40th IEEE Real-Time Systems Symposium (RTSS 2019)

Necessary Feasibility Analysis for Mixed-Criticality Task Systems on Uniprocessor

Hoon Sung Chwa
DGIST

Hyeongboo Baek
Incheon National University

Jinkyu Lee
Sungkyunkwan University

Highlights

The paper provides

tight necessary feasibility tests

for **mixed-criticality (MC)** systems on uniprocessor

the first study that yields non-trivial results
for MC necessary feasibility:

- Reducing a set of MC task sets whose feasibility is unknown by existing studies
- Identifying unique issues of developing necessary feasibility tests for MC systems

“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?

“Feasibility” of timing guarantees

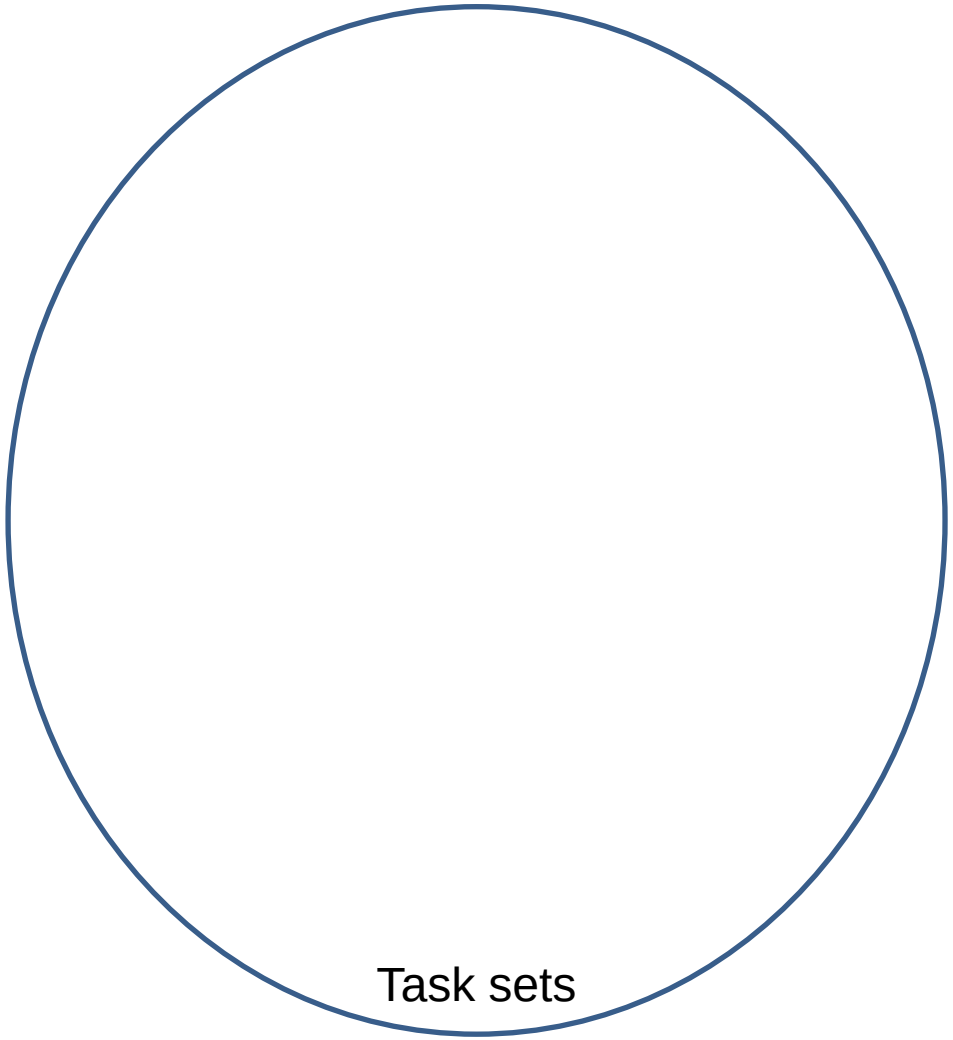
- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility
    (Finding “Yes” answers)
  - Addressing necessary feasibility
    (Finding “No” answers)

“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)

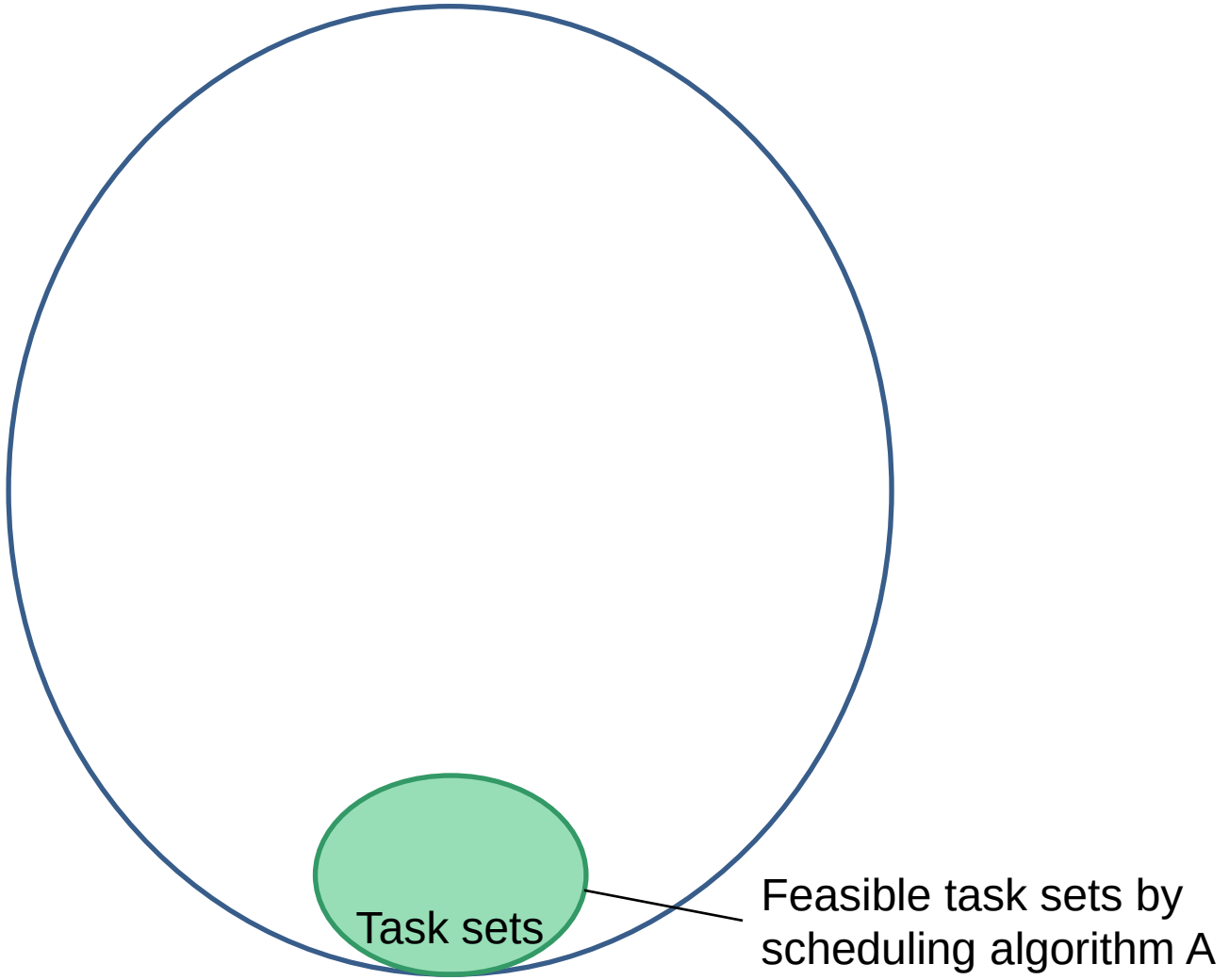
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)



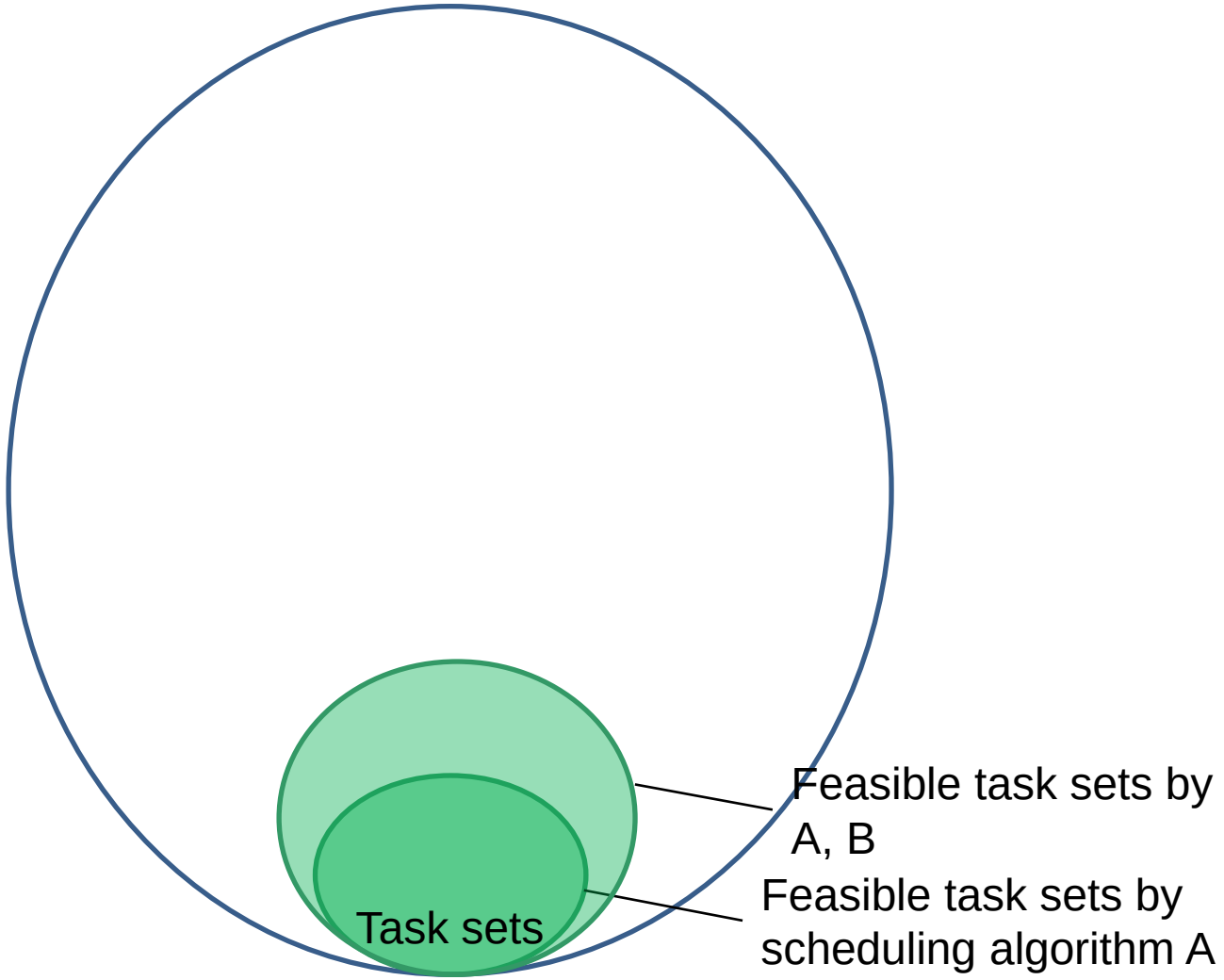
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)



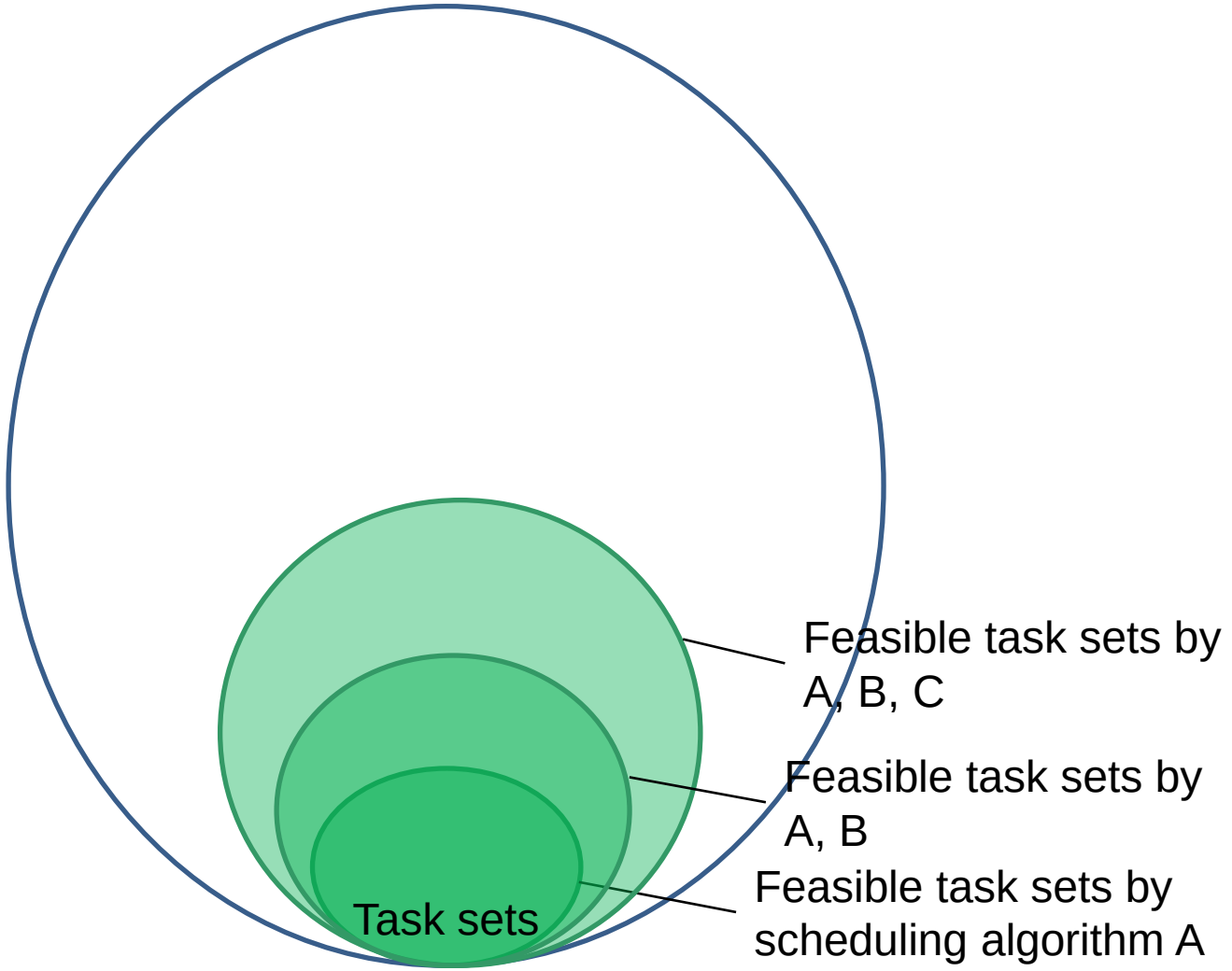
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)



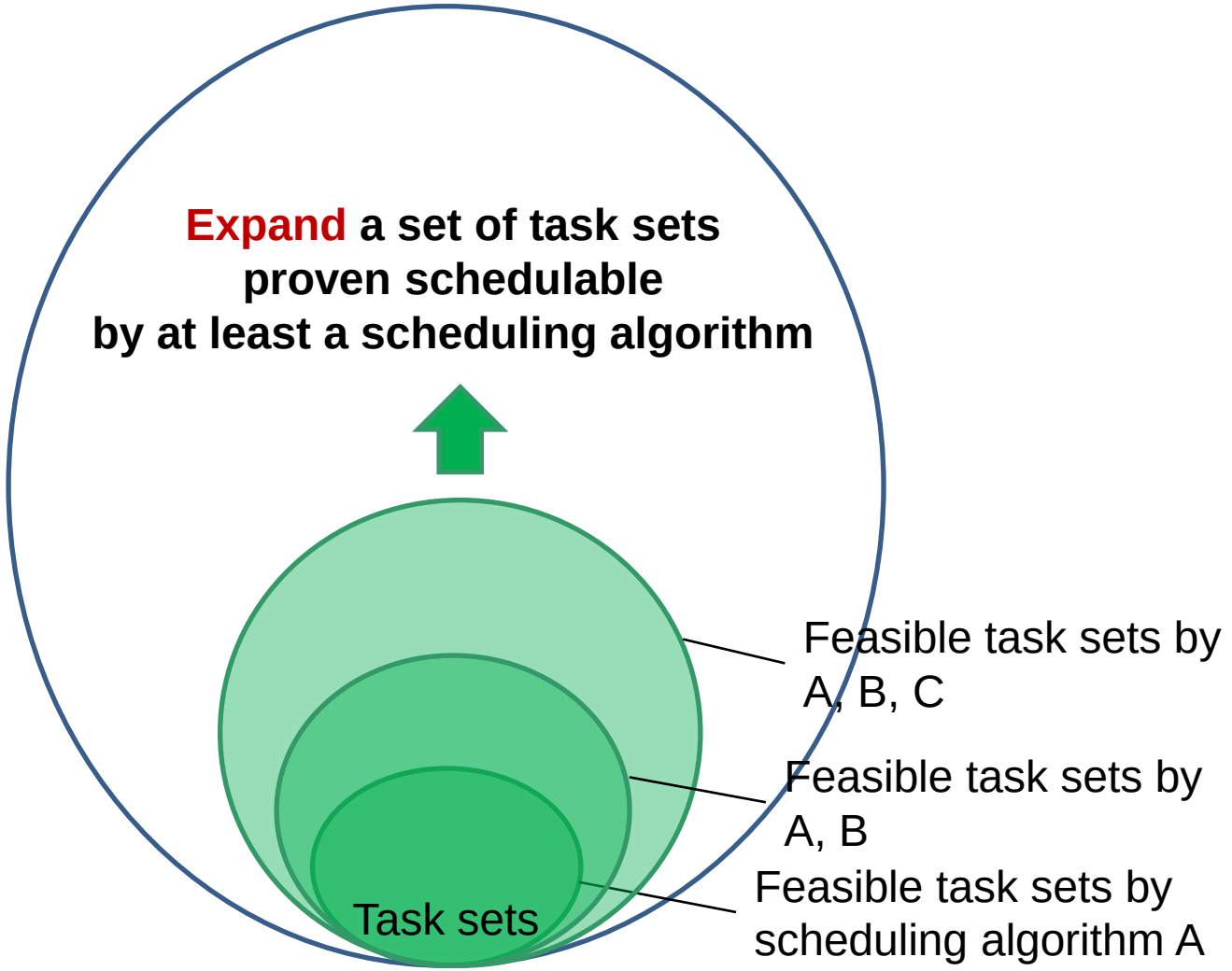
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)



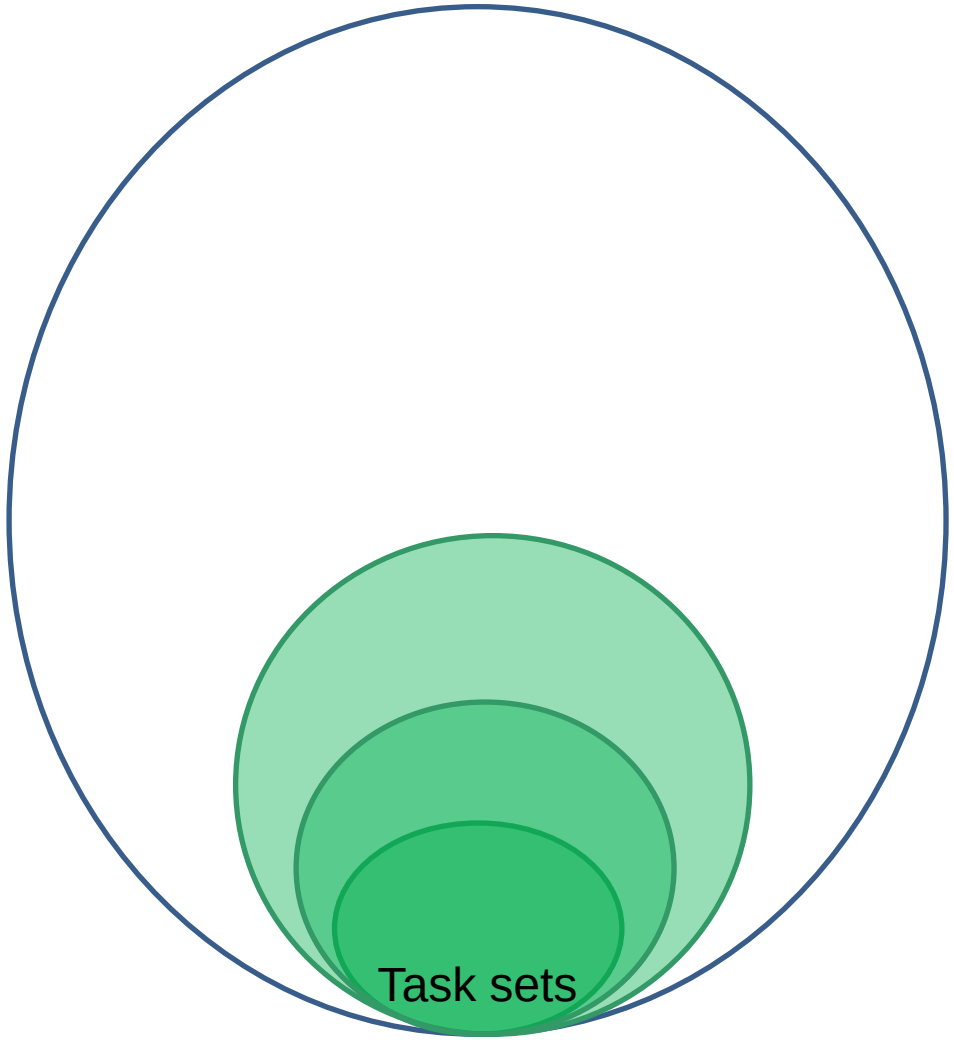
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)



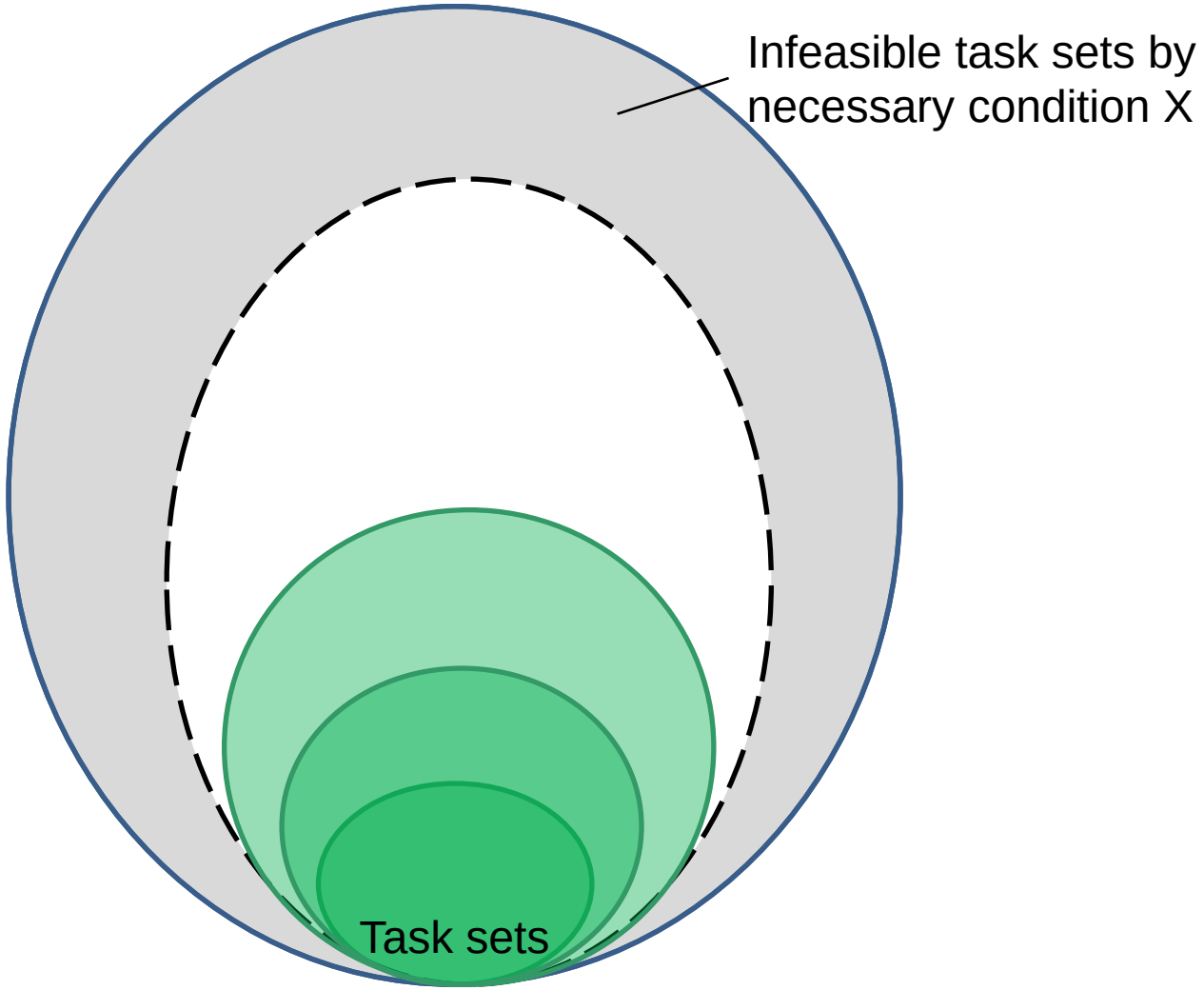
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)
    - Derive conditions of task sets that are never schedulable by any scheduling algorithm



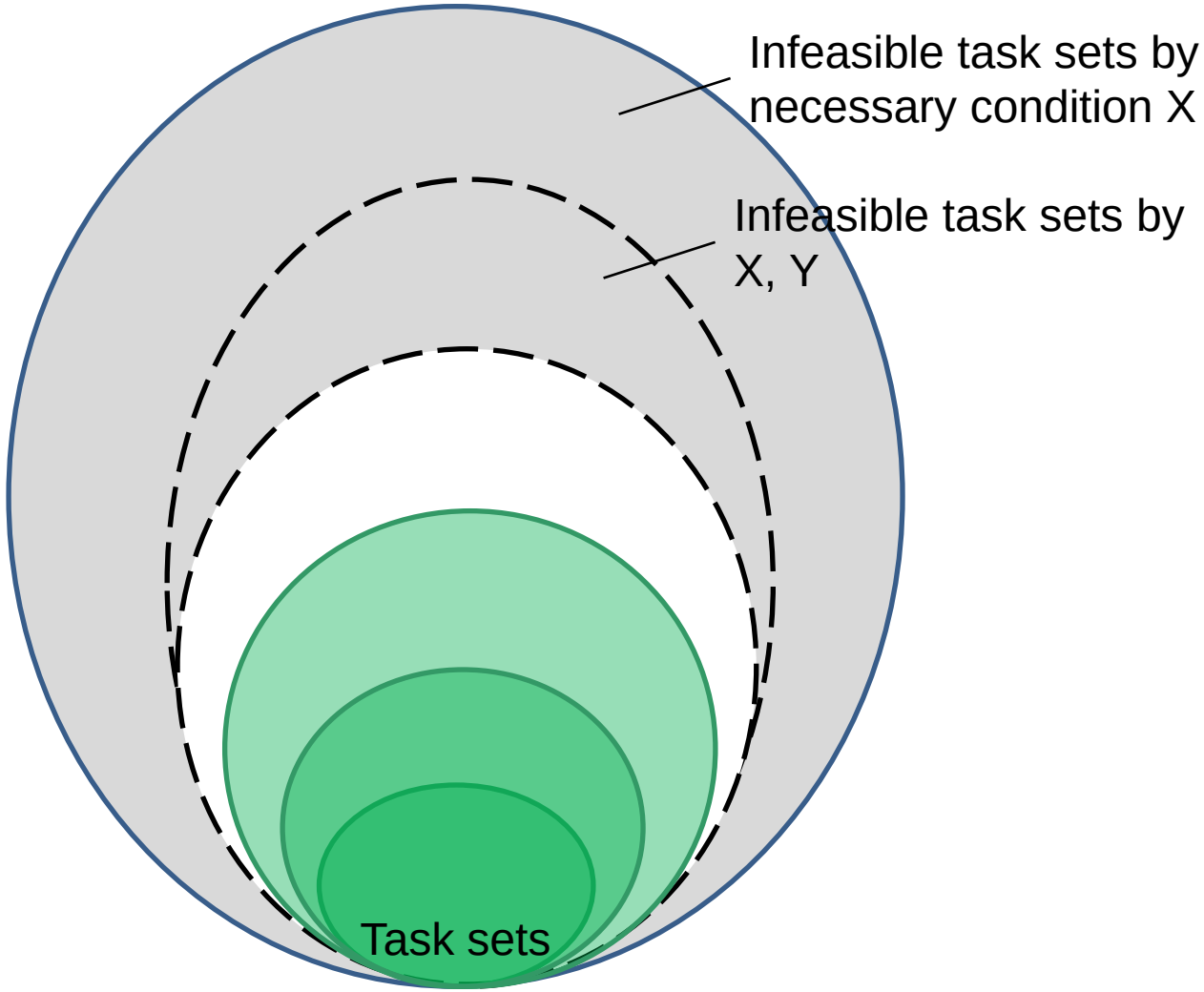
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)
    - Derive conditions of task sets that are never schedulable by any scheduling algorithm



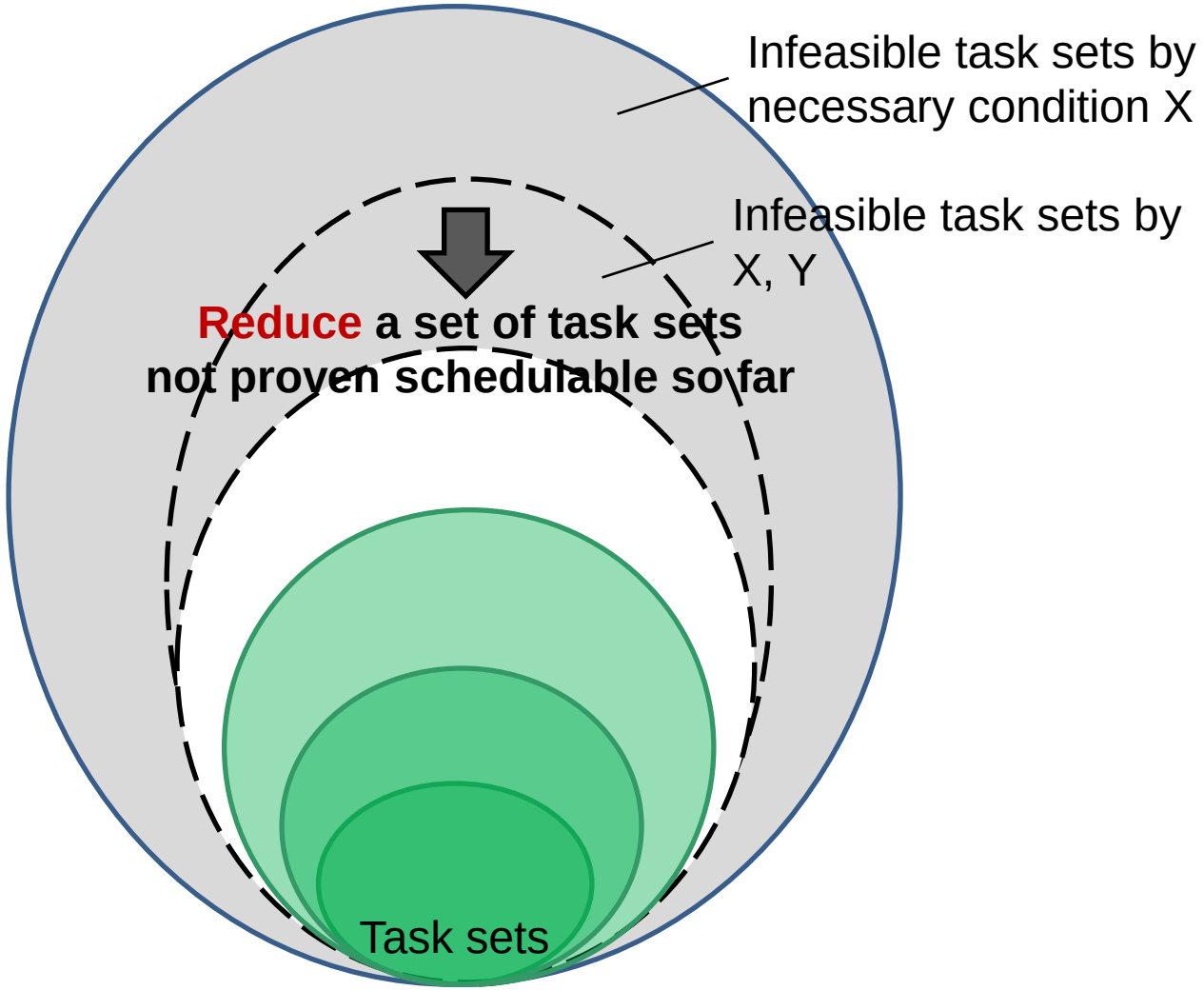
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)
    - Derive conditions of task sets that are never schedulable by any scheduling algorithm



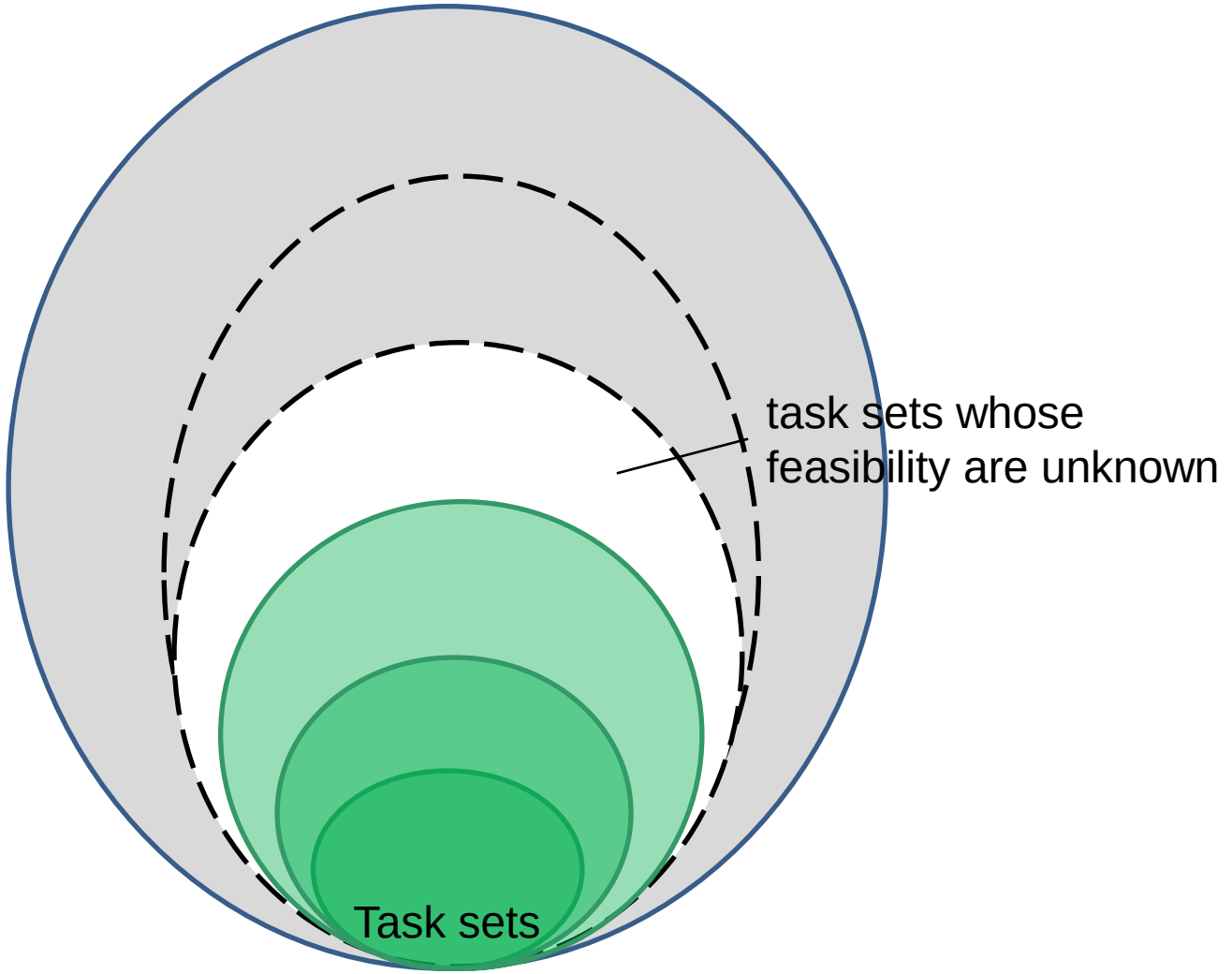
“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)
    - Derive conditions of task sets that are never schedulable by any scheduling algorithm



“Feasibility” of timing guarantees

- Is it possible to successfully schedule every instance of all tasks without missing any deadlines?
- Two directions
  - Addressing sufficient feasibility (Finding “Yes” answers)
    - Develop new scheduling algorithms and their schedulability analysis
  - Addressing necessary feasibility (Finding “No” answers)
    - Derive conditions of task sets that are never schedulable by any scheduling algorithm



State of the Art

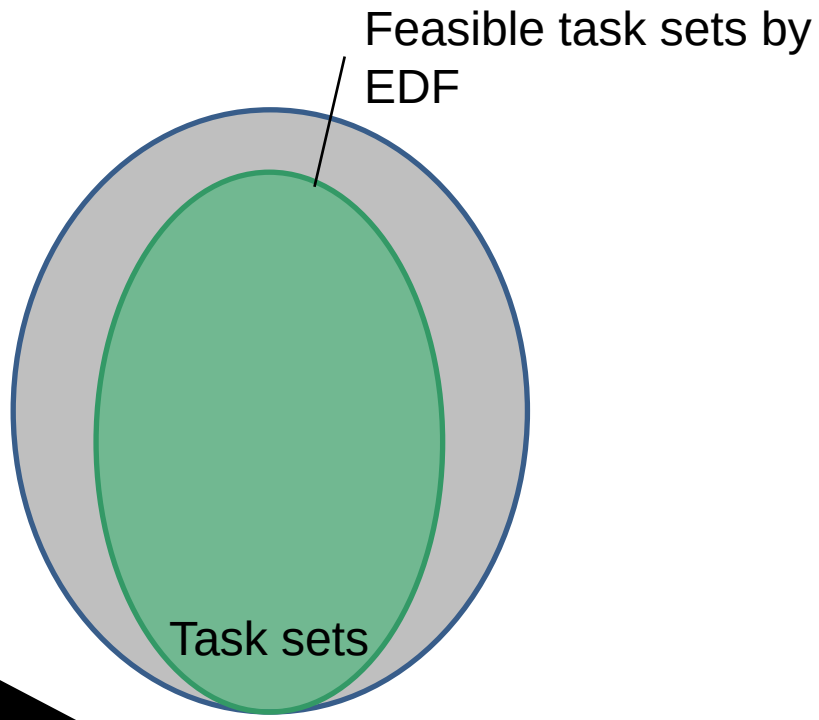
Single-Criticality Task Systems

Mixed-Criticality Task Systems

State of the Art

Single-Criticality Task Systems

+ Exact feasibility analysis
+ Optimal scheduling algorithm (EDF)

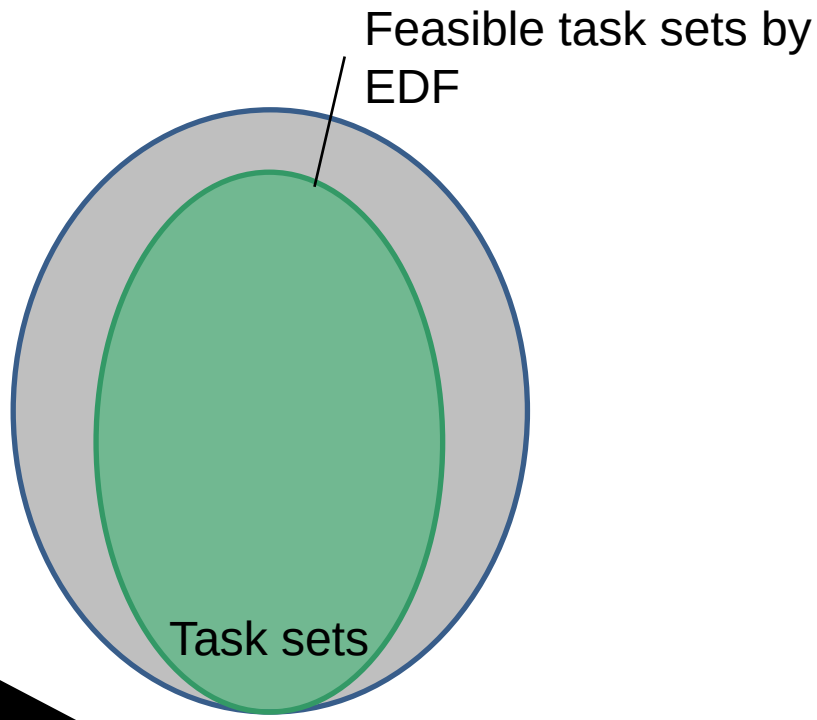


Mixed-Criticality Task Systems

State of the Art

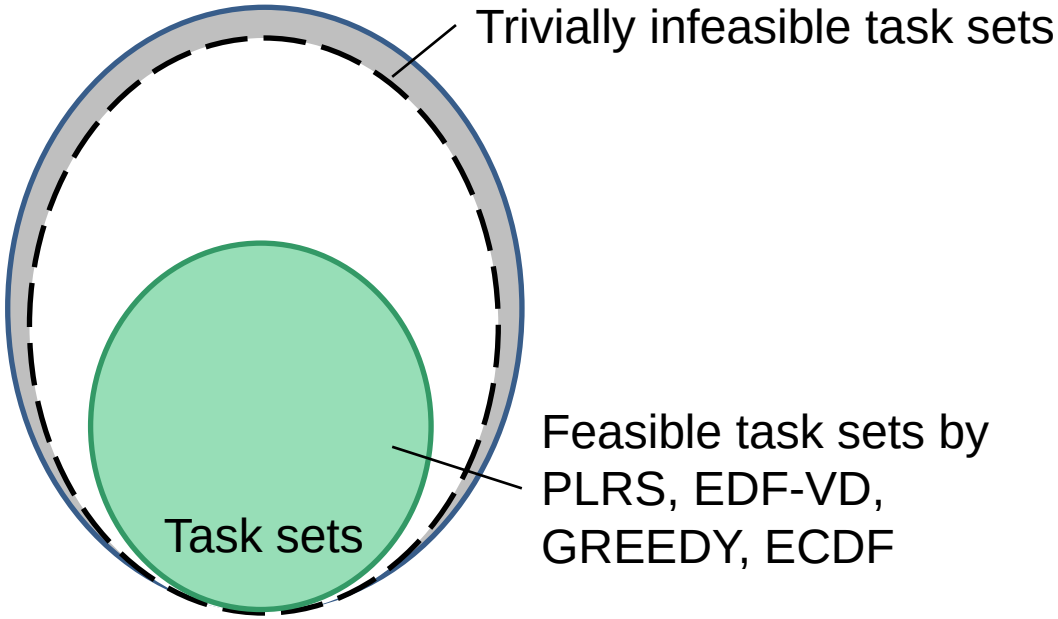
Single-Criticality Task Systems

+ Exact feasibility analysis
+ Optimal scheduling algorithm (EDF)



Mixed-Criticality Task Systems

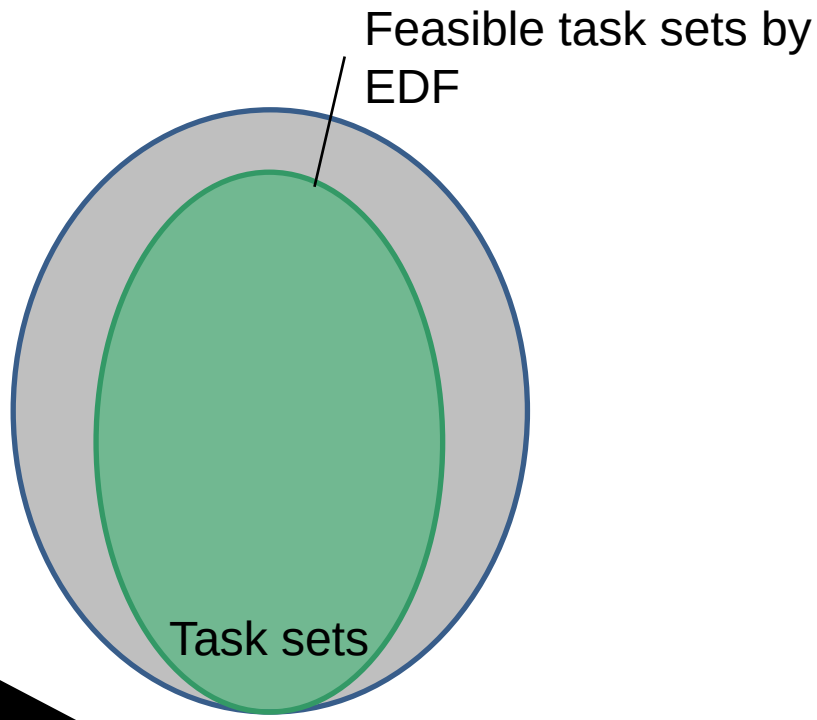
+ MC-specific scheduling algorithms
- No optimal scheduling algorithm
- Only a few existing necessary feasibility condition



State of the Art

Single-Criticality Task Systems

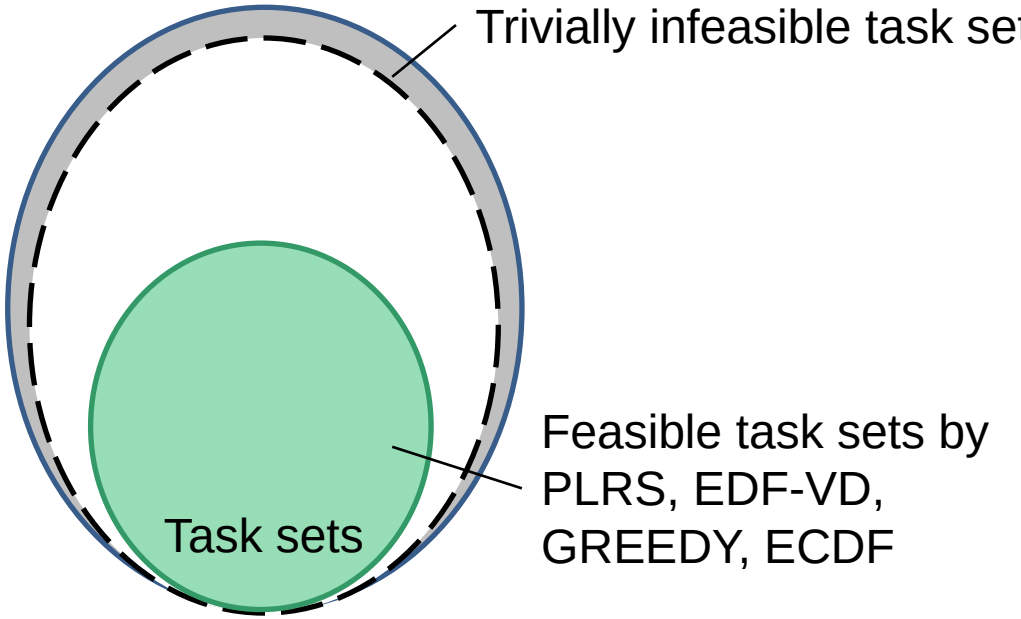
+ Exact feasibility analysis
+ Optimal scheduling algorithm (EDF)



Mixed-Criticality Task Systems

+ MC-specific scheduling algorithms
- No optimal scheduling algorithm
- Only a few existing necessary feasibility condition

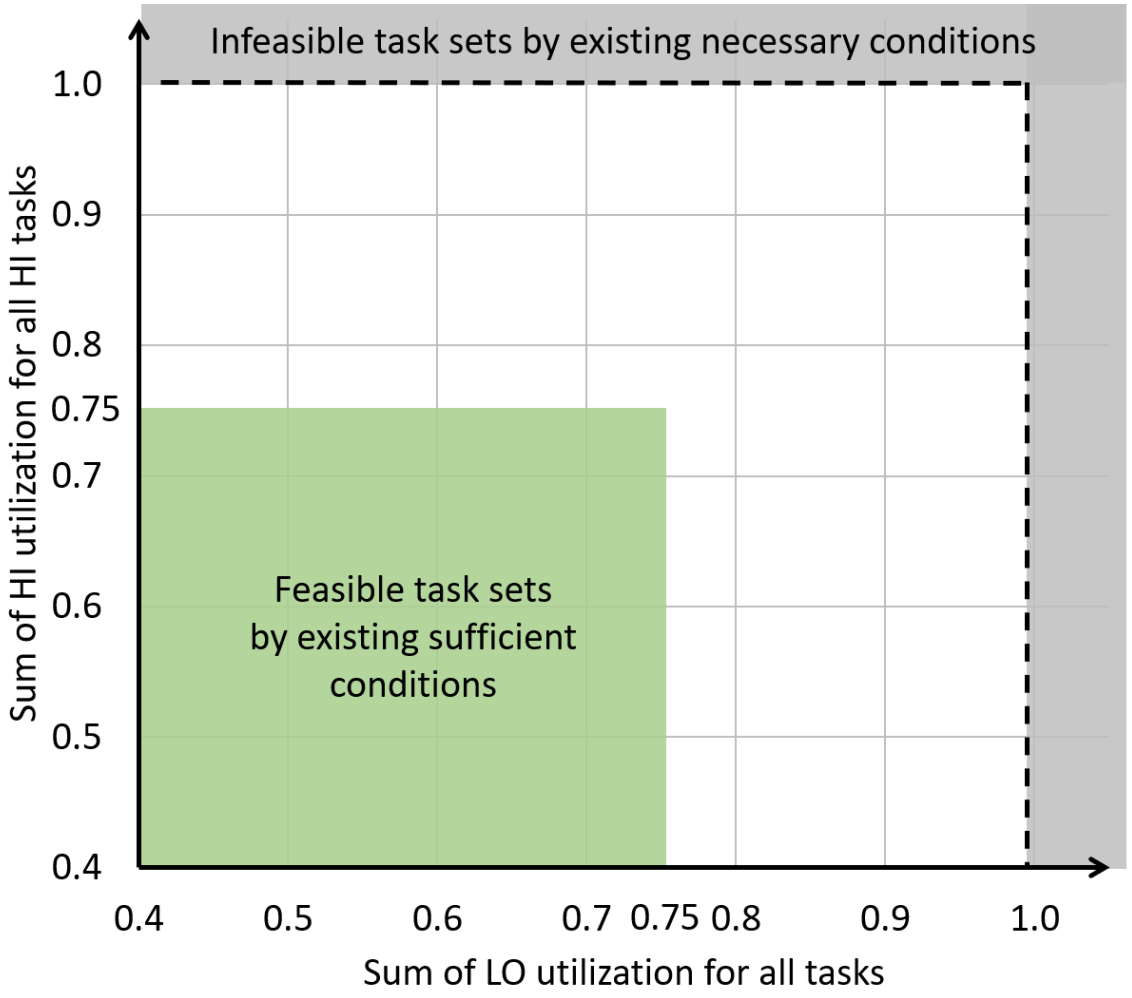
Still huge gap between the task sets proven feasible and that proven infeasible



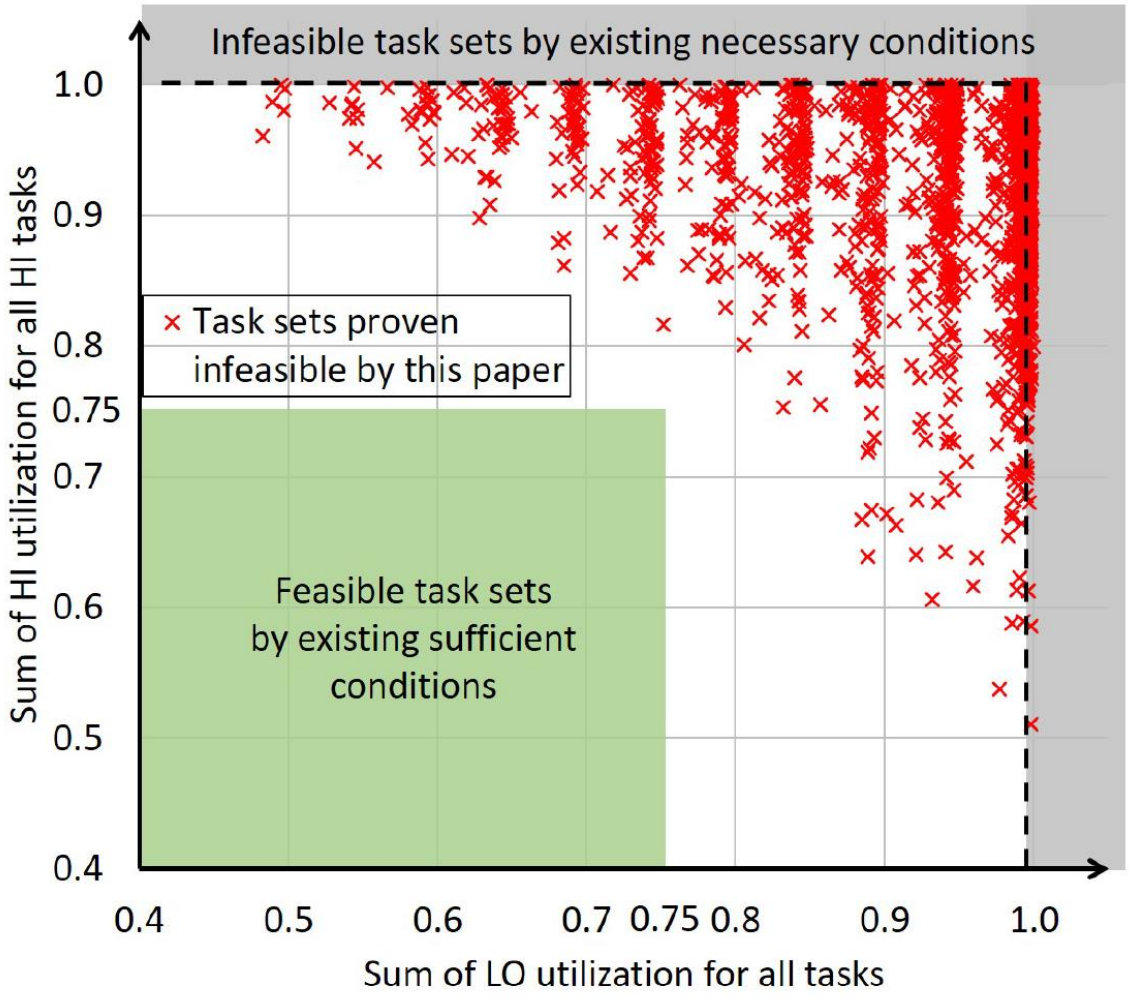
Goal

- Develop **necessary feasibility tests** that cover a broader range of infeasible MC task sets on a uniprocessor
  - Determining MC-feasibility for dual-criticality task systems is known to be NP-hard

Contributions of This Work

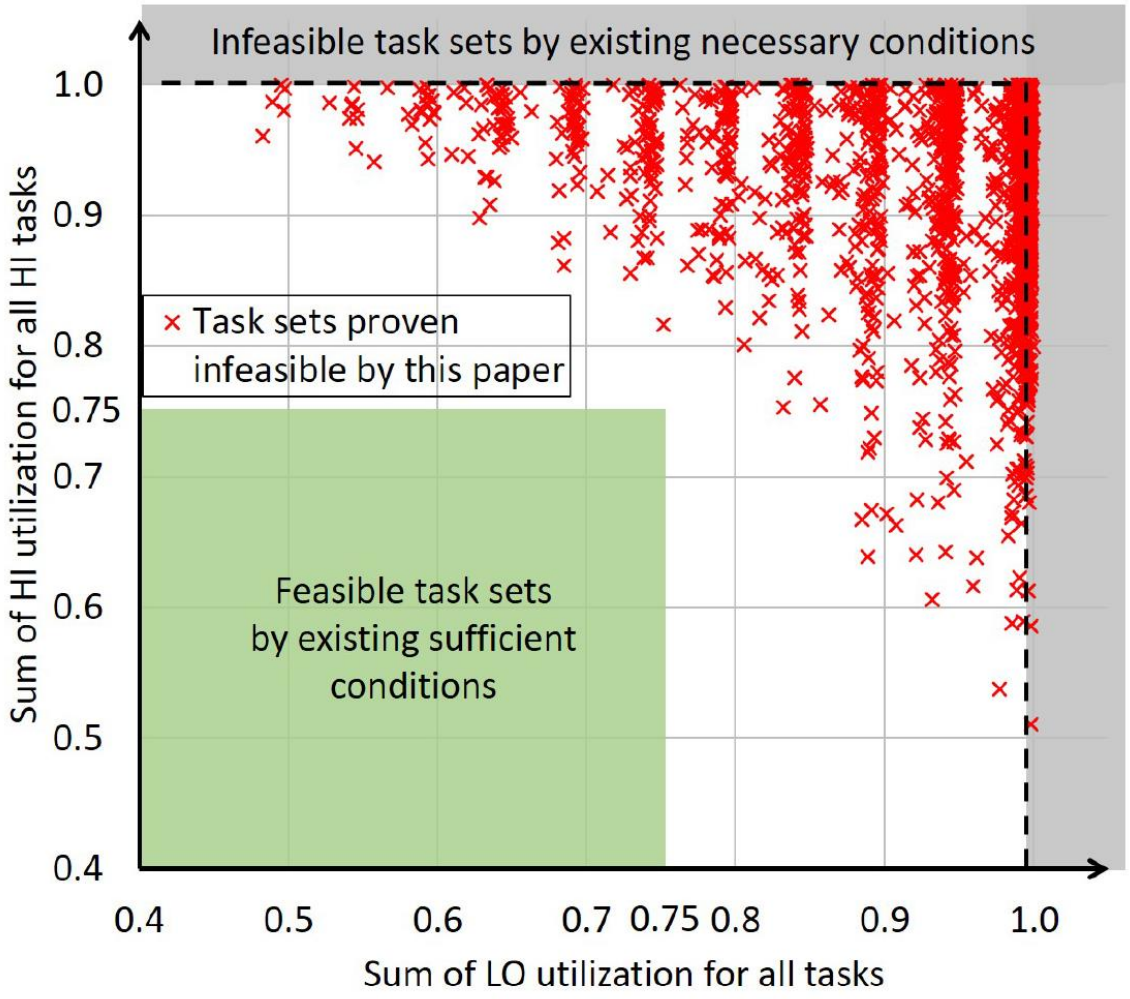


Contributions of This Work



The first study that yields non-trivial results for MC necessary feasibility

Contributions of This Work



The first study that yields non-trivial results for MC necessary feasibility

- Explore unique issues specific to MC task systems for developing necessary feasibility tests
- Identify new challenges posed by such unique issues of MC task systems
- Establish foundations of necessary feasibility tests for MC task systems

System Model

- Dual-criticality systems (Vestal's task model)
  - **Task** $\boldsymbol{\tau_i = \left(T_i, \chi_i, C_i^{LO}, C_i^{HI}, D_i\right)}$, where
    - $\chi_i \in \{LO, HI\}$;
      - LO – low-critical task, HI – high-critical task
    - C_i^{LO}: LO WCET, C_i^{HI}: HI WCET
      - for LC task $C_i^{LO} = C_i^{HI}$ and for HC task$C_i^{LO} \leq C_i^{HI}$
  - **Job** $\mathbf{J_i^q} = \left(r_i^q, \gamma_i^q\right)$, where
    - r_i^q: the release time of the job
    - $\gamma_i^q \in (0, C_i^{HI}]$: the execution requirement
  - **Scenario** for a given task set τ
    - A collection of release times and execution requirements of jobs invoked by tasks in τ

System Model

- Dual-criticality systems (Vestal's task model)
  - **Task** $\tau_i = (T_i, \chi_i, C_i^{LO}, C_i^{HI}, D_i)$, where
    - $\chi_i \in \{LO, HI\}$;
      - LO – low-critical task, HI – high-critical task
    - C_i^{LO}: LO WCET, C_i^{HI}: HI WCET
      - for LC task $C_i^{LO} = C_i^{HI}$ and for HC task $C_i^{LO} \leq C_i^{HI}$
  - **Job** $\mathbf{J_i^q} = (r_i^q, \gamma_i^q)$, where
    - r_i^q: the release time of the job
    - $\gamma_i^q \in (0, C_i^{HI}]$: the execution requirement
  - **Scenario** for a given task set τ
    - A collection of release times and execution requirements of jobs invoked by tasks in τ

MC-feasible (MC-infeasible)
- If *every* scenario is feasible (If there exists *at least one* scenario that is not feasible

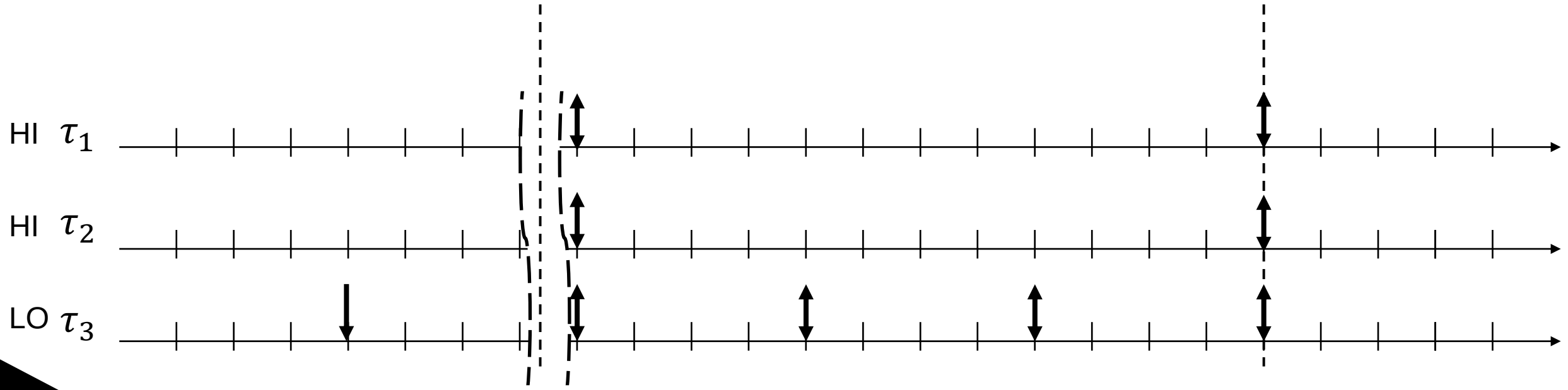
- **Feasible scenario**
  - If there exists a schedule that satisfies

i) every job finishes its execution time before its deadline when exhibiting the LO behavior

ii) every HI job finishes its execution time before its deadline when exhibiting the HI behavior

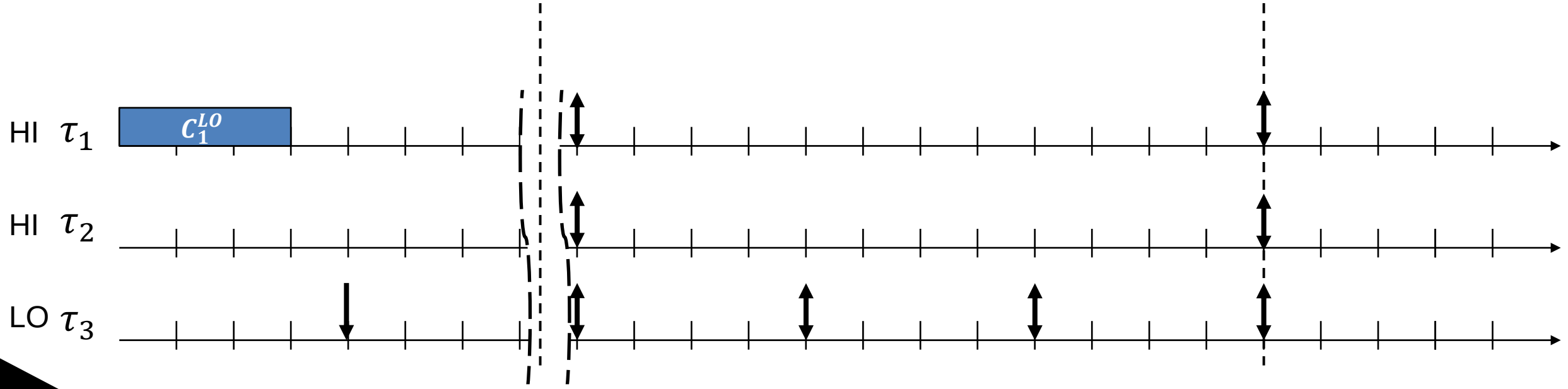
Unique Characteristics of MC Task Systems

- Existence of the mode change



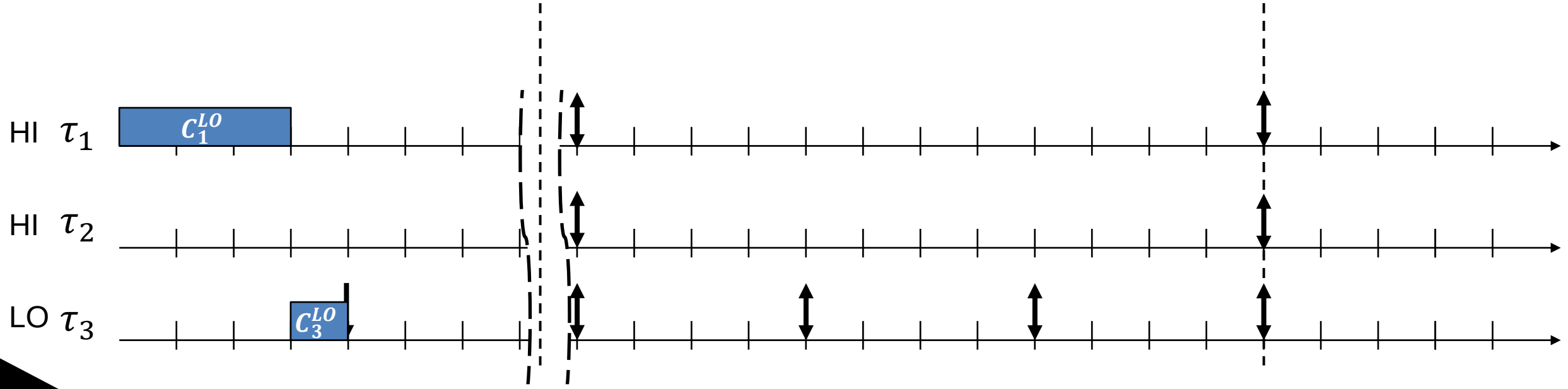
Unique Characteristics of MC Task Systems

- Existence of the mode change



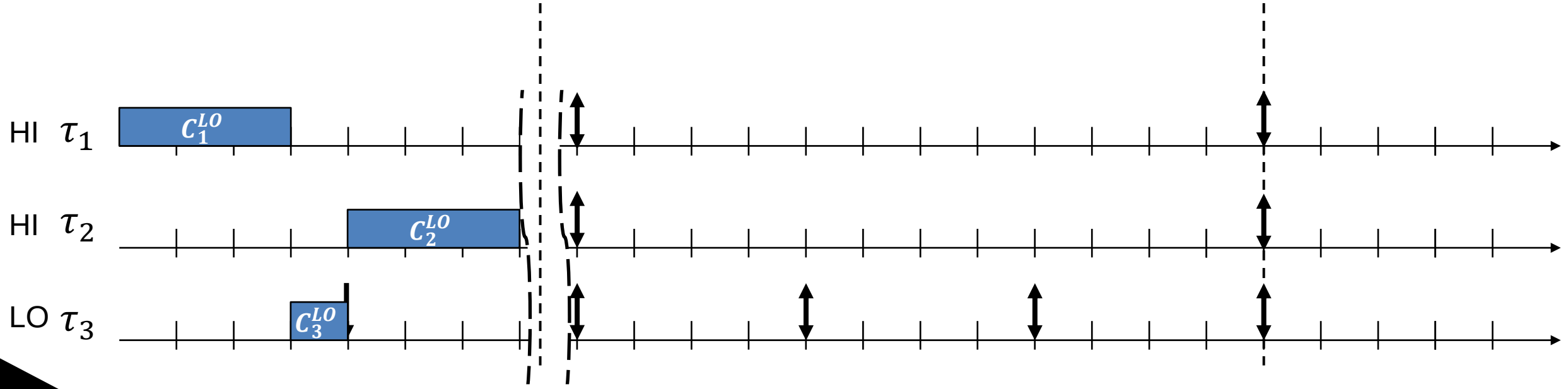
Unique Characteristics of MC Task Systems

- Existence of the mode change



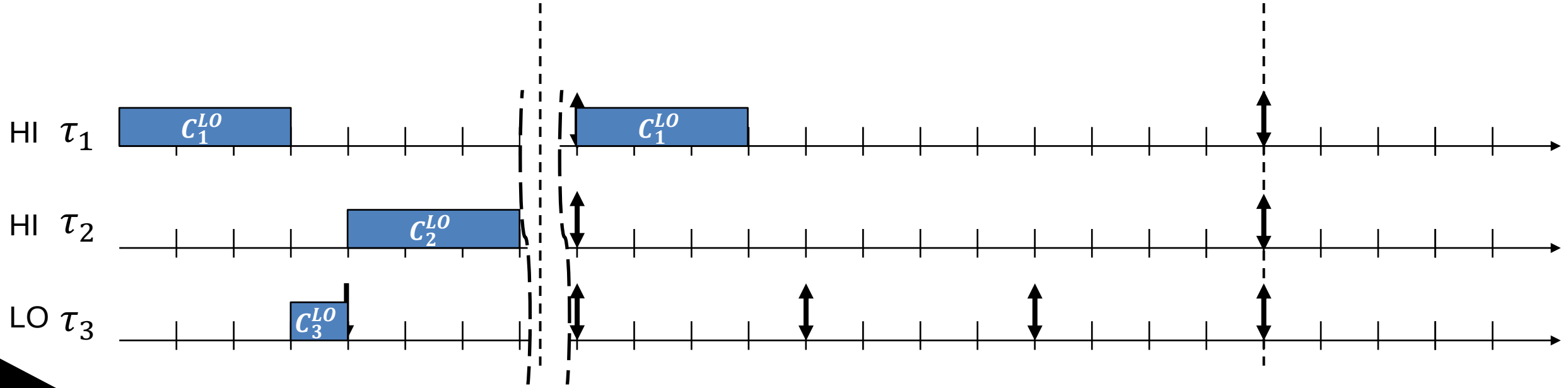
Unique Characteristics of MC Task Systems

- Existence of the mode change



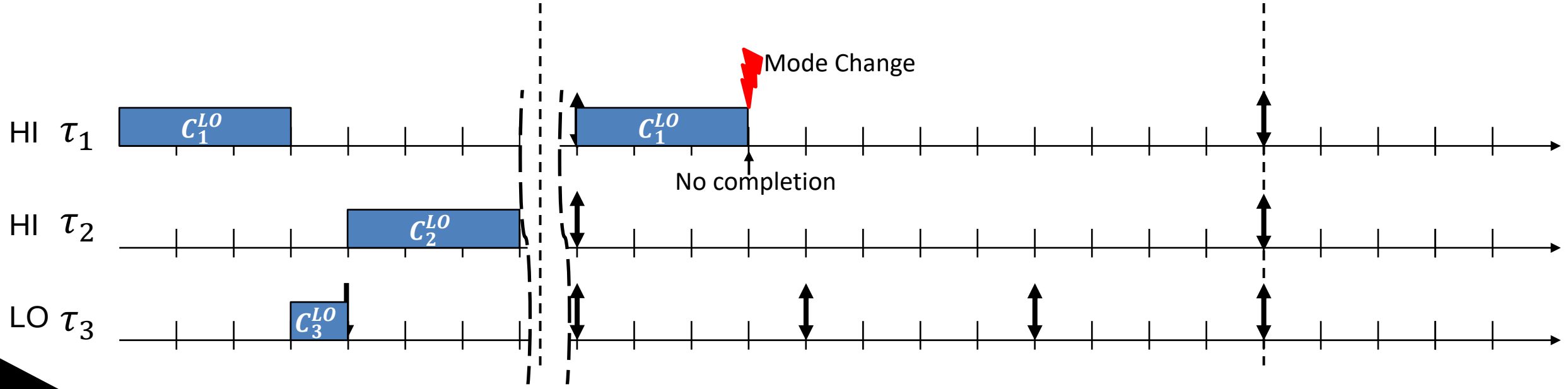
Unique Characteristics of MC Task Systems

- Existence of the mode change



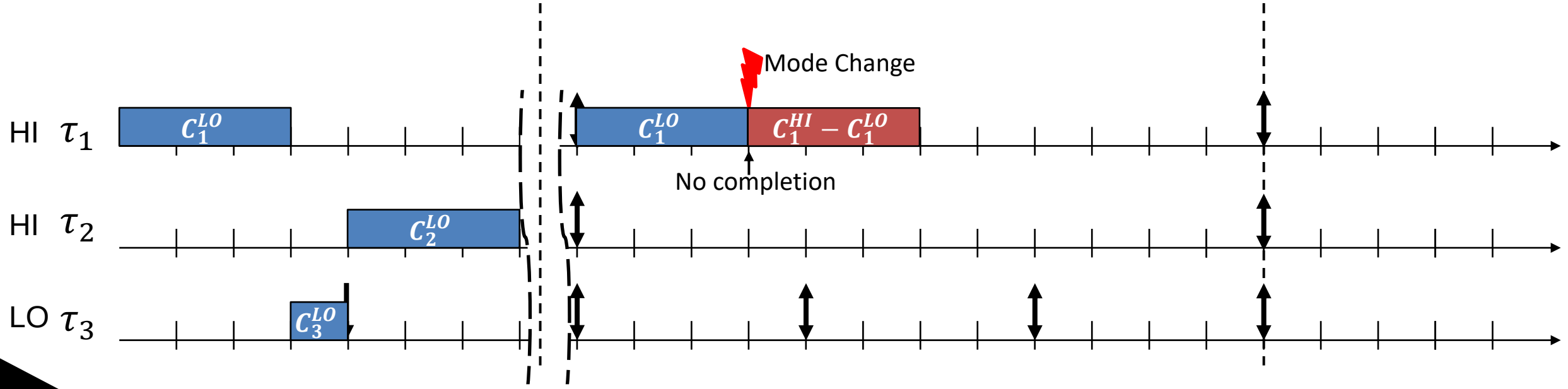
Unique Characteristics of MC Task Systems

- Existence of the mode change



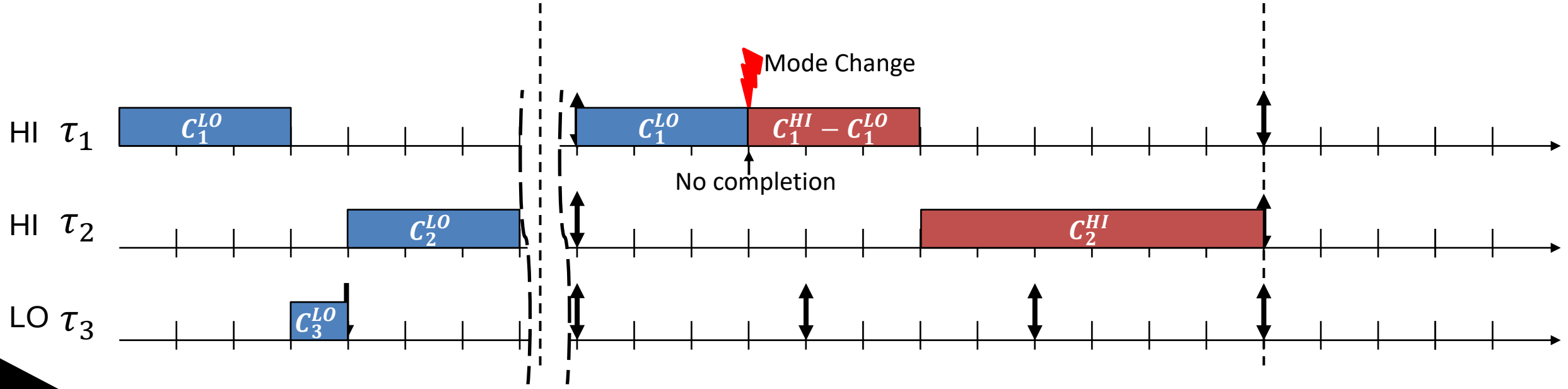
Unique Characteristics of MC Task Systems

- Existence of the mode change



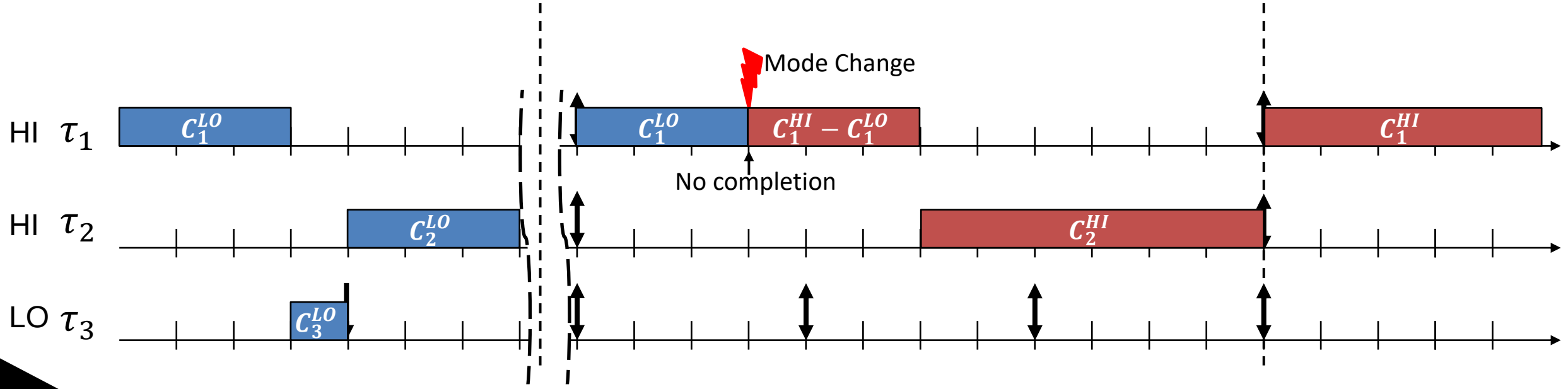
Unique Characteristics of MC Task Systems

- Existence of the mode change



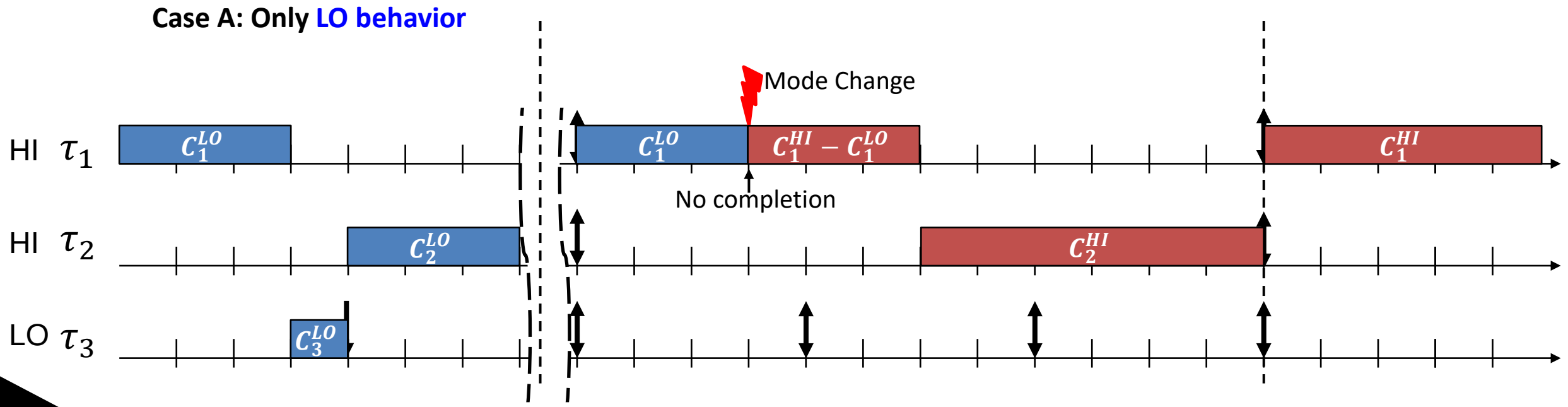
Unique Characteristics of MC Task Systems

- Existence of the mode change



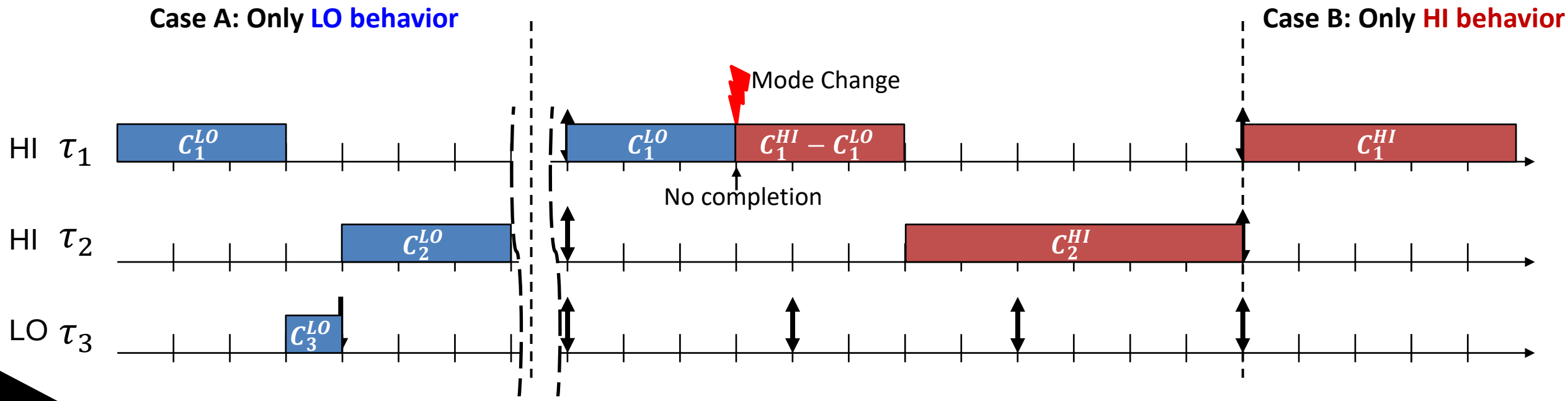
Unique Characteristics of MC Task Systems

- Existence of the mode change



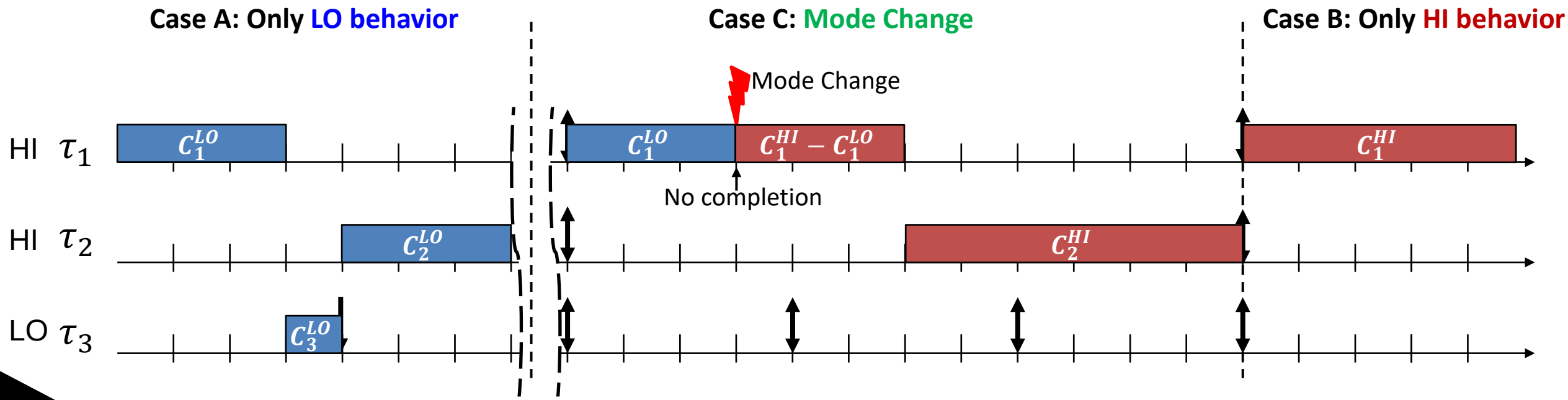
Unique Characteristics of MC Task Systems

- Existence of the mode change



Unique Characteristics of MC Task Systems

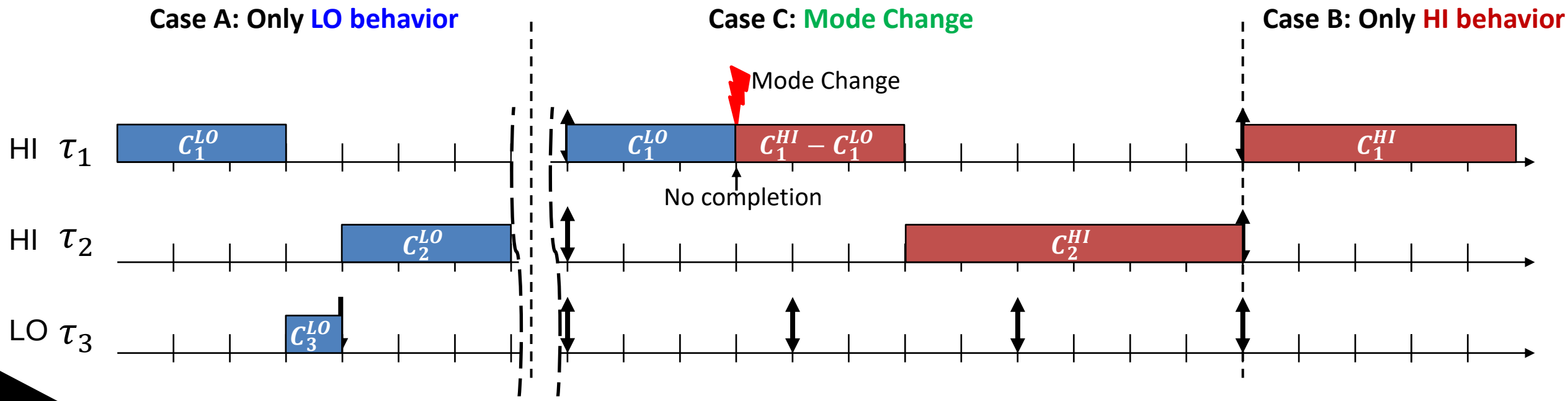
- Existence of the mode change



Unique Characteristics of MC Task Systems

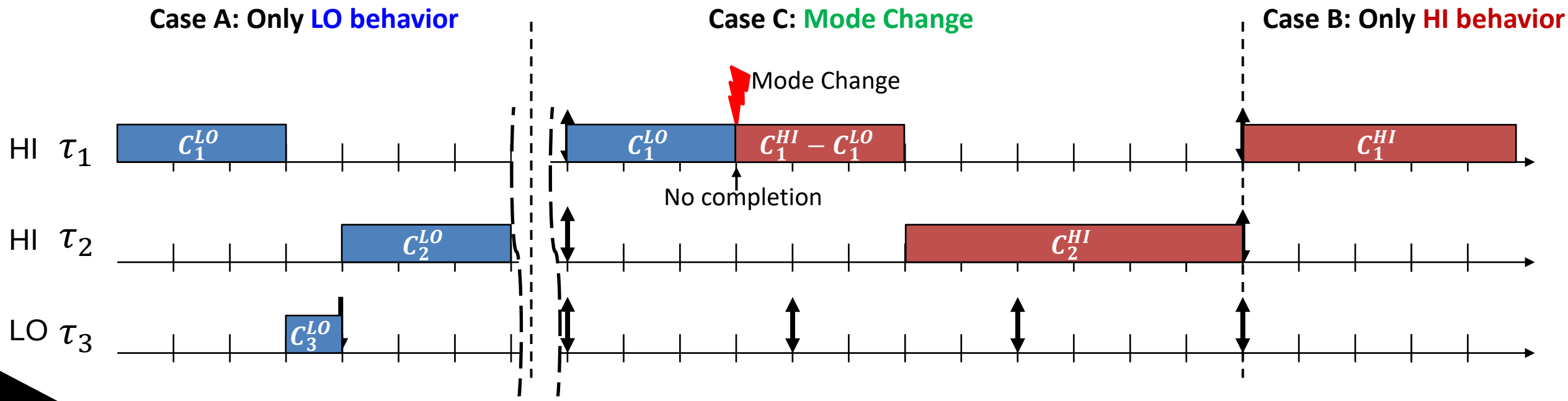
- Existence of the mode change

  C1. The **demand** **varies** depending on the system behavior

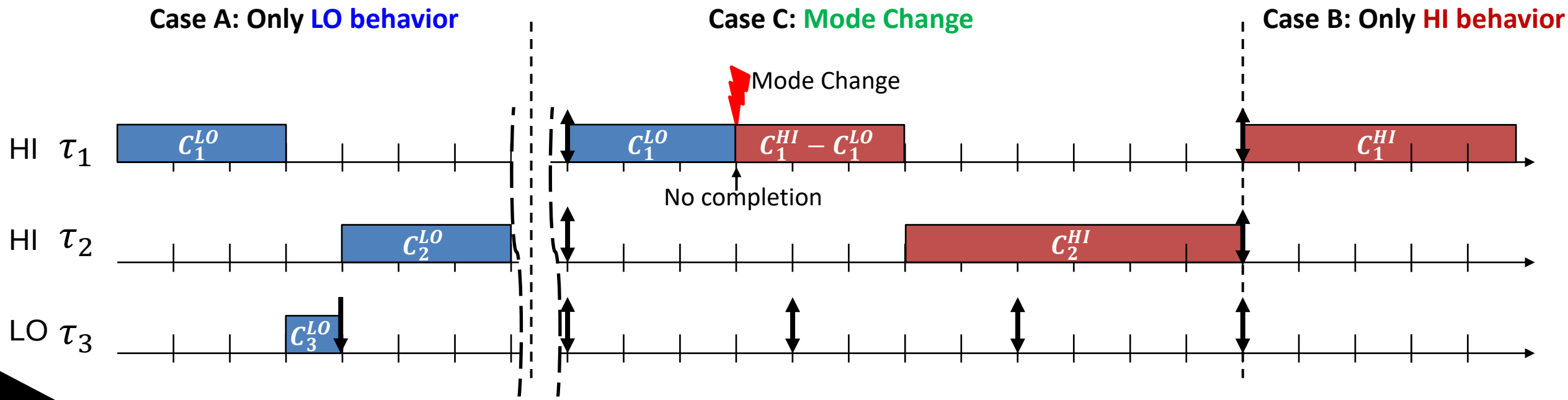


Unique Characteristics of MC Task Systems

- Existence of the mode change
  - C1. The **demand varies** depending on the system behavior
  - C2. It is **impossible to know** beforehand **when the mode change occurs**

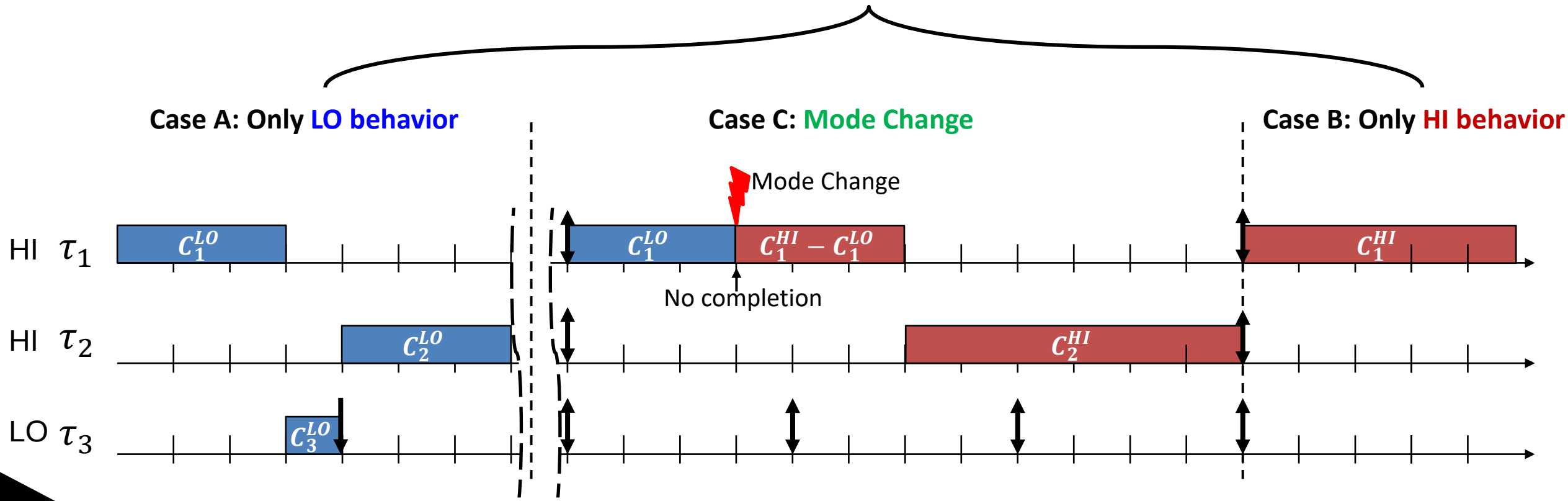


Necessary Feasibility of MC Task Systems



Necessary Feasibility of MC Task Systems

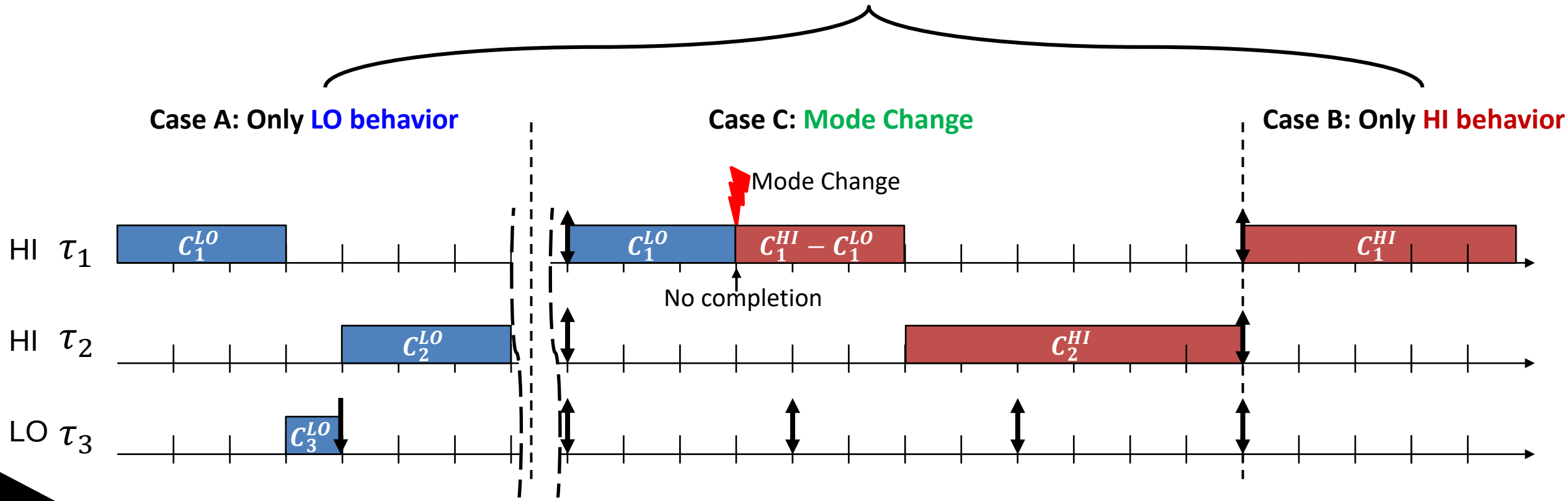
each scenario of Cases A & B ≡ a scenario of a single-criticality task system



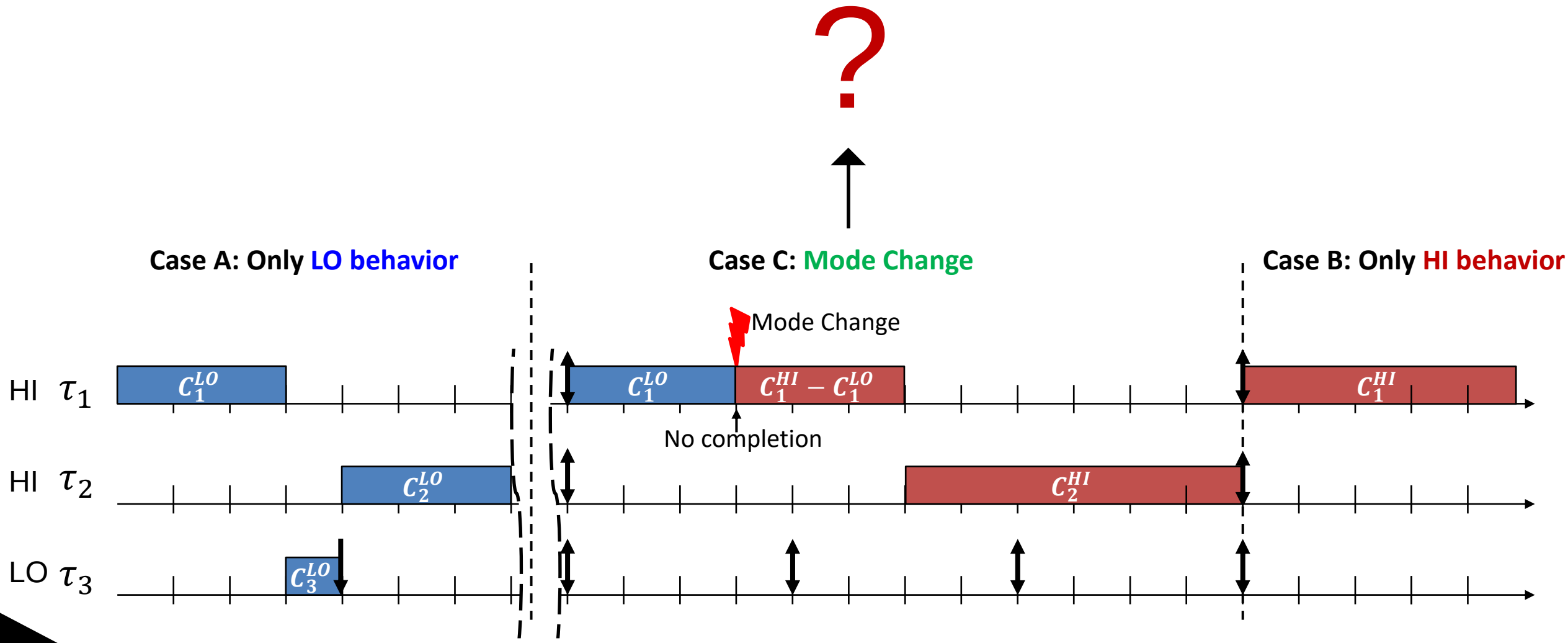
Necessary Feasibility of MC Task Systems

the demand > the supply ⇒ infeasible

each scenario of Cases A & B ≡ a scenario of a single-criticality task system



Necessary Feasibility of MC Task Systems

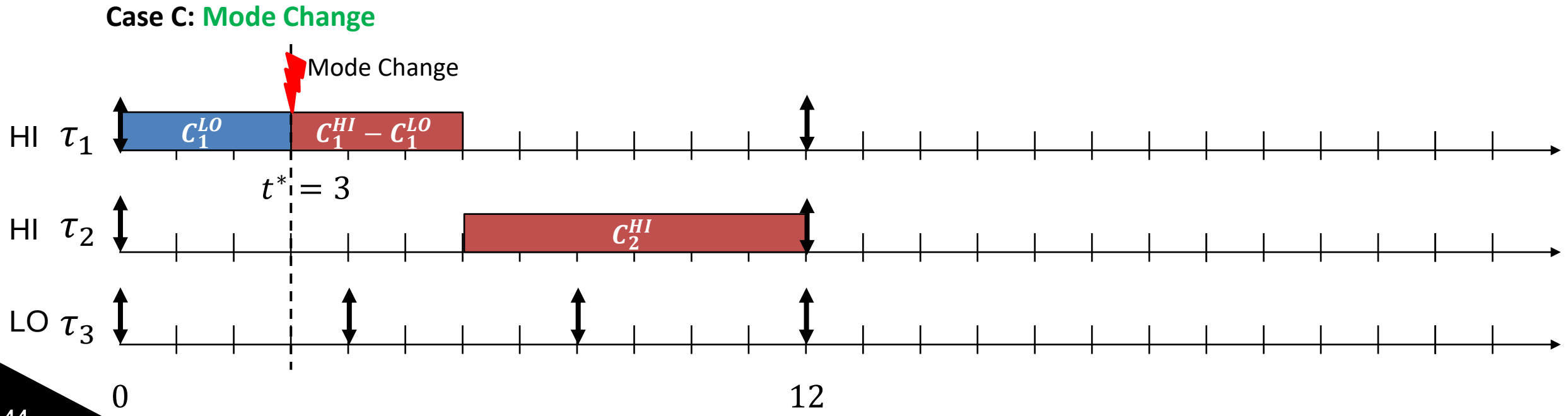


Necessary Feasibility of MC Task Systems

When the mode change occurs at $t^* = 3$

- The demand of τ_1 in [0,12] = C_1^{HI}
- The demand of τ_2 in [0,12] = C_2^{HI}
- The demand of τ_3 in [0,12] = 0

The demand in [0,12] = 12 $\leq$ the supply in [0,12]



Necessary Feasibility of MC Task Systems

When the mode change occurs at $t^* = 3$

- The demand of τ_1 in [0,12] = C_1^{HI}
- The demand of τ_2 in [0,12] = C_2^{HI}
- The demand of τ_3 in [0,12] = 0

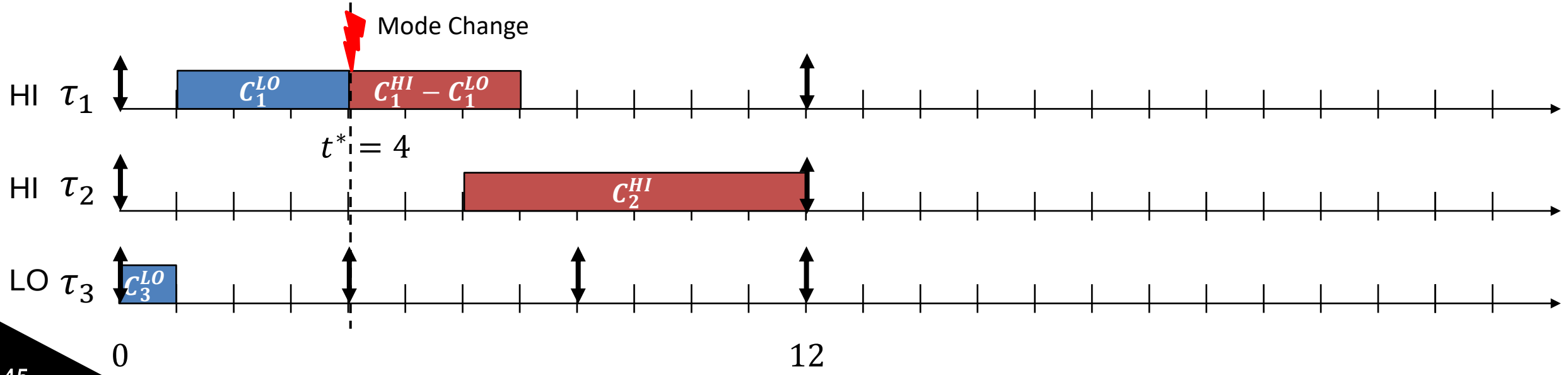
The demand in [0,12] = 12 $\leq$ the supply in [0,12]

When the mode change occurs at $t^* = 4$

- The demand of τ_1 in [0,12] = C_1^{HI}
- The demand of τ_2 in [0,12] = C_2^{HI}
- The demand of τ_3 in [0,12] = 1

The demand in [0,12] = 13 $>$ the supply in [0,12]

Case C: Mode Change



Key Observations and Challenges

Key observations

O1. The contribution of each LO job to the demand varies with the mode change instant

O2. It is impossible to calculate the demand without specifying the mode change instant

O3. The demand > the supply in a case does not necessarily yield infeasibility of the scenario

Key Observations and Challenges

Key observations

O1. The contribution of each LO job to the demand varies with the mode change instant

O2. It is impossible to calculate the demand without specifying the mode change instant

O3. The demand > the supply in a case does not necessarily yield infeasibility of the scenario

Challenges

Q1. How to characterize and calculate the demand in an interval that changes depending on the mode change instant? (from O1 & O2)

Q2. What is the meaning of the demand > the supply in an interval when the mode change instant is given? (from O3)

Q3. How to derive a necessary feasibility condition without assuming the mode change instant is given? (from O2 & O3)

Our Approach

Q1. How to characterize and calculate the demand ?

Q2. What is the meaning of the demand > the supply ?

Q3. How to derive a necessary feasibility condition ?

Our Approach

Q1. How to characterize and calculate the demand ?

- Specify **a range of mode change instant** without the target scheduling algorithm (Lemma 4)
- **Select two sub-intervals** based on a mode change instant t^*
- **Calculate the demand** in the target sub-intervals (Lemmas 5,6,7)

Q2. What is the meaning of the demand > the supply ?

Q3. How to derive a necessary feasibility condition ?

Our Approach

Q1. How to characterize and calculate the demand ?

- Specify **a range of mode change instant** without the target scheduling algorithm (Lemma 4)
- **Select two sub-intervals** based on a mode change instant t^*
- **Calculate the demand** in the target sub-intervals (Lemmas 5,6,7)

Q2. What is the meaning of the demand > the supply ?

- Compare the total demand with the total supply in the target sub-intervals
- **Judge the infeasibility of the mode change instant t^*** (Lemma 8)

Q3. How to derive a necessary feasibility condition ?

Our Approach

Q1. How to characterize and calculate the demand ?

- Specify **a range of mode change instant** without the target scheduling algorithm (Lemma 4)
- **Select two sub-intervals** based on a mode change instant t^*
- **Calculate the demand** in the target sub-intervals (Lemmas 5,6,7)

Q2. What is the meaning of the demand > the supply ?

- Compare the total demand with the total supply in the target sub-intervals
- **Judge the infeasibility of the mode change instant t^*** (Lemma 8)

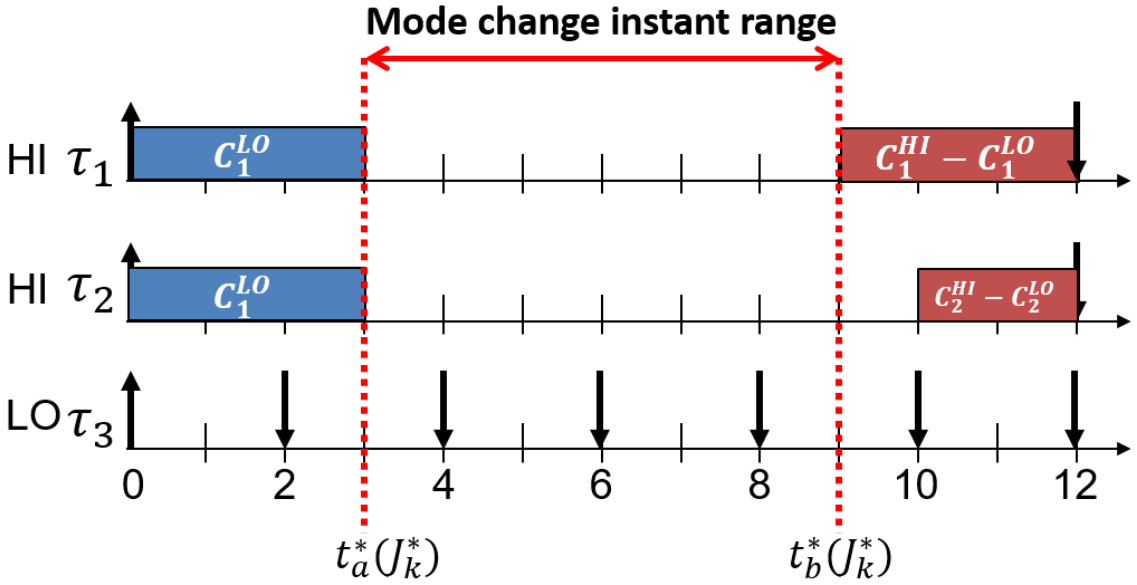
Q3. How to derive a necessary feasibility condition ?

- Repeat Lemma 8 for all t^* in the mode change instant range
- **Check there exists no feasible mode change instant** (Infeasibility of the task set) (Theorem 1)

Our Approach

(a) Target J_k^*
(the job with the earliest release time among all HI jobs whose execution requirement is strictly larger than LO WCET)

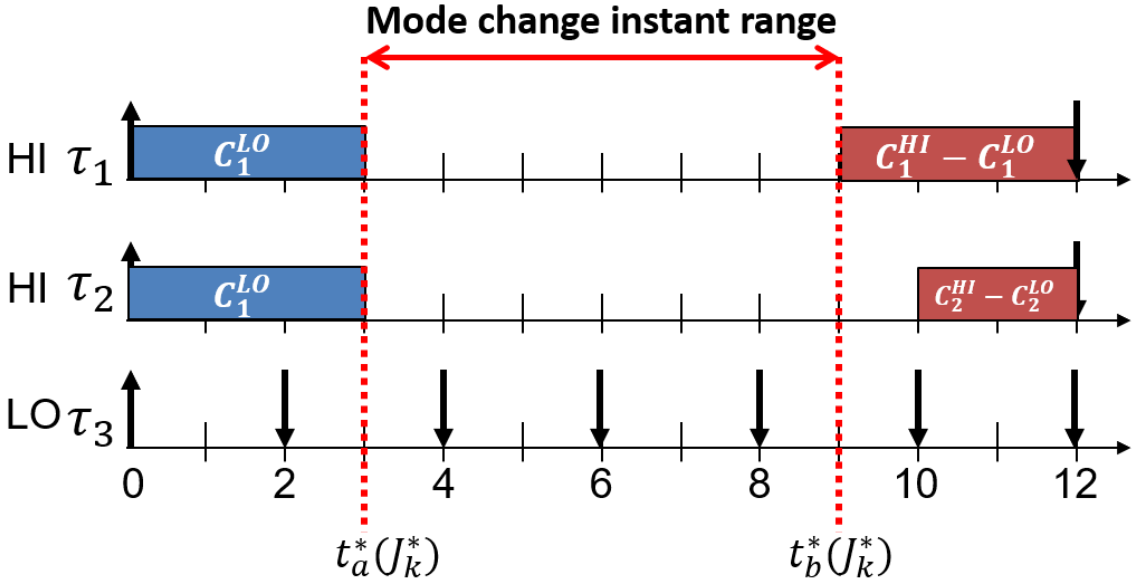
(b) Specify mode change instant range in $t^* \in$ [3,9]
(*Lemma 4*)



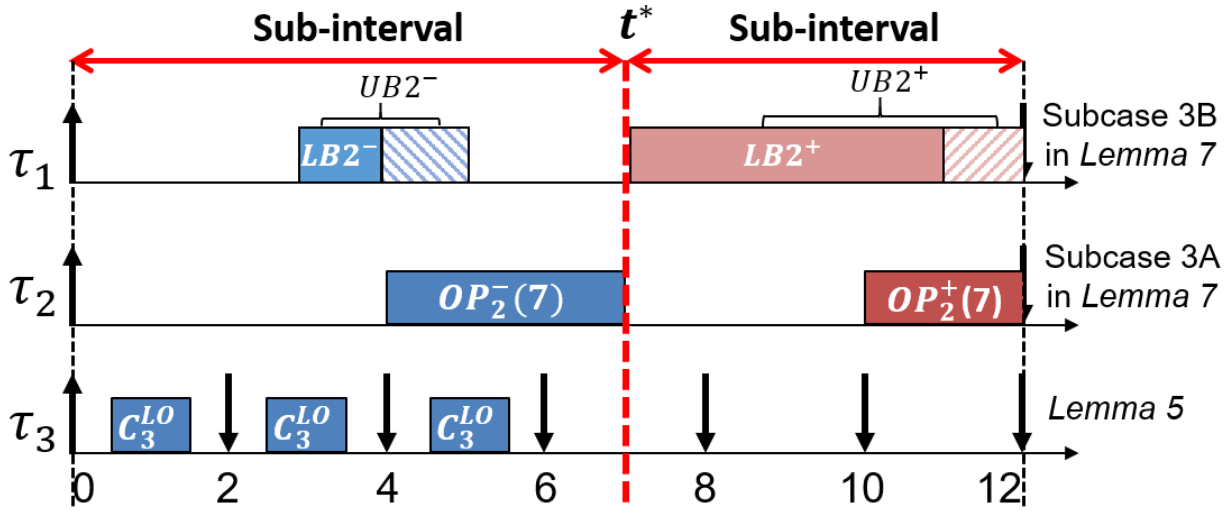
Our Approach

(a) Target J_k^*
(the job with the earliest release time among all HI jobs whose execution requirement is strictly larger than LO WCET)

(b) Specify mode change instant range in $t^* \in$ [3,9] (*Lemma 4*)



(c) Given t^*, select sub-intervals [0,7] and [7,12]

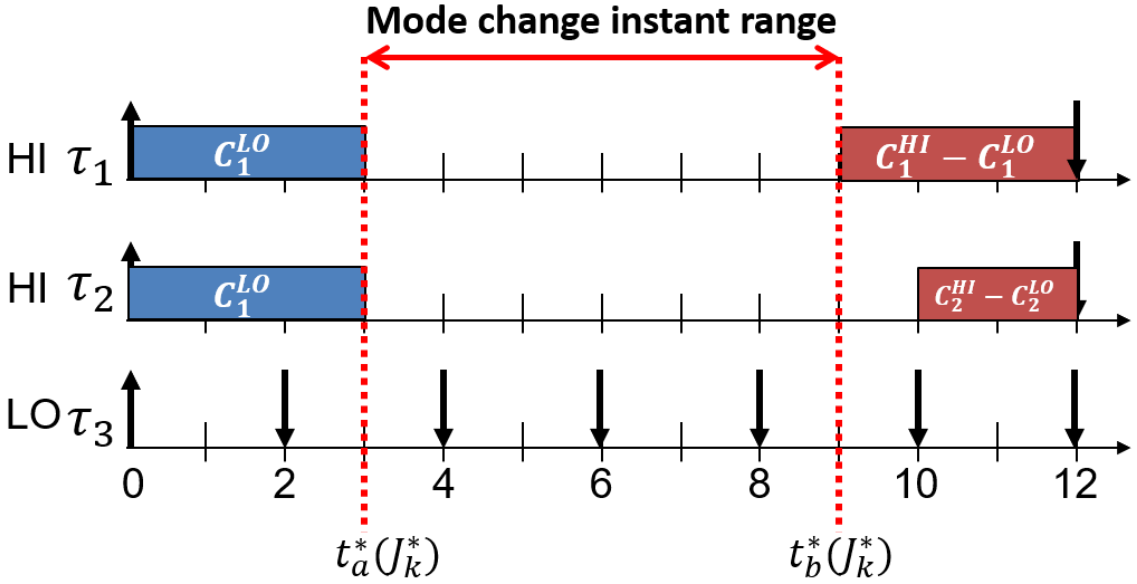


Our Approach

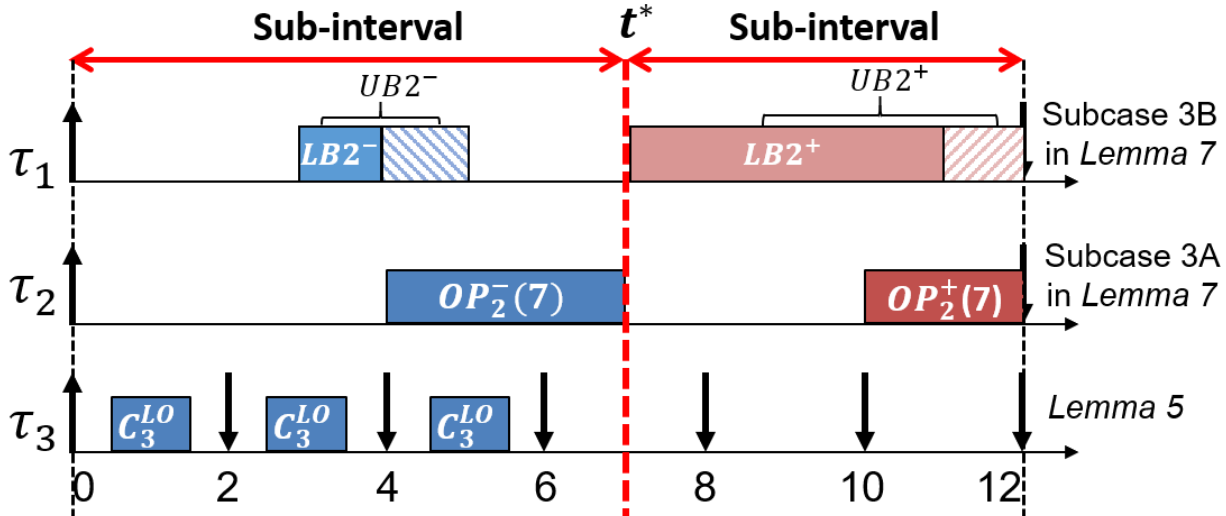
(a) Target J_k^*

(the job with the earliest release time among all HI jobs whose execution requirement is strictly larger than LO WCET)

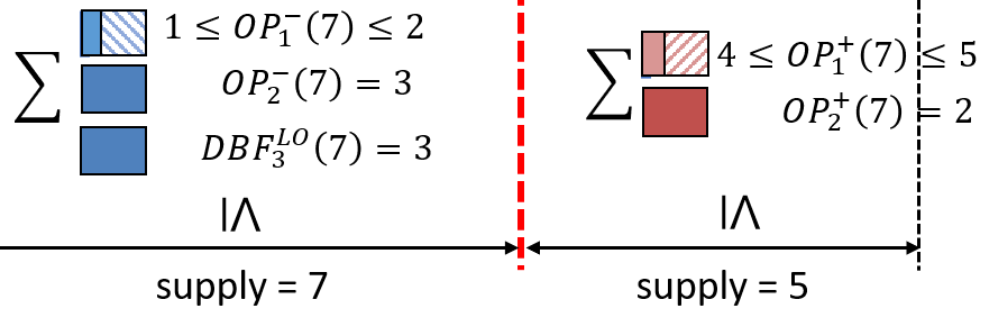
(b) Specify mode change instant range in $t^* \in$ [3,9] (*Lemma 4*)



(c) Given t^*, select sub-intervals [0,7] and [7,12]



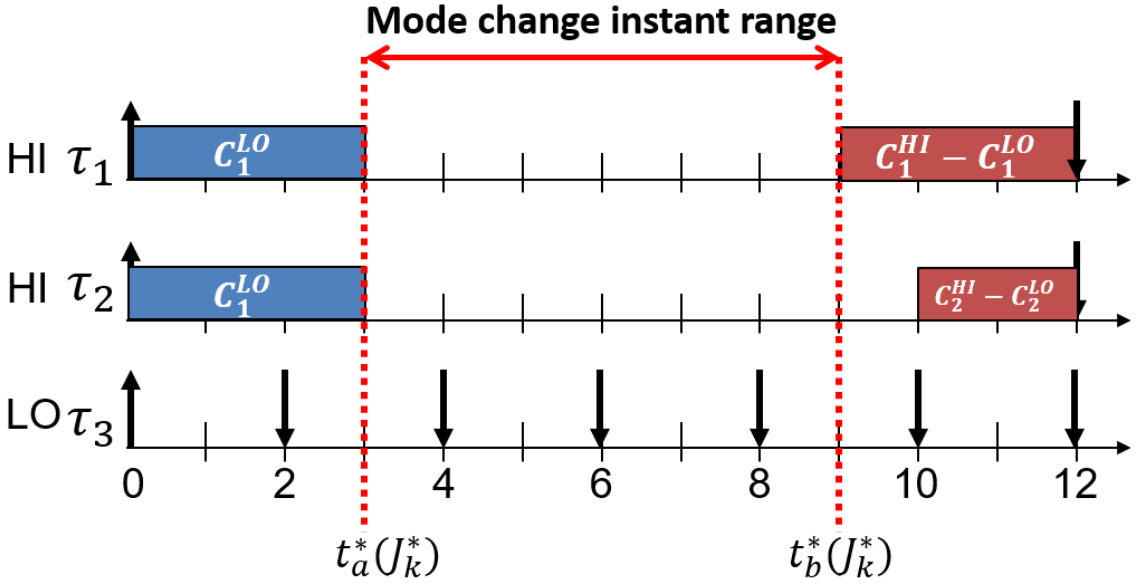
(d) Calculate demand in the sub-intervals (Lemmas 5,6,7)



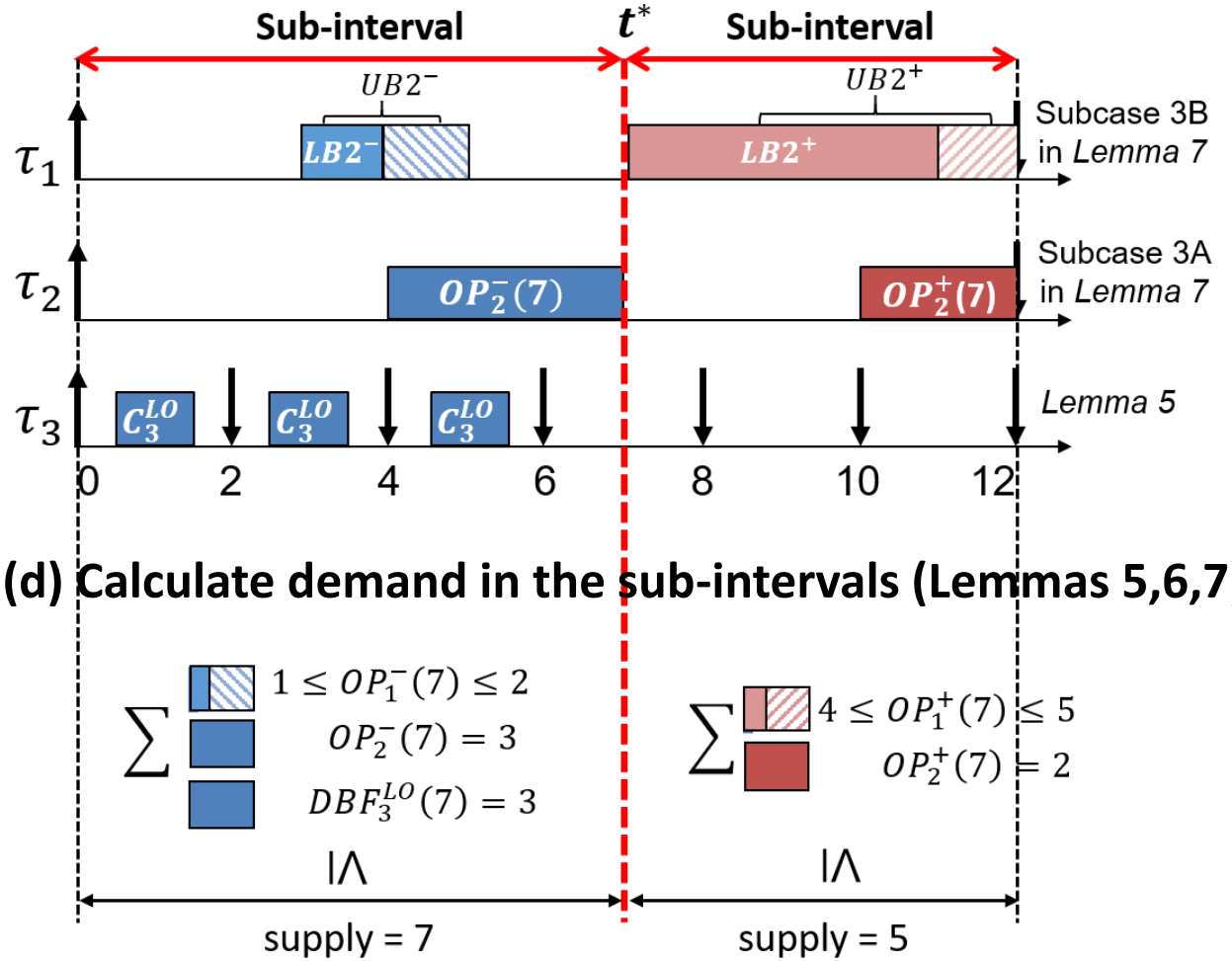
Our Approach

(a) Target J_k^*
(the job with the earliest release time among all HI jobs whose execution requirement is strictly larger than LO WCET)

(b) Specify mode change instant range in $t^* \in$ [3,9] (*Lemma 4*)



(c) Given t^*, select sub-intervals [0,7] and [7,12]



(d) Calculate demand in the sub-intervals (Lemmas 5,6,7)

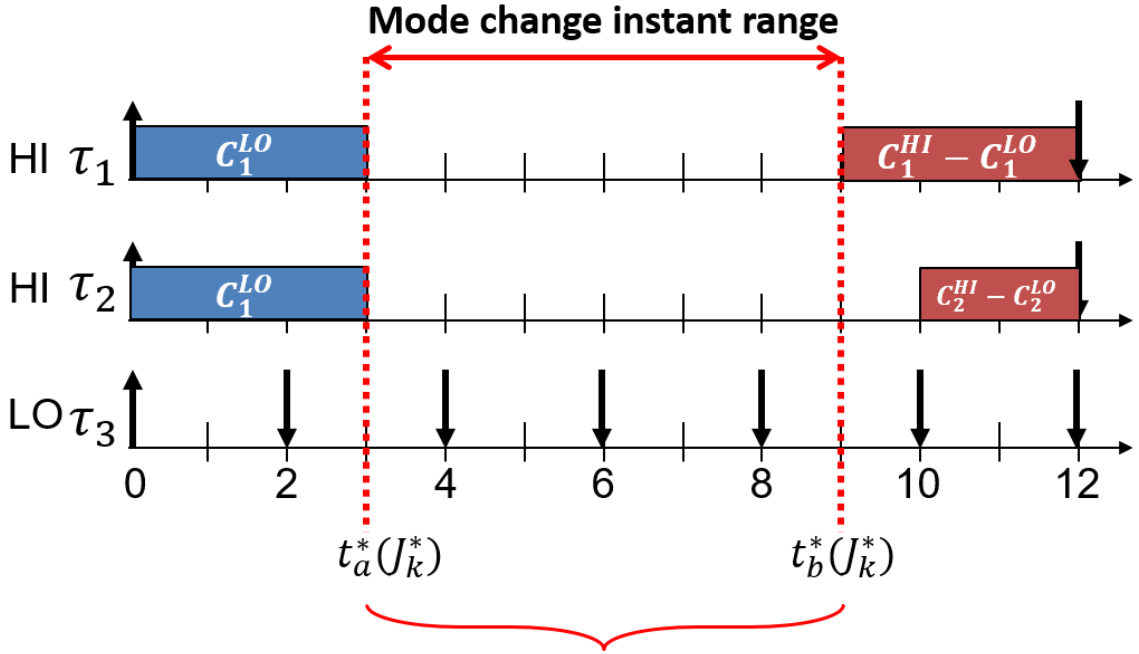
(e) Check infeasibility of the mode change instant t^* (Lemma 8)

Our Approach

(a) Target J_k^*

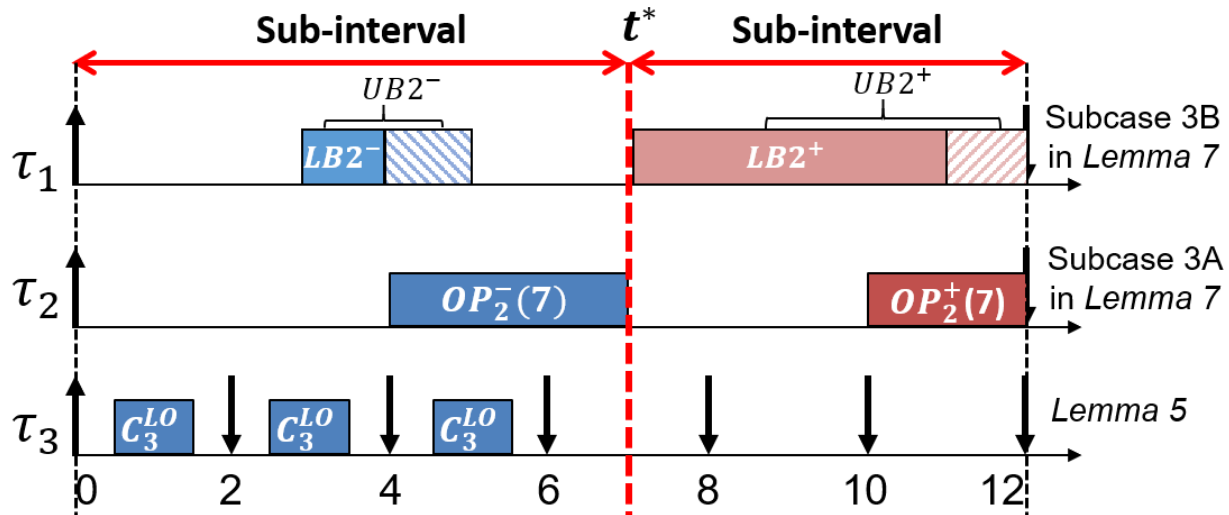
(the job with the earliest release time among all HI jobs whose execution requirement is strictly larger than LO WCET)

(b) Specify mode change instant range in $t^* \in [3,9]$ (*Lemma 4*)

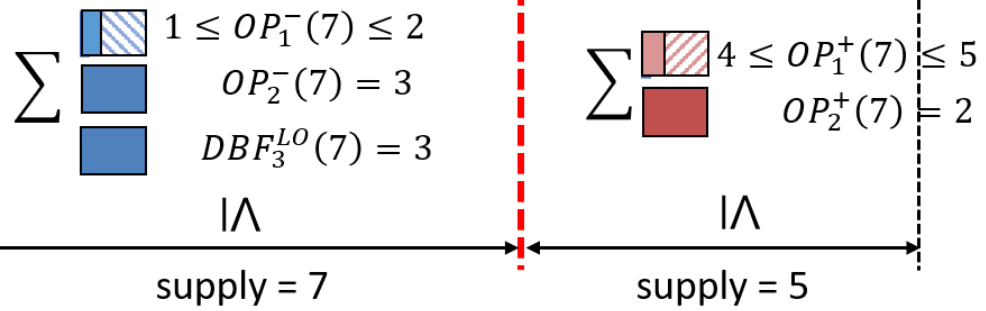


(f) Repeat for all $t^* \in [3,9]$, check no feasible mode change instant implying the task set's infeasibility (*Theorem 1*)

(c) Given t^*, select sub-intervals [0,7] and [7,12]



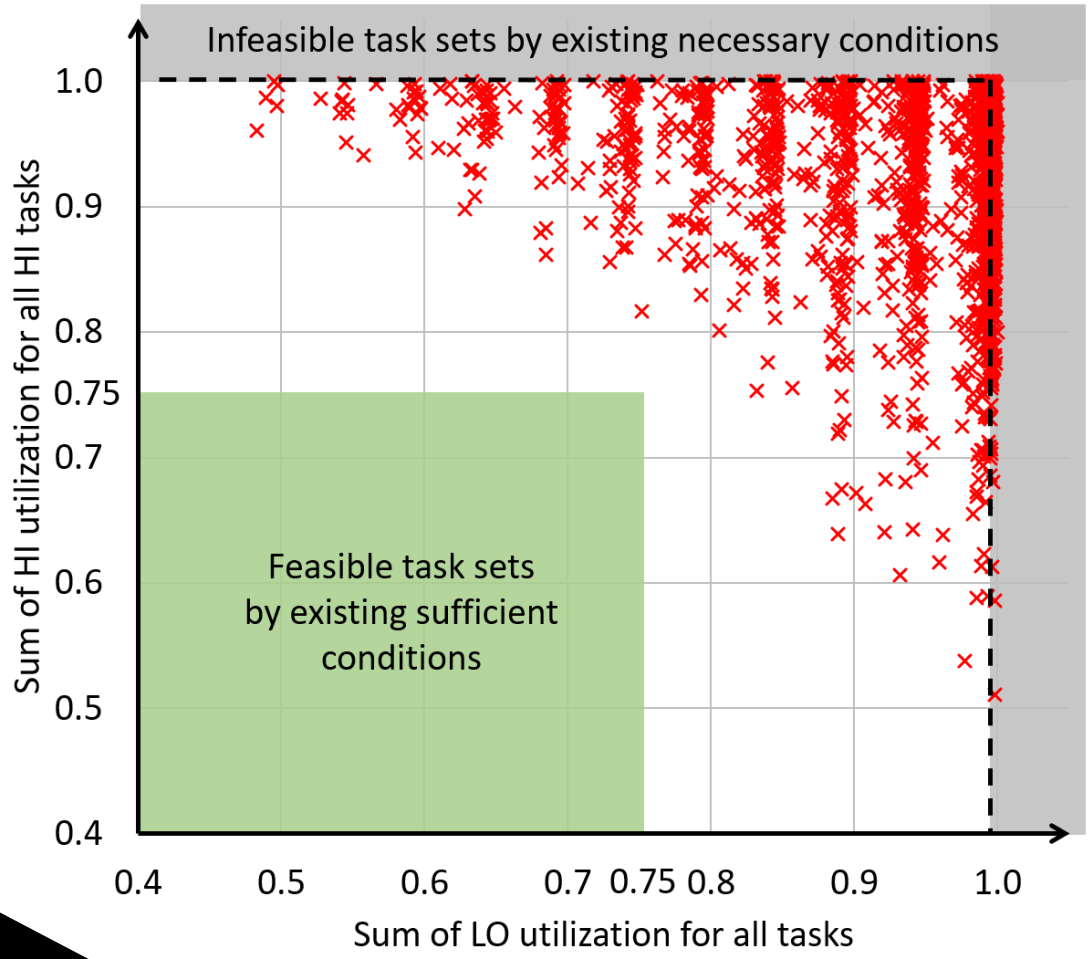
(d) Calculate demand in the sub-intervals (Lemmas 5,6,7)



(e) Check infeasibility of the mode change instant t^* (Lemma 8)

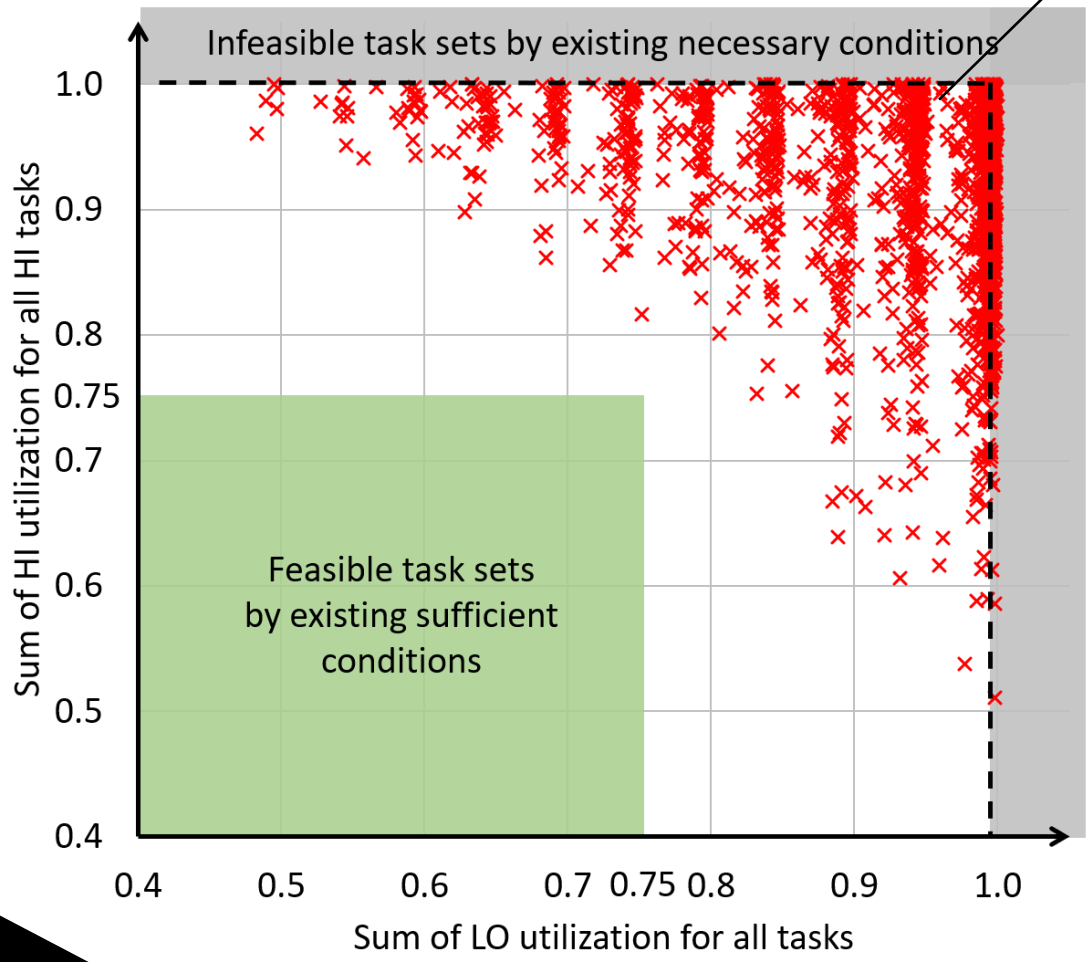
Evaluation

- Implicit-deadline task sets



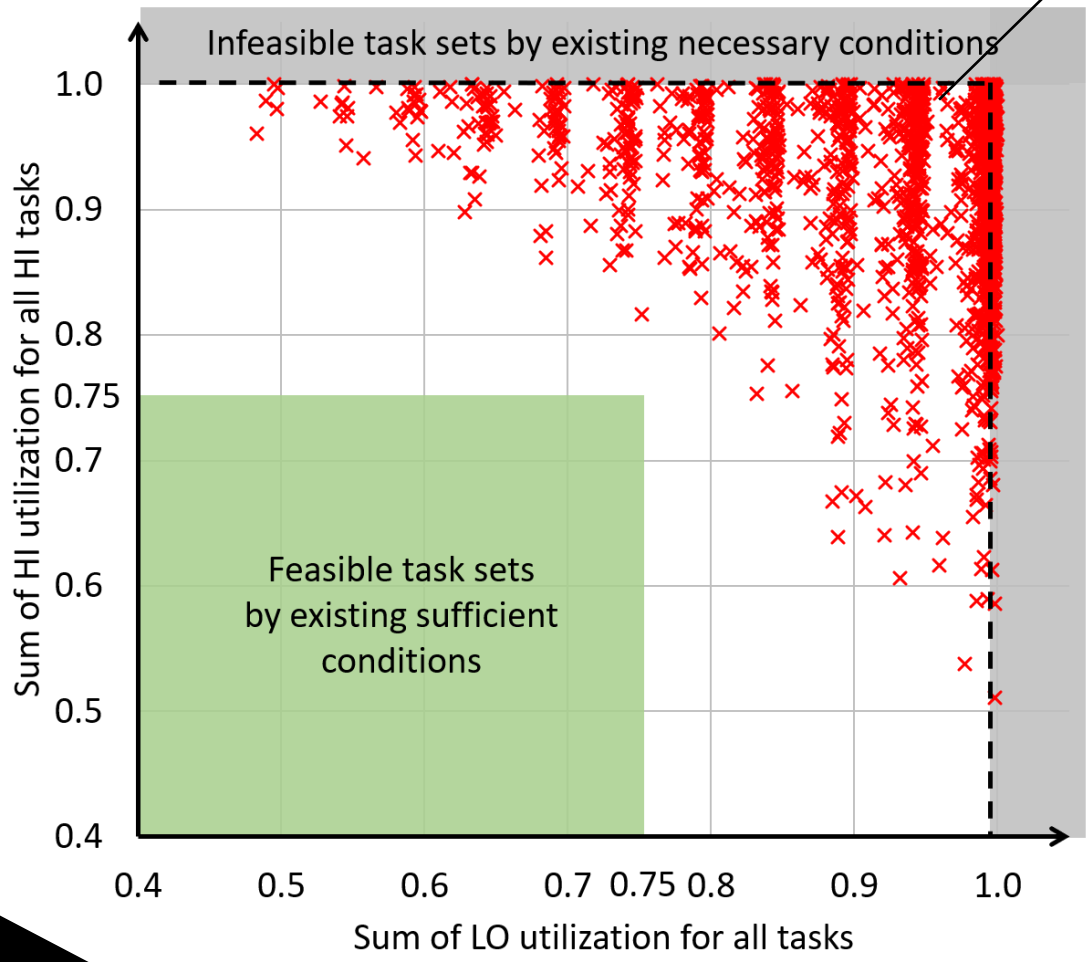
Evaluation

- Implicit-deadline task sets



Evaluation

- Implicit-deadline task sets

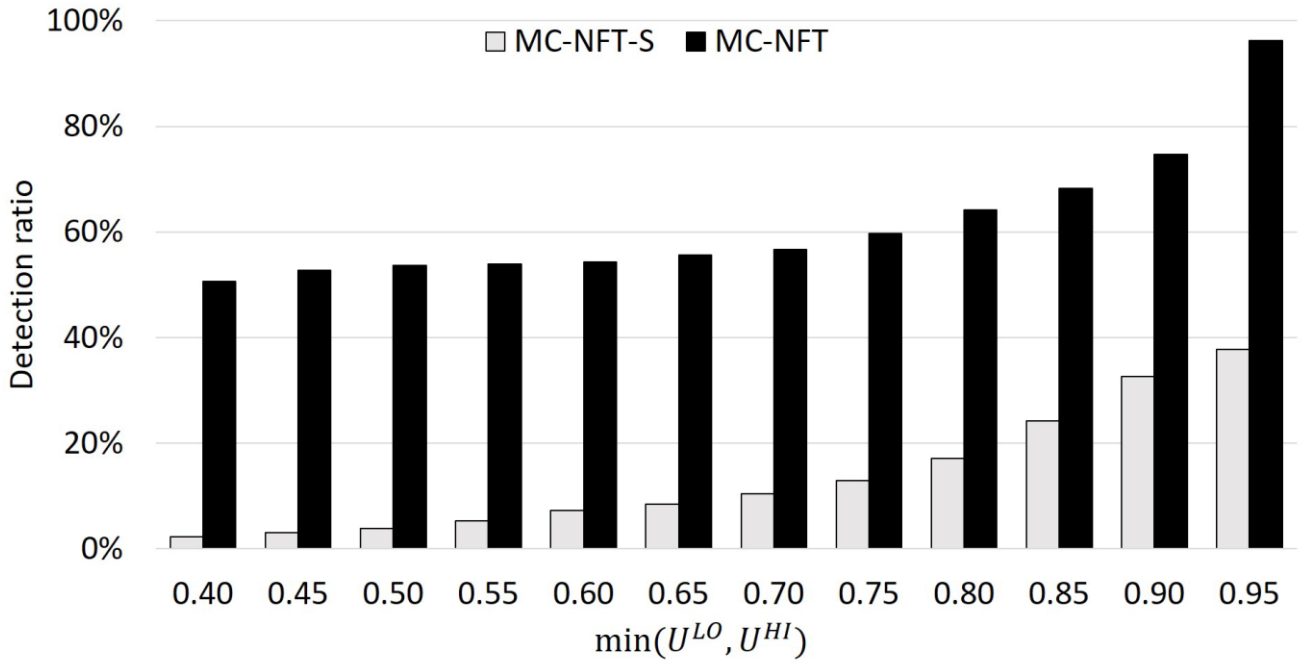


- Find several additional infeasible task sets **over a wider range** of LO and HI total utilization which have been proven neither feasible nor infeasible by any existing studies

- Identify significantly more infeasible task sets as LO and HI total utilization become close to 1.0

Evaluation

- Constrained-deadline task sets

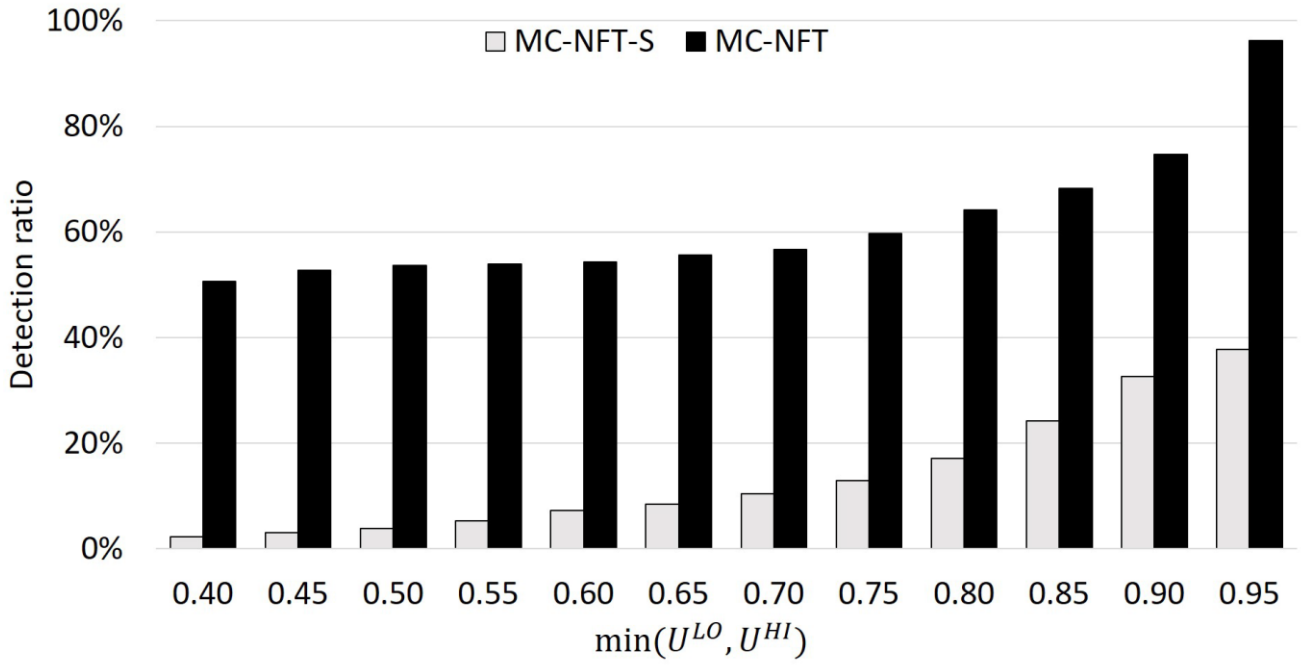


- Exhibit high capability in finding infeasible task sets
  - **MC-NFT**: **56%** task sets proven infeasible
  - **MC-NFT-S**: **8.2%** task sets proven infeasible
    among task sets which have been proven neither feasible nor infeasible by any existing studies

MC-NFT: collective necessary feasibility test in Theorem 2
MC-NFT-S: simplified version of MC-NFT in Theorem 3

Evaluation

- Constrained-deadline task sets



- Exhibit high capability in finding infeasible task sets
  - **MC-NFT**: **56%** task sets proven infeasible
  - **MC-NFT-S**: **8.2%** task sets proven infeasible
    among task sets which have been proven neither feasible nor infeasible by any existing studies

Benefit of dealing with unique issues in MC task systems

MC-NFT: collective necessary feasibility test in Theorem 2
MC-NFT-S: simplified version of MC-NFT in Theorem 3

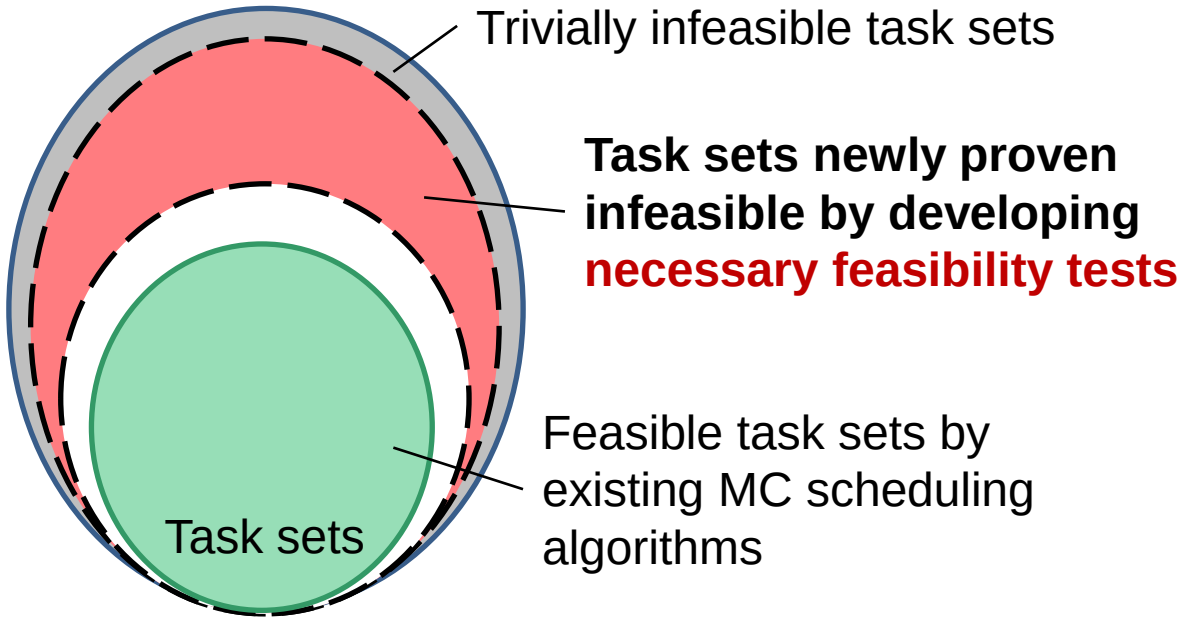
Conclusion

The paper provides

tight necessary feasibility tests for **mixed-criticality (MC)** systems on uniprocessor

Conclusion

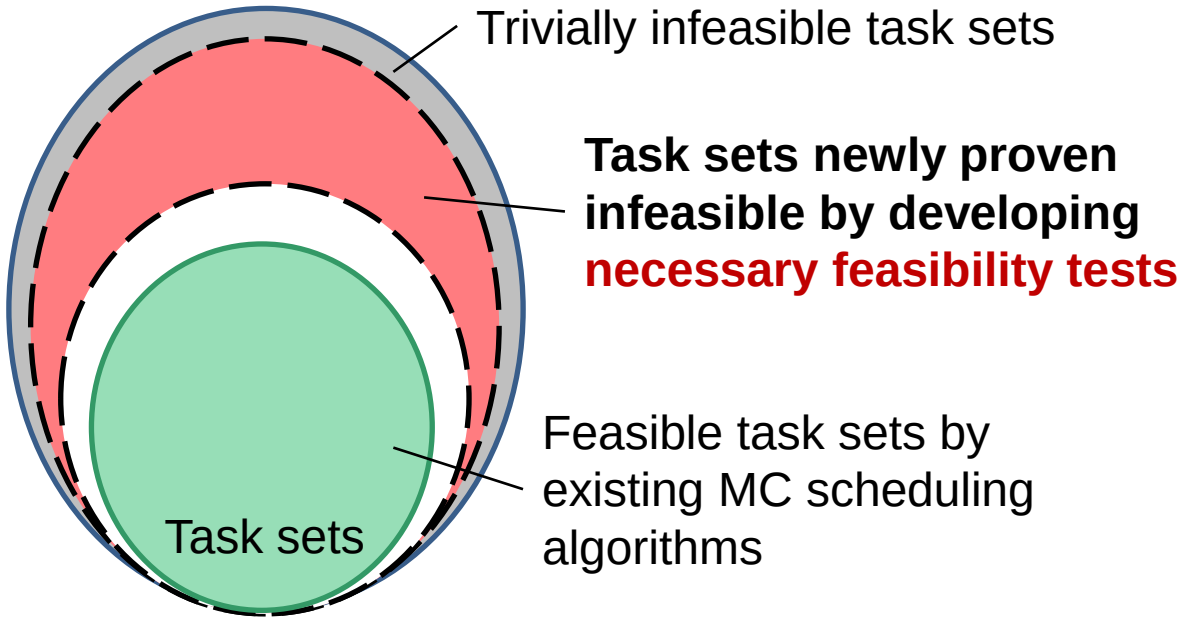
The paper provides
tight necessary feasibility tests for **mixed-criticality (MC)** systems on uniprocessor



The first study that yields non-trivial results for MC necessary feasibility

Conclusion

The paper provides
tight necessary feasibility tests for **mixed-criticality (MC)** systems on uniprocessor

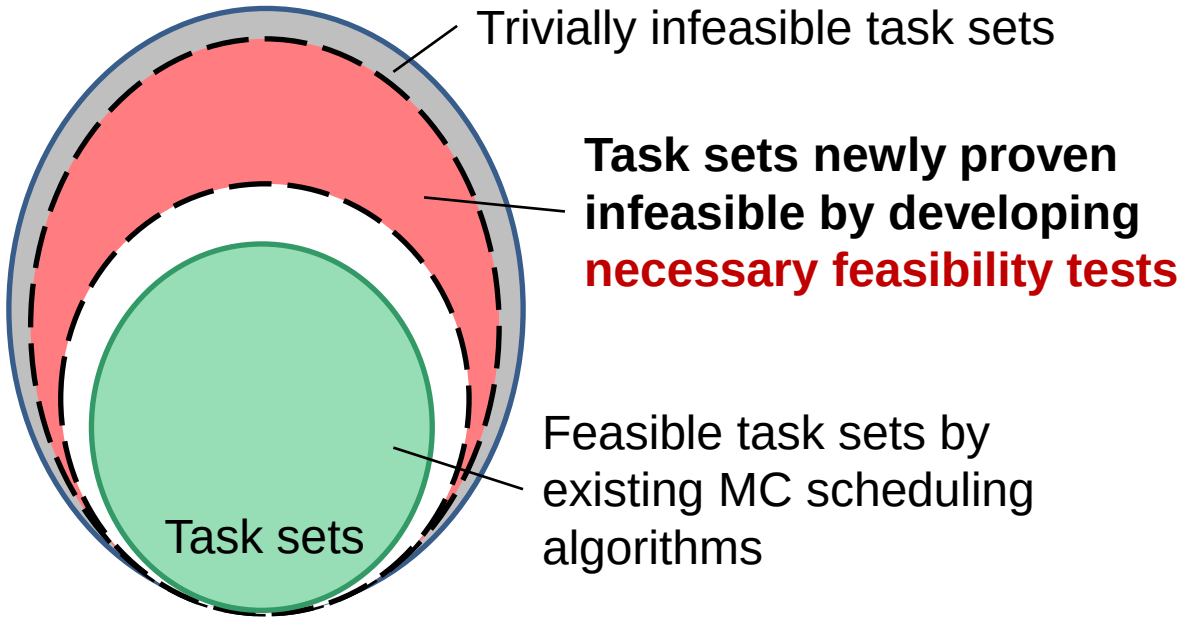


The first study that yields non-trivial results for MC necessary feasibility

- Investigate characteristics of MC systems in terms of necessary feasibility
- Identify new challenges posed by such characteristics of MC task systems
- Establish foundations of necessary feasibility tests for MC task systems

Conclusion

The paper provides
tight necessary feasibility tests for **mixed-criticality (MC)** systems on uniprocessor



The first study that yields non-trivial results for MC necessary feasibility

- Investigate characteristics of MC systems in terms of necessary feasibility
- Identify new challenges posed by such characteristics of MC task systems
- Establish foundations of necessary feasibility tests for MC task systems

Reduce a gap between the task sets proven feasible and that proven infeasible