EEE499 - Real-Time Embedded System Design

Schedulability Part 2

RM Least Upper Bound (RM LUB)

$$U = \sum_{i=0}^{m} (e_i / p_i) \leq m(2^{\frac{1}{m}} - 1)$$

U: Utilization of the CPU that is achievable

e_i: Execution time of task i

m: Total number of tasks sharing common CPU resources

p_i: Release period of task i

Response Time Analysis

- a disadvantage of utilization based schedulability testing for RM LUB is that it is sufficient but not necessary

- to supplement this test, we introduce the notion of **Response Time Analysis**

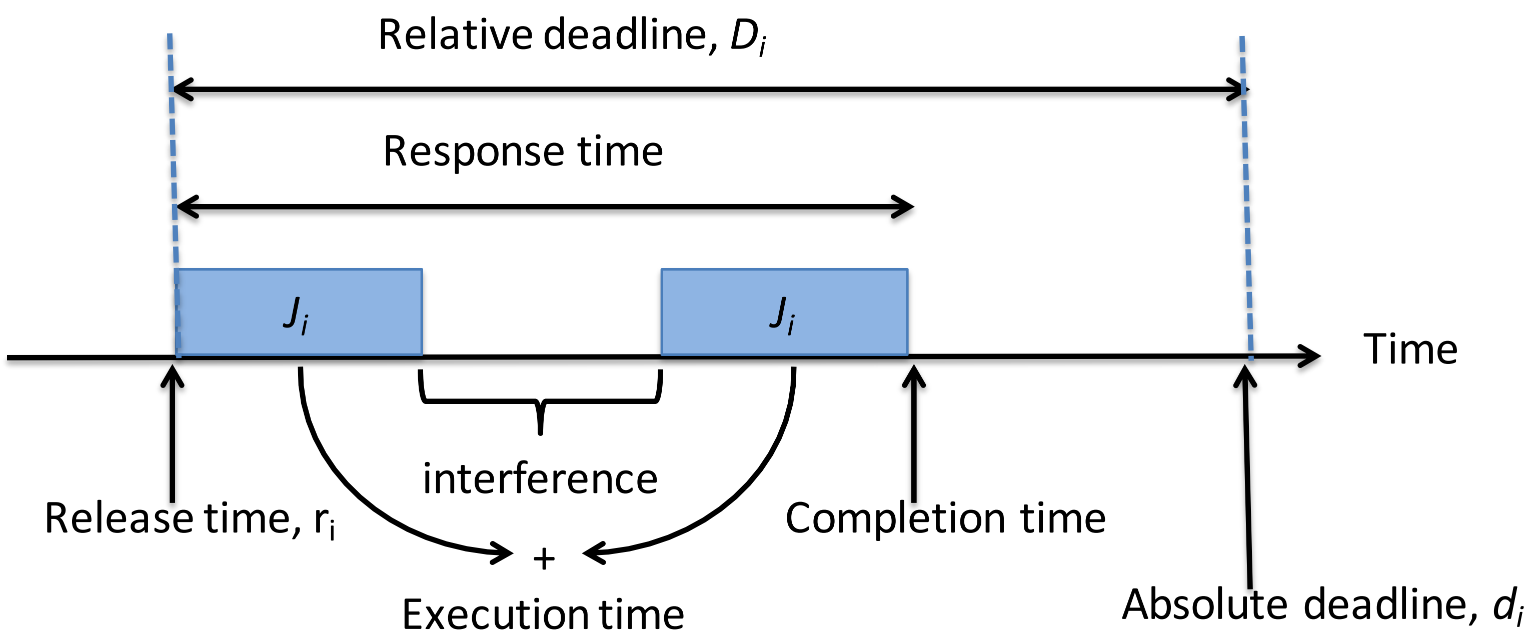
Response Time Analysis

Response time analysis allows to:

- predicts the worst case response time for each task
- compare each task's response time to its deadline

If all worst case response times are less than their respective deadlines, the system is schedulable

Response Time



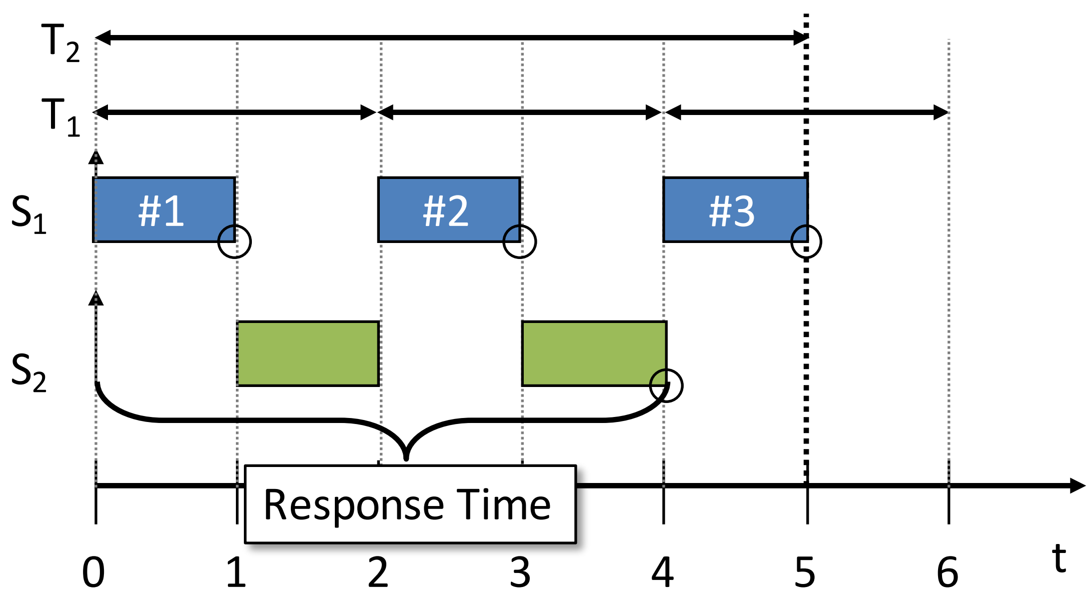
Response Time

$S_1 = (2,1)$

$S_2 = (5,2)$

$$R_2 = e_2 + \overbrace{e_{1,1} + e_{1,2}}^{I_1}$$

$$R_2 = 2 + 1 + 1 = 4$$



Response Time

Response time of a task is defined to be the sum of its own worst case execution time and its maximum interference

$$R_i = e_i + I_i \qquad (1)$$

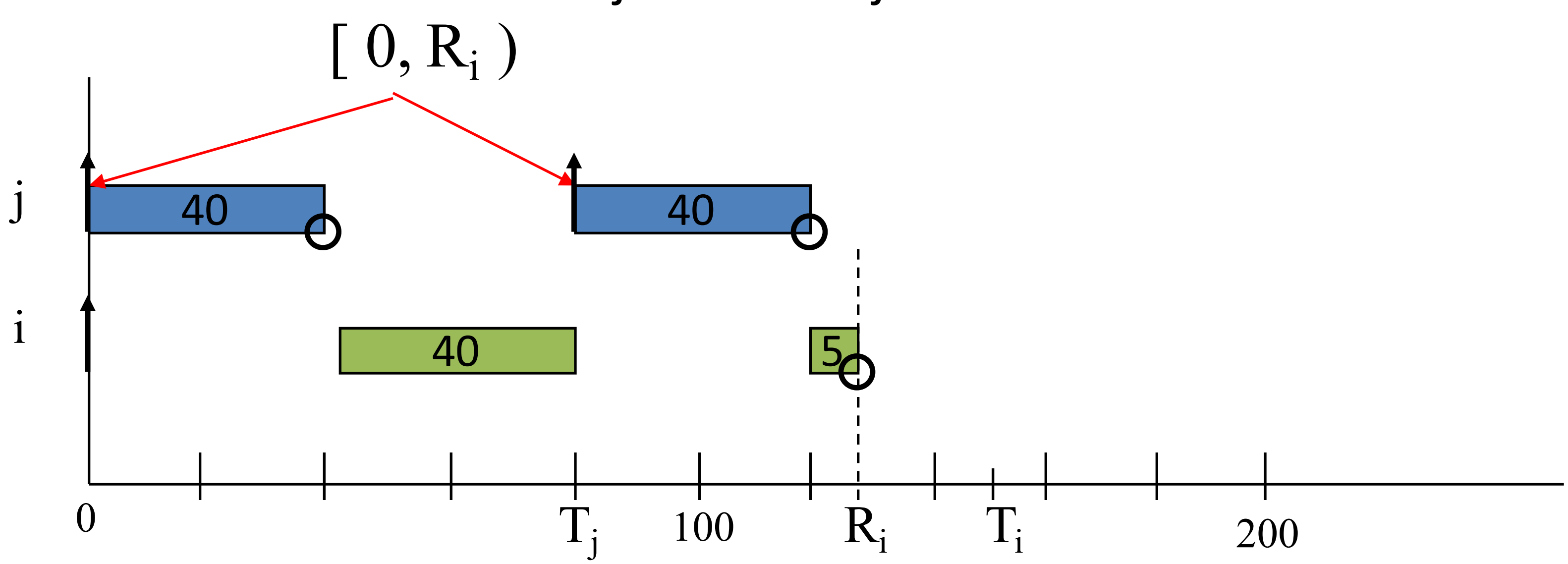
where I_i is the maximum interference* that task i can experience in any time interval $[t, t + R_i)$

*The condition for maximum interference occurs when all higher priority tasks are released at the same time as task i, the *critical instant*.

Counting Releases

- consider task i and a higher priority task j
- now, the **number of releases** of task j in time interval 0 to R_i can be derived as follows:

$$\#\text{ of releases}_j = \lceil R_i / p_j \rceil \qquad (2)$$



Determining Interference

- from this, the maximum interference of task j on task i in interval 0 to R_i is given by:

$$\text{interference}_{\max\,[0, R_i)} = \lceil R_i/p_j \rceil e_j \qquad (3)$$

- but there may be other higher priority tasks, therefore:

$$I_i = \sum_{j=1}^{k} \lceil R_i/p_j \rceil e_j \qquad (4)$$

where P(j) < P(k)

Calculating Response Time

- substituting equation (4) into (1) gives the general expression for response time:

$$R_i = e_i + \sum_{j=1}^{k} \left\lceil {R_i}/{p_j} \right\rceil e_j \qquad (5)$$

- note the following issue (problem):
  - R_i is on both sides of this equation

Understanding Response Time

(Example 1 from last class)

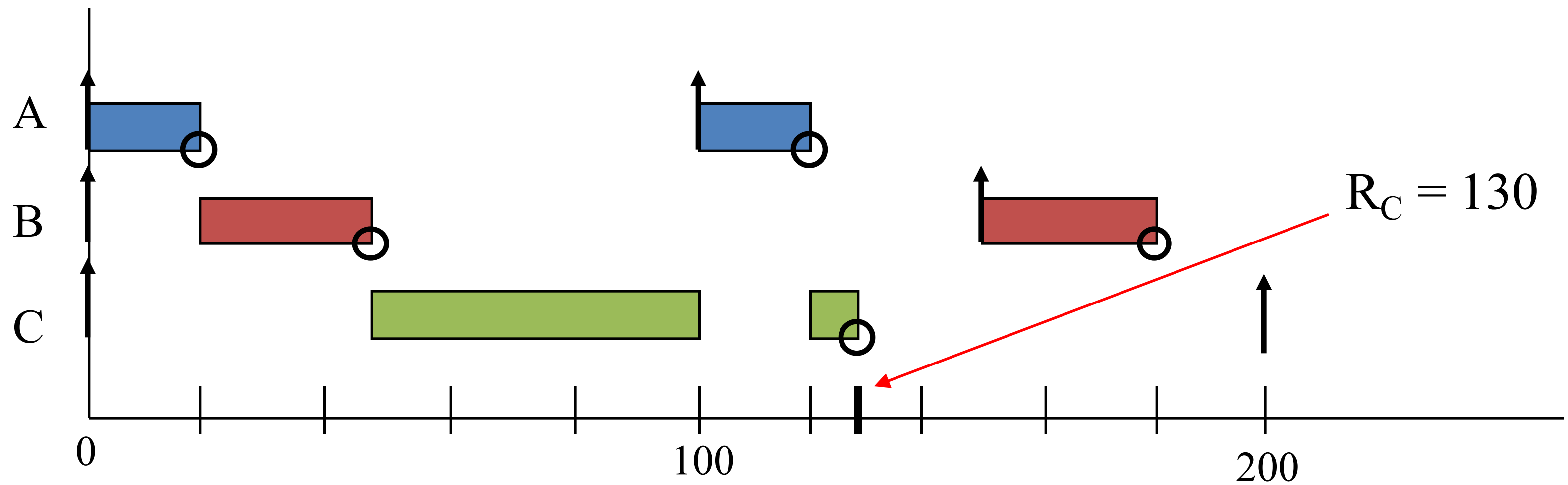
$$R_C = e_C + \lceil R_C/p_A \rceil e_A + \lceil R_C/p_B \rceil e_B$$

$$= 60 + 2(20) + 1(30)$$

$$= 130$$

C_A, T_A C_B, T_B C_C, T_C

System = {(20, 100), (30, 150), (60, 200)}



Calculating Response Time

- essentially, form a recursive relationship with Equation (5) and solve iteratively:

$$w_i^{n+1} = e_i + \sum_{j=1}^{k} \left\lceil {w_i^n}/{p_j} \right\rceil e_j \qquad (6)$$

where initial (seed) value $w_i^0 = e_i$

- the algorithm is then to solve for successive values of w_i^{n+1} until $w_i^{n+1} = w_i^n$, then solution found -> $R_i = w_i^n$

  if $R_i > D_i$, then task i can not meet its deadline

Calculating Response Time Example 1

Tâche	e_i	p_i	P_i	U_i
A	40	80		
B	10	40		
C	5	20		

Is it schedulable?

Calculating Response Time

Tâche	e_i	p_i	P_i	U_i
A	40	80	3	
B	10	40	2	
C	5	20	1	

Is it schedulable?

Calculating Response Time

Tâche	e_i	p_i	P_i	U_i
A	40	80	3	0.5
B	10	40	2	0.25
C	5	20	1	0.25

Is it schedulable?

$$U = 1 > 0.779$$

Simple Task Model

- Assumptions:
  1. Tasks are periodic and the period is constant
  2. Completion-time < period
  3. Tasks are independent
  4. Runtime is known and deterministic
  5. all system overheads are negligible or deemed to be included in task computation times
  6. Critical instant - defined as the maximum load condition when all tasks release together

- Constraints
  1. Deadline = period
  2. fixed set of tasks
  3. Preemptive

Scheduling with Aperiodic Tasks

- the simple task model that we have been able to deal with thus far is restrictive in several ways. Not being able to handle aperiodic tasks is a major restriction.
- one approach is to make aperiodic (or sporadic) tasks resemble periodic tasks
  - consider that an asynchronous task's minimum inter-arrival time can be treated like a period, T
  - with just this assumption one can use response time analysis for both types of tasks

Scheduling with Aperiodic Tasks

- the simple task model assumption that $D=T$ is unrealistic for aperiodic tasks
  - typically, an aperiodic task will occur infrequently (large inter-arrival time) but must be serviced quickly ($D < T$)
  - therefore priority assignment based upon the period (T) will usually not satisfy the requirement to meet the deadline (D)

Deadline Monotonic Priority Ordering

- deadline monotonic priority ordering (DMPO) scheme is introduced as follows:
  - the shorter the task deadline, the higher the priority

Tasks	e_i	p_i	D_i	P_i	R_i
1	3	20	5		
2	3	15	7		
3	4	10	10		
4	3	20	20		

Deadline Monotonic Priority Ordering (Example 2)

- deadline monotonic priority ordering (DMPO) scheme is introduced as follows:
  - the shorter the task deadline, the higher the priority

Tasks	e_i	p_i	D_i	P_i	R_i
1	3	20	5	1	
2	3	15	7	2	
3	4	10	10	3	
4	3	20	20	4	

Deadline Monotonic Priority Ordering

- deadline monotonic priority ordering (DMPO) scheme is introduced as follows:
  - the shorter the task deadline, the higher the priority

Tasks	e_i	p_i	D_i	P_i	R_i
1	3	20	5	1	3
2	3	15	7	2	6
3	4	10	10	3	10
4	3	20	20	4	20

Response Time Analysis Example 3

Tasks	e_i	p_i	P_i
1	3	7	
2	3	12	
3	5	18	

- use RM scheduling
- apply the utilization based schedulabilty test
- use Response Time Analysis to determine whether the system is schedulable

References

[1] Siewert, S. Pratt, J. Real-Time Embedded Components and Systems with Linux and RTOS. Mercury Learning and Information, 2016.

[2] Burns, A. and Wellings, A., "*Real-Time Systems and Programming Languages*", Chapter 13, Addison Wesley, 1997

[3] TimeSys Corp, "*The Concise Handbook of Real-Time Systems*", Version 1.0, 1999