THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Mixed-criticality scheduling: improved resource-augmentation results

Sanjoy Baruah, **Haohan Li**
The University of North Carolina

Leen Stougie
Vrije Universiteit

Outline

- Motivation
  - Background for certification

Outline

- Motivation
  - Background for certification
- Model
  - Definition of mixed-criticality system
  - Hardness of feasibility test

Outline

- Motivation
  - Background for certification
- Model
  - Definition of mixed-criticality system
  - Hardness of feasibility test
- Solution
  - Why EDF and criticality-monotonic fail
  - OCBP algorithm

Motivation

- An example for classic real-time tasks

Motivation

- An example for classic real-time tasks

J_1			
J_2			
J_3			
J_4			

Motivation

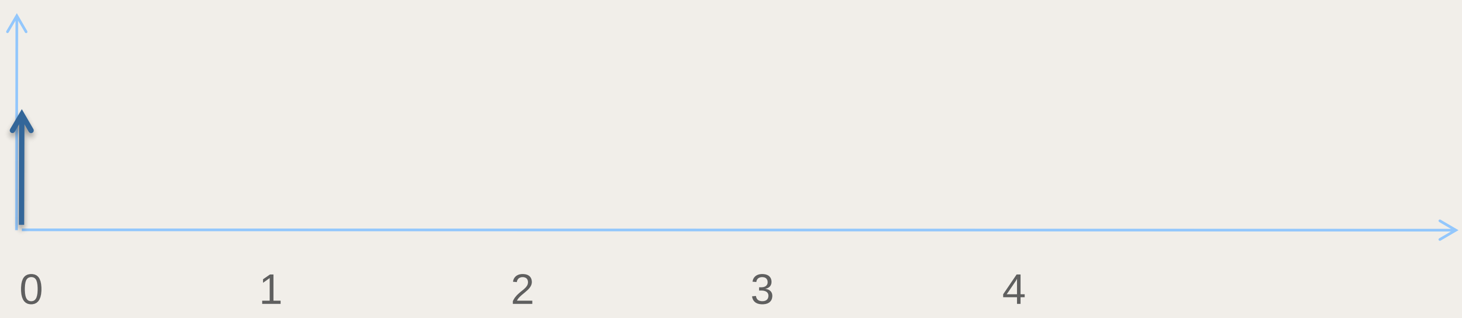
- An example for classic real-time tasks

	Release time(A_i)		
J_1			
J_2			
J_3			
J_4			

Motivation

- An example for classic real-time tasks

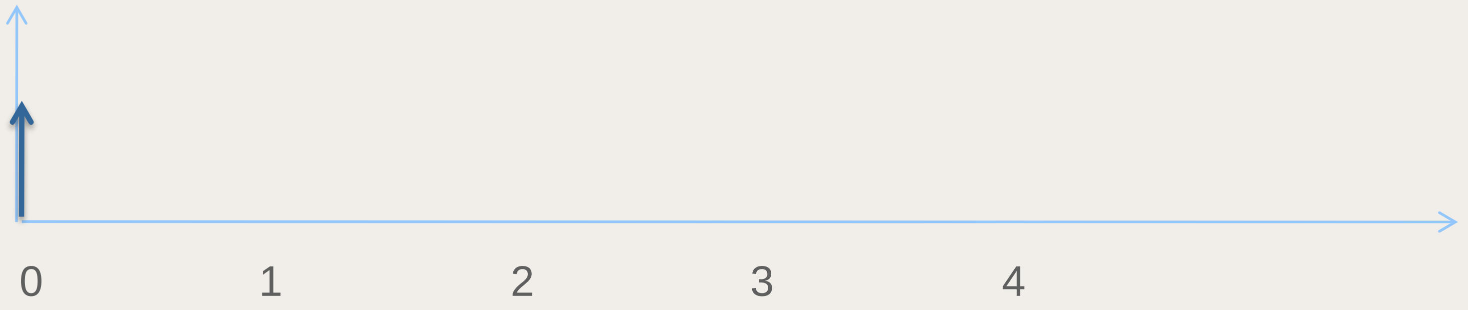
	Release time(A_i)		
J_1	0		
J_2	0		
J_3	0		
J_4	0		



Motivation

- An example for classic real-time tasks

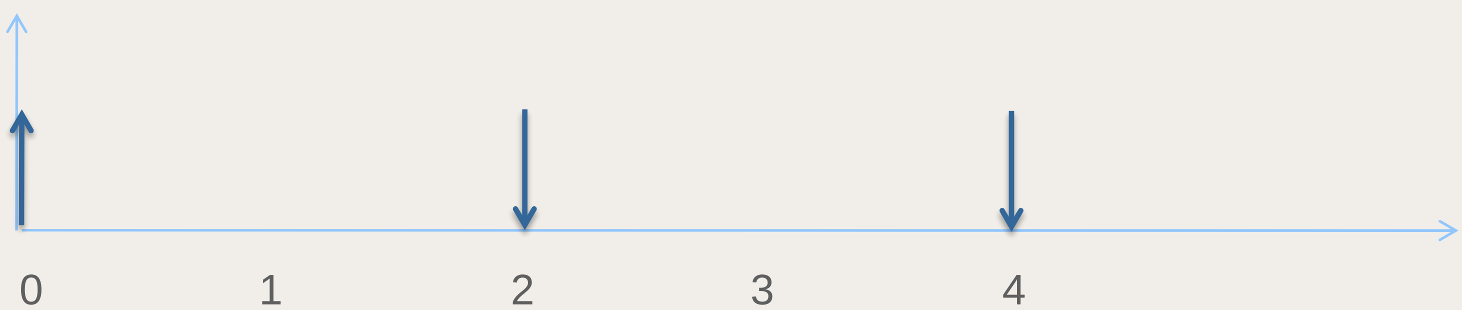
	Release time(A_i)	Deadline(D_i)	
J_1	0		
J_2	0		
J_3	0		
J_4	0		



Motivation

- An example for classic real-time tasks

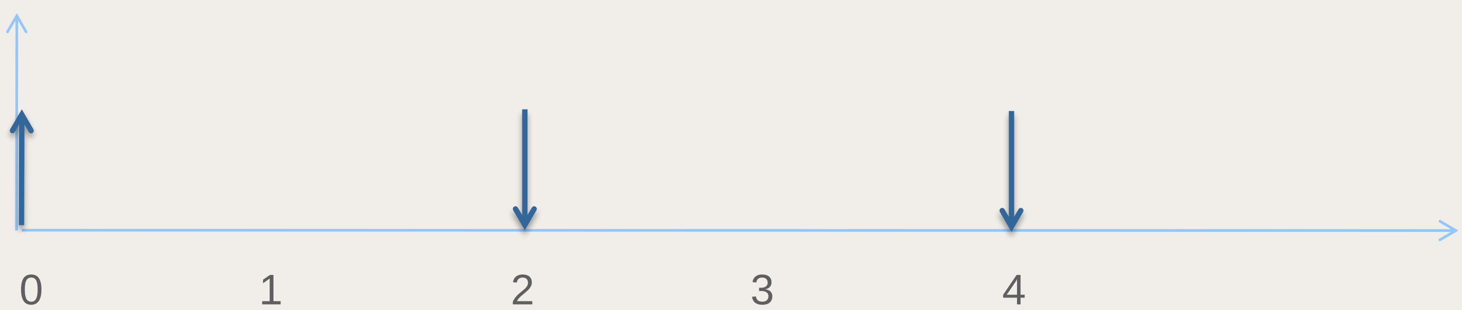
	Release time(A_i)	Deadline(D_i)	
J_1	0	2	
J_2	0	4	
J_3	0	4	
J_4	0	4	



Motivation

- An example for classic real-time tasks

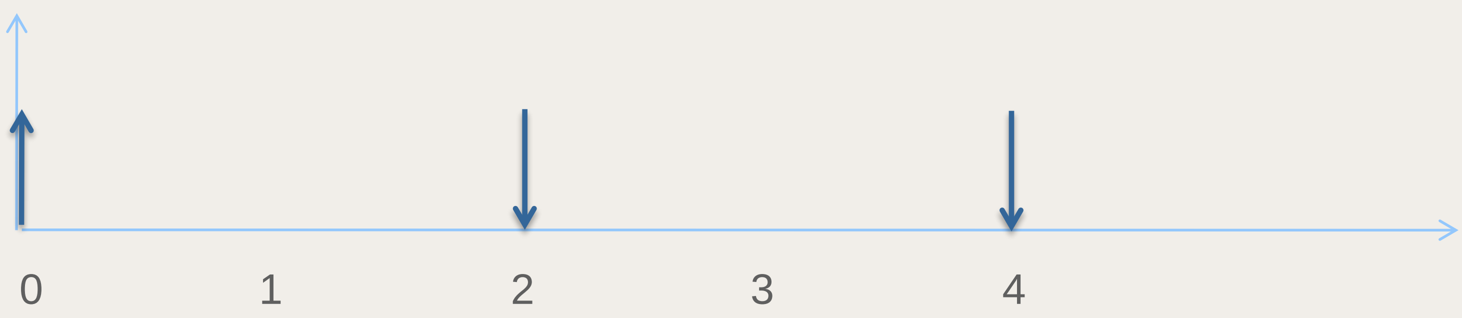
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	
J_2	0	4	
J_3	0	4	
J_4	0	4	



Motivation

- An example for classic real-time tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1



Motivation

- An example for classic real-time tasks

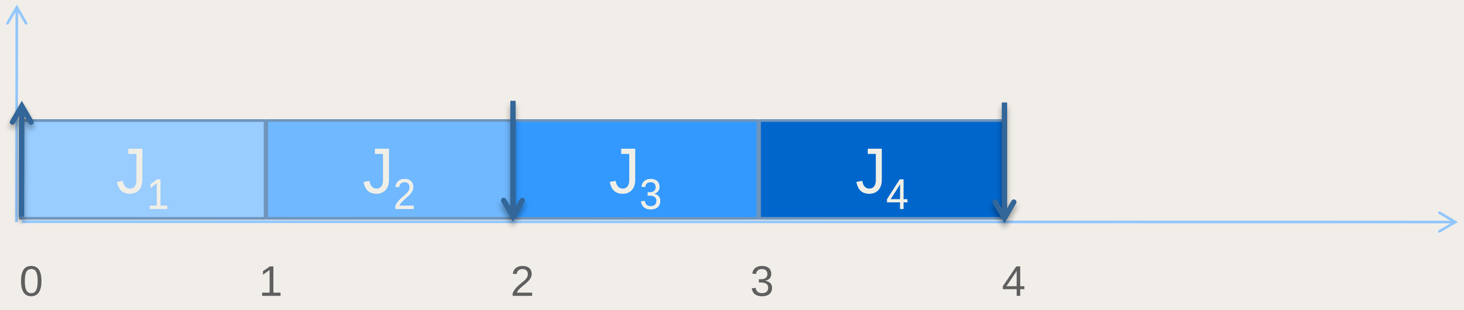
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1

- We can schedule them using earliest-deadline-first(EDF) strategy optimally

Motivation

- An example for classic real-time tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1



Motivation

- Execution time is estimated by system-designer

Motivation

- Execution time is estimated by system-designer
- With different tools we'll get different estimations

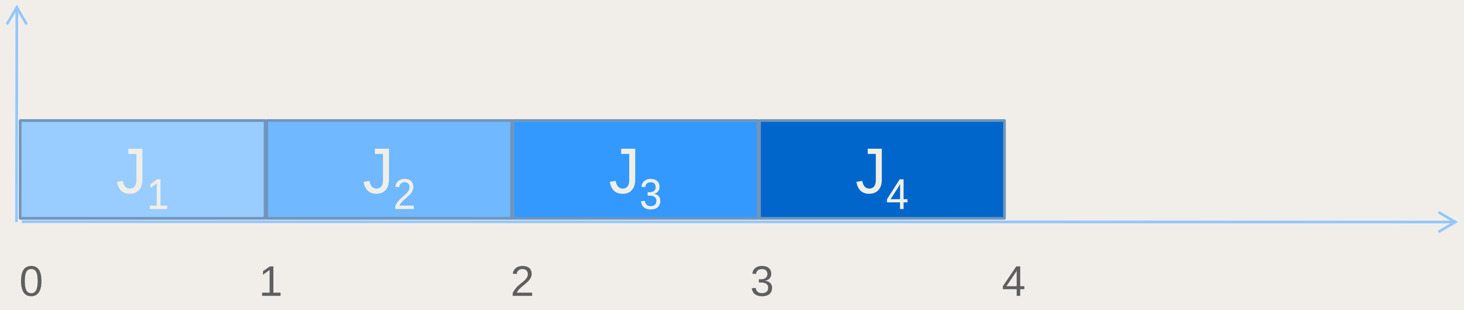
Motivation

- Execution time is estimated by system-designer
- With different tools we'll get different estimations
- Sometimes a part of the tasks must pass certifications from authorities, who would give a pessimistic estimation

Motivation

- An example for mixed-criticality tasks

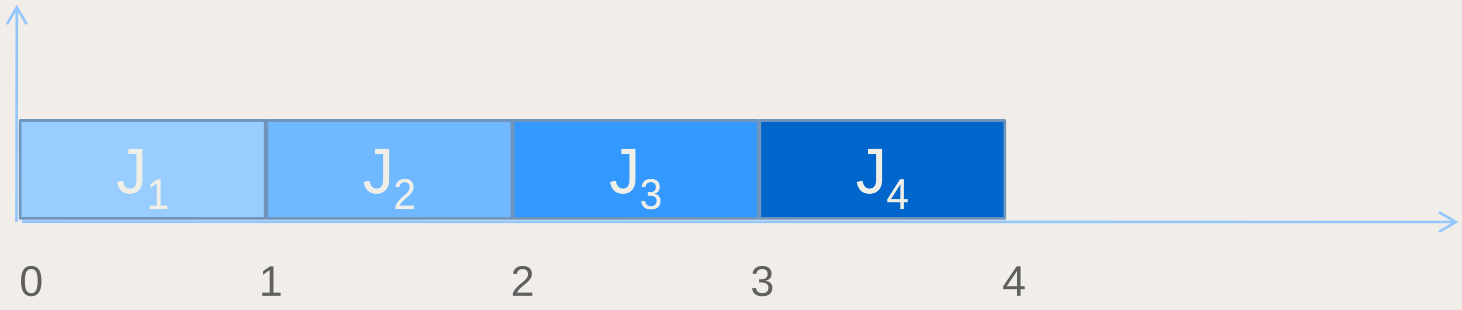
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1



Motivation

- An example for mixed-criticality tasks

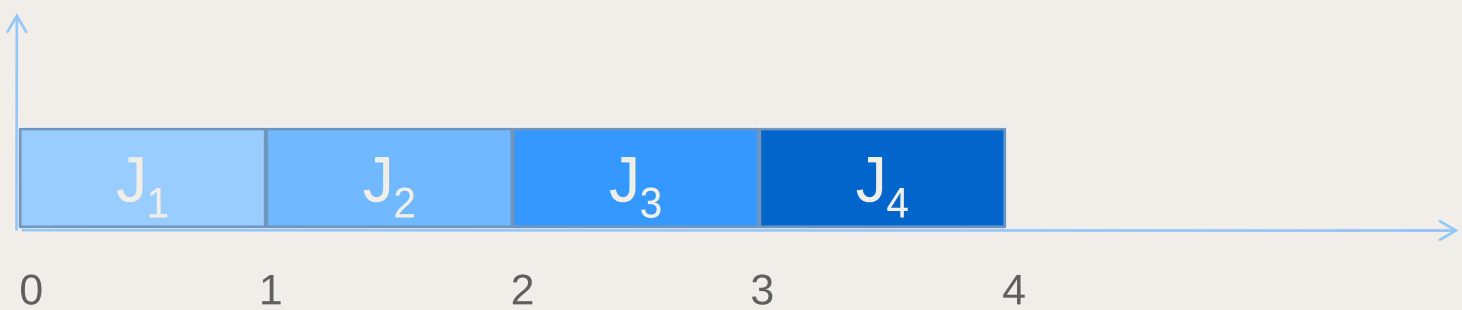
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1



Motivation

- An example for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



Motivation

- An example for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

Motivation

- An example for mixed-criticality tasks

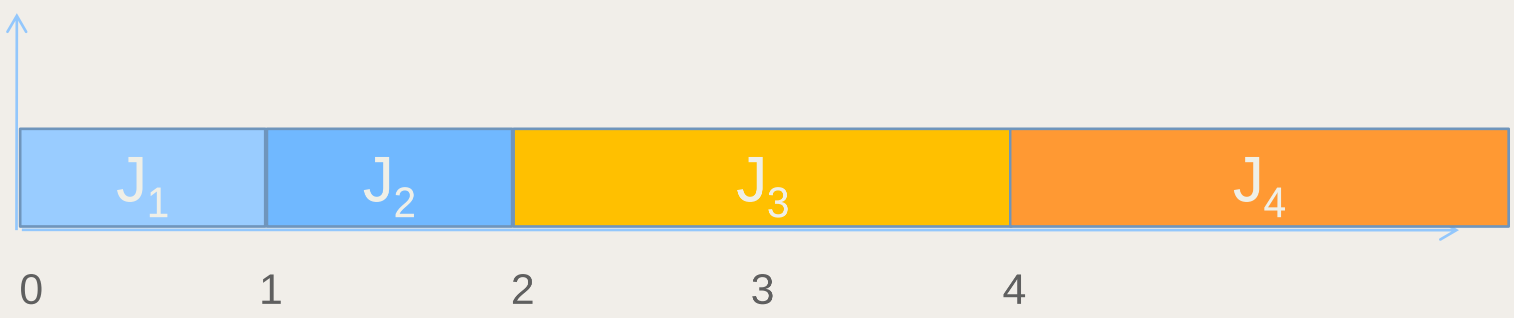
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

Motivation

- An example for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

These two tasks don't have to be certified

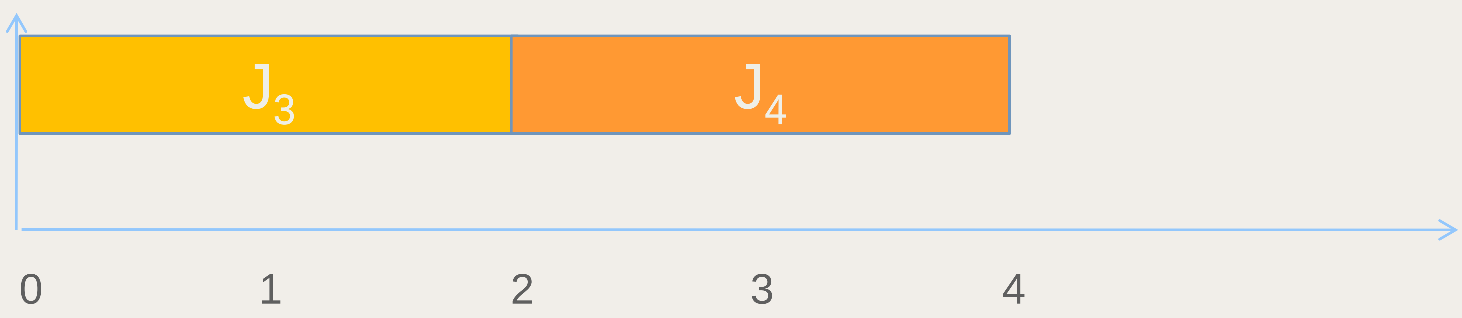


Motivation

- An example for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

These two tasks don't have to be certified

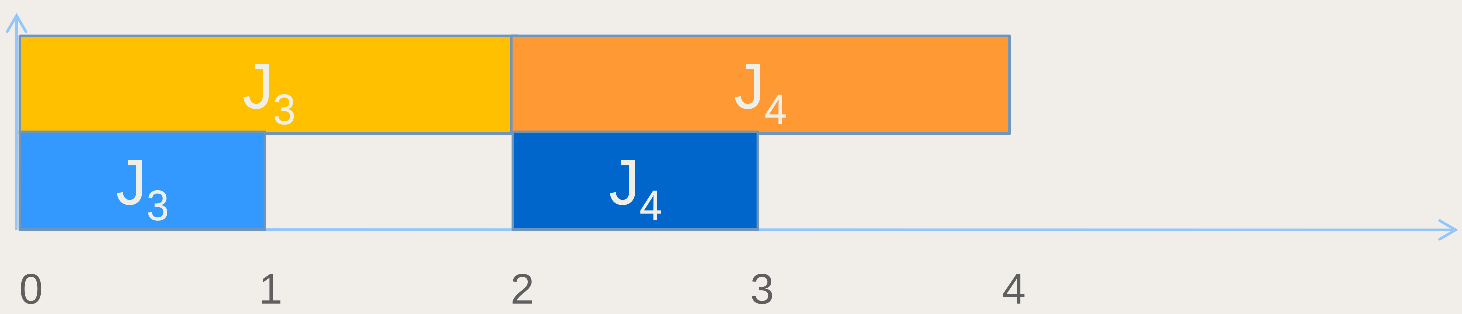


Motivation

- An example for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

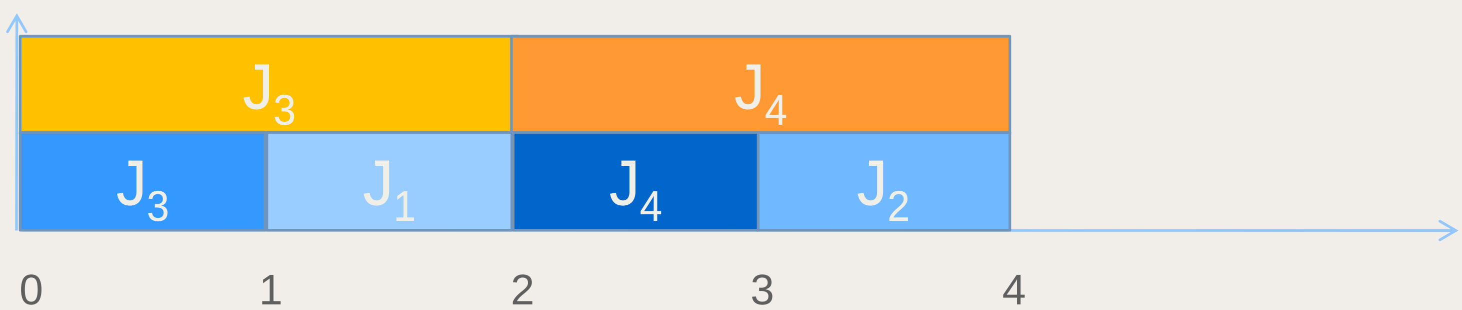
These two tasks don't have to be certified



Motivation

- An example for mixed-criticality tasks

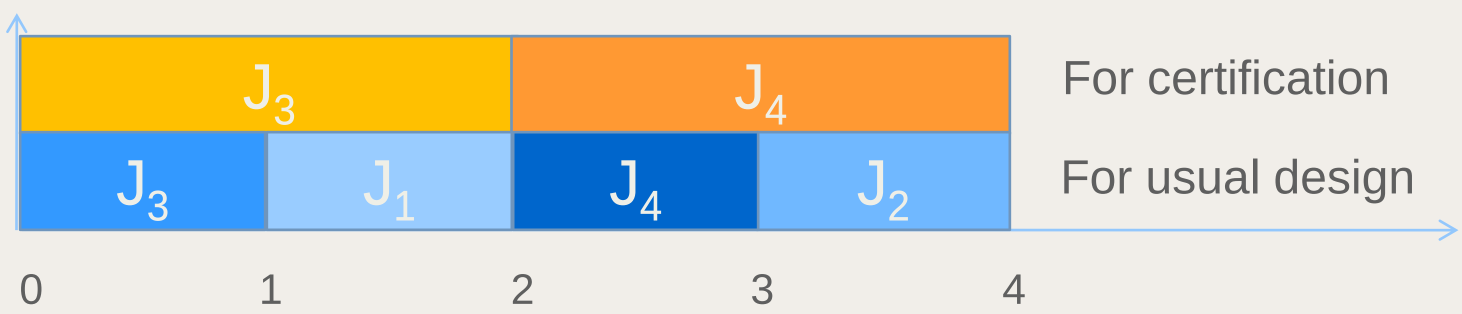
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



Motivation

- An example for mixed-criticality tasks

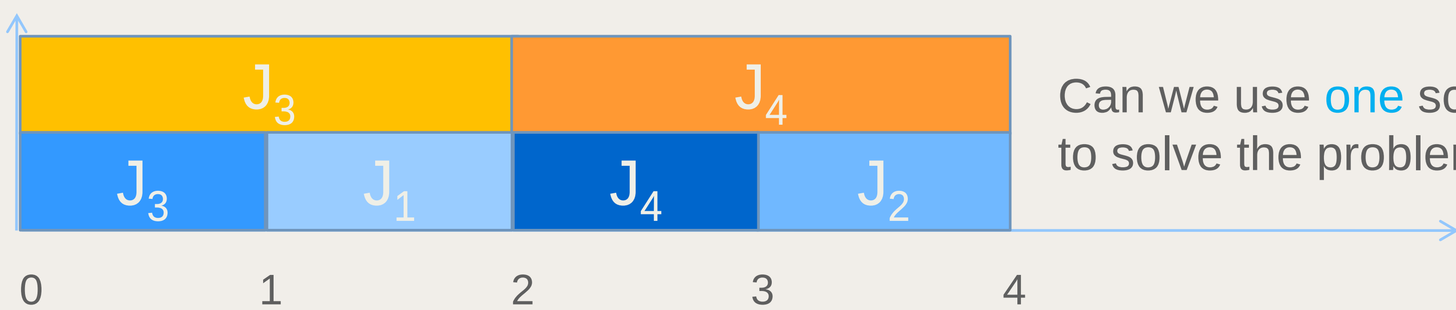
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



Motivation

- An example for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

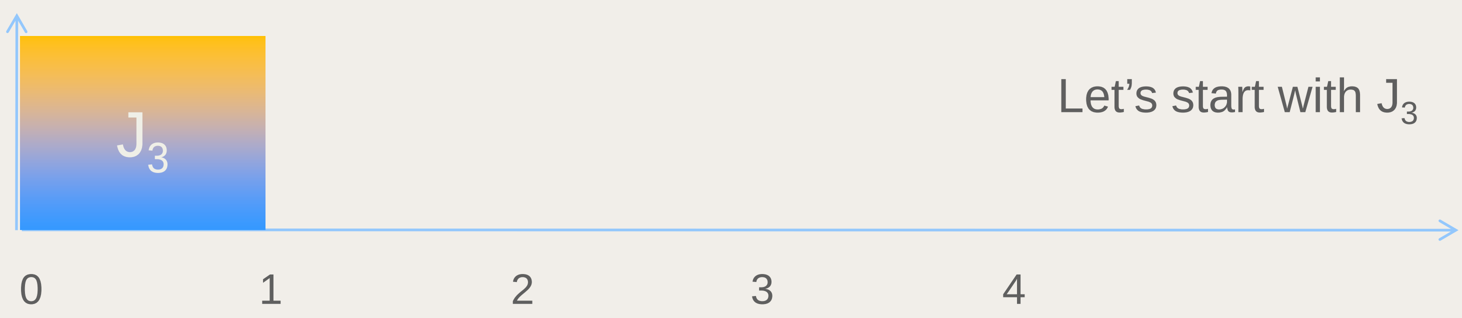


Can we use one schedule to solve the problem?

Motivation

- A solution for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

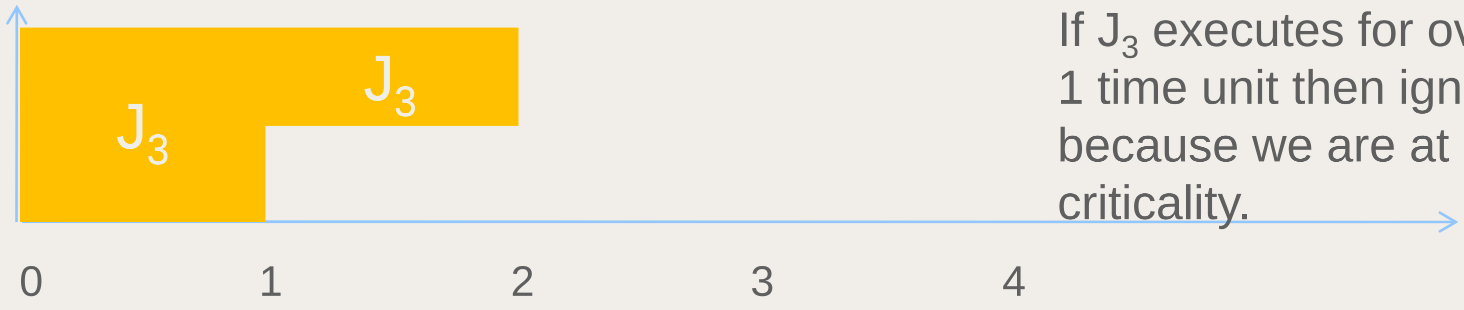


Let's start with J_3

Motivation

- A solution for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

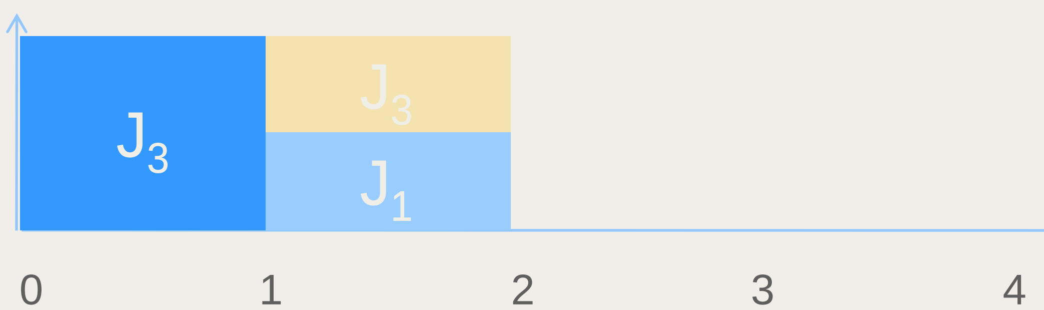


If J_3 executes for over 1 time unit then ignore J_1, because we are at high criticality.

Motivation

- A solution for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



If J_3 stops by time 1, we'll execute J_1, because we are at low criticality.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Motivation

- A solution for mixed-criticality tasks

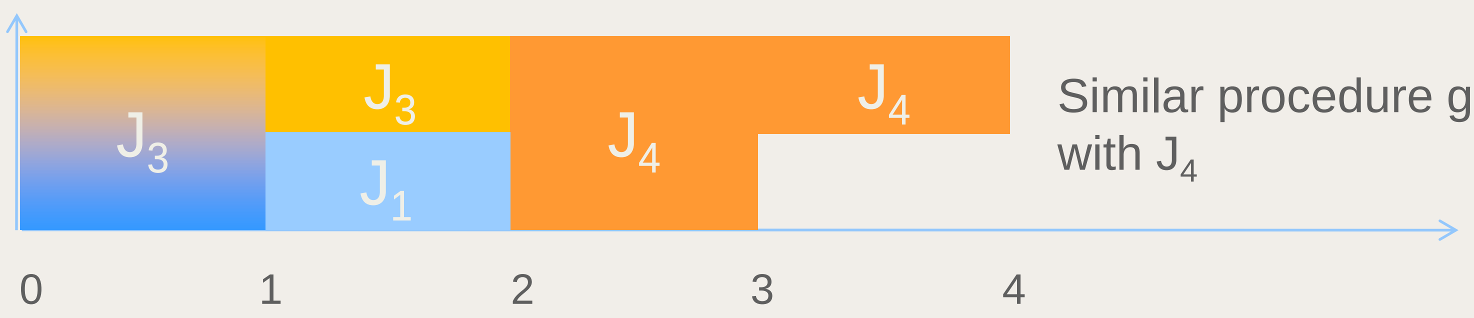
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

Similar procedure goes with J_4

Motivation

- A solution for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

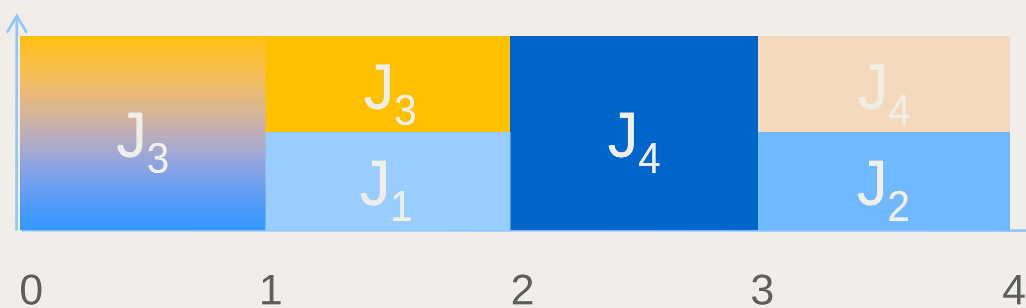


Similar procedure goes with J_4

Motivation

- A solution for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2

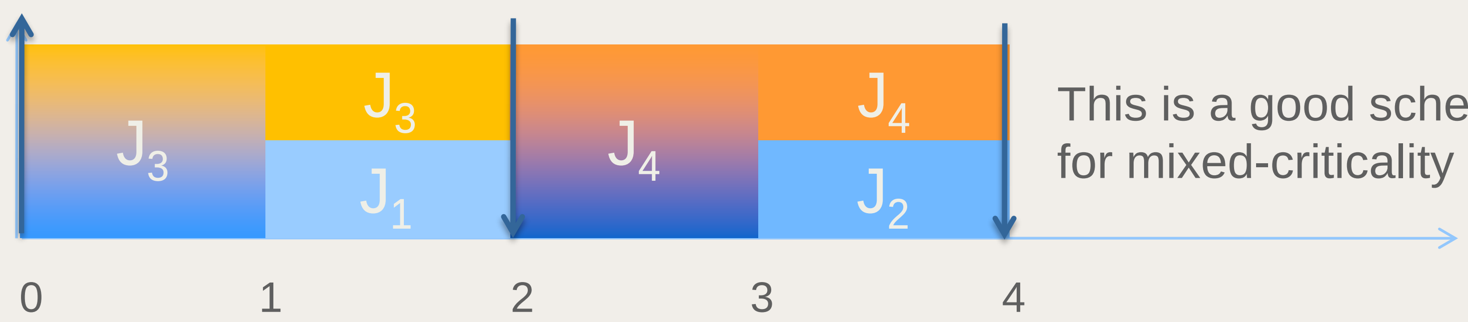


Similar procedure goes with J_4

Motivation

- A solution for mixed-criticality tasks

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



This is a good schedule for mixed-criticality tasks

Model

- On the base of classic real-time task model, we add a parameter x_i, denoting the **criticality** of this task.

	Release time(A_i)	Deadline (D_i)	Criticality (x_i)	Execution time for low-criticality	Execution time for high-criticality
J_1	0	2	1	1	1
J_2	0	4	1	1	1
J_3	0	4	2	1	2
J_4	0	4	2	1	2

Model

- We assume that
  - Defined execution time means worst-case execution time. The actual execution time will only be known when a task really executes;
  - Execution time for low-criticality is no greater than the time for high criticality.

Model

- We define a task set as mixed-criticality schedulable(MC-schedulable) if there exists a schedule that:
  - If every task's execution time is no greater than defined execution time at low criticality, every task will meet the deadline;
  - If at least one criticality-2 task's execution time is greater than defined execution time at low criticality, every criticality-2 task will meet the deadline.

Model

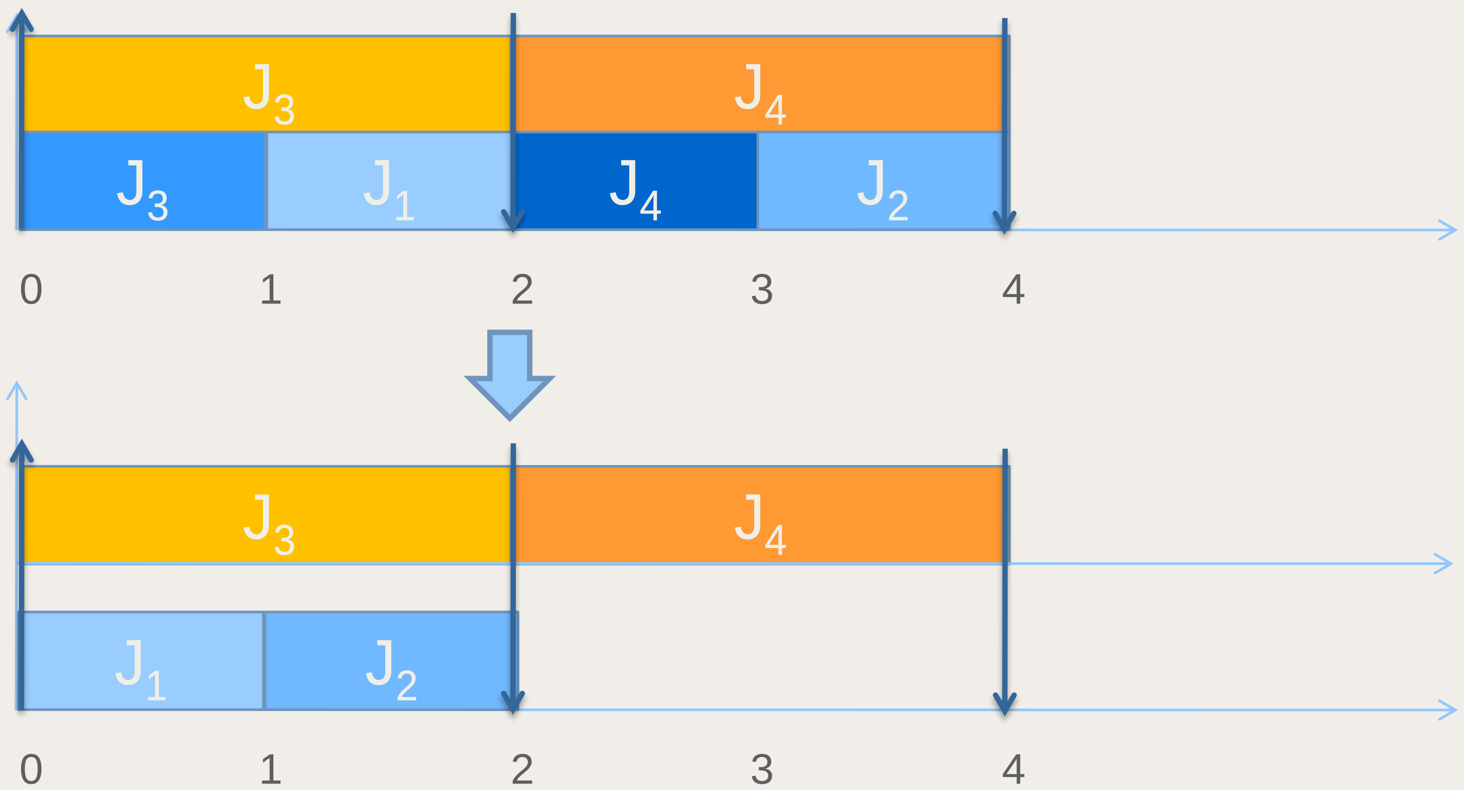
- The MC-schedulability testing is NP-hard in the strong sense even if:
  - There are only two criticalities;
  - Every job's release time is the same.

Solution

- Because MC-schedulability test is highly intractable, we focus on the following approximation test:
  - If an instance is MC-schedulable on a processor, this instance will pass our proposed test on any Φ times faster processor, or to say, a Φ-speed processor.

Solution

- A schedulability test with Φ=2 is trivial by processor-sharing strategy.

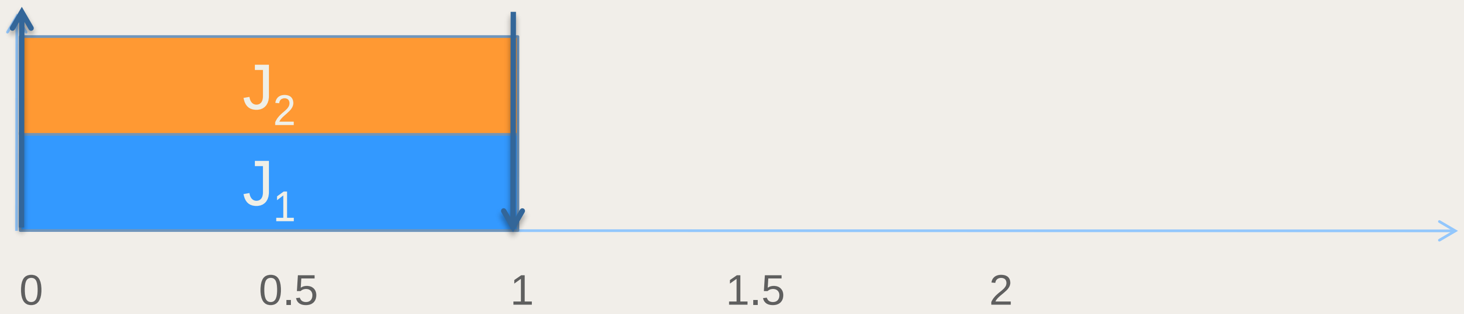


Solution

- Classical schedule algorithm:
  - Earliest-deadline-first: EDF $\Phi=2$

	Release time(A_i)	Deadline (D_i)	Criticality (x_i)	Execution time for low-criticality	Execution time for high-criticality
J_1	0	1	1	1	1
J_2	0	1	2	0	1

  - This is MC-schedulable:



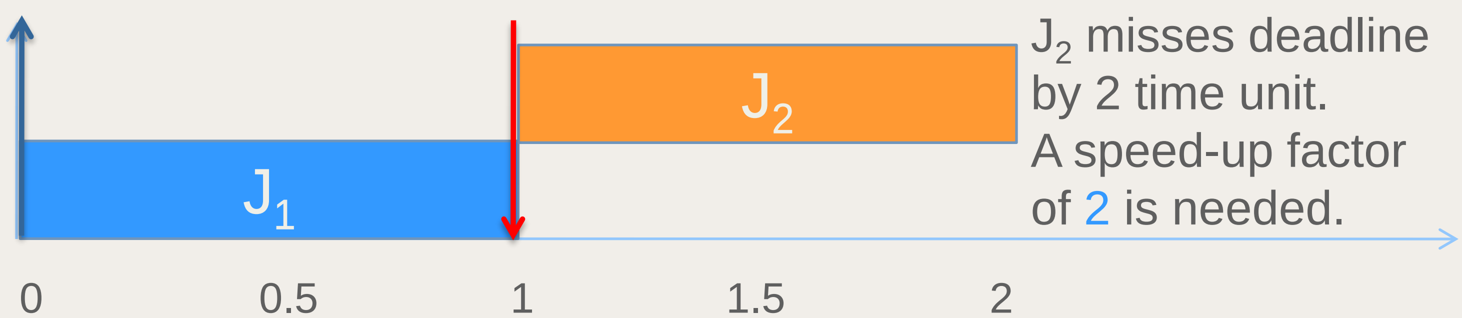
THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Solution

- Classical schedule algorithm:
  - Earliest-deadline-first(EDF): Φ=2

	Release time(A_i)	Deadline (D_i)	Criticality (x_i)	Execution time for low-criticality	Execution time for high-criticality
J_1	0	1	1	1	1
J_2	0	1	2	0	1

  - But EDF will need a speed-up factor of 2:



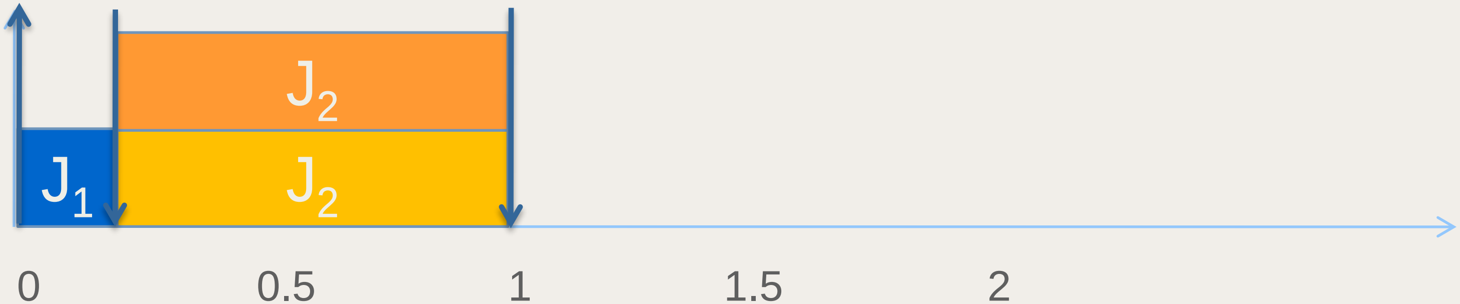
J_2 misses deadline by 2 time unit.
A speed-up factor of 2 is needed.

Solution

- Classical schedule algorithm:
  - Criticality monotonic: $\Phi=\infty$

	Release time(A_i)	Deadline (D_i)	Criticality (x_i)	Execution time for low-criticality	Execution time for high-criticality
J_1	0	0.1	1	0.1	0.1
J_2	0	1	2	0.9	0.9

  - This is MC-schedulable:

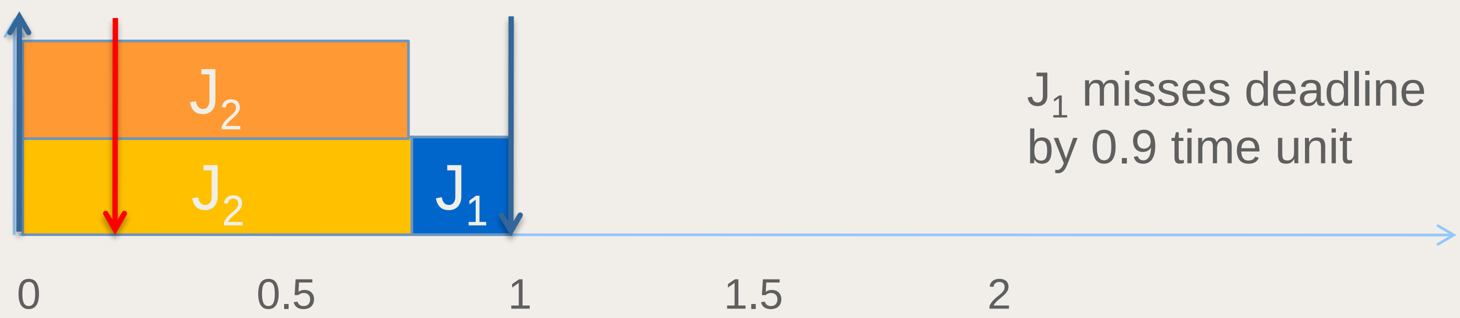


Solution

- Classical schedule algorithm:
  - Criticality monotonic: $\Phi=\infty$

	Release time(A_i)	Deadline (D_i)	Criticality (x_i)	Execution time for low-criticality	Execution time for high-criticality
J_1	0	0.1	1	0.1	0.1
J_2	0	1	2	0.9	0.9

  - But criticality-monotonic will fail:



J_1 misses deadline by 0.9 time unit

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):
  - The algorithm seeks one task to be the lowest priority if:
    - This task is criticality-1, and all the other tasks have used criticality-1 execution time, and this task can still meet its deadline.
    - This task is criticality-2, and all the other tasks have used criticality-2 execution time, and this task can still meet its deadline.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Solution

- Our previous result is:
  - OCBP algorithm will need at most Φ=1.618 speed-up factor to schedule any MC-schedulable instance with only 2 criticalities.
    - It was accepted by 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010.

Solution

- In this paper we've shown the situation where there are n different criticalities.
- $\Phi_n = n$ is still sufficient by processor-sharing.
- We assume that:
  - The tasks are not repetitive;
  - Every criticality-(i+k) task will also be criticality-i task;
  - Execution time at higher criticality is no smaller than execution time at lower criticality.

Solution

- The result is:
  - OCBP will schedule any MC-schedulable instance with n criticalities on a Φ_n-speed processor, where

$$\phi_n = \frac{1+\sqrt{1+4\phi_{n-1}^2}}{2}$$

  - The result will converge to $\Phi_n = n/2$ with increasing n.

Solution

- The result is:

n(trivial)	Φ_n	optimal
1	1.0000	1
2	1.6180	1
3	2.1935	1
4	2.7498	1
5	3.2949	1
6	3.8326	1
7	4.3651	1
8	4.8936	1
9	5.4191	1
10	5.9421	1

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Future work

- Extend the current result to periodic/sporadic real-time task model;
- Tighten the current result to an optimal lower bound for OCBP algorithm;
- Explore new algorithms to schedule mixed-criticality system.

Thank you

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL