5$
  - $w_3^1 = 5 + \lceil 5/12 \rceil 3 + \lceil 5/7 \rceil 3 = 11$
  - $w_3^2 = 5 + \lceil 11/12 \rceil 3 + \lceil 11/7 \rceil 3 = 14$
  - $w_3^3 = 5 + \lceil 14/12 \rceil 3 + \lceil 14/7 \rceil 3 = 17$
  - $w_3^4 = 5 + \lceil 17/12 \rceil 3 + \lceil 17/7 \rceil 3 = 20$
  - $w_3^5 = 5 + \lceil 20/12 \rceil 3 + \lceil 20/7 \rceil 3 = 20$
  - => $R_3 = 20 <= 20$ OK

Process	Period, *T*	Computation time, *C*	Priority, *P*
Task_1	7	3	3
Task_2	12	3	2
Task_3	20	5	1



Response time analysis – Example (2)

- Task set that fails the U-based test
  - U = 40/80+10/40+5/20 = 1/2+1/4+1/4 = 1 > 0.78
- Response time test is ok
  - R1 = 80 <= 80 OK
  - R2 = 15 <= 40 OK
  - R3 = 5 <= 20 OK

Process	Period, *T*	Computation time, *C*	Priority, *P*
Task_1	80	40	1
Task_2	40	10	2
Task_3	20	5	3

EDF

- Dynamic priority assignment
- Same idea as for aperiodic tasks
  - Tasks are selected according to their absolute deadlines
  - Tasks with earlier deadlines are given higher priorities
  - It is intrinsically preemptive
- More powerful than RM
- It works optimally for periodic as well as aperiodic tasks

EDF vs. RM – Example

Process	*T*	*C*
T_1	5	2
T_2	7	4

EDF schedule

Draw the schedule

RM schedule

Draw the schedule

EDF schedulability test [Liu&Layland 73]

- Schedulability of a periodic task set scheduled by EDF can be verified through the **processor utilization factor U**

- A set of periodic tasks is schedulable with EDF iff $\sum_{i=1}^{n}\left(\frac{C_i}{T_i}\right) \leq 1$

- Sufficient and necessary condition

EDF schedulability test – Example

- Processor utilization of the task set
  - U = 2/5 + 4/7 = 34/35 = 0.97
  - U > 0.82
    - Schedulability not guaranteed under RM
  - U <1
    - Schedulability guaranteed under EDF

Process	Period, *T*	WCET, *C*
T_1	5	2
T_2	7	4

Deadline monotonic (DM)

Assumption up to now

- relative deadline = period

DM scheduling weakens this assumption

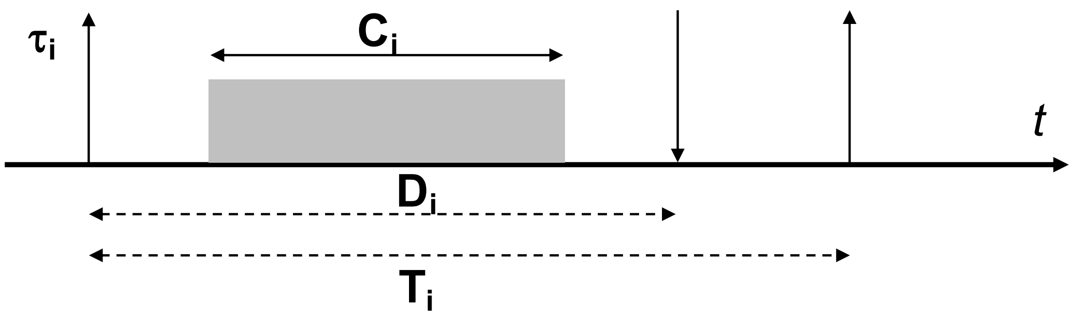
- Static algorithm with preemption

For DM each periodic tasks t_i is characterized by four parameters:

- Relative deadline D_i (equal for all instances)
- Worst case computation time C_i (equal for all instances)
- Period T_i
- Phase ϕ_i

DM scheduling

- DM = generalization of RM
  - RM optimal for D = T
  - DM extends this optimality for D < T
- Priority of a process inversely proportional to its deadline (but still static!)
  - Given tasks τ_i and τ_j, $D_i < D_j \Rightarrow P_i > P_j$



DM scheduling – Example

- Task set not schedulable with RM but schedulable with DM

Process	Period, *T*	Deadline, D	Computation time, *C*	Priority, *P*	Response time, *R*
Task_1	20	5	3	4	3
Task_2	15	7	3	3	6
Task_3	10	10	4	2	10
Task_4	20	20	3	1	20

Draw the schedule

DM schedulability analysis

- Schedulability can be tested replacing the period with the deadlines in the definition of U

$$U = \sum_{i=1 \ldots n} C_i / D_i$$

- Too pessimistic! (U overestimated)

DM schedulability analysis

- Actual guarantee test based on a modified response time analysis
- Intuitively:
  for each τ_i , the sum of its processing time and the interference (preemption) imposed by higher priority tasks must be $\leq D_i$

$$\boldsymbol{C_i + I_i \leq D_i}$$

$$\forall i:\ 1 \leq i \leq n$$

$$I_i = \Sigma_{(j=1\ldots i-1)} \lceil R_i / T_j \rceil C_j$$

EDF for D<T

- EDF applies also to the case D < T
- But schedulability test based on the **processor demand criterion**
- The processor demand of a task τ_i in any interval [t, t+L] is the amount of processing time required by τ_i in [t, t+L] that has to be completed at or before t+L
  - i.e., that has to be executed with deadlines $\leq$ t+L

Processor demand for EDF

- Applicable also to the case D=T
- In general, the schedulability of the task set is guaranteed iff the *cumulative processor demand* in any interval [0, L] ≤ L (the interval length):

$$C_P(0,L) = \sum_{i=1}^{n} \left\lfloor \frac{L}{T_i} \right\rfloor C_i. \quad \leq L$$

Processor demand for EDF

- In the case D<T

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

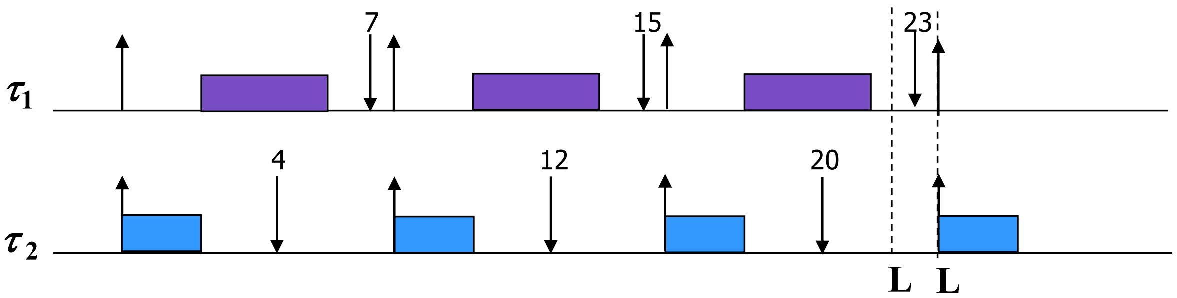
Number of checkpoints is actually limited

Number of completions between 0 and L-D_i

Processor demand for EDF – Example

- Schedulability test (L=21)
  - $(\lfloor(21-7)/8\rfloor+1) \bullet 3 + (\lfloor(21-4)/8\rfloor+1) \bullet 2 = 6 + 6 = 12 < 21$ OK
- Schedulability test (L=24)
  - $(\lfloor(24-7)/8\rfloor+1) \bullet 3 + (\lfloor(24-4)/8\rfloor+1) \bullet 2 = 9 + 6 = 15 < 24$ OK

Task	T	D	C
τ_1	8	7	3
τ_2	8	4	2



Periodic task scheduling: summary

RM is optimal among fixed priority assignments (with D=T)

EDF is optimal among dynamic priority assignments

Deadlines = Periods

- guarantee test in O(n) using processor utilization, applicable to EDF and RM (only sufficient condition)

Deadlines < periods

- polynomial time algorithms for guarantee test
- fixed priority (DM): response time analysis
- dynamic priority (EDF): processor demand

Periodic task scheduling – Summary

	$D_i = T_i$	$D_i \le T_i$
Static Priority	**RMA** Processor utilization approach $\sum_{i=1}^{n}\left(\frac{C_i}{T_i}\right) \le n(2^{1/n} - 1)$	**DMA** Response time approach $\forall i,\ R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \le D_i$
Dynamic Priority	**EDF** Processor utilization approach $\sum_{i=1}^{n}\left(\frac{C_i}{T_i}\right) \le 1$	**EDF** Processor demand approach $\forall L > 0,\quad L \ge \sum_{i=1}^{n}\left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1\right) C_i$