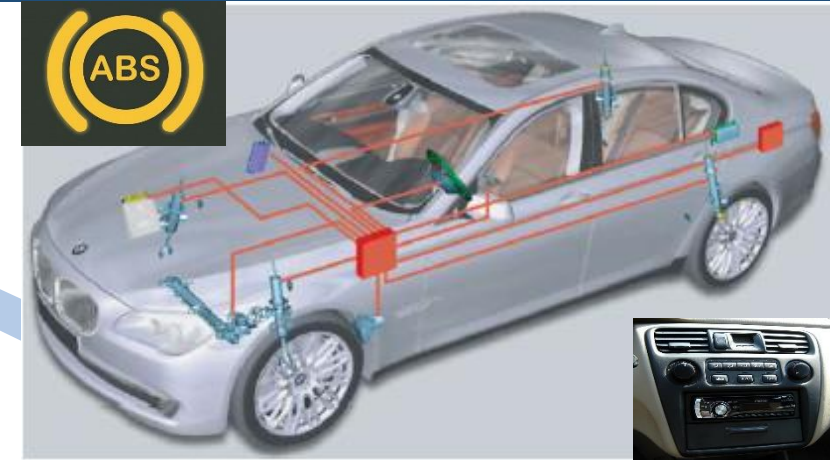
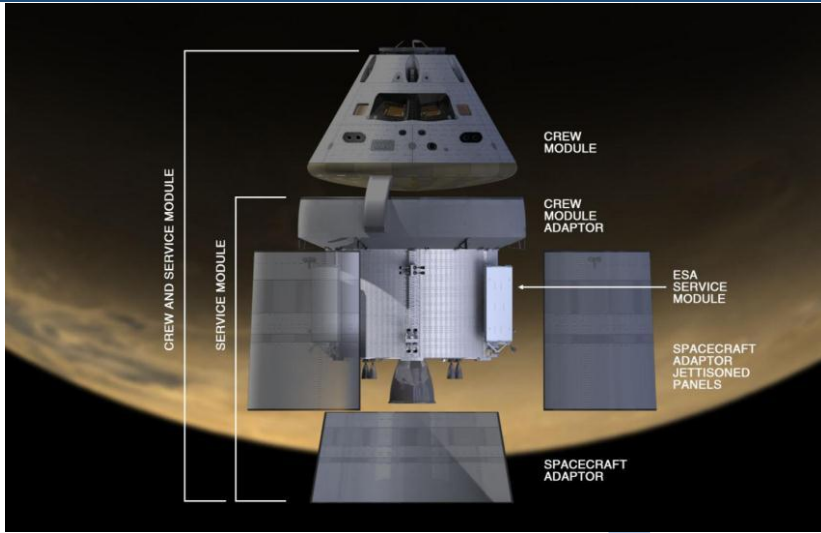


Criticality- and Requirement-aware Bus Arbitration for Multi-core Mixed Criticality Systems

Mohamed Hassan and Hiren Patel

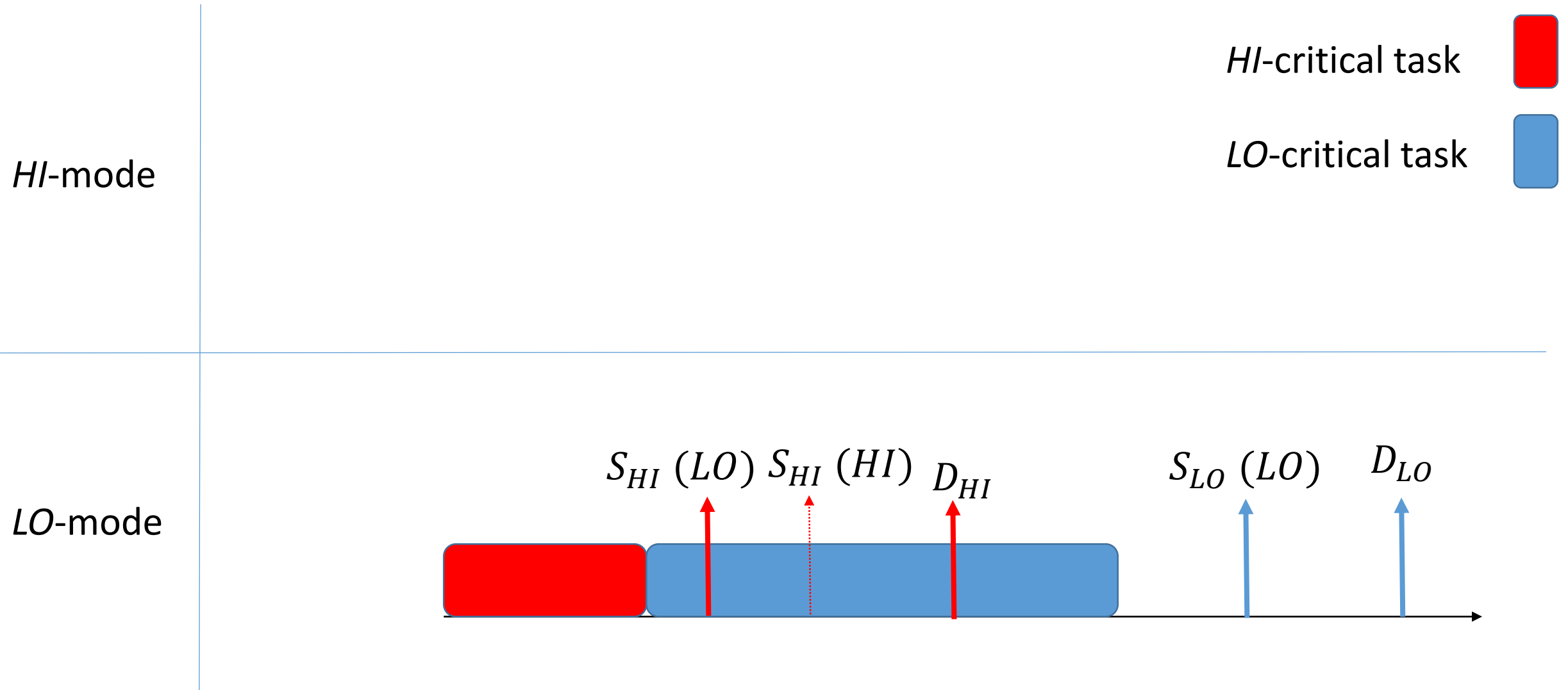
MCS: Physical Criticality Side



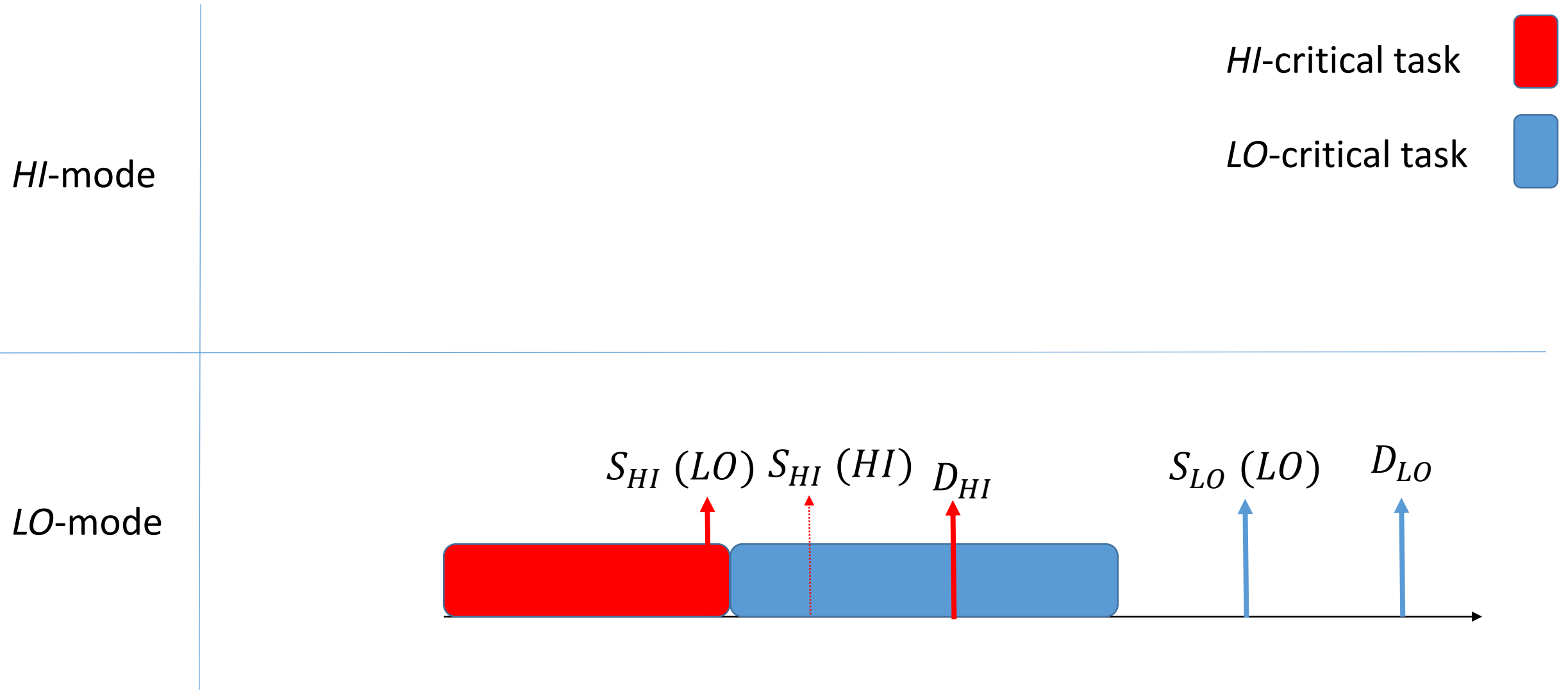
MCS



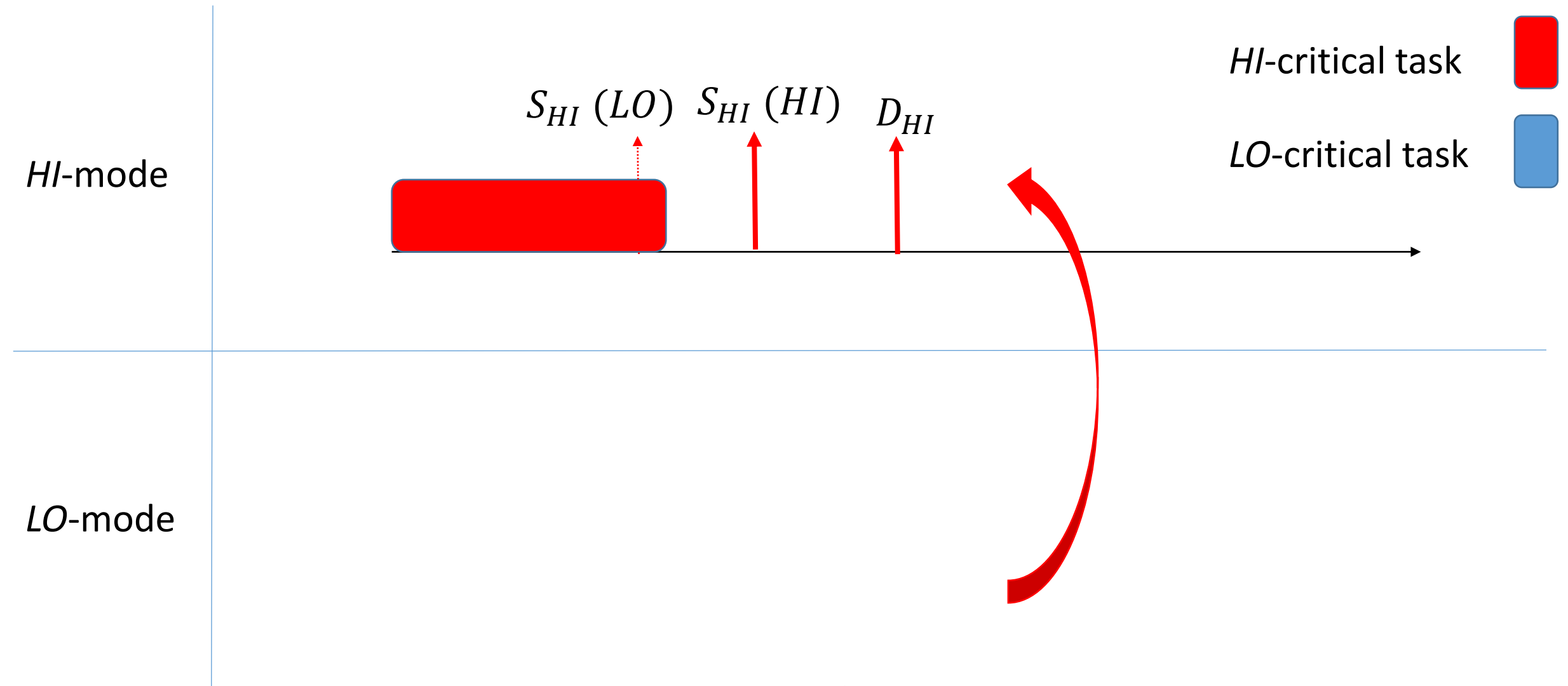
MCS: Schedulability Side



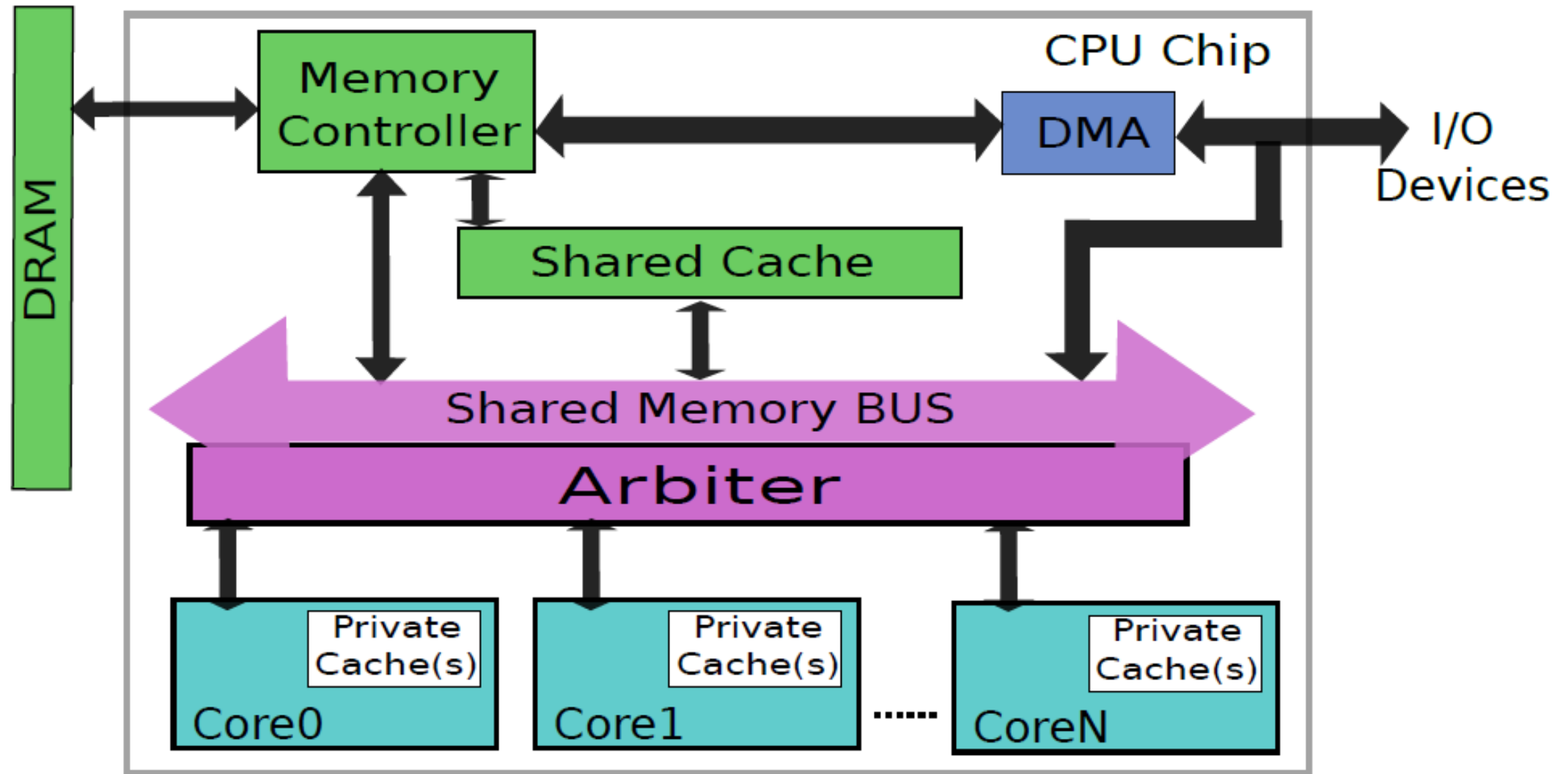
MCS: Schedulability Side



MCS: Schedulability Side

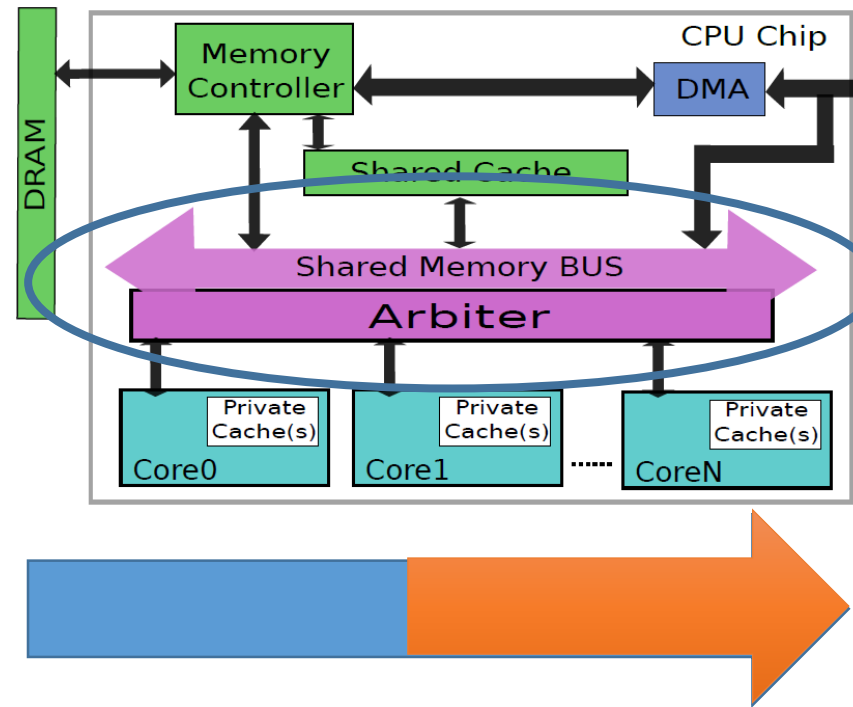


Multi-core MCS



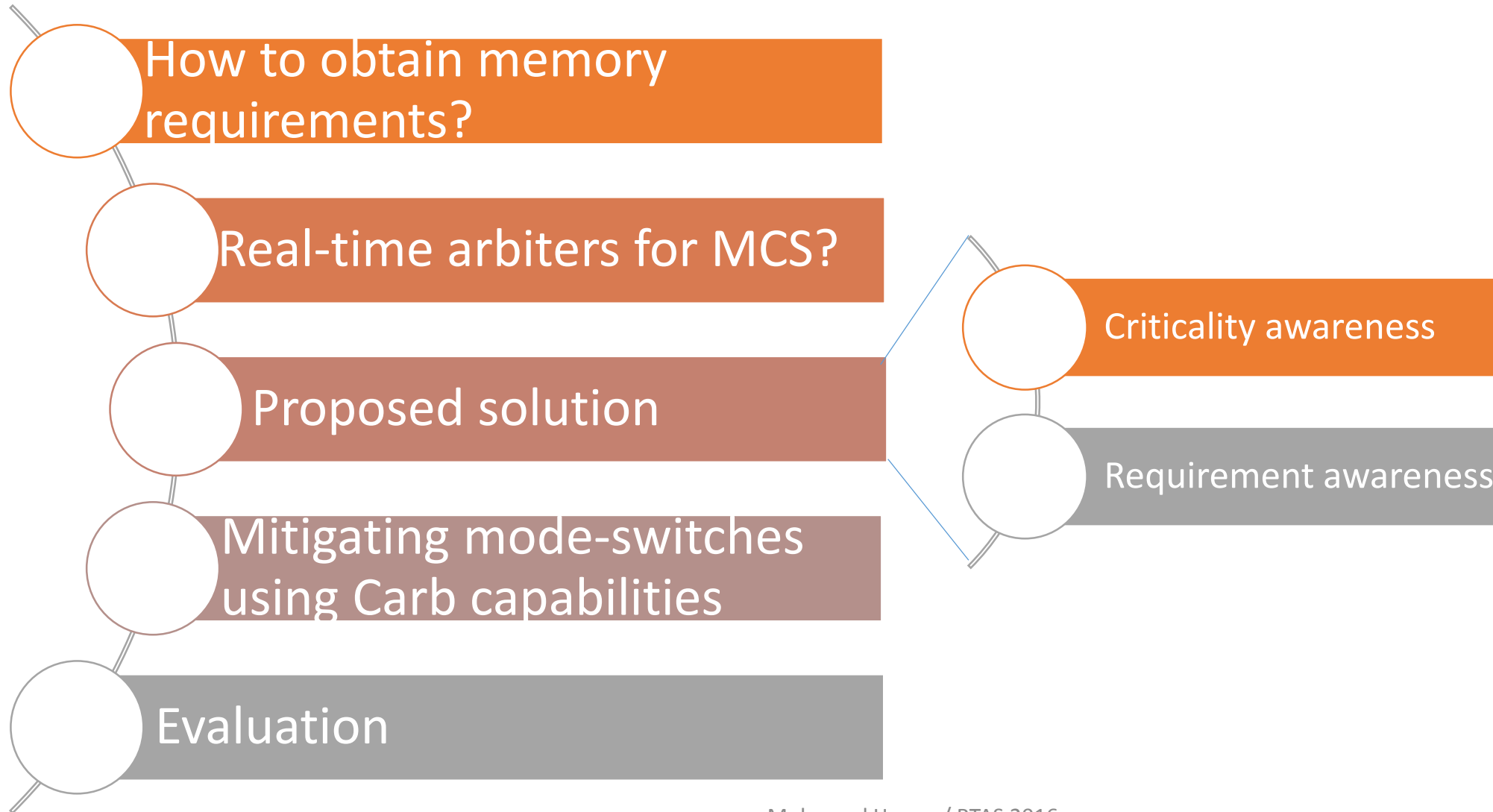
Problem Statement

Arbitrate accesses to the shared memory bus such that memory latency requirements of all tasks are satisfied



Mitigate mode switches in MCS utilizing dynamic memory re-arbitration

Outline



Traditional MCS model

- Originally proposed for single-core MCS

$task = \langle CL, Deadline, WCET(l) \rangle$

Calculated in isolation (no interference amongst cores)

Traditional MCS model

- Originally proposed for single-core MCS

$$task = \langle CL, Deadline, WCET(l) \rangle$$

Calculated in isolation (no interference amongst cores)

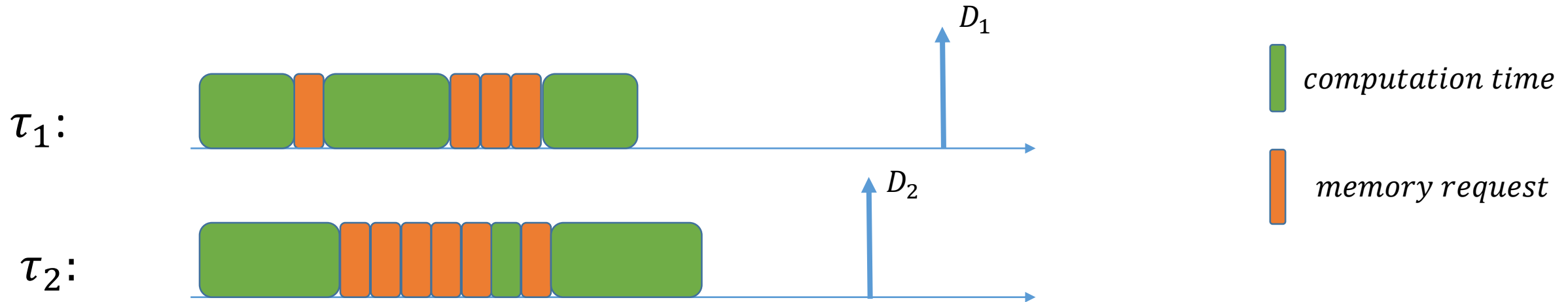
To account for interferences in multi-core

$$task = \langle CL, Deadline, WCCT(l), WC \#requests, WC \text{ interference} \rangle$$

Obtained by static analysis or simulation

$WC \text{ memory time} \times \#requests$

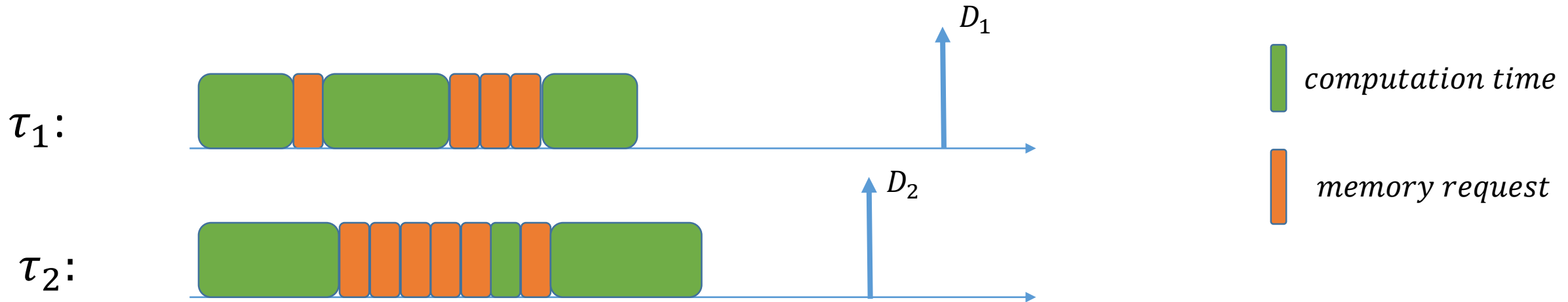
Execution Time Decomposition



Schedulability condition:

$$f(WCET) \leq f(deadline)$$

Execution Time Decomposition



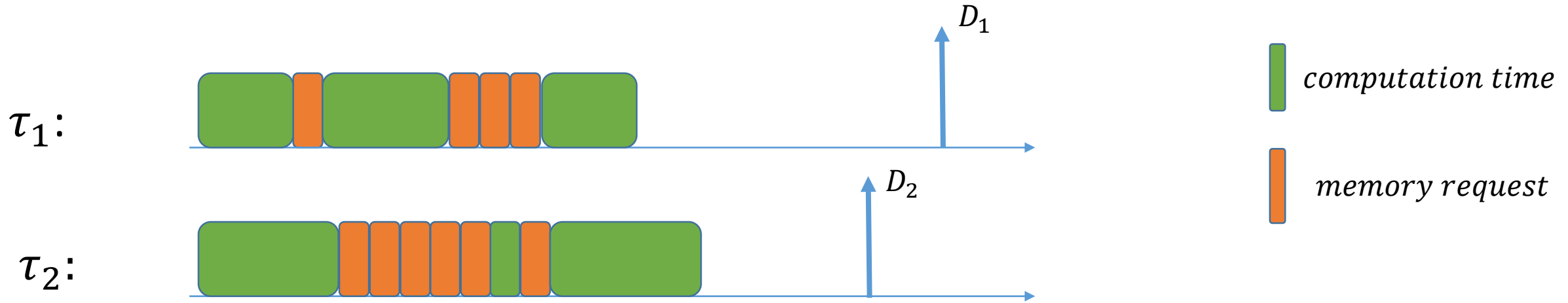
Schedulability condition:

$$f(WCET) \leq f(deadline)$$



$$f(WCCT, WC \text{ memory time}, \#mem \text{ requests}) \leq f(deadline)$$

Execution Time Decomposition



Schedulability condition:

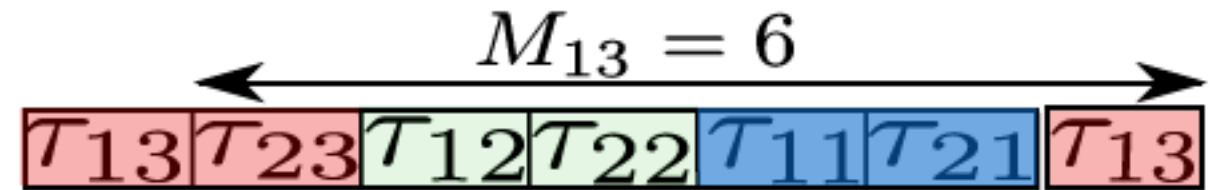
$$f(WCET) \leq f(deadline)$$



$$f(WCCT, \text{WC memory time}, \# \text{memory requests}) \leq f(deadline)$$

Real-time Arbiters for MCS?

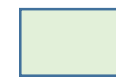
	Requirement Awareness	Criticality Awareness
RR		



Highest
criticality



Medium
criticality

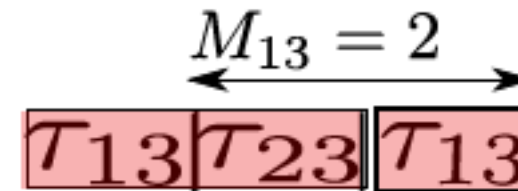
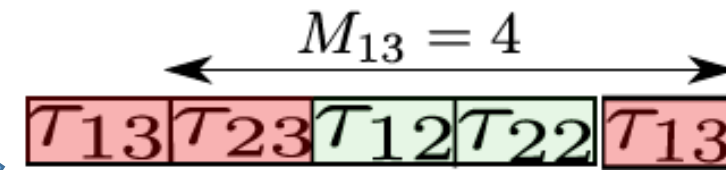


Lowest
criticality



Real-time Arbiters for MCS?

	Requirement Awareness	Criticality Awareness
RR		
PRR (ISCA 2009)		



Highest
criticality



Medium
criticality

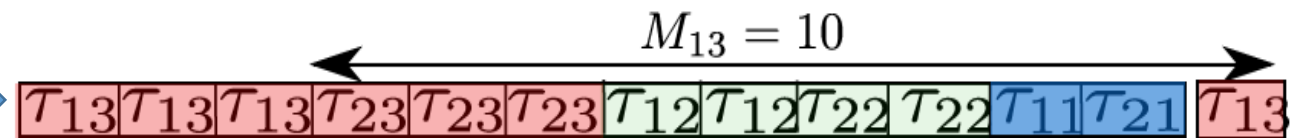
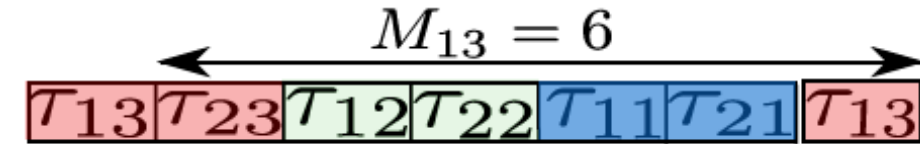


Lowest
criticality



Real-time Arbiters for MCS?

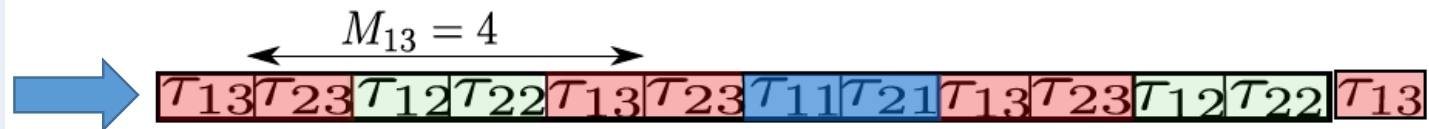
	Requirement Awareness	Criticality Awareness
RR		
PRR		
WRR/TDM		



Highest criticality 
 Medium criticality 
 Lowest criticality 

Real-time Arbiters for MCS?

	Requirement Awareness	Criticality Awareness
RR		
PRR		
WRR/TDM		
HRR (RTSS 2011)/ Distributed TDM (RTAS2015)		



Highest
criticality



Medium
criticality



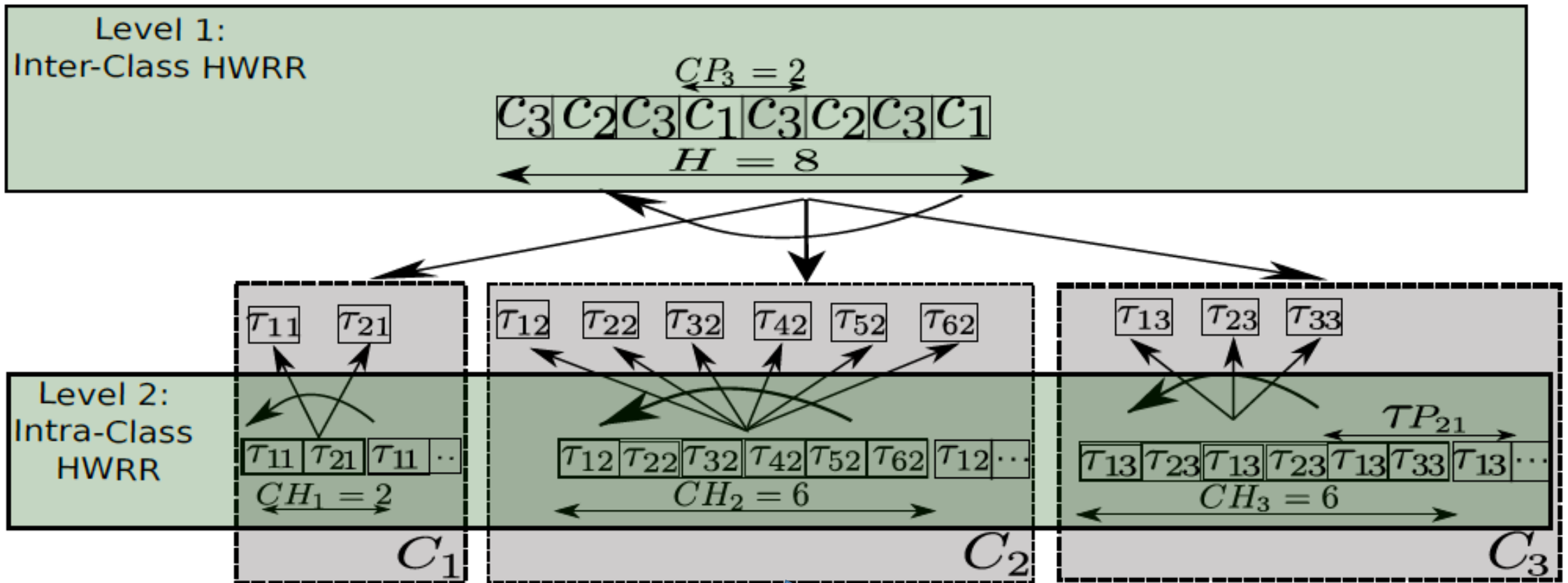
Lowest
criticality



Real-time Arbiters for MCS?

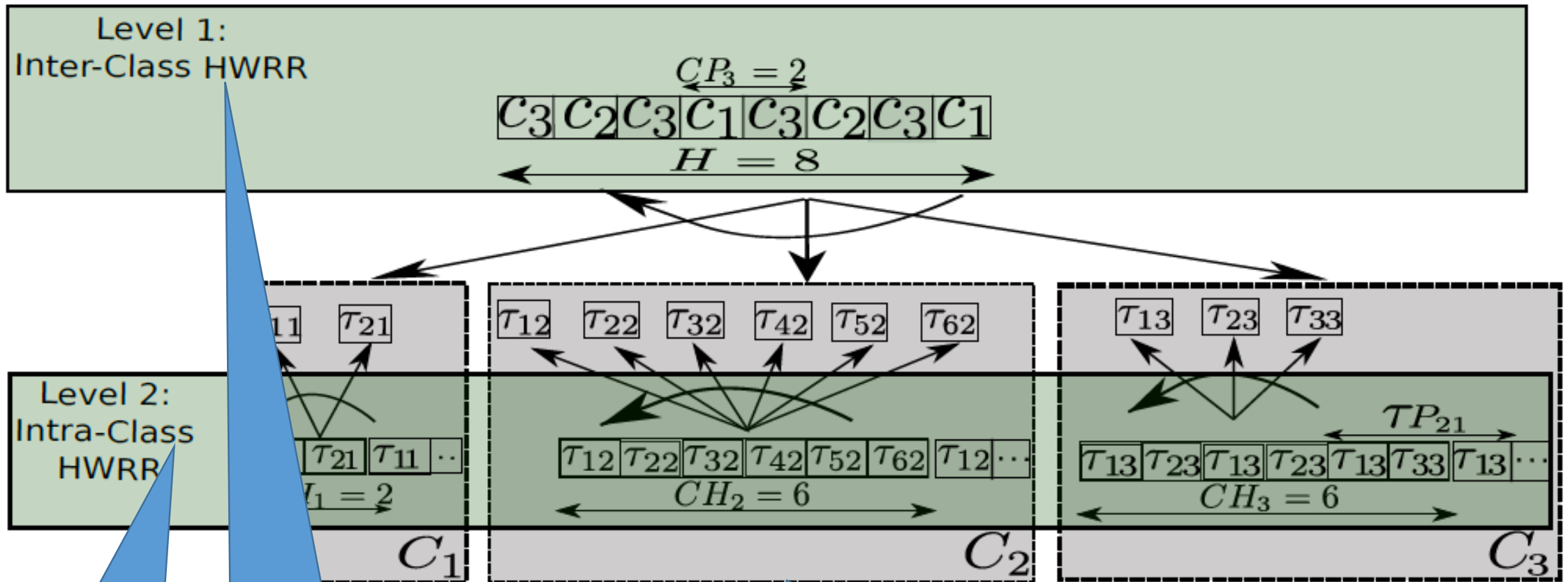
	Requirement Awareness	Criticality Awareness
RR		
PRR		
WRR/TDM		
HRR (RTSS 2011)/ Distributed TDM (RTAS2015)		
C Arb		

CArb Arbitration



(1) Tasks of same criticality combined into same class

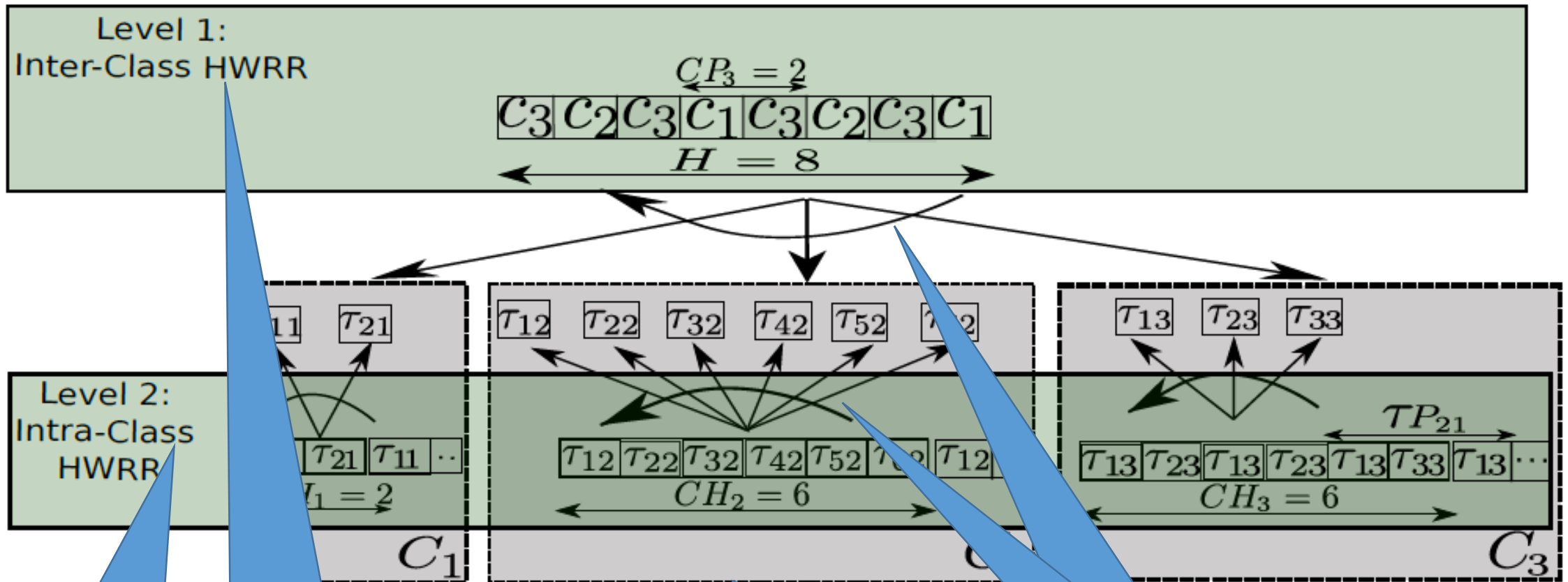
CArb Arbitration



(2) Two-tier Hierarchical Arbitration To separate the interference across classes

(1) Tasks of same criticality combined into same class

C Arb Arbitration

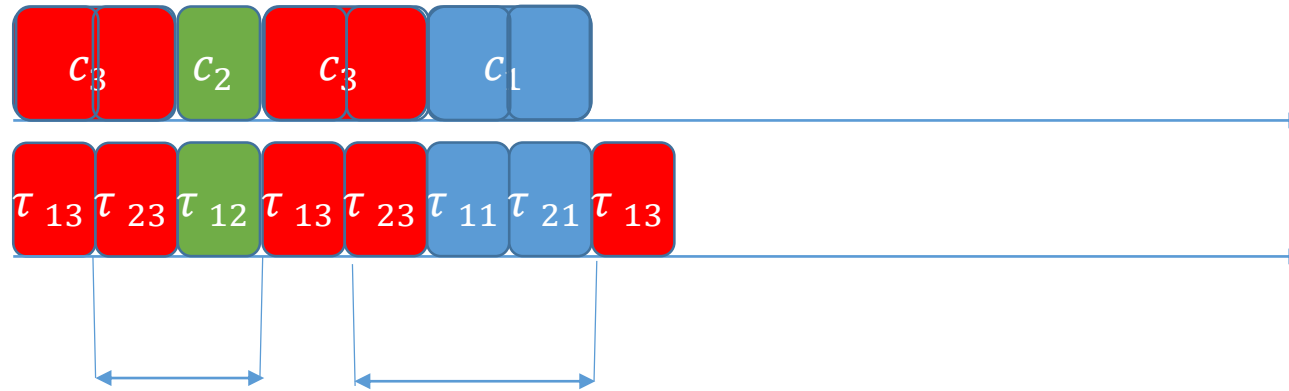


(2) Two-tier Hierarchical Arbitration To separate the interference across classes

(1) Tasks of same criticality combined into same class

(3) Harmonic WRR with optimal assignment to satisfy requirements

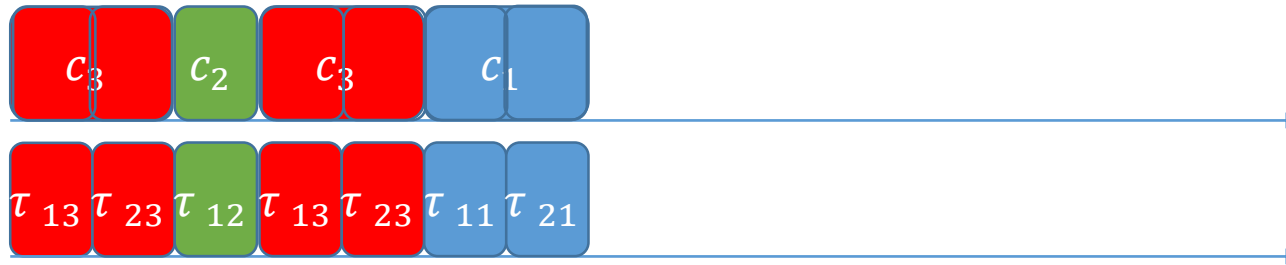
Criticality Awareness



Interference decomposition:

- Intra-class (same criticality): τ_{13} suffers a WC interference of 1 slot from τ_{23}
- Inter-class (other criticalities): τ_{13} suffers a WC interference of 2 slots from c_1 or 1 slot from c_2

Criticality Awareness



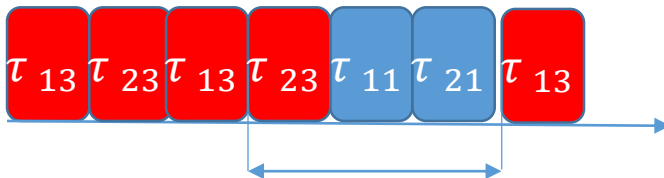
Dynamic re-arbitration

- What is the right arbitration decision to decrease interference on τ_{13} ?

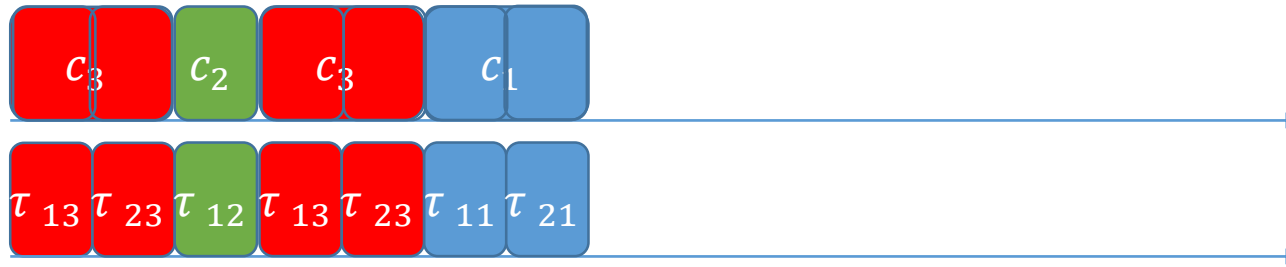
Solution 1: throttling tasks of class c_2

Intra-interference = 2

has no effect at all on WC!



Criticality Awareness



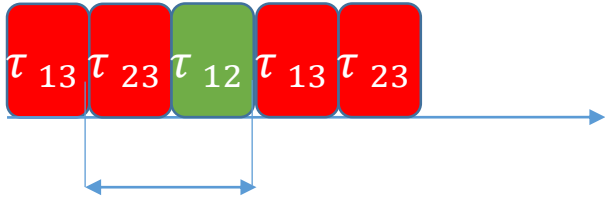
Dynamic re-arbitration

- What is the right arbitration decision to decrease interference on τ_{13} ?

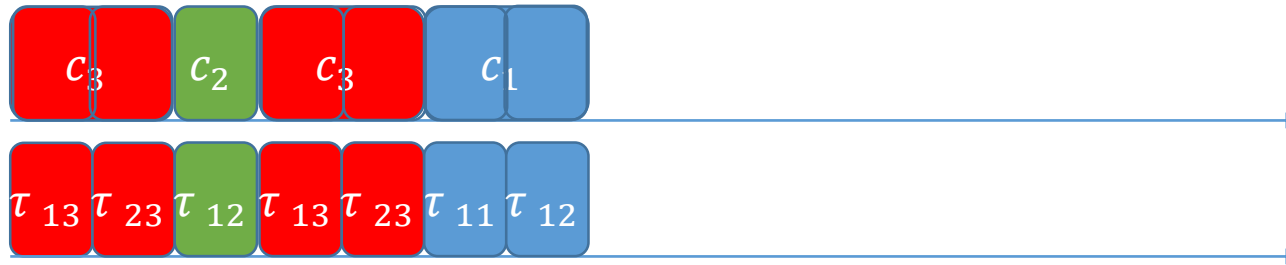
Solution 2: throttling all tasks of class c_1

unnecessarily conservative!

Intra-interference = 1



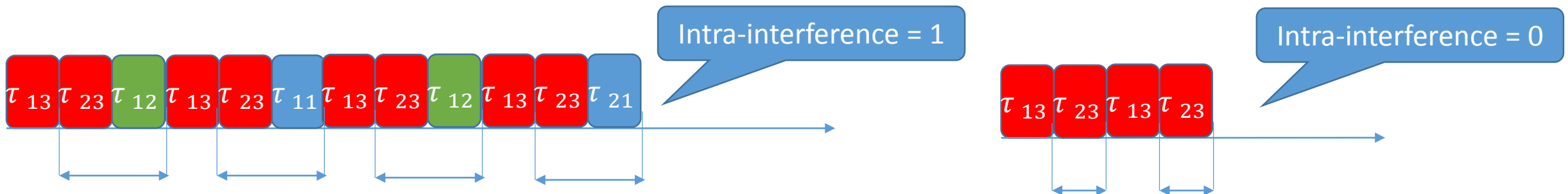
Criticality Awareness



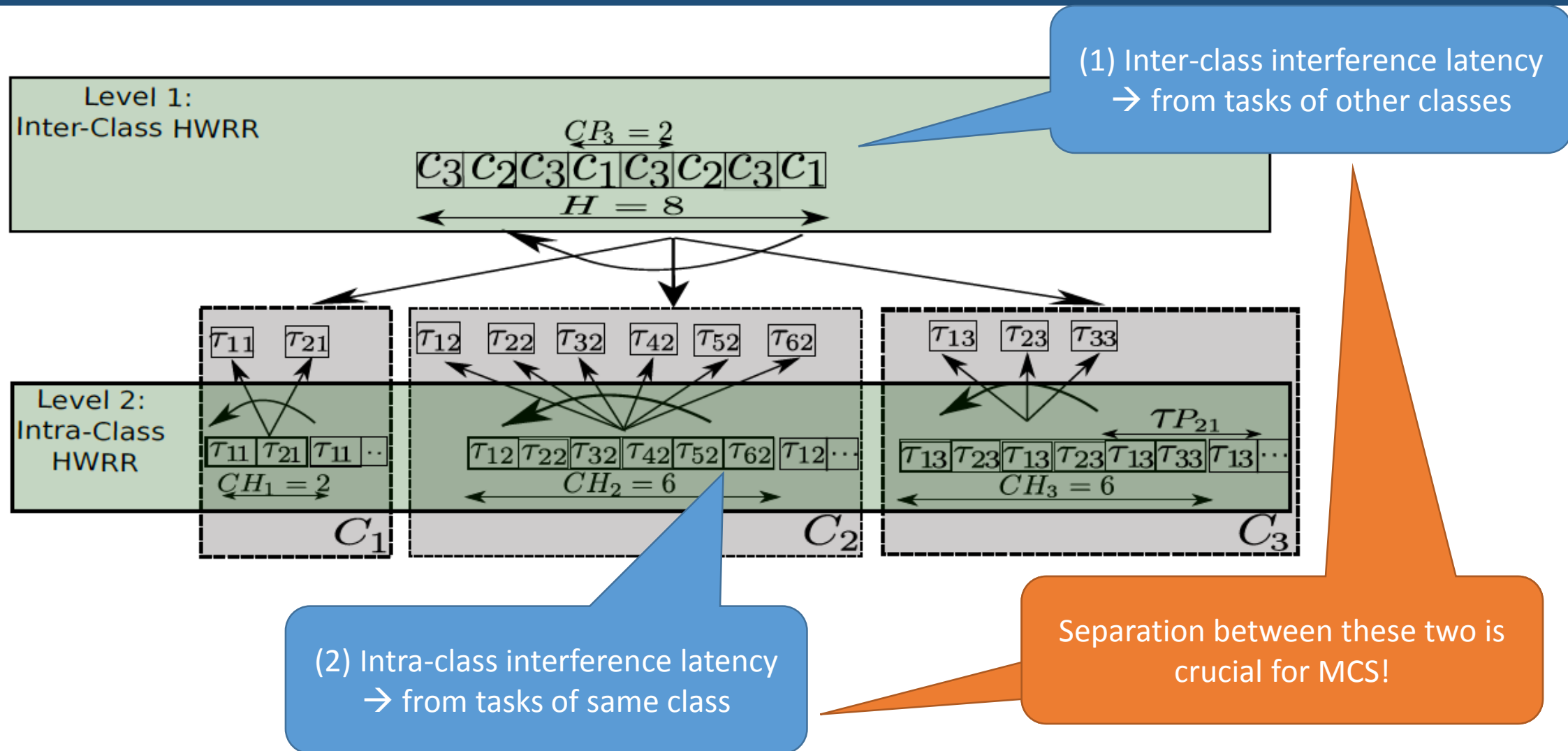
Dynamic re-arbitration

- What is the right arbitration decision to decrease interference on τ_{13} ?

How much “decrease” is enough ?

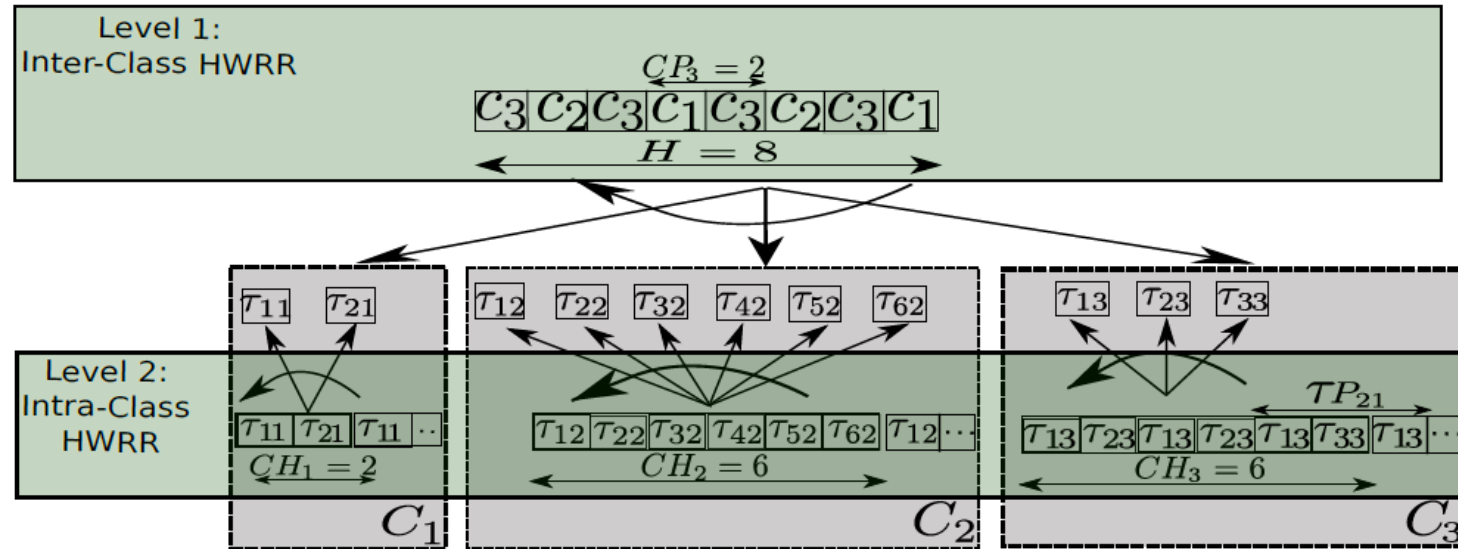


Requirement Awareness-WC latency



Requirement Awareness-Optimal Assignment

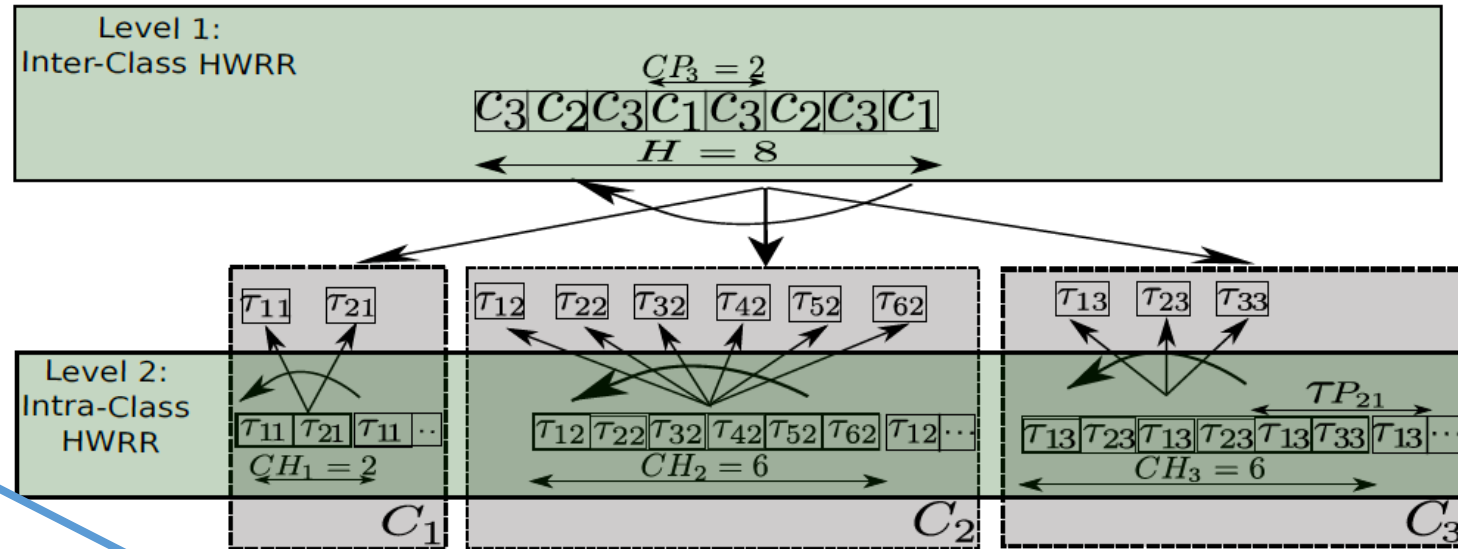
Target:
minimize schedule
hyperperiod



Requirement Awareness-Optimal Assignment

Target:
minimize schedule
hyperperiod

(C.1) memory
latency requirement



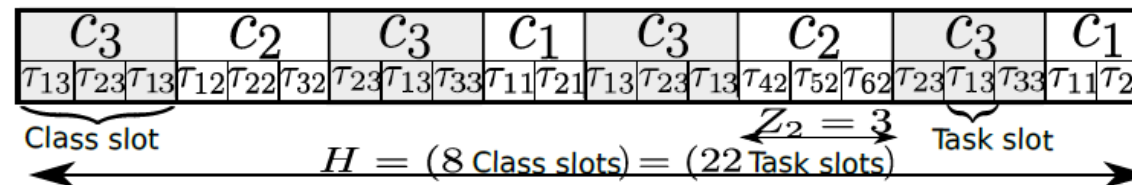
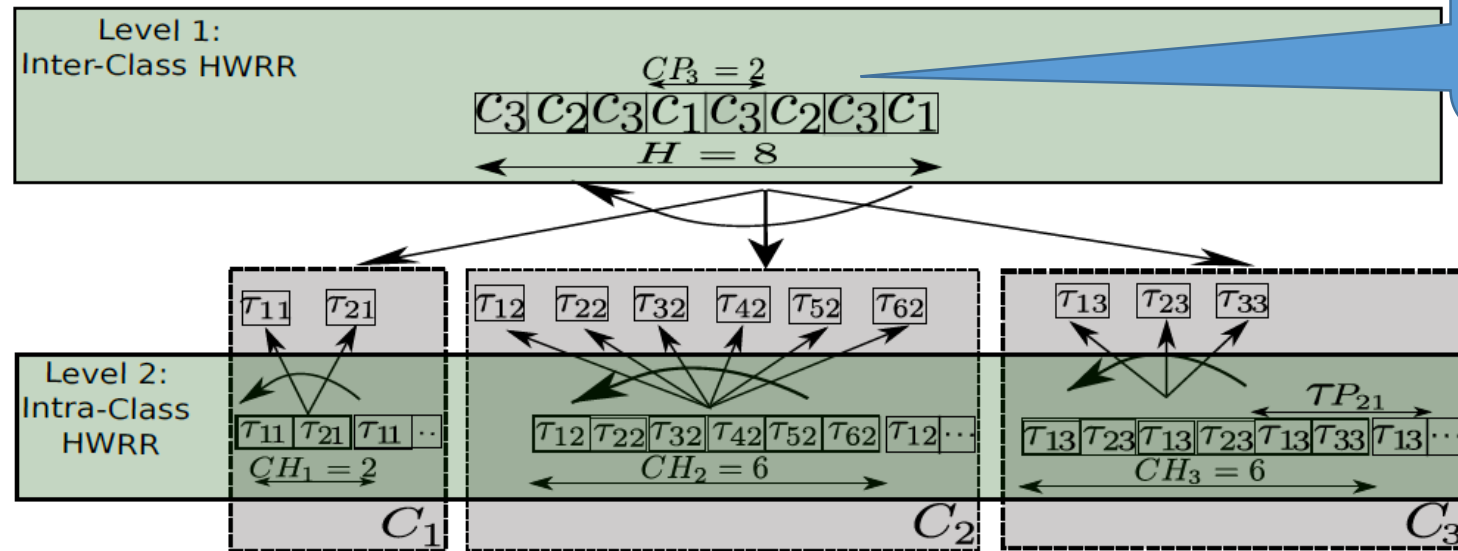
$$f(WCCT, \text{WC memory time}, \# \text{memory requests}) \leq f(\text{deadline})$$

Requirement Awareness-Optimal Assignment

Target:
minimize schedule
hyperperiod

(C.1) memory
latency requirement

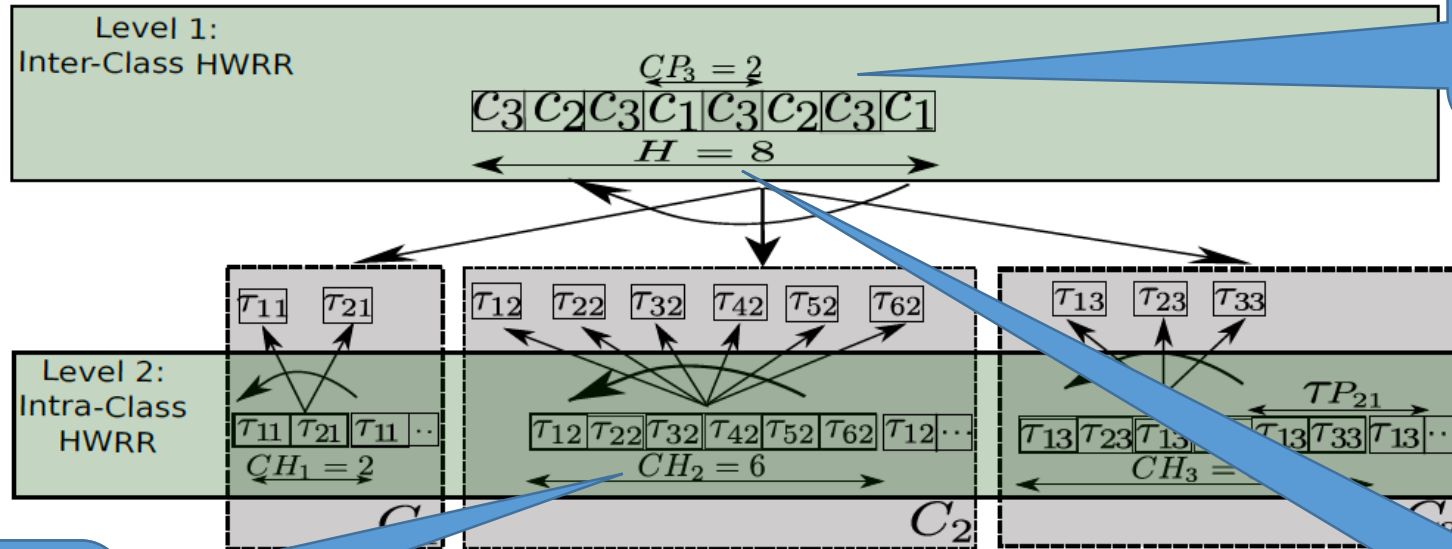
(C.2) Starvation:
 $H \geq CP_l \geq 2$
Starving C_l Starving
other classes



Requirement Awareness-Optimal Assignment

Target:
minimize schedule
hyperperiod

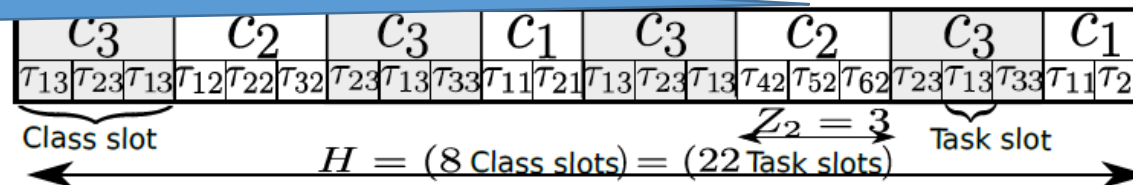
(C.1) memory
latency requirement



(C.2) Starvation:
 $H \geq CP_l \geq 2$
Starving C_l Starving other classes

(C.6) Periodicity:
total number of task slots
of a class is integer
multiple of its hyperperiod

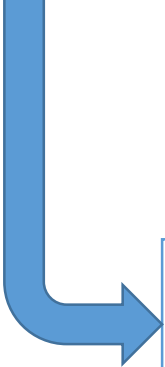
(C.4) Periodicity:
 H is integer multiple of
class periods



Λ : Function of the Criticality Level?

WC computation time and WC memory requests are obtained by same methods: *timing analysis* or *simulations*

WC computation time is a function of the CL $\rightarrow$ why not WC memory requests?

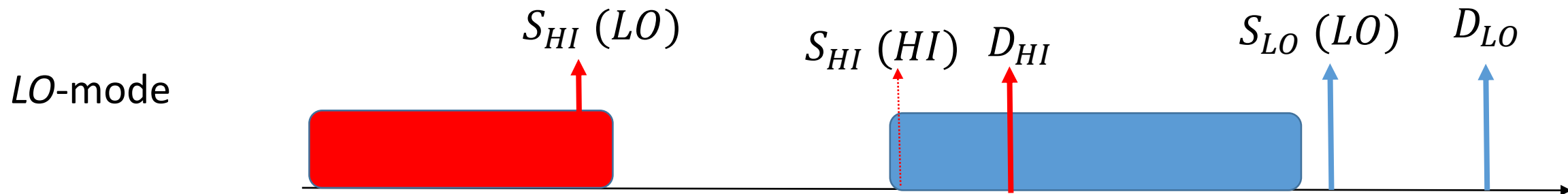
- 
- CArb: run the optimization framework per CL (operation mode)
 - Execute a distinct schedule per operation mode

Dynamic Re-arbitration- The Problem

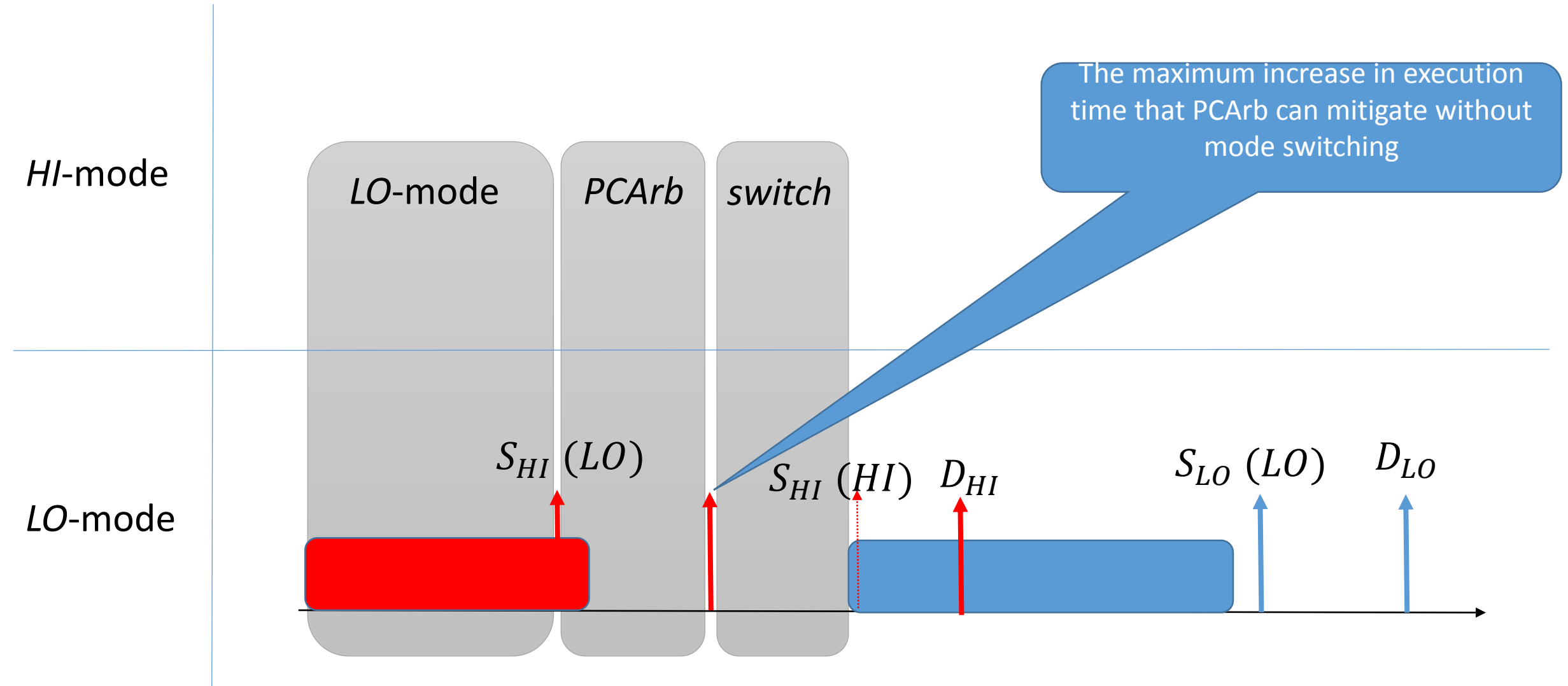
- Problems with traditional approach:
 1. Suspending l -critical tasks at the $(l + 1)$ -mode entails having no guarantees for those tasks
 - They are still critical tasks! (WMC 2013)
 2. Mode switching at the OS scheduling level results in huge overheads (RTAS 2015)
 - Minimizing those switches is highly desirable!

Scheme1: Prioritized CArb

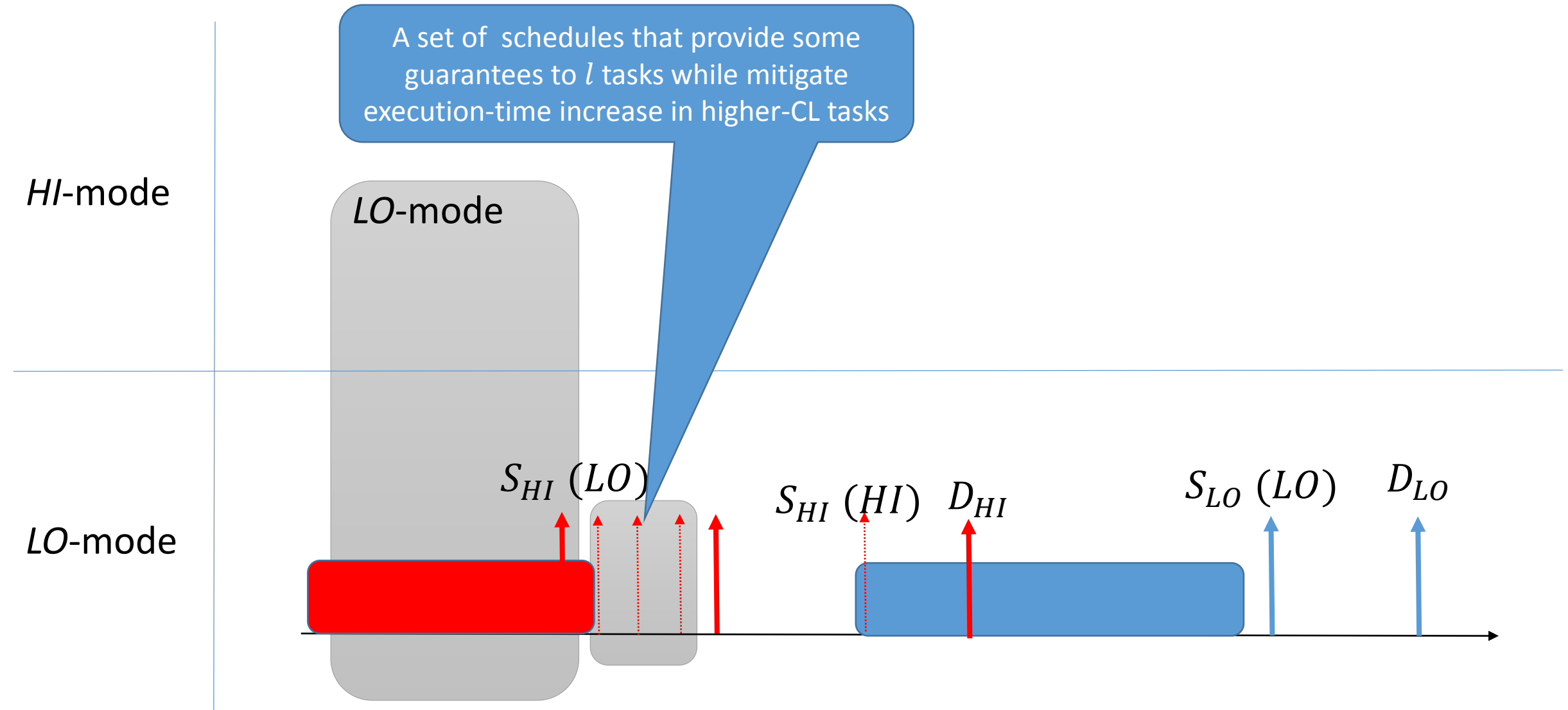
- Don't suspend l tasks.
- Allow them to access memory only on slack slots $\rightarrow$ **eliminated their memory interference**
 - total execution time = computation time + interference delay



Scheme1: Prioritized CArb



Scheme2: Prioritized CArb Is A Special Case!



Evaluation: Avionics case-study

Use-case requirements					Processor Scheduling using [21]			Optimal CARb parameters	
τ_{jl}	D_{jl} (ms)	S_{jl} (ms)	Λ_{jl}	Partition	Core		memory access requirements	τw_{jl}	(CW_l, Z_l)
τ_{14}	25	1.06	500	1	1		$M_{14} \leq 5.02\mu s$	6	(3, 4)
τ_{24}	50	3.09	500	2	1		$M_{24} \leq 8.11\mu s$	3	
τ_{34}	100	2.7	500	3	1		$M_{34} \leq 23.17\mu s$	1	
τ_{44}	200	1.09	500	4	1		$M_{44} \leq 45.96\mu s$	2	
τ_{13}	25	0.94	1000	5	2		$2M_{13} + M_{23} + M_{33} \leq 6.45\mu s$	6	(6, 4)
τ_{23}	50	1.57	1000					4	
τ_{33}	50	1.68	1000					4	
τ_{43}	50	4.5	1000	6	2	3/5	$4M_{43} + 4M_{53} + 2M_{63} + M_{73} \leq 35.28\mu s$	3	
τ_{53}	50	2.94	1000					3	
τ_{63}	100	1.41	1000					3	
τ_{73}	200	6.75	1000					1	
τ_{12}	50	5.4	4000	7	3	0.4	$M_{12} \leq 1.77\mu s$	1	(3, 1)
τ_{11}	50	2.4	2000	8	3	0.6	$4M_{11} + M_{21} + 4M_{31} + M_{41} \leq 28.77$	5	(12, 5)
τ_{21}	200	0.94	2000					2	
τ_{31}	50	1.06	2000					5	
τ_{41}	200	2.28	2000					3	
τ_{51}	25	4.75	3000	9	4	1	$8M_{51} + 2M_{61} + M_{71} + 2M_{81} + 4M_{91} \leq 24.17$	20	
τ_{61}	100	12.87	3000					6	
τ_{71}	200	0.47	3000					3	
τ_{81}	100	1.24	3000					6	
τ_{91}	50	1.62	3000					10	

TDM across partitions and RM within partition(Sha/RTCSA 2004)

21 task and 4 classes

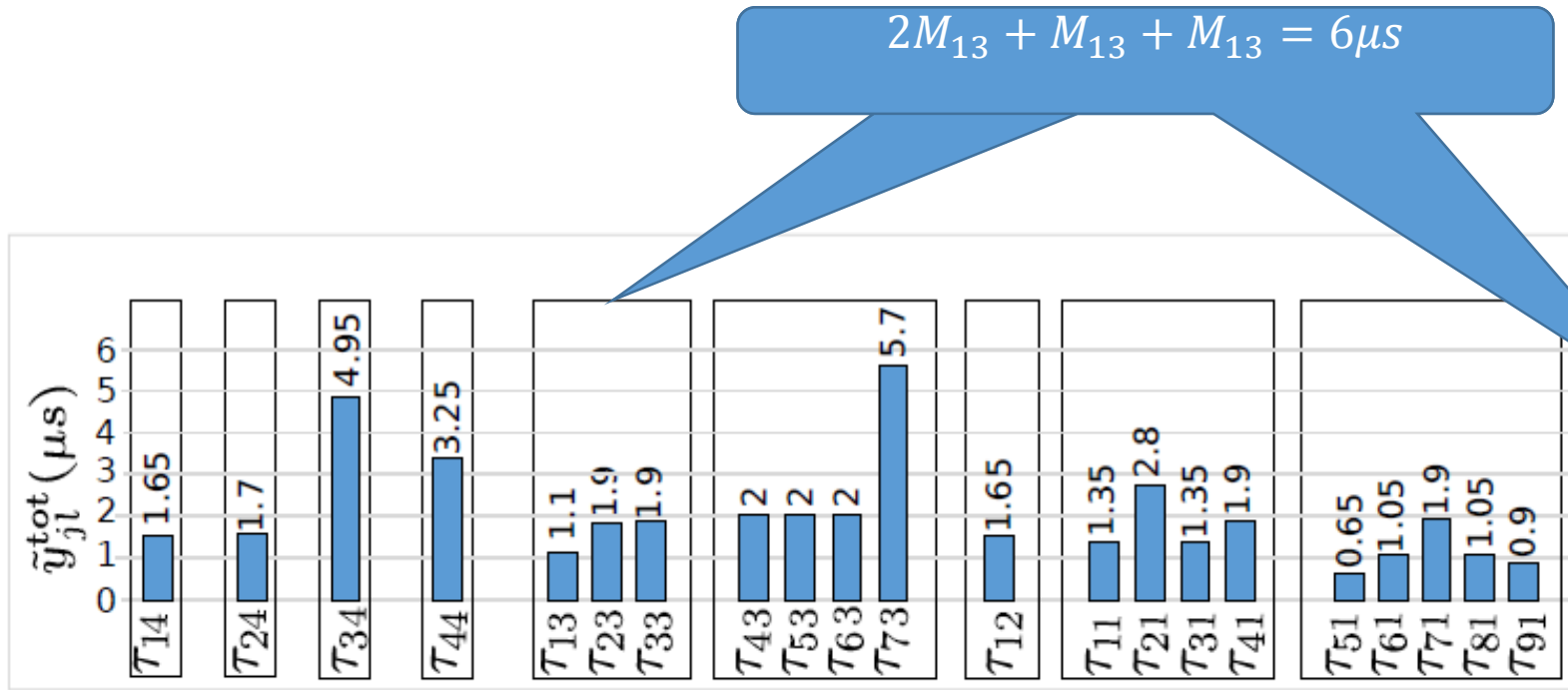
9 partitions

Mohamed Hassan

Derive memory requirements per class

Run optimization solver to obtain CARb parameters

Evaluation: Avionics case-study



memory access requirements
$M_{14} \leq 5.02\mu s$
$M_{24} \leq 8.11\mu s$
$M_{34} \leq 23.17\mu s$
$M_{44} \leq 45.96\mu s$
$2M_{13} + M_{23} + M_{33} \leq 6.45\mu s$
$4M_{43} + 4M_{53} + 2M_{63} + M_{73} \leq 35.28\mu s$
$M_{12} \leq 1.77\mu s$
$4M_{11} + M_{21} + 4M_{31} + M_{41} \leq 28.77$
$8M_{51} + 2M_{61} + M_{71} + 2M_{81} + 4M_{91} \leq 24.17$

Evaluation: Dynamic Re-arbitration

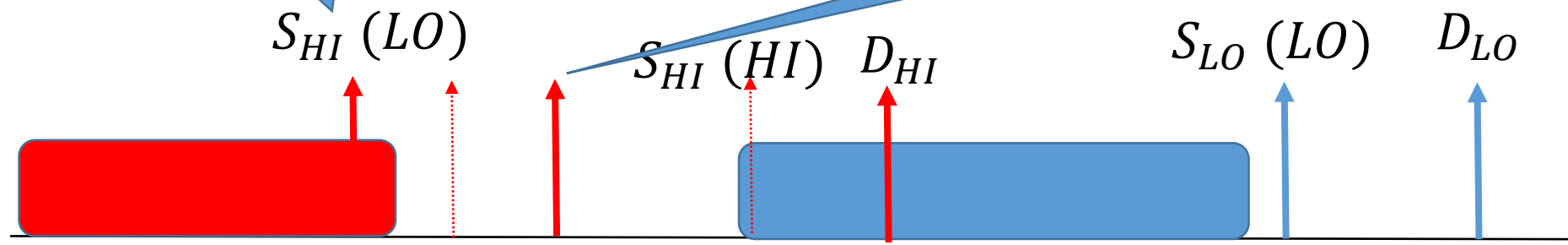
τ	D (ms)	S (ms)	Partition	Core	U	Λ	$M(\mu s)$	τw	CW	Z
τ_{13}	5	1	1	1	0.5	2000	$M_{13} \leq 0.22$	1	4	1
τ_{23}	5	1	2	1	0.5	2000	$M_{23} \leq 0.22$	1		
τ_{12}	5	2	3	2	1	1000	$2M_{12} + M_{22} \leq 1.28$	1	2	1
τ_{22}	10	3				1000		1		
τ_{11}	10	2	4	3	1	2000	$3M_{11} + 2M_{21} \leq 2.9$	1	2	1
τ_{21}	15	8				2000		1		

6 task
and 3
classes

Evaluation: Dynamic Re-arbitration

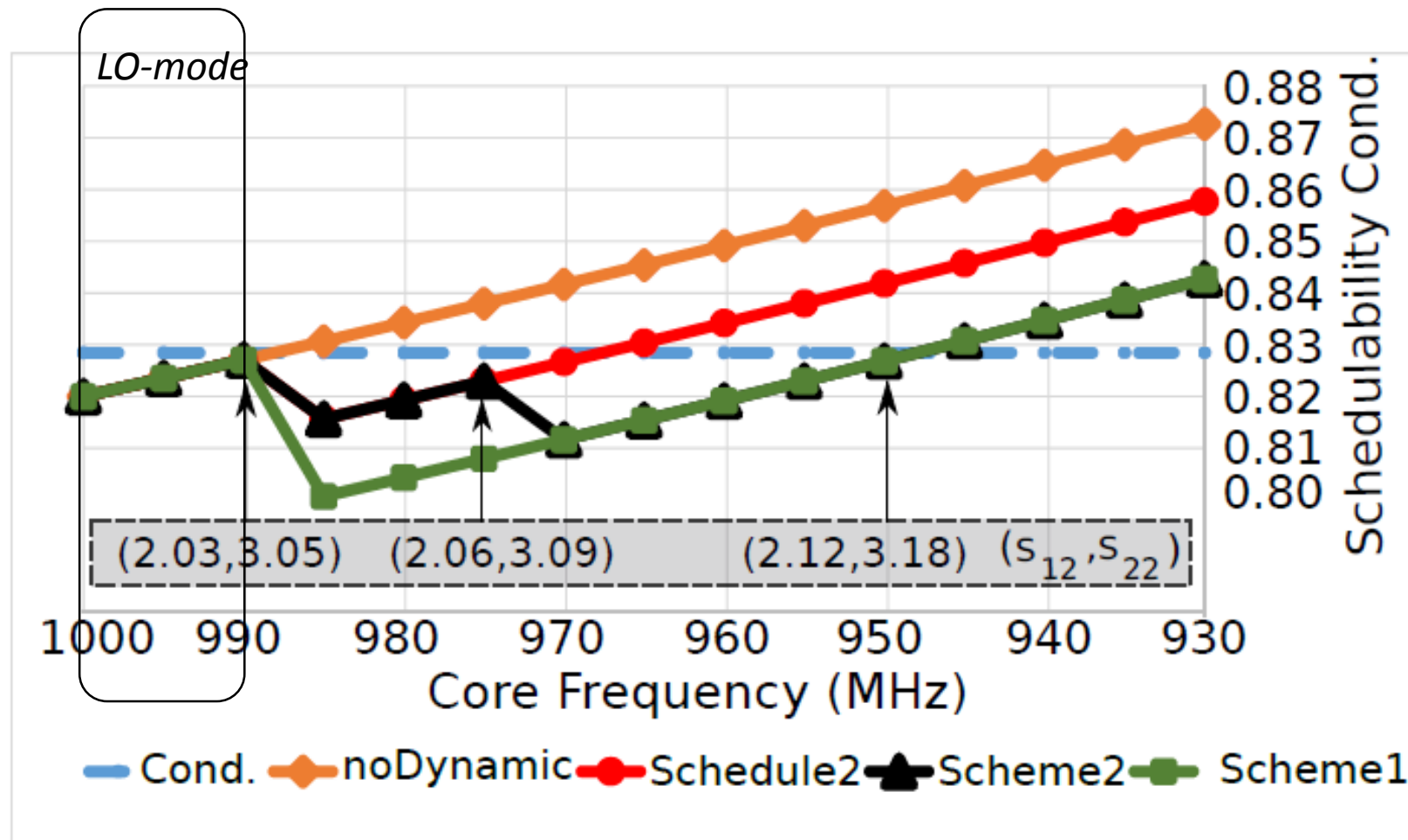
Lower WCET estimate

The maximum increase in execution time that PCArb can mitigate without mode switching



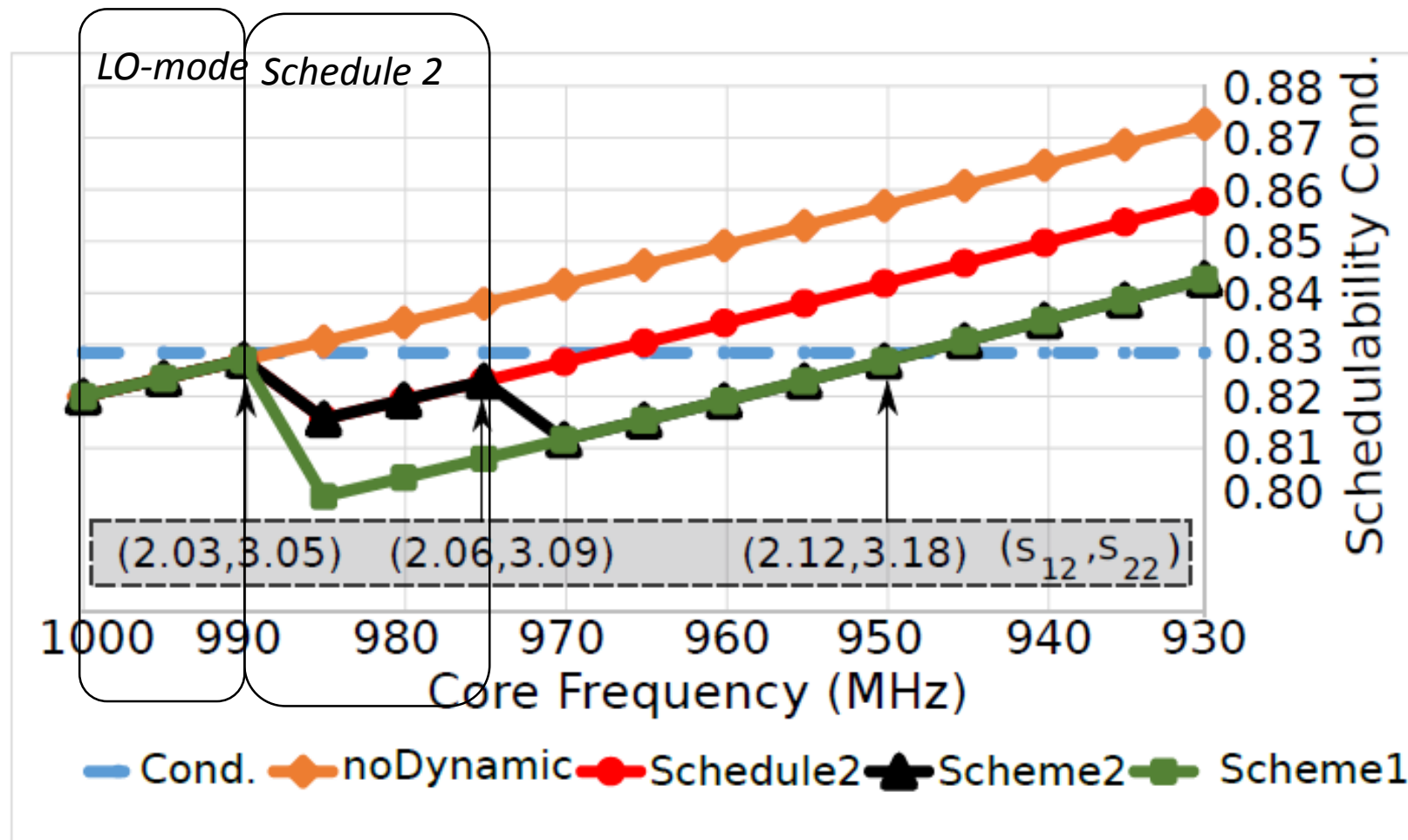
Evaluation: Dynamic Re-arbitration

$$f(WCCT, WC \text{ memory time}, \# \text{memory requests}) \leq f(\text{deadline})$$



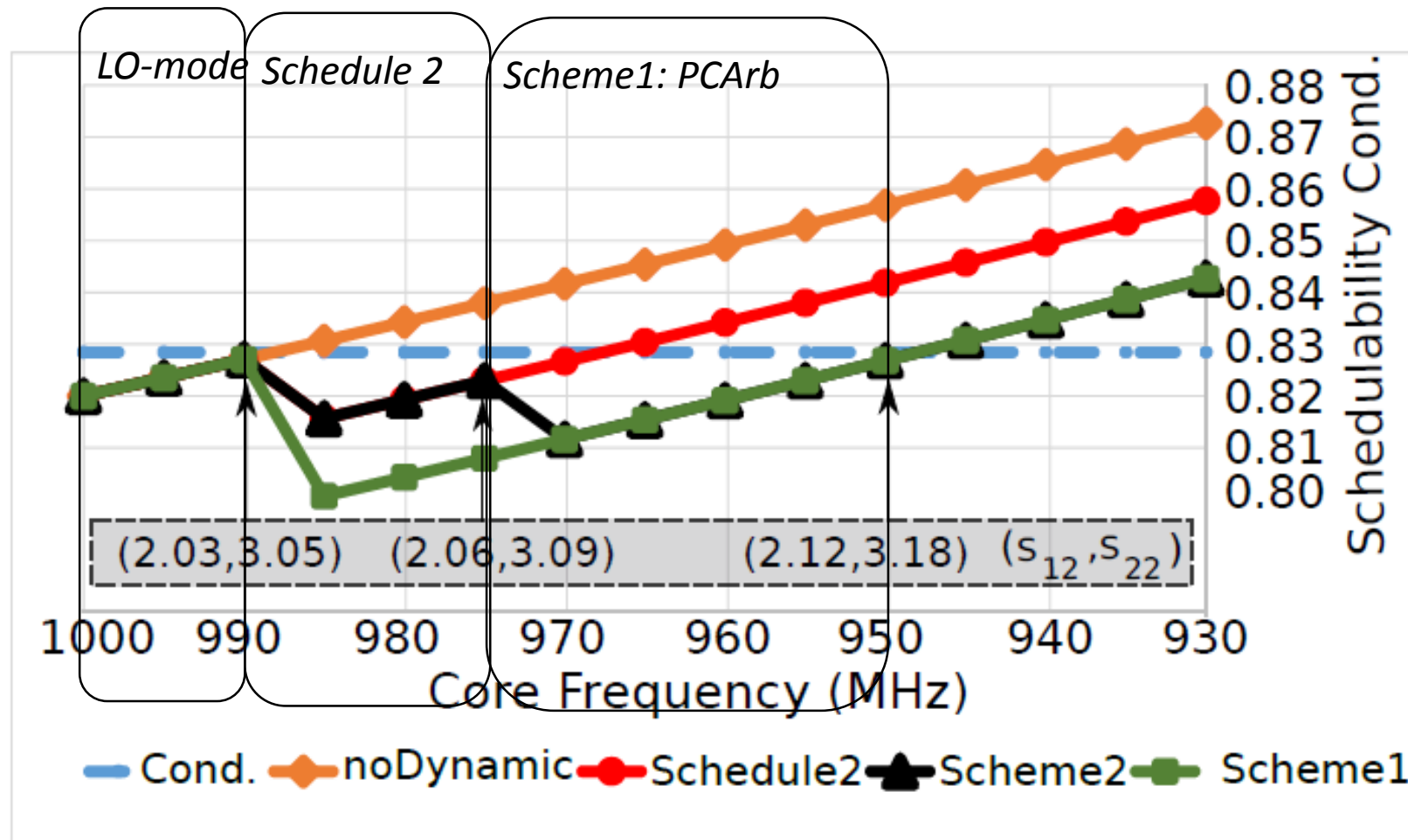
Evaluation: Dynamic Re-arbitration

$$f(WCCT, WC \text{ memory time}, \# \text{memory requests}) \leq f(\text{deadline})$$



Evaluation: Dynamic Re-arbitration

$$f(WCCT, WC \text{ memory time}, \# \text{memory requests}) \leq f(\text{deadline})$$



Summary



CArb: Criticality- and Requirement-aware Bus Arbiter for MCS

How to decompose WCET to derive memory latency requirements

Optimal harmonic arbitration to satisfy memory latency requirements

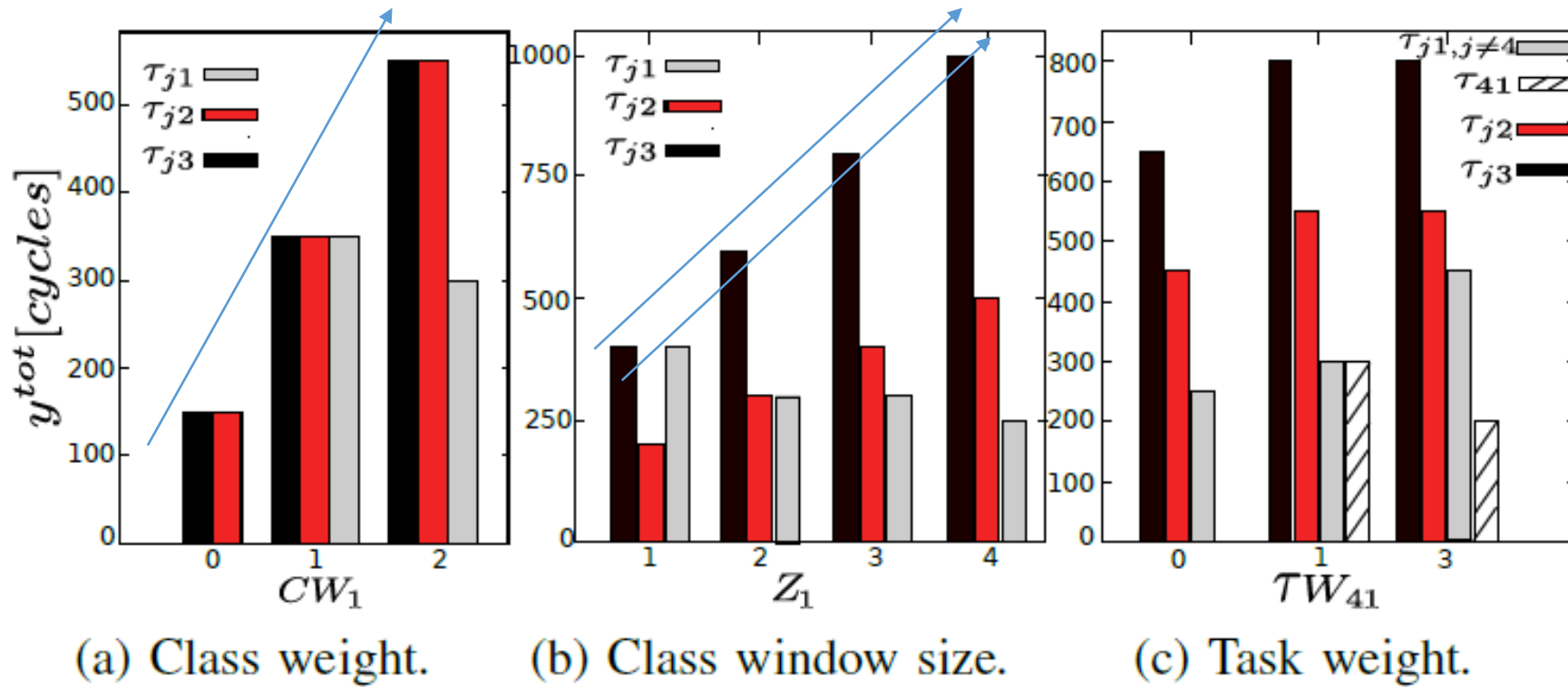
Separates interference amongst different criticalities

Benefits of a criticality-aware arbiter

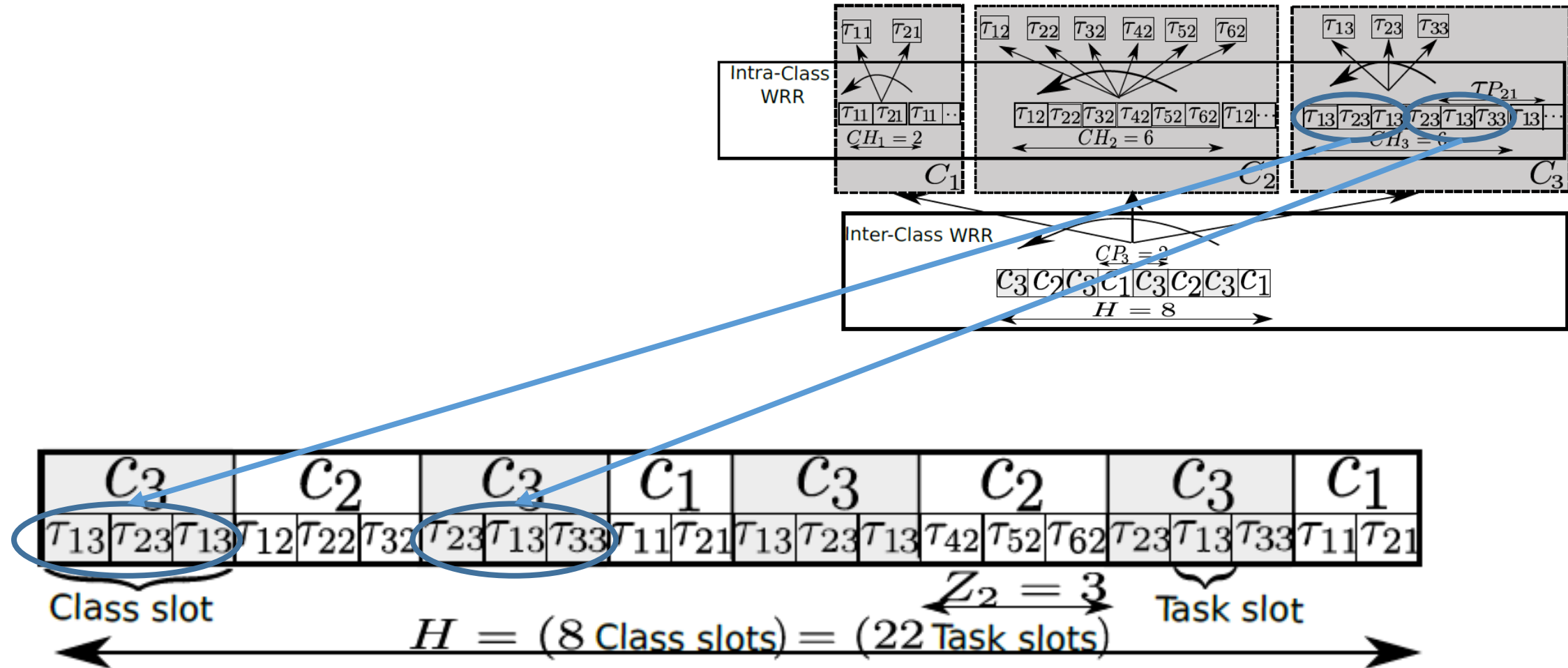
*mitigate mode switching

* prevent unnecessary suspension of tasks

Evaluation: Synthetic Experimentation

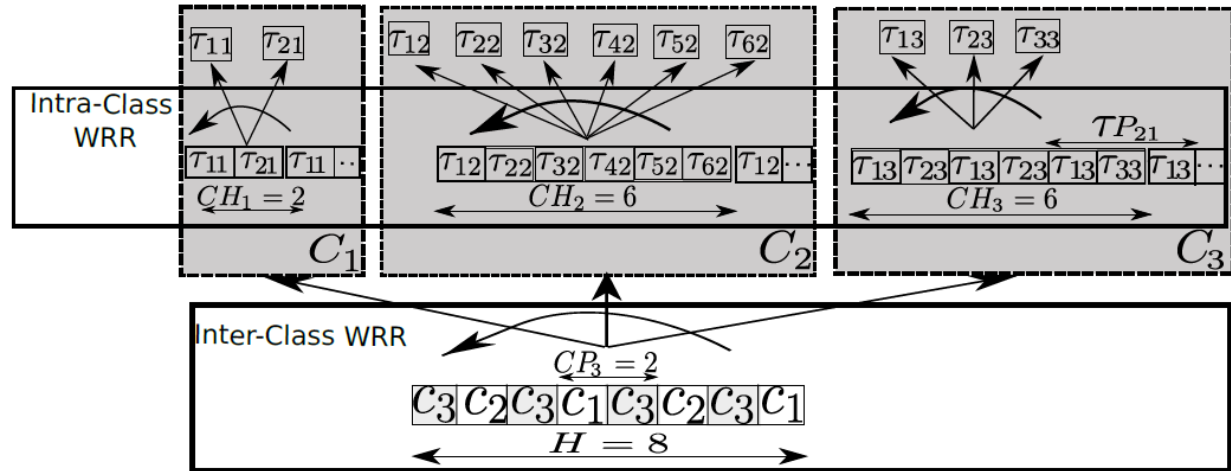


Final CArb Schedule



Area Overhead

- 5 classes and 100 task/class requires only 440B.



	CW	Z	→		τ_{13}	τ_{23}	τ_{33}	CH_3			
C_3	4	3		TW	3	2	1	6			
C_2	2	3	→		τ_{12}	τ_{22}	τ_{32}	τ_{42}	τ_{52}	τ_{62}	CH_2
C_1	2	2		TW	1	1	1	1	1	1	6
C_1	2	2	→		τ_{11}	τ_{21}	CH_1				
H	8			TW	1	1	2				