THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Towards the Design of Certifiable Mixed-Criticality systems

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

Leen Stougie
Vrije Universiteit

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Outline

- **Motivation**
  - Certification requirements in embedded systems
- **Model**
  - Definition of mixed-criticality system
  - Hardness of feasibility test
- **Solution**
  - Why EDF and criticality-monotonic fail
  - OCBP: A new algorithm

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

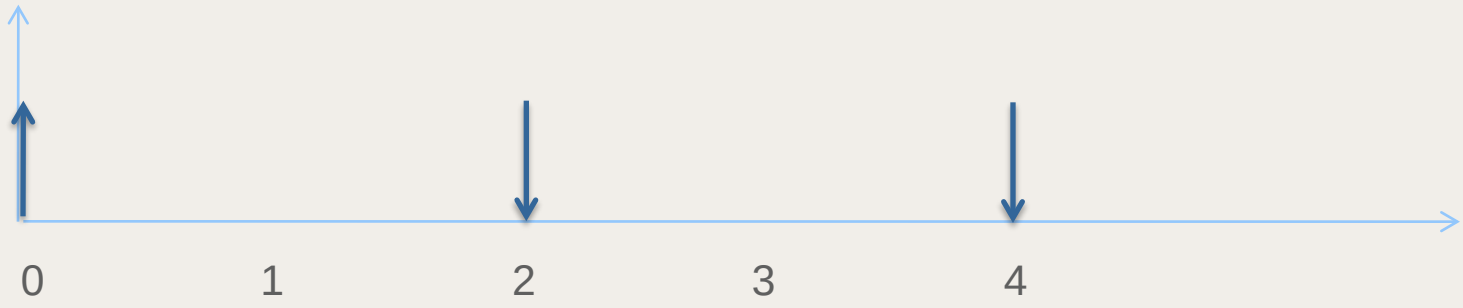
Motivation

- An example for classic real-time jobs
  - Uniprocessor
  - Preemptive
  - Hard real-time
  - Given finite instance of jobs
    - One-pass job set
    - Known release times and deadlines

Motivation

- An example for classic real-time jobs

	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 jobs

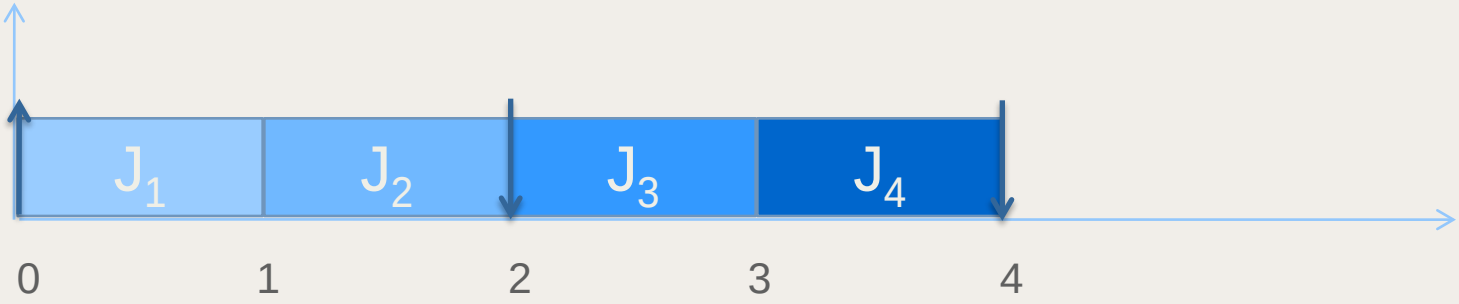
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1

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

Motivation

- An example for classic real-time jobs

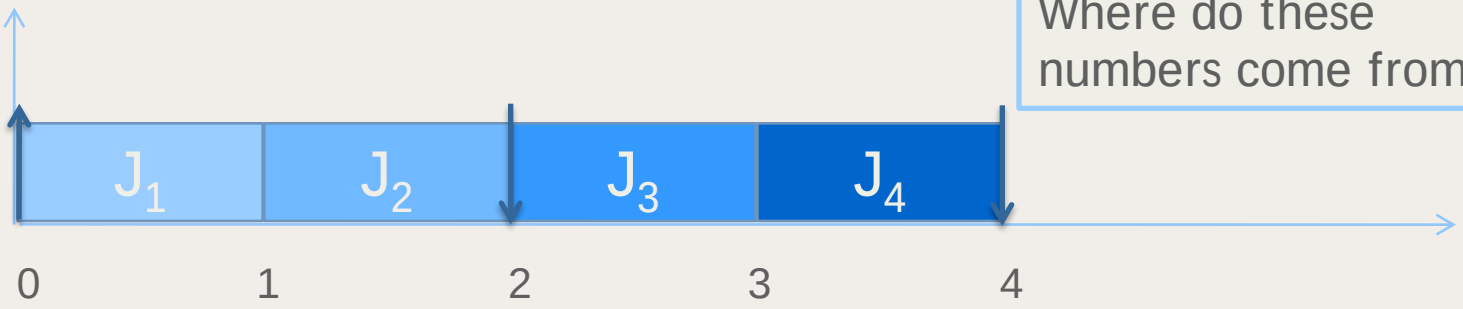
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1



Motivation

- An example for classic real-time jobs

	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



Where do these numbers come from?

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

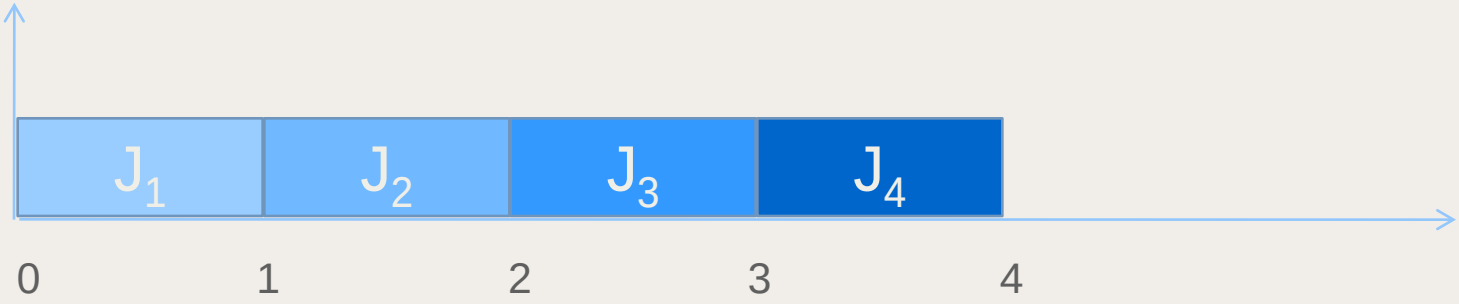
Motivation

- Execution time is estimated
- With different tools we'll get different estimations
- Sometimes a part of the system must pass certification from authorities. They will simulate the system to check validity.
- Authorities may use more pessimistic estimations.

Motivation

- An example for mixed-criticality jobs

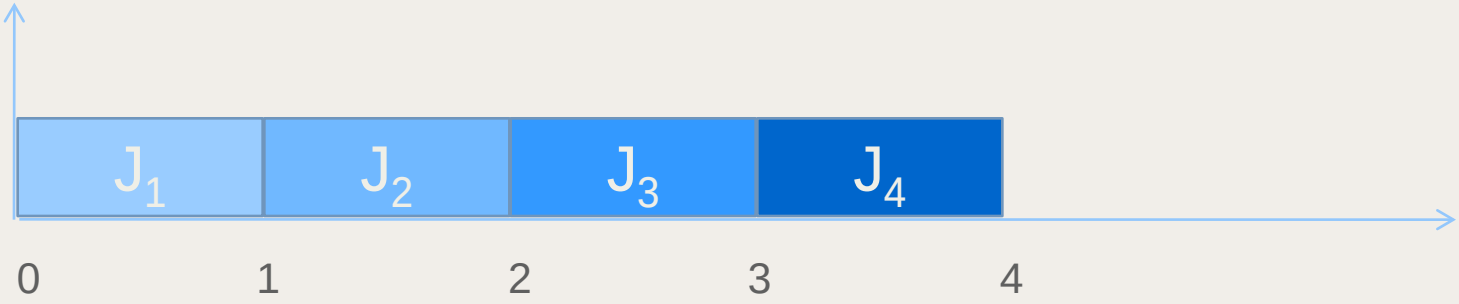
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	1
J_4	0	4	1



Motivation

- An example for mixed-criticality jobs

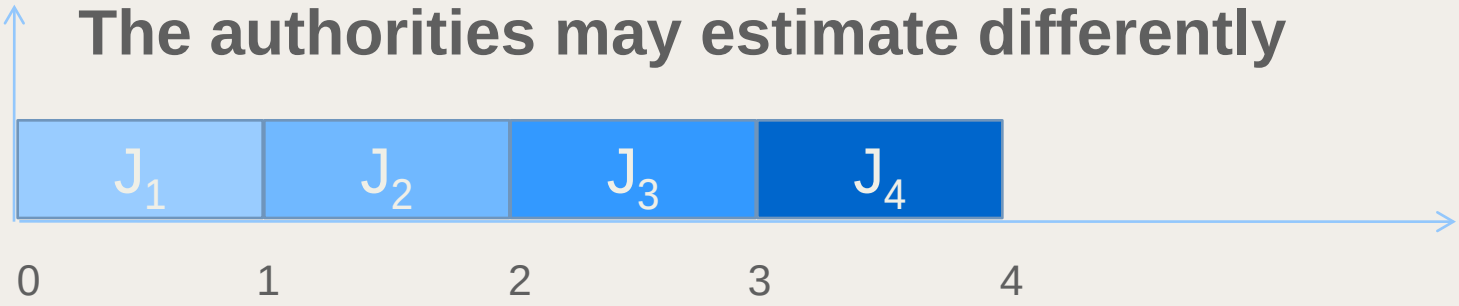
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	Not critical, like camera, radio, heater		
J_2			
J_3	Safety-critical, like flight control system		
J_4			



Motivation

- An example for mixed-criticality jobs

	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

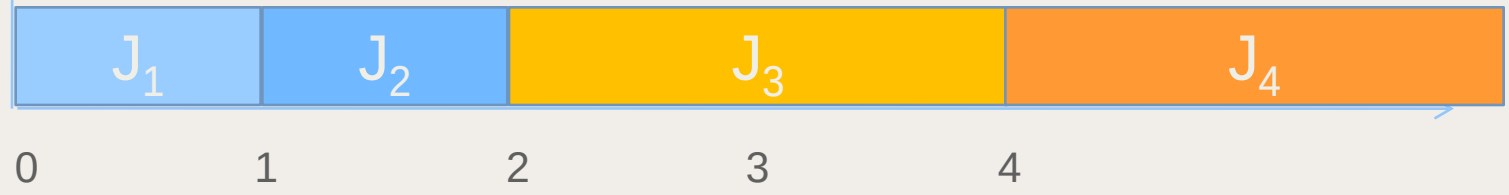


Motivation

- An example for mixed-criticality jobs

	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

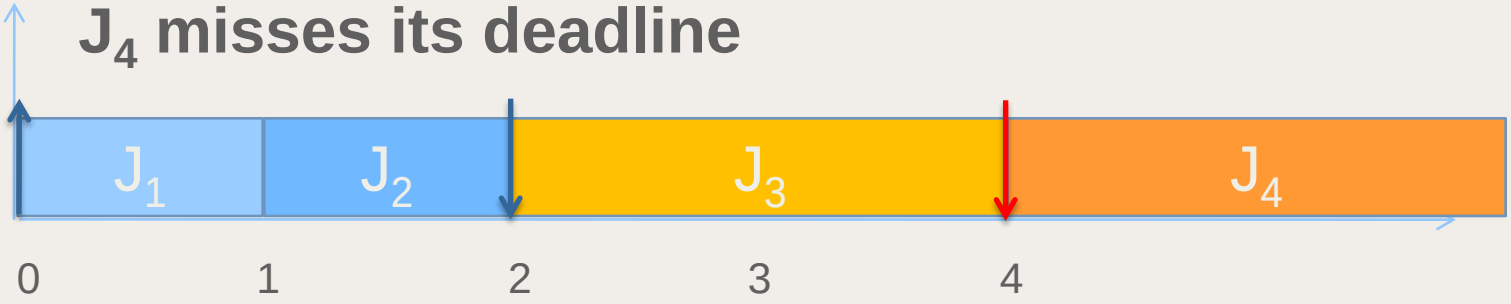
With previous EDF schedule:



Motivation

- An example for mixed-criticality jobs

	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 jobs

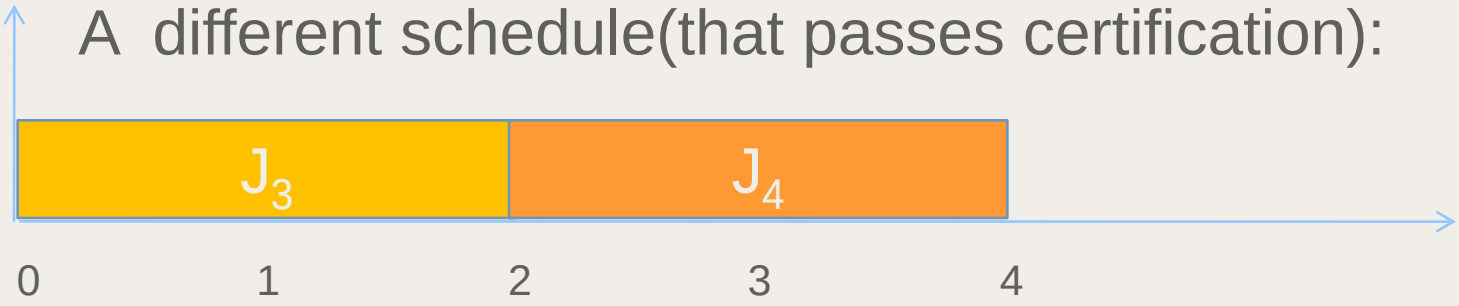
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
These two jobs don't have to be certified			
J_3	0	4	2
J_4	0	4	2

Motivation

- An example for mixed-criticality jobs

	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
These two jobs don't have to be certified			
J_3	0	4	2
J_4	0	4	2

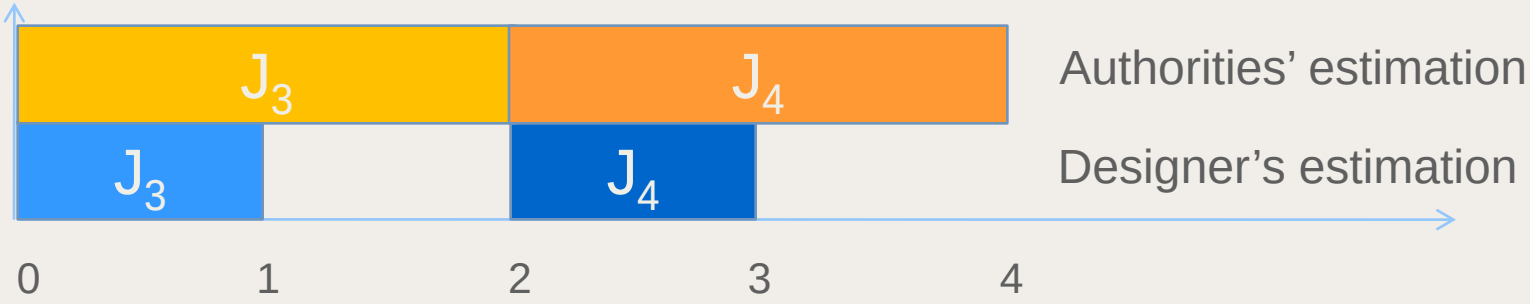
A different schedule(that passes certification):



Motivation

- An example for mixed-criticality jobs

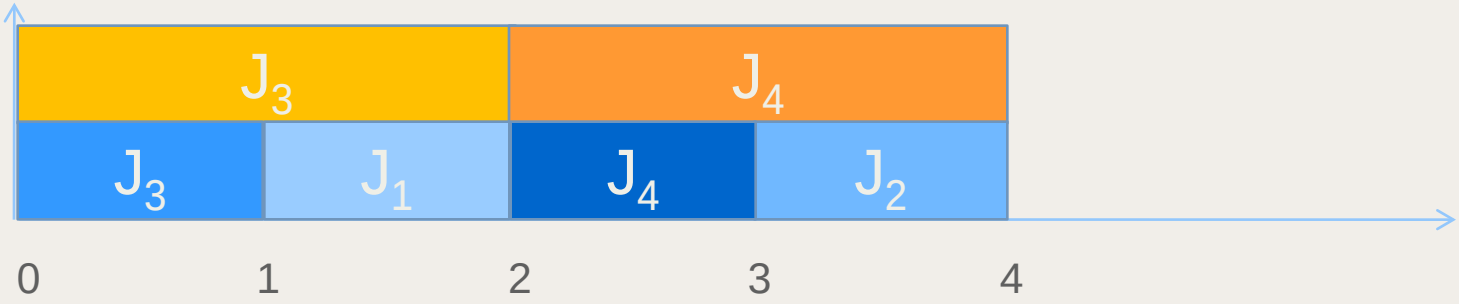
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1 0 2 1 These two jobs don't have to be certified J_2 0 4 1			
J_3	0	4	2
J_4	0	4	2



Motivation

- An example for mixed-criticality jobs

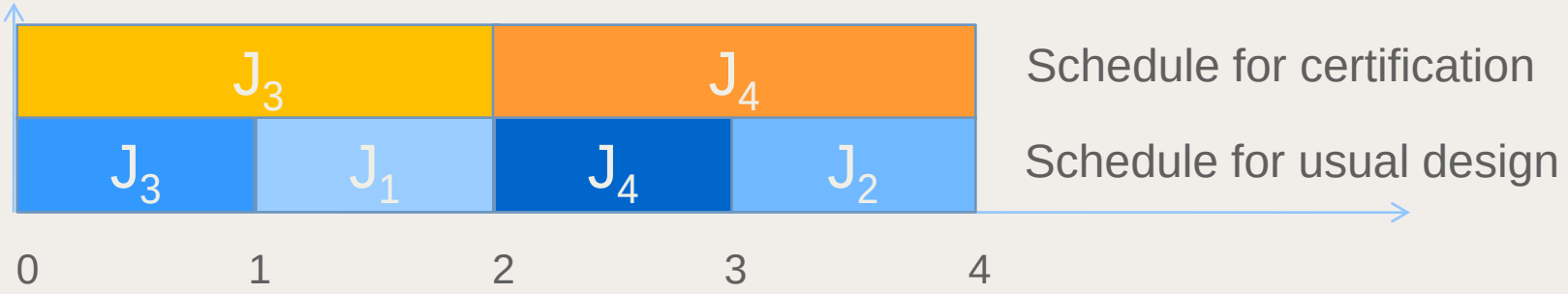
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



Motivation

- An example for mixed-criticality jobs

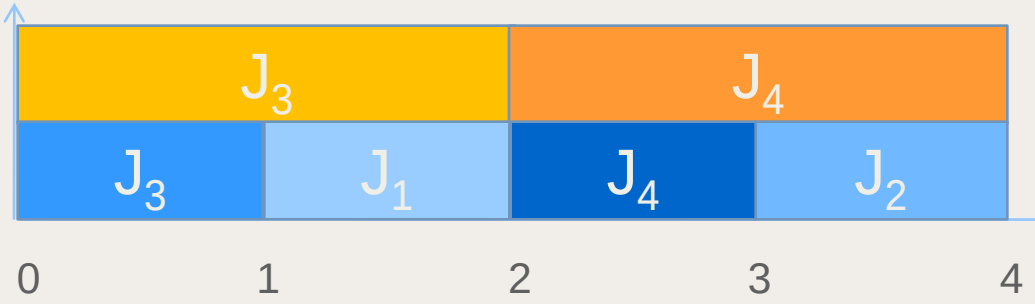
	Release time(A_i)	Deadline(D_i)	Execution time(C_i)
J_1	0	2	1
J_2	0	4	1
J_3	0	4	2
J_4	0	4	2



Motivation

- An example for mixed-criticality jobs

	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

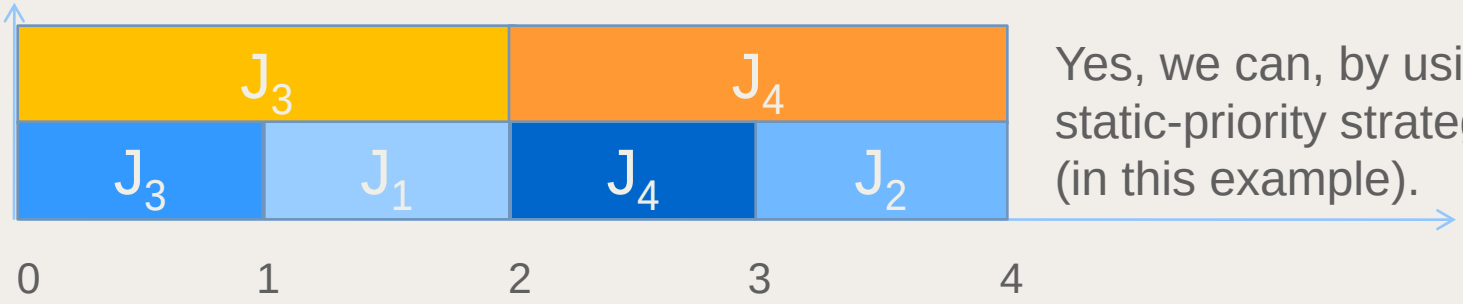


Can we use one strategy to solve the problem?

Motivation

- An example for mixed-criticality jobs

	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

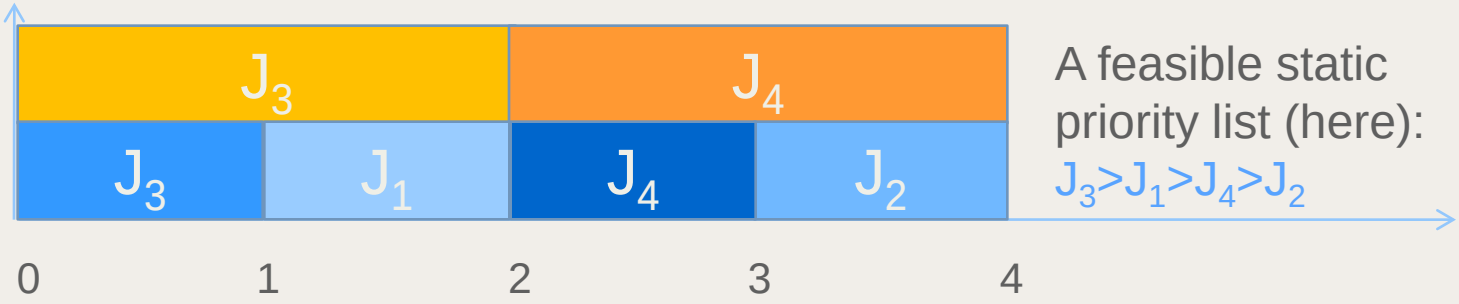


Yes, we can, by using a static-priority strategy (in this example).

Motivation

- An example for mixed-criticality jobs

	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



A feasible static priority list (here): $J_3 > J_1 > J_4 > J_2$

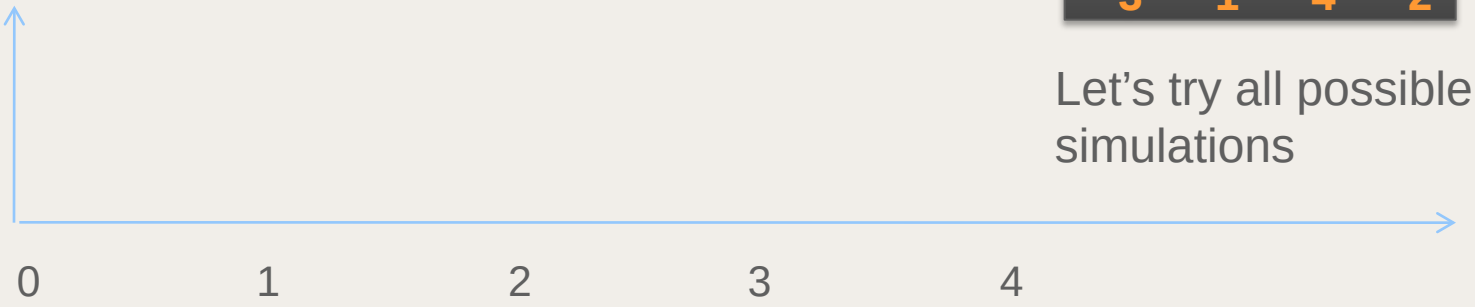
Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$

Let's try all possible simulations



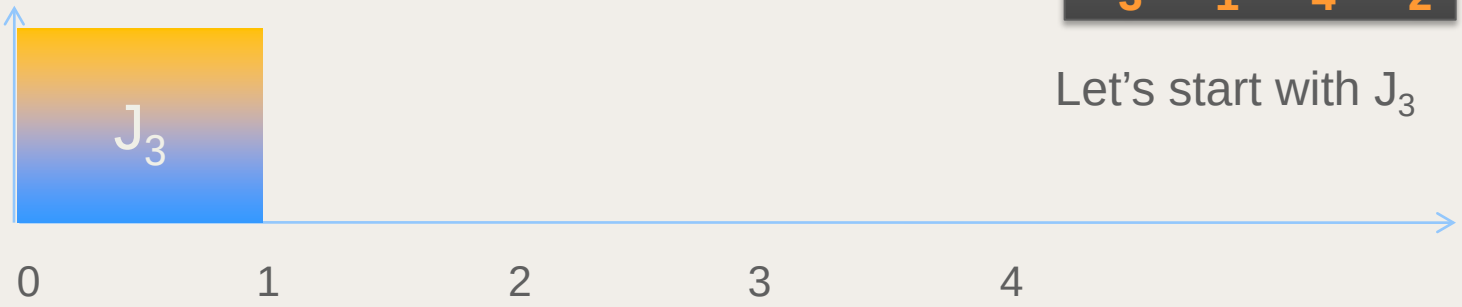
Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$

Let's start with J_3

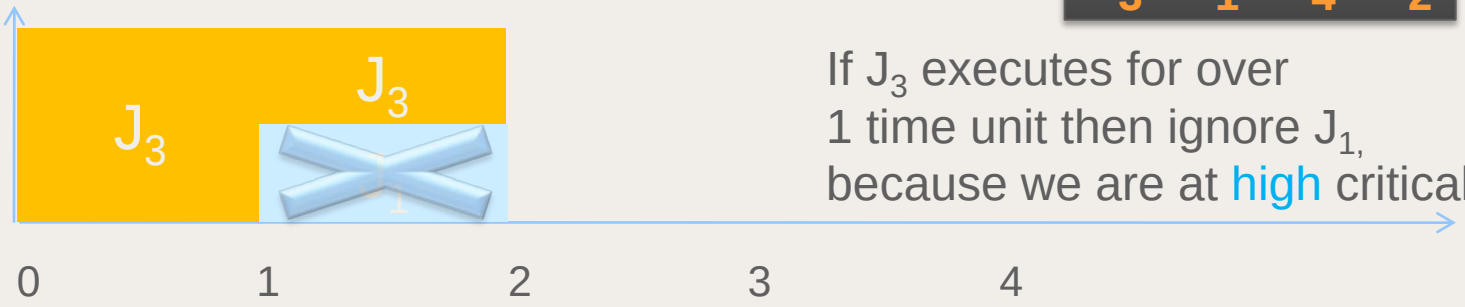


Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$



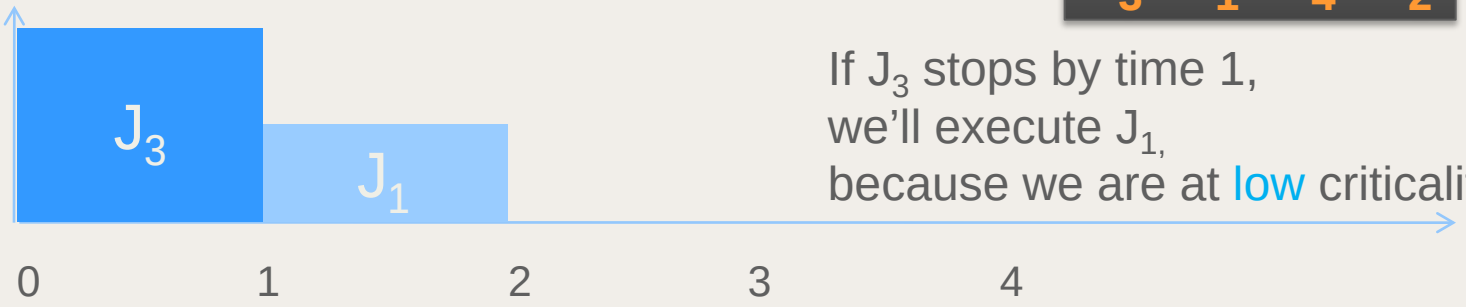
If J_3 executes for over 1 time unit then ignore J_1, because we are at high criticality.

Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$



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

Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$

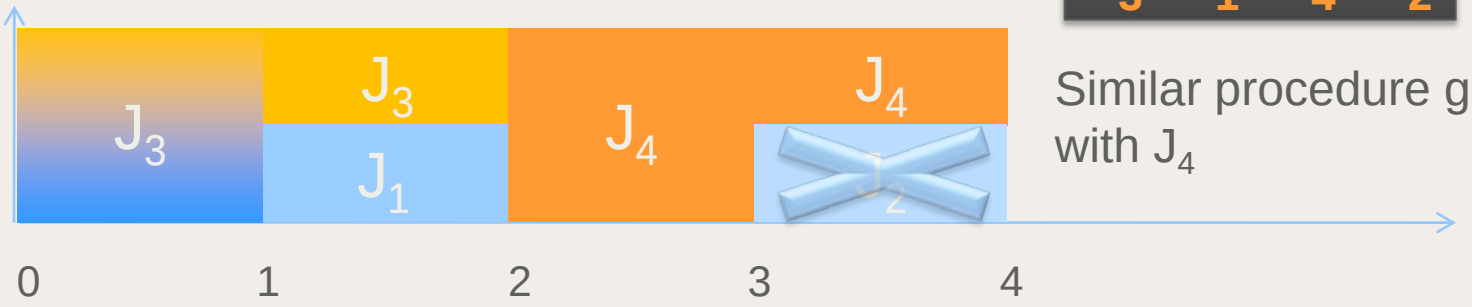
Similar procedure goes with J_4

Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$



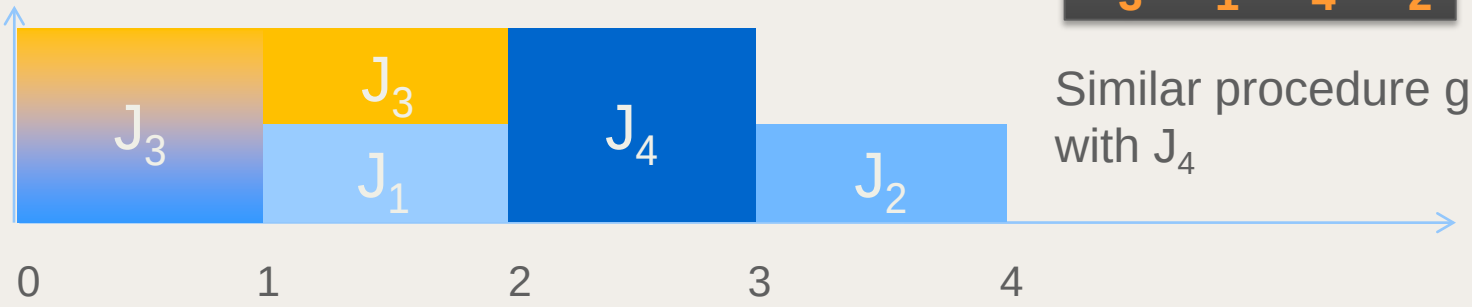
Similar procedure goes with J_4

Motivation

- A solution for mixed-criticality jobs

	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

$J_3 > J_1 > J_4 > J_2$

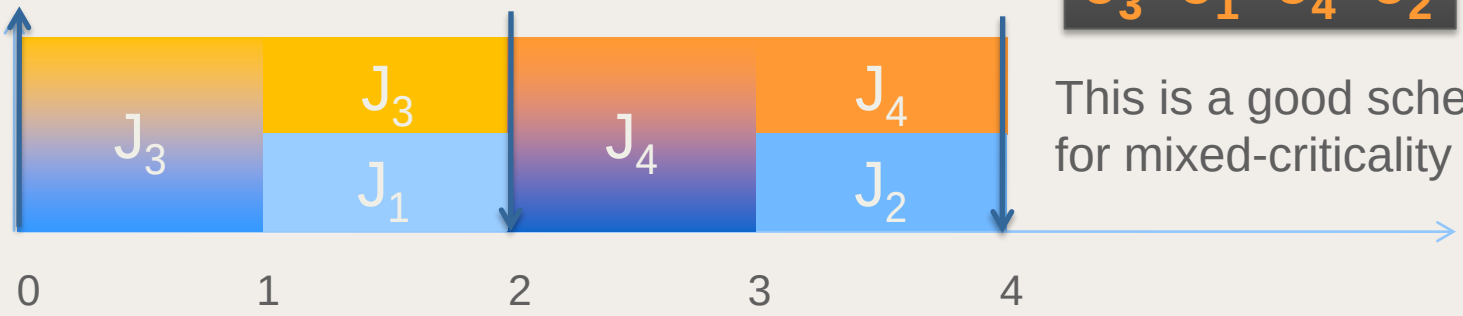


Similar procedure goes with J_4

Motivation

- A solution for mixed-criticality jobs

	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



$J_3 > J_1 > J_4 > J_2$

This is a good schedule for mixed-criticality jobs

Model

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

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

Model

- We define a job set as mixed-criticality schedulable(MC-schedulable) if there exists a schedule such that:
  - If every job uses less than specified execution time at low criticality, every job will meet its deadline;
  - If at least one high-criticality job uses more than specified execution time at low criticality, every high-criticality job will meet its deadline.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Intractability Result

- Determining whether a given instance is MC-schedulable is NP-hard in the strong sense even if:
  - Every job's release time is exactly the same;
  - Jobs are preemptive.

Solution

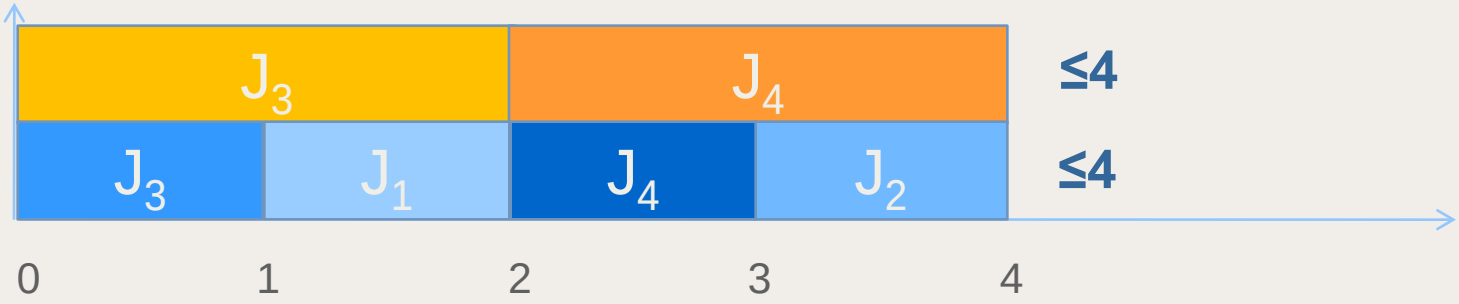
- Processor Speed-up Factor
  - A scheduling algorithm has a processor speed-up factor Φ if
    - it can schedule any MC-schedulable instance
    - on a processor Φ times as fast
    - without any knowledge of the optimal schedule
      - The optimal schedule may even be *clairvoyant*
  - We use speed-up factor as a measurement
    - Lower means better, $\Phi=1$ means optimal.

Solution

- We seek scheduling algorithms with low processor speed-up factor Φ:
  - An algorithm can schedule any MC-schedulable (or full-utilized) instance on a Φ -speed processor.

Solution

- A schedulability test with $\Phi=2$ is trivial by worst-case reservation strategy.
  - Because the summation of time demands in each criticality can not exceed the overall available processor time.



Solution

- A schedulability test with $\Phi=2$ is trivial by worst-case reservation strategy.
  - Because the summation of time demands in each criticality can not exceed the overall available processor time.

Solution

- A schedulability test with $\Phi=2$ is trivial by worst-case reservation strategy.
  - Because the summation of time demands in each criticality can not exceed the overall available processor time.

Solution

- Classical scheduling algorithms
  - Earliest deadline first(EDF): $\Phi=2$
    - It's no better than worst-case reservation
  - Criticality monotonic: $\Phi=\infty$
    - It can need arbitrarily high speed-up factor to meet all deadlines.

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):
  - It's very similar to "Audsley's Approach"
  - Repeatedly determine which remaining job can be assigned with lowest-priority.

Solution

- Own-Criticality-Based-Priority algorithm:
  - *J* may be assigned with lowest priority if it meets its deadline as the lowest-priority job, when all other jobs executes for their worst-case execution time at *J*'s criticality.
    - If *J* is of high criticality, it will assume all other jobs use maximum possible execution time;
    - If *J* is of low criticality, it will assume all other jobs use low-criticality execution time.
      - **Otherwise we can just drop *J*.**

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

  - For J_2, if all other jobs use low-criticality time, total time demand is 4, J_2 can be the lowest-priority job.

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

  - For J_1, if all other jobs use low-criticality time, total demand is 4, too. J_1 can not be the lowest-priority job.

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

  - For J_1, if all other jobs use low-criticality time, total demand is 4, too. J_1 can not be the lowest-priority job.

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

  - For J_4, if all other jobs use high-criticality time, total demand is 5, J_4 can be the lowest-priority job with $\Phi=1.2$.

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

  - J_1 and J_3 can both be the lowest-priority job.

Solution

- Own-Criticality-Based-Priority algorithm (OCBP algorithm):

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

  - Final priority order:
    - $J_1 > J_3 > J_4 > J_2$, or $J_3 > J_1 > J_4 > J_2$.

$\Phi = 1.2$

Result

- Our result is:
  - OCBP algorithm will need at most $\Phi=1.618$ speed-up factor to schedule any MC-schedulable instance with 2 criticalities.
  - OCBP algorithm runs in polynomial time.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Future work

- Extend the current result to periodic/sporadic real-time job model;
- Consider practical issues, like jitters, context-switches, or interruptions;
- Explore new algorithms to schedule mixed-criticality systems.

Thank you

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL