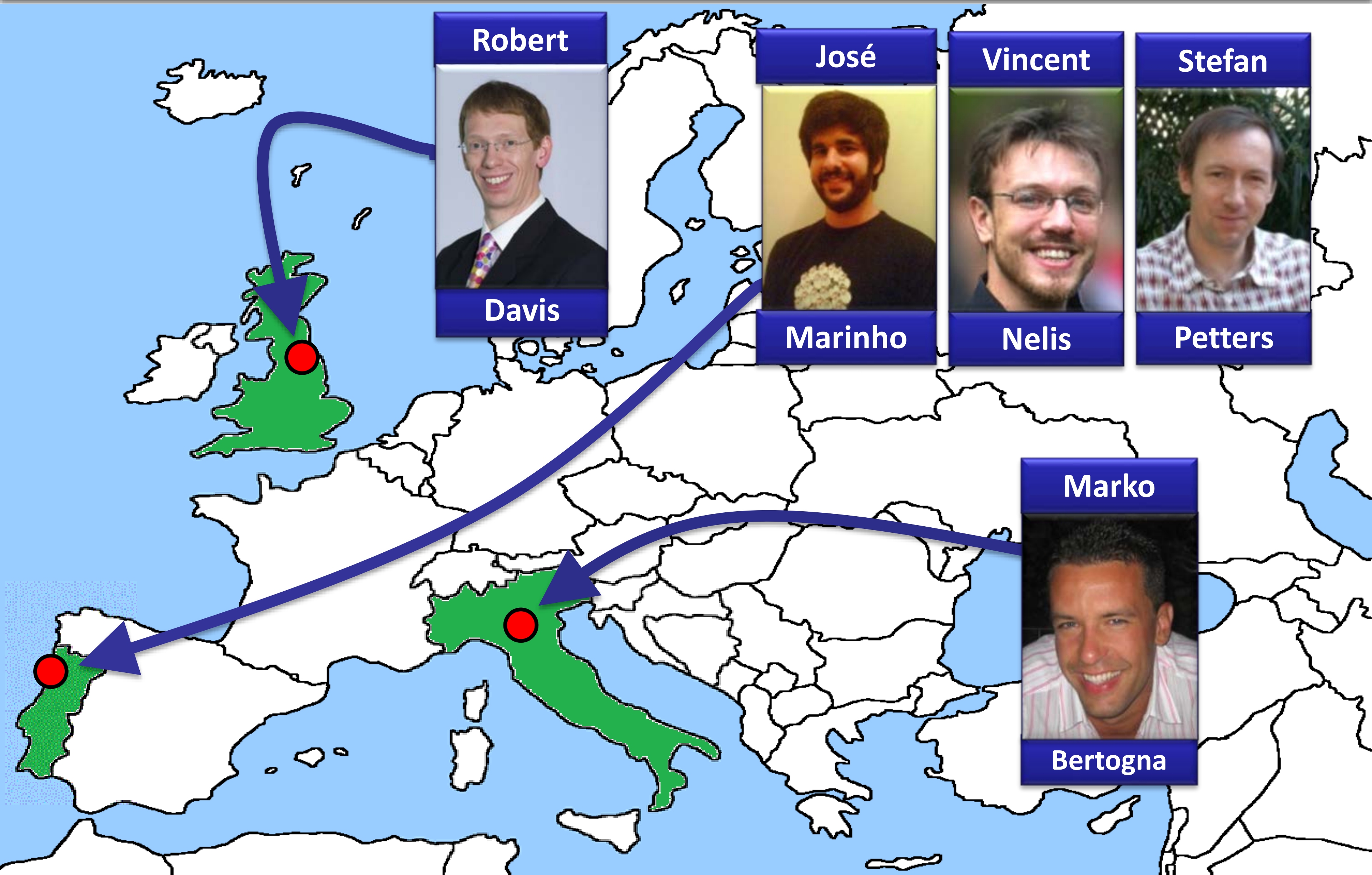
# Limited Pre-emptive Global Fixed Task Priority

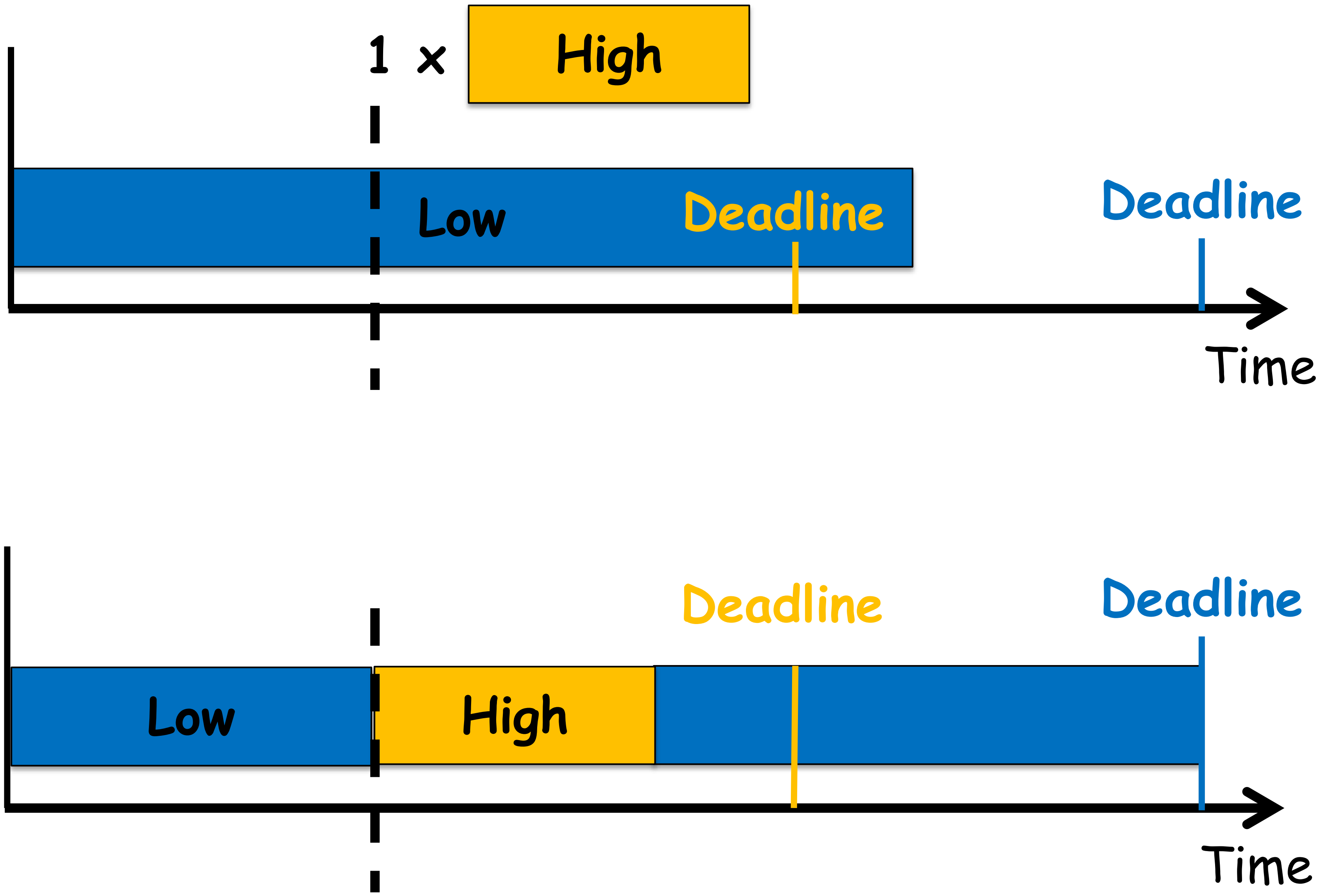


# In a nutshell…

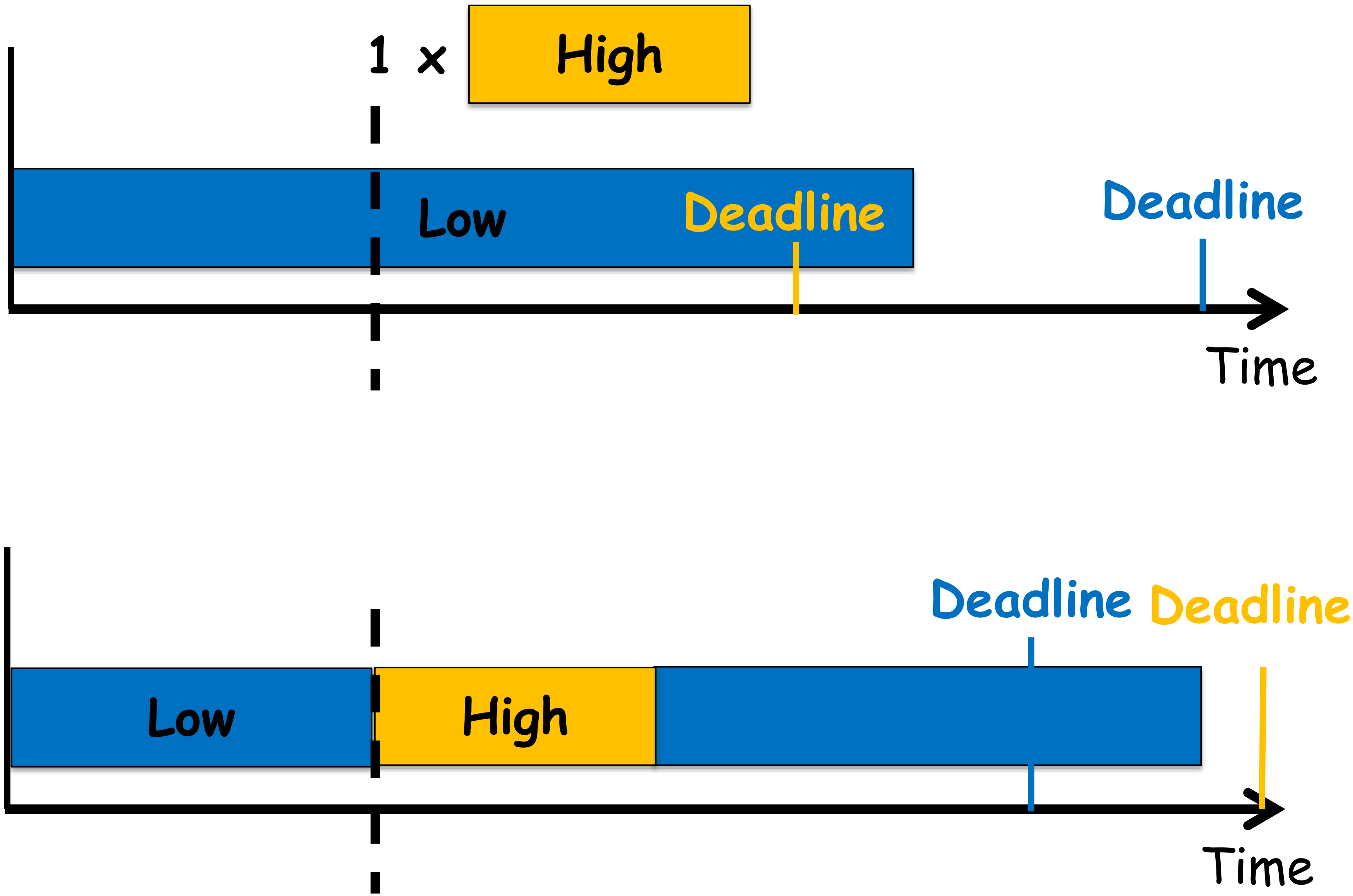
We studied the preemptive scheduling problem on multicores

We focused on the **limited preemptive model**, which is somewhat between fully preemptive and non preemptive approaches

# Preemption or no preemption?



# Preemption or no preemption?



# In a nutshell…

We studied the preemptive scheduling problem on multicores

We focused on the **limited preemptive model**, which is somewhat between fully preemptive and non preemptive approaches

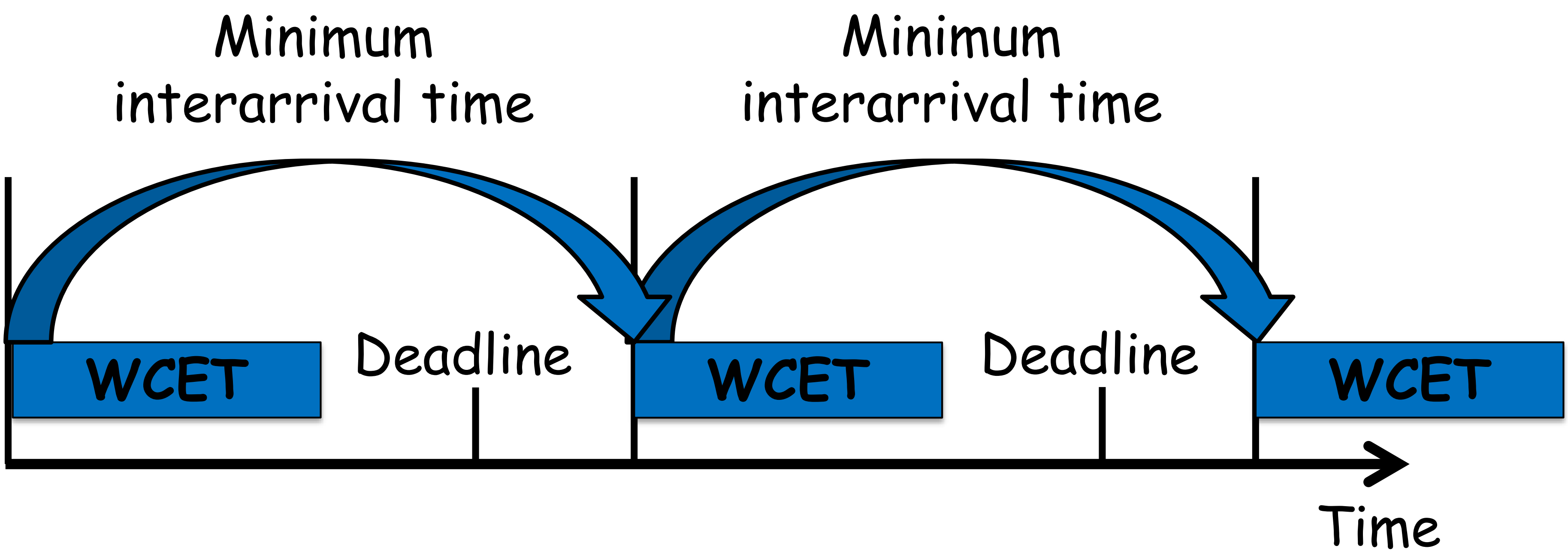
We came up with **two different scheduling techniques** and derived **schedulability analyses** for each of them

We played around with the parameters to further **reduce the number of preemption and increase the schedulability**

# System model & assumptions

## Application model:

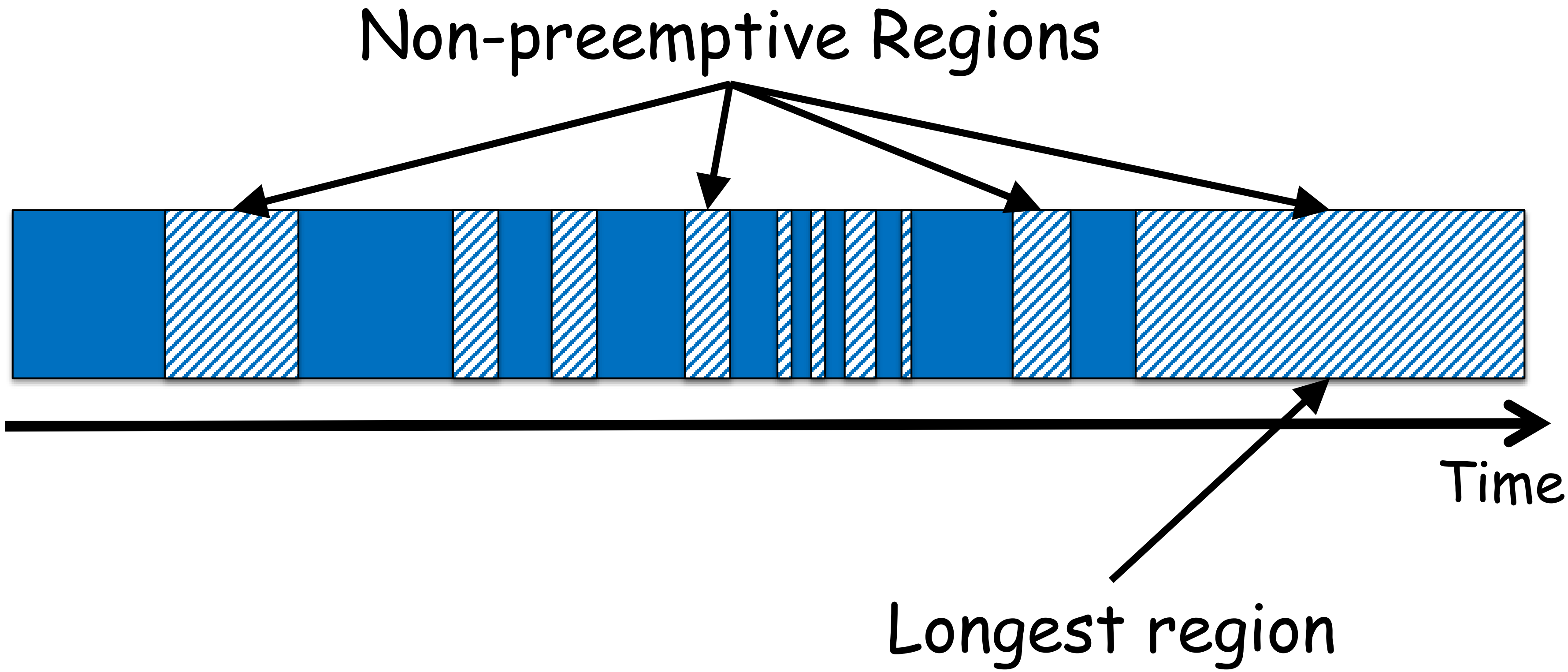
Sporadic constrained-deadline tasks



# System model & assumptions

## Scheduler model:

Global Fixed Task Priority Scheduling
Fixed Non-Preemptive Region

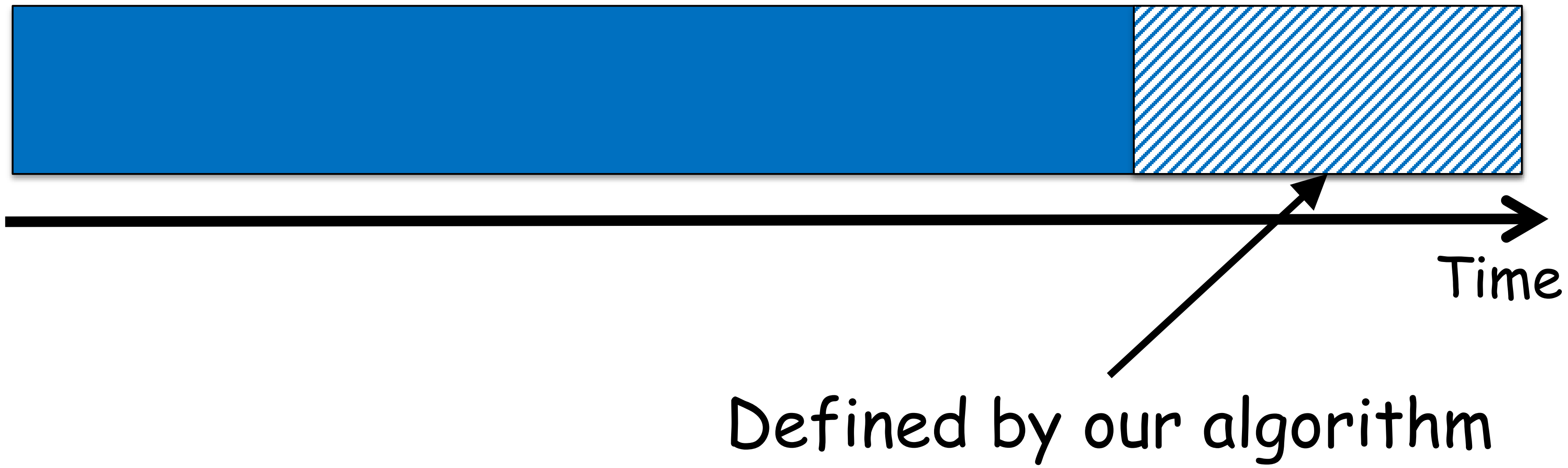


# System model & assumptions

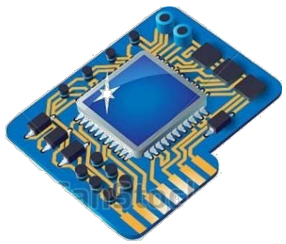
## Scheduler model:

Global Fixed Task Priority Scheduling

Fixed Non-Preemptive Region



# System model & assumptions



## Platform model:

Identical multiprocessor platform

m processors

## Assumptions:

No migration cost,

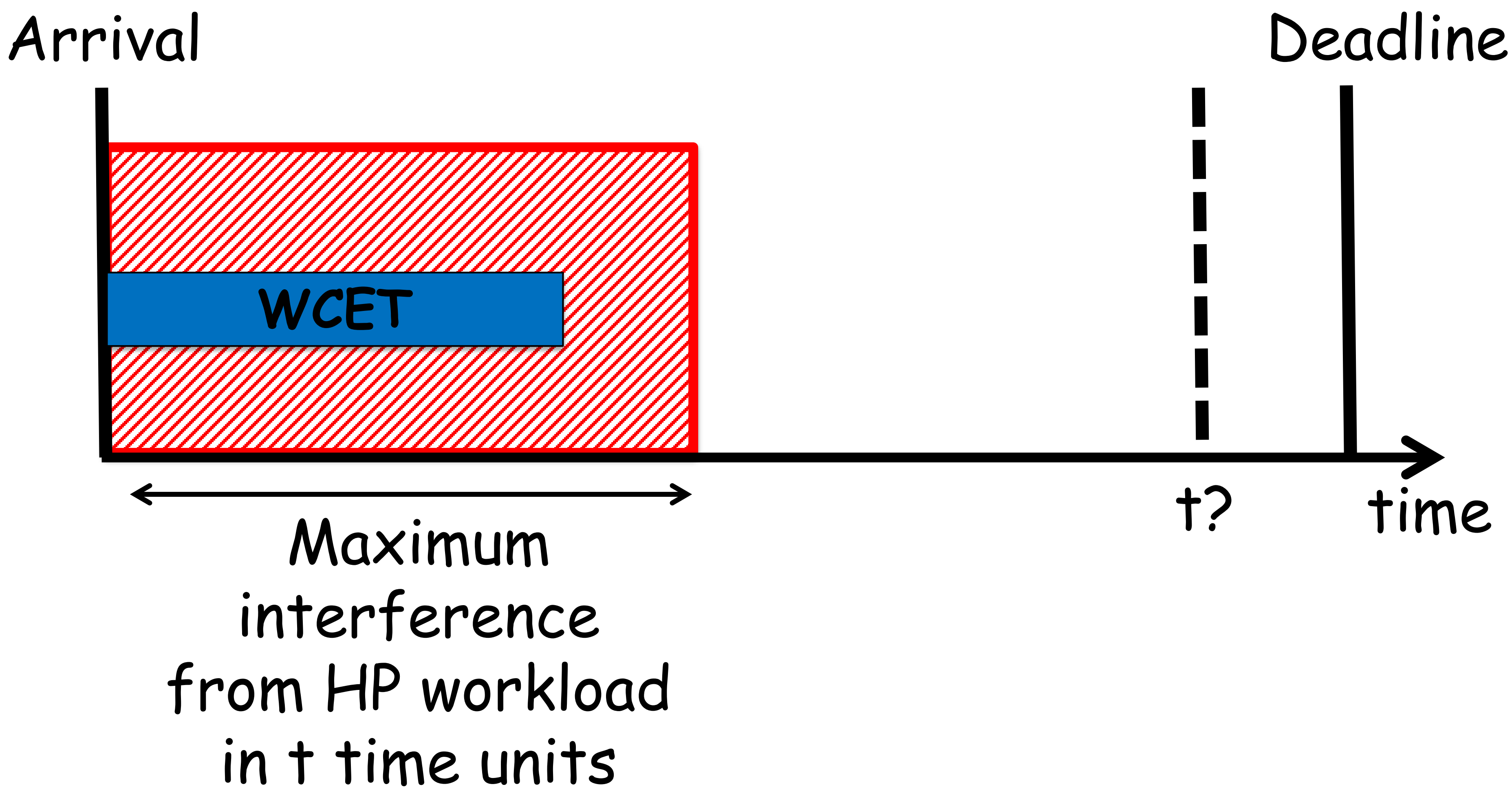
No preemption cost

Independant tasks

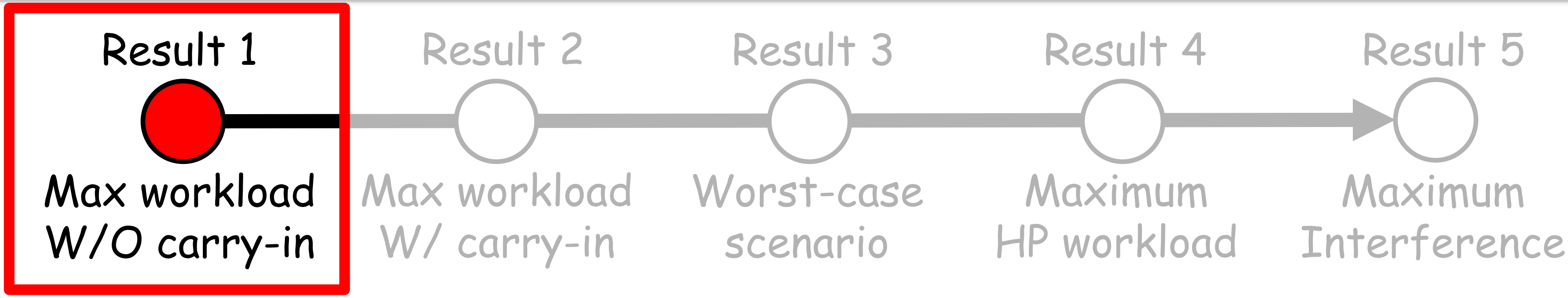
Focus on the two new scheduling techniques that we propose and the impact on the schedulability

A bit of history...

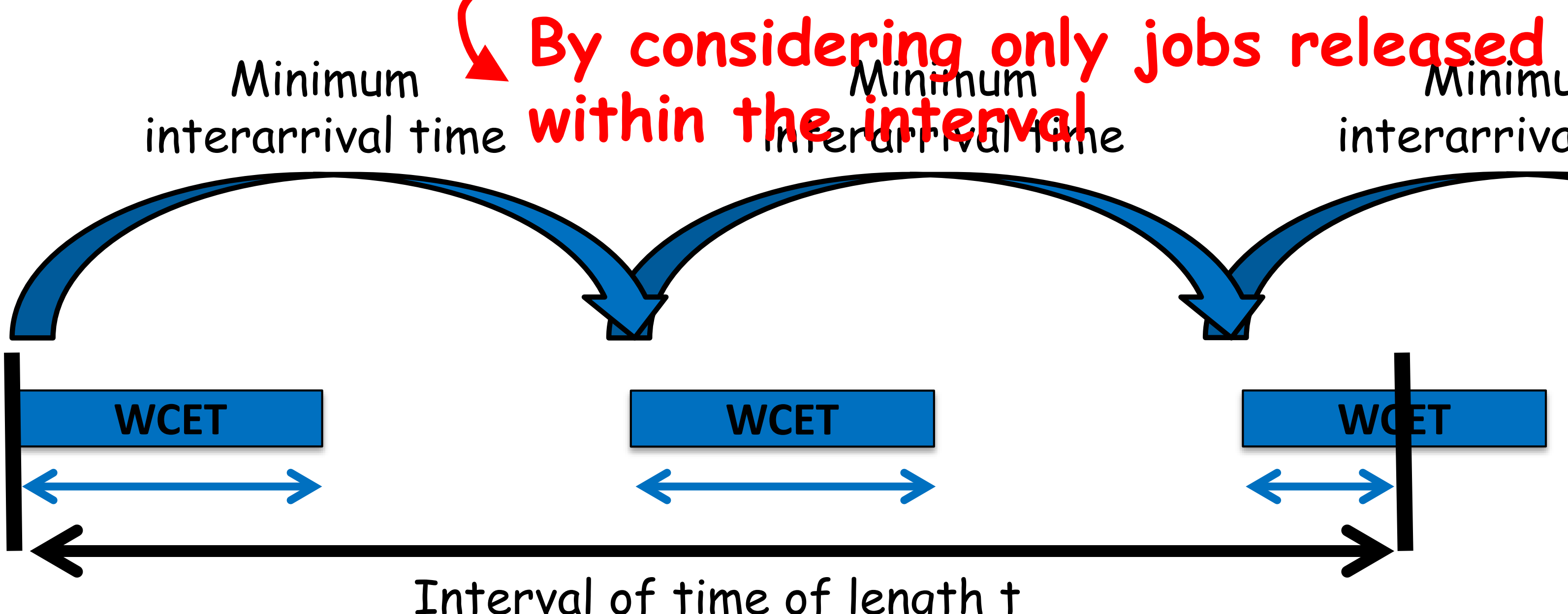
# Background on fully-preemptive schedulability analysis



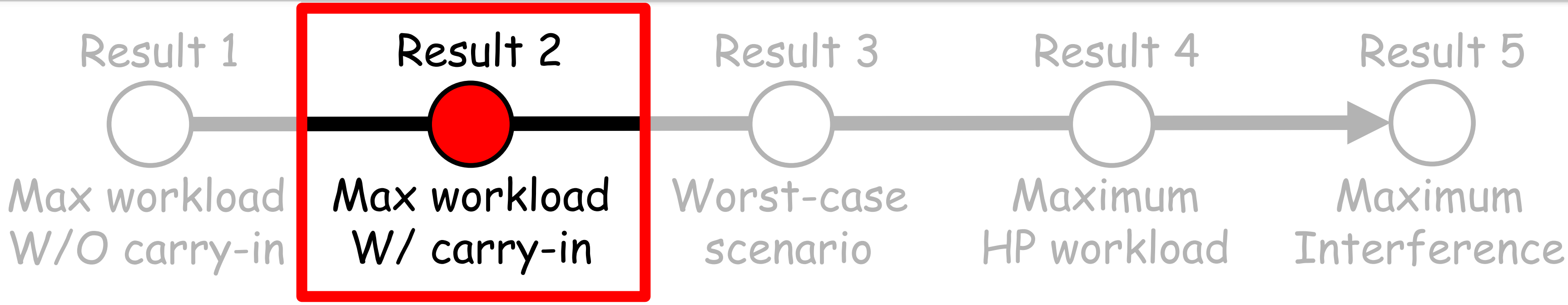
# Background on fully-preemptive schedulability analysis



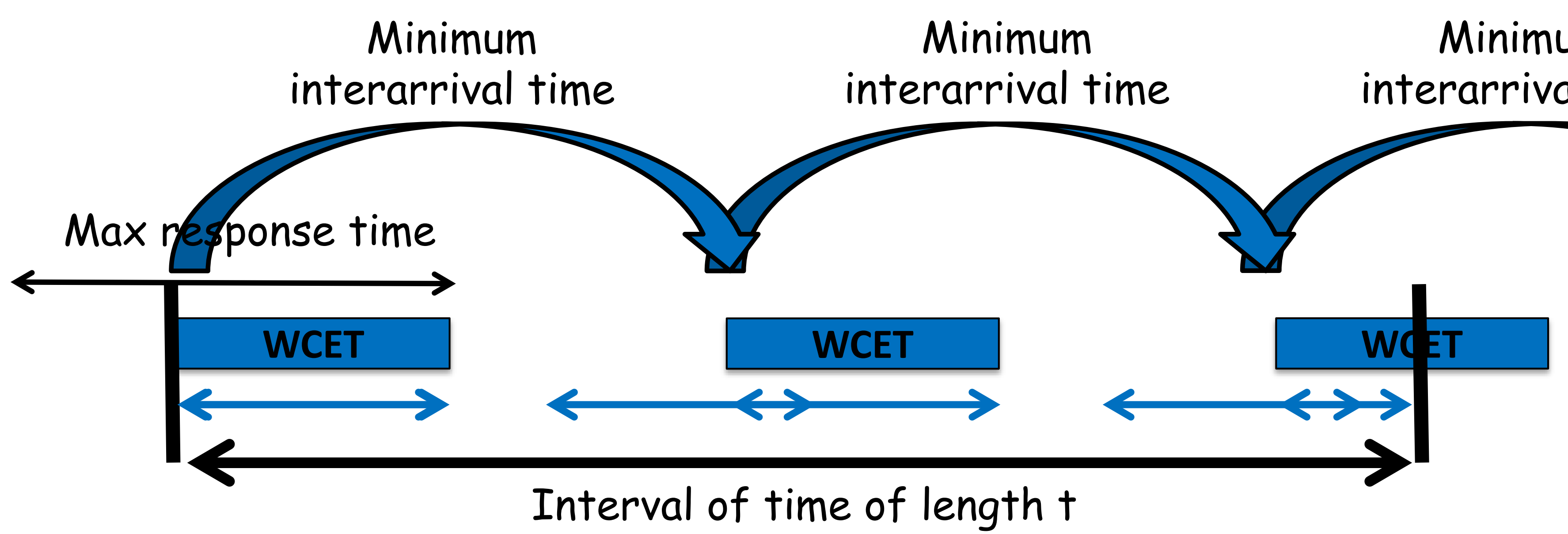

Result 1: Maximum workload **without** carry-in job



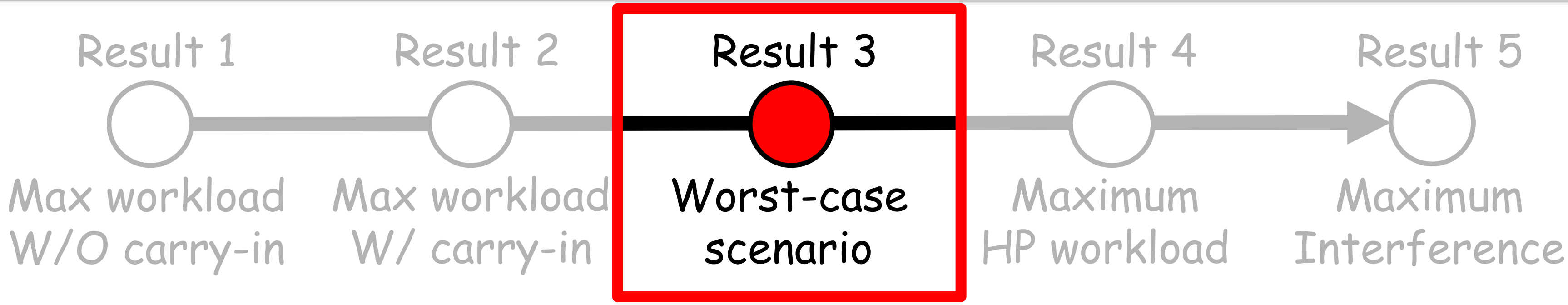
# Background on fully-preemptive schedulability analysis



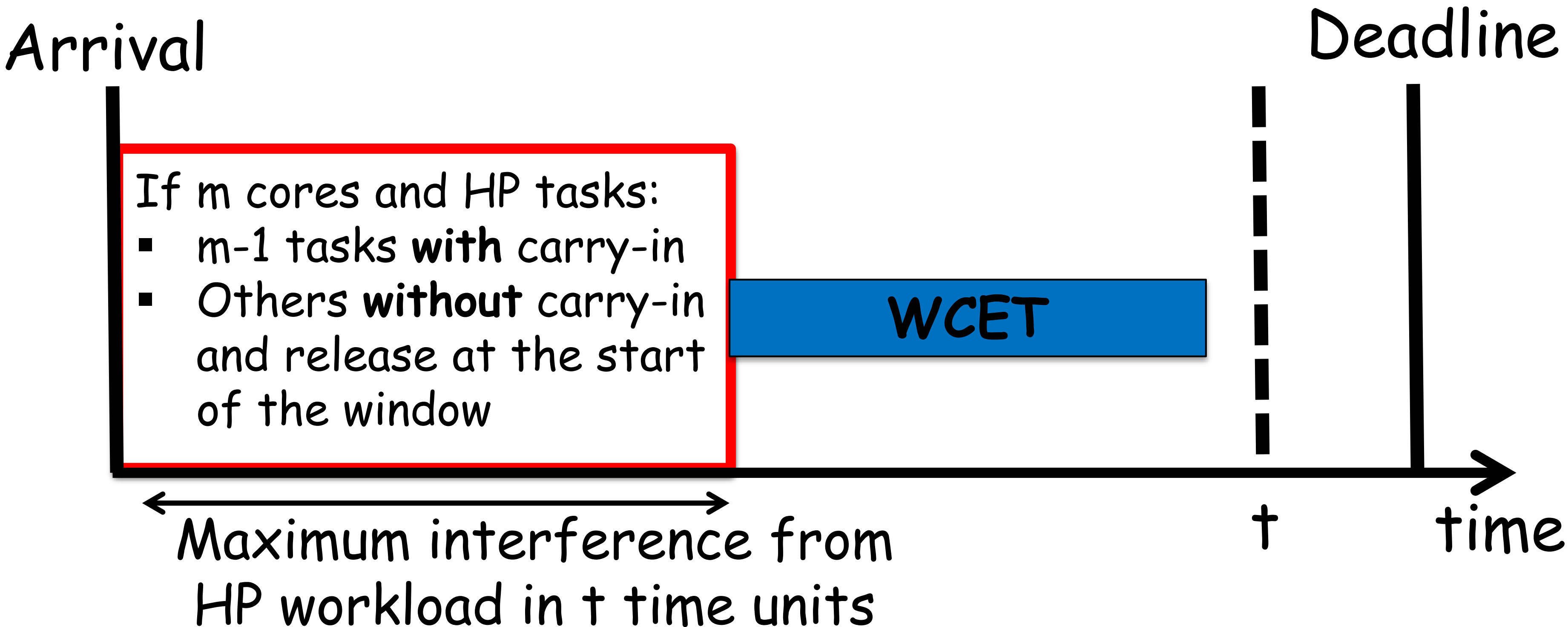

Result 2: Maximum workload **with** carry-in



# Background on fully-preemptive schedulability analysis




## Result 3: Worst-case interference scenario



# Background on fully-preemptive schedulability analysis


## Result 4: Maximum Higher Priority Workload

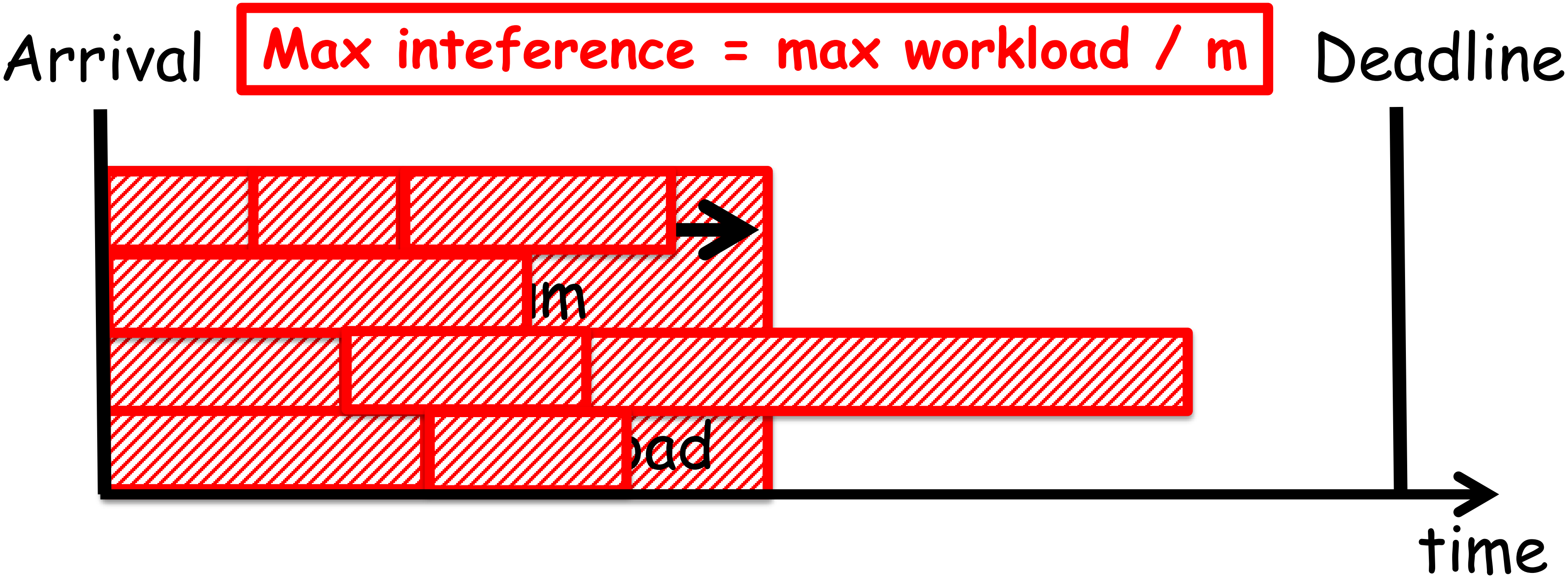
1. Worst-case interference scenario
2. Maximum task workload without carry-in job
3. Maximum task workload with carry-in job

⇒ Maximum workload that can be generated within the given time window

# Background on fully-preemptive schedulability analysis

Result 1 — Max workload W/O carry-in
Result 2 — Max workload W/ carry-in
Result 3 — Worst-case scenario
Result 4 — Maximum HP workload
**Result 5 — Maximum Interference**

## Result 4: Maximum interference



# Preemptive techniques on multicores

**Definition of fully preemptive scheduling on multicores**

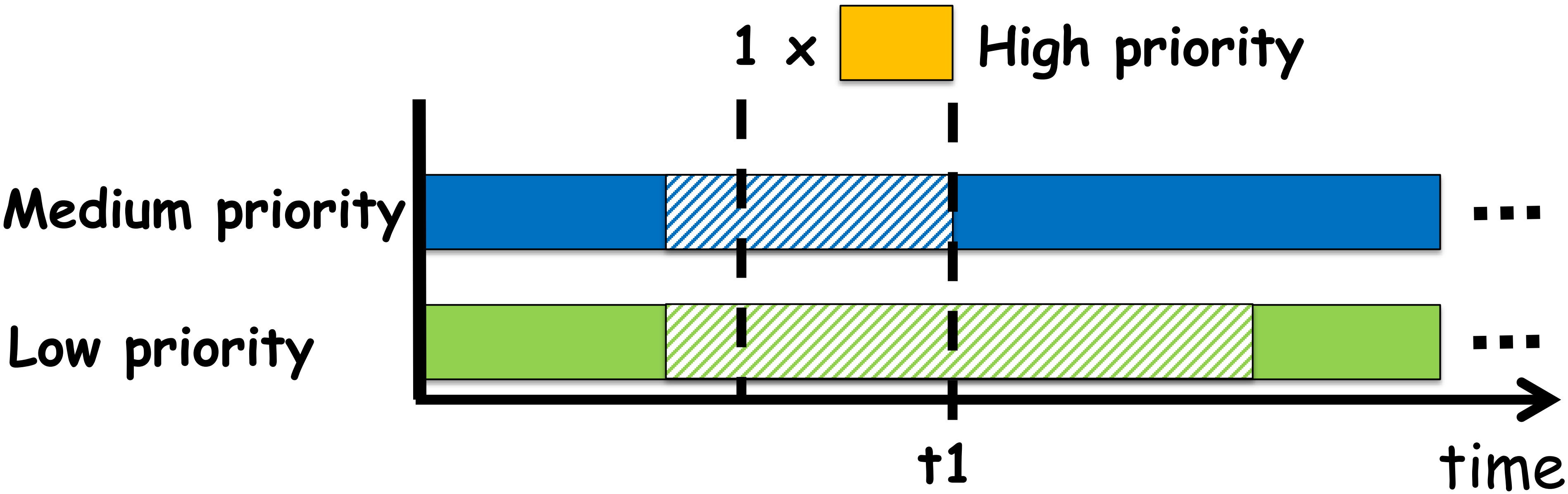
"~~policy where at any time~~, the m highest priority tasks execute on the m processors of the platform."

**In limited preemptive scheduling**

# Preemptive techniques on multicores

**Definition of fully preemptive scheduling on multicores**

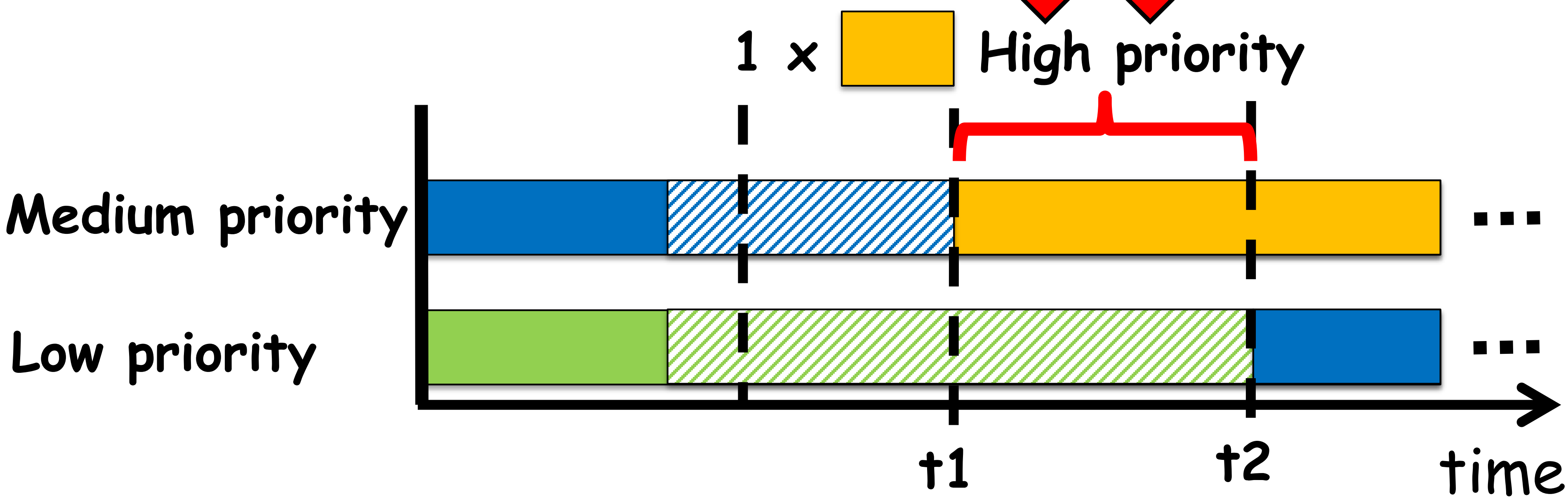
"~~policy where at any time~~, the m highest priority tasks execute on the m processors of the platform."



# Preemptive techniques on multicores

**Definition of fully preemptive scheduling on multicores**

"~~policy where at any time~~, the m hi[illegible] priority tasks execute on the m processors of th[illegible]orm."

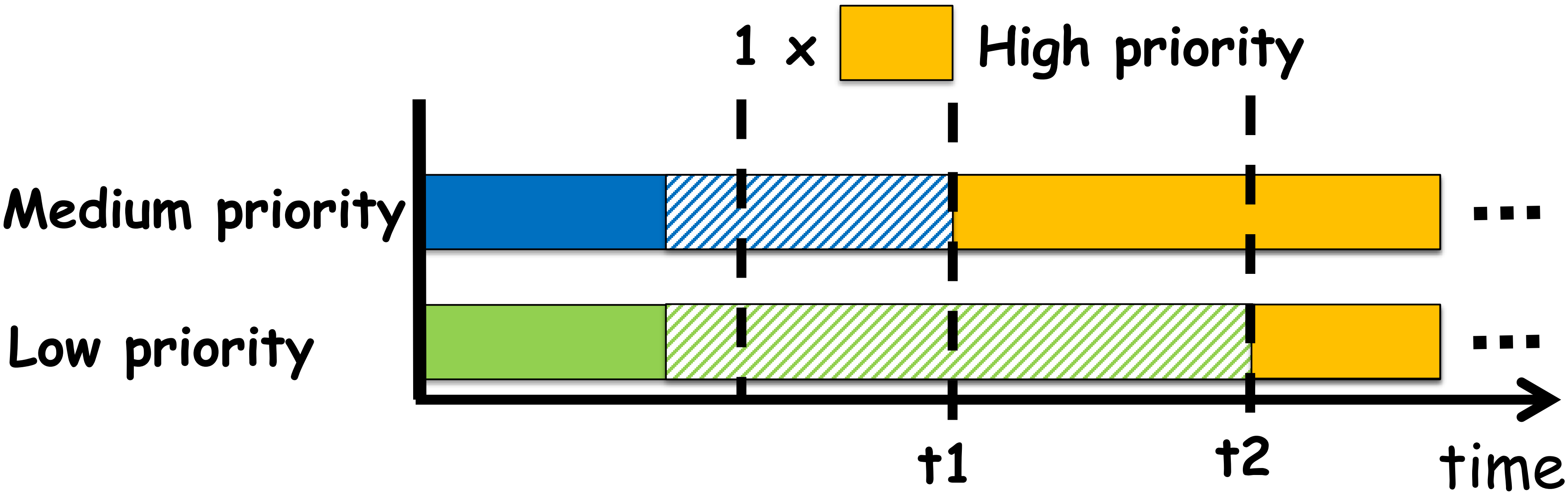



**Regular Deferred Scheduling (RDS)**: New arriving higher priority workload preempts the first preemptible lower priority workload.

# Preemptive techniques on multicores

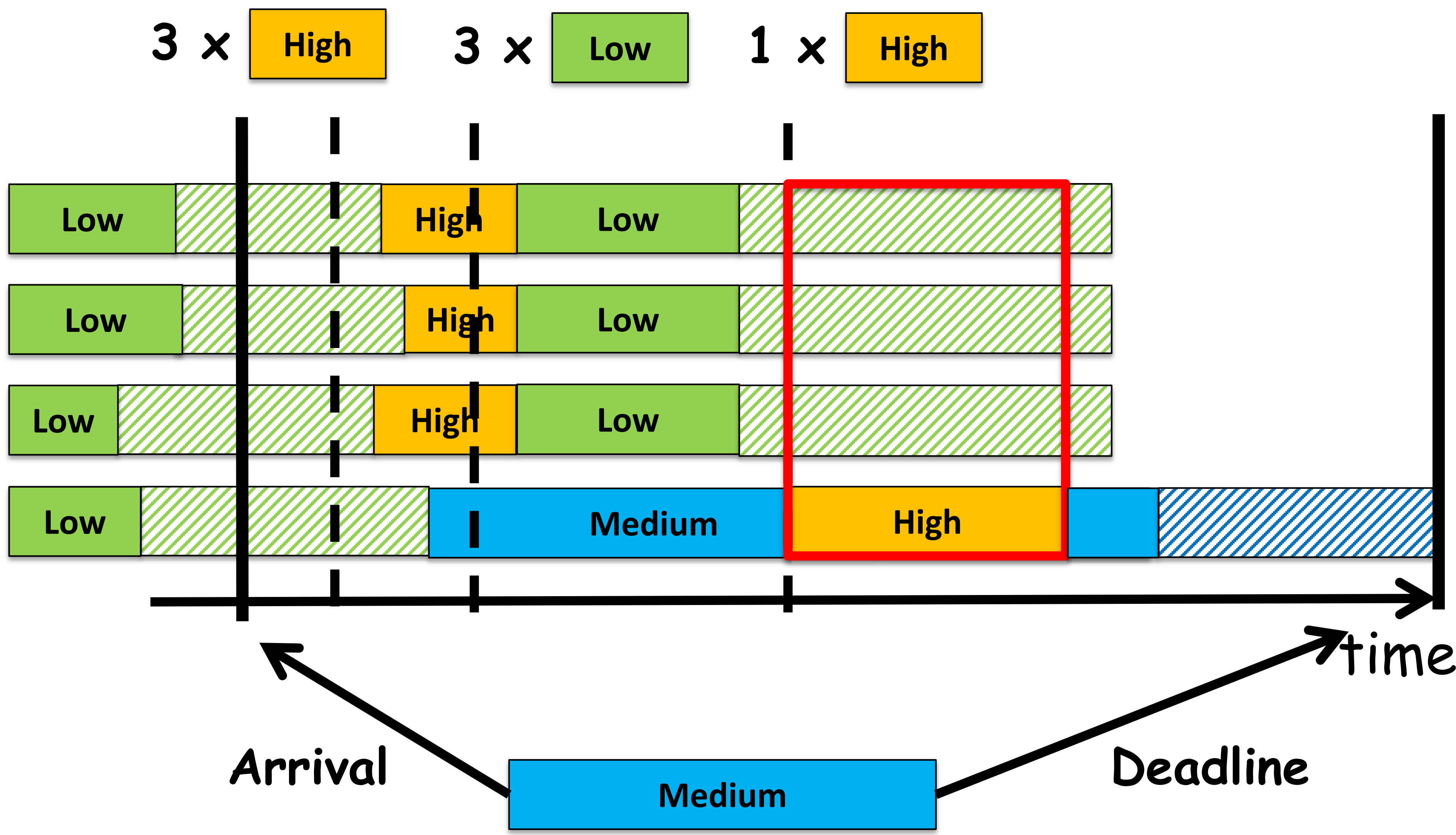
**Definition of fully preemptive scheduling on multicores**

"~~policy where at any time~~, the m highest priority tasks execute on the m processors of the platform."




**Adapted Deferred Scheduling (ADS)**: New arriving higher priority workload preempts the lowest priority job once it gets preemptible.

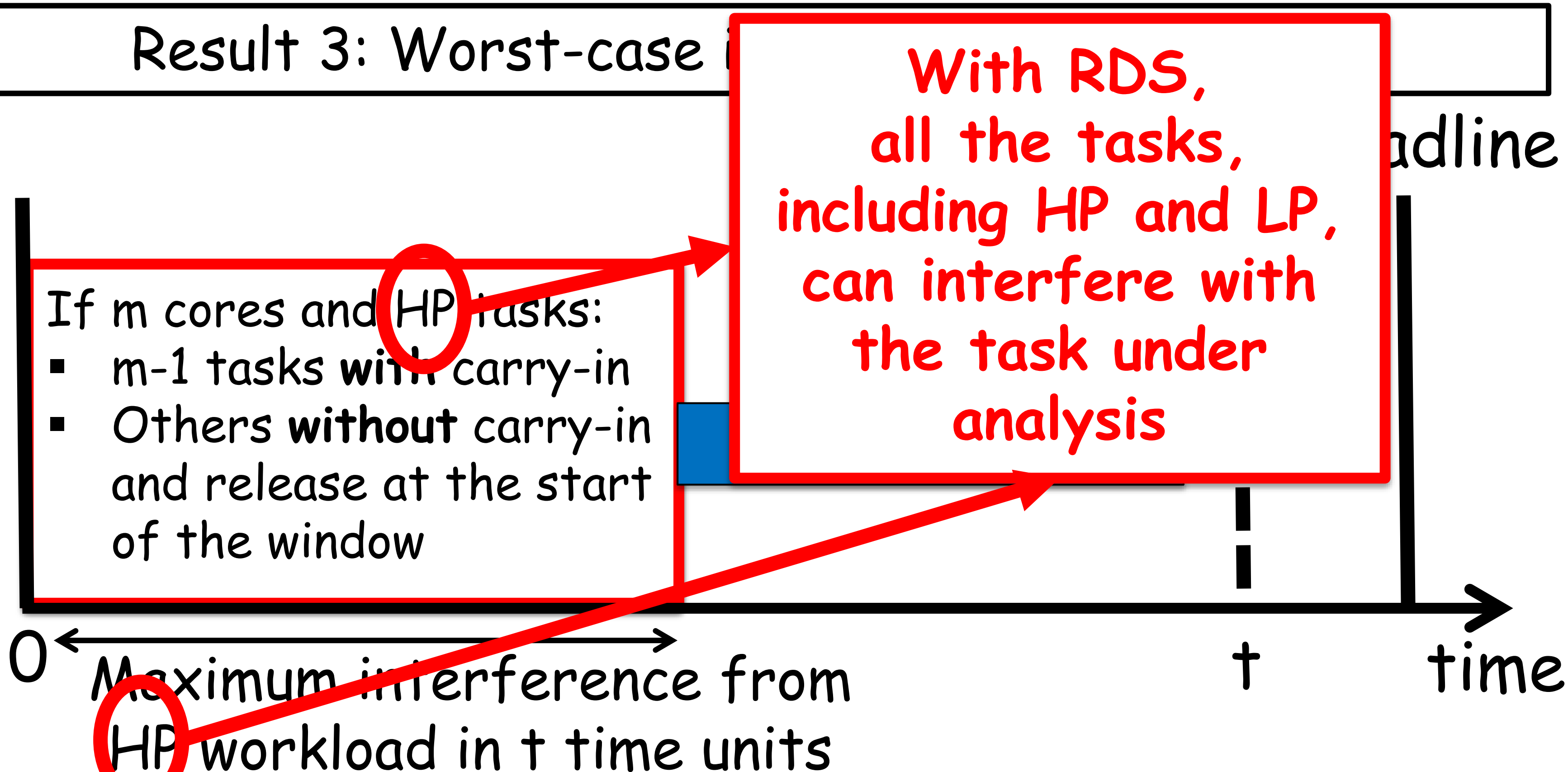
# What's wrong with RDS?



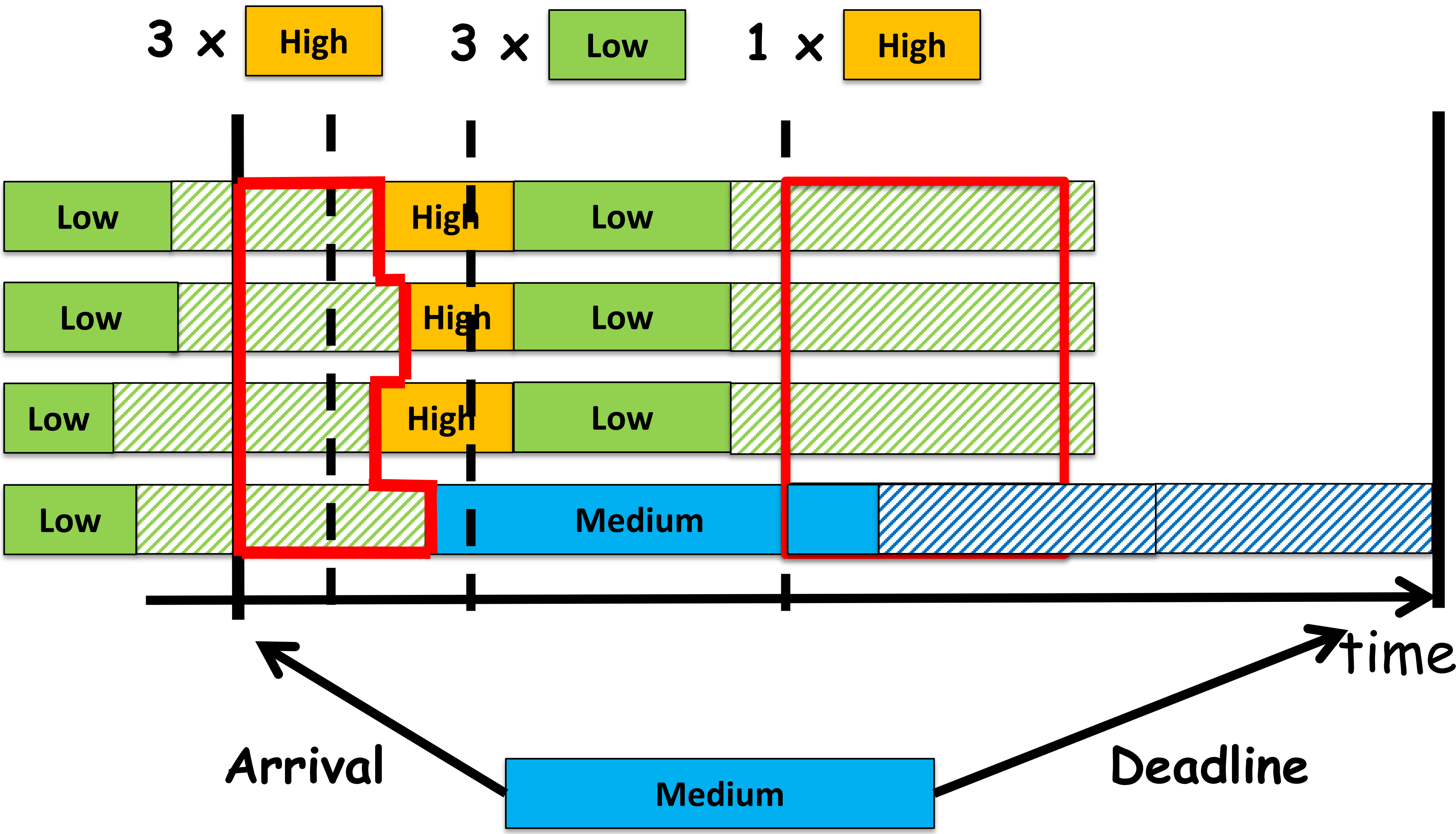
# What's wrong with RDS

| Result 1 | Result 2 | Result 3 | Result 4 | Result 5 |
|---|---|---|---|---|
| Max workload W/O carry-in | Max workload W/ carry-in | **Worst-case scenario** | Maximum HP workload | Maximum Interference |

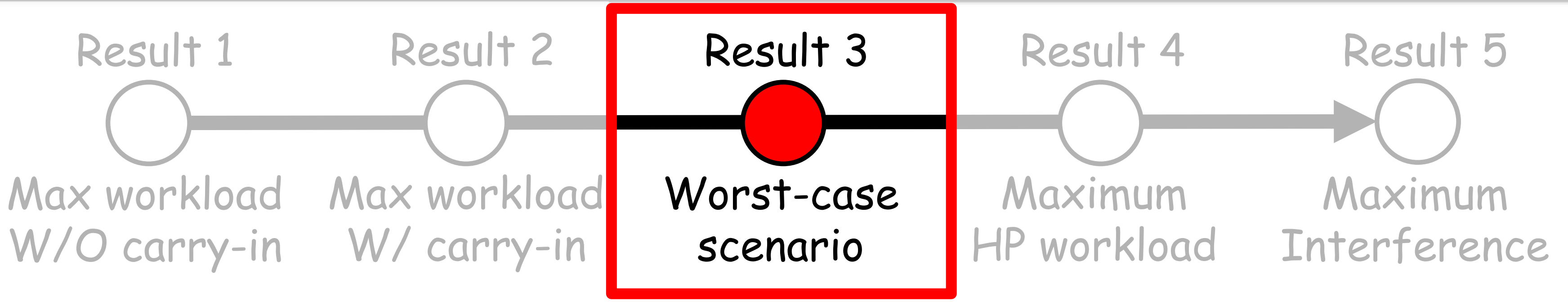
Result 3: Worst-case i



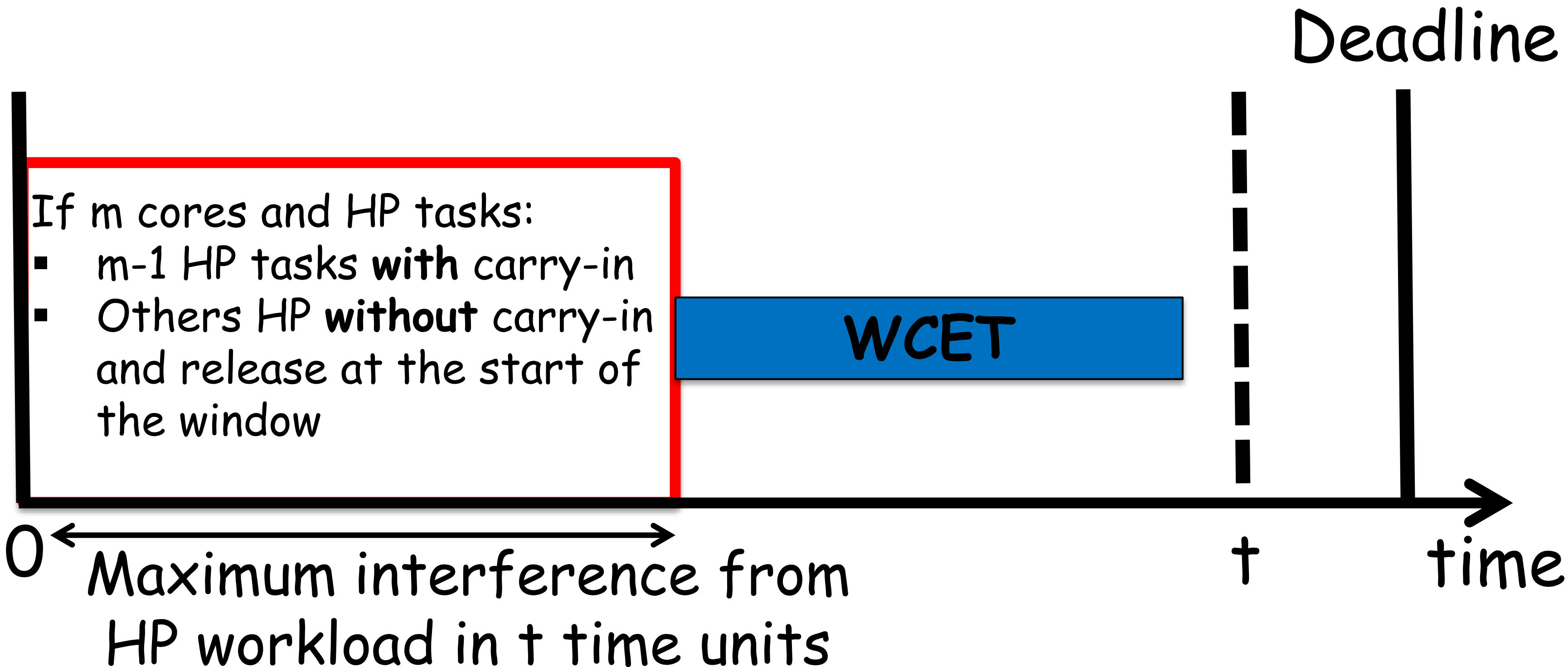
# And what about ADS?



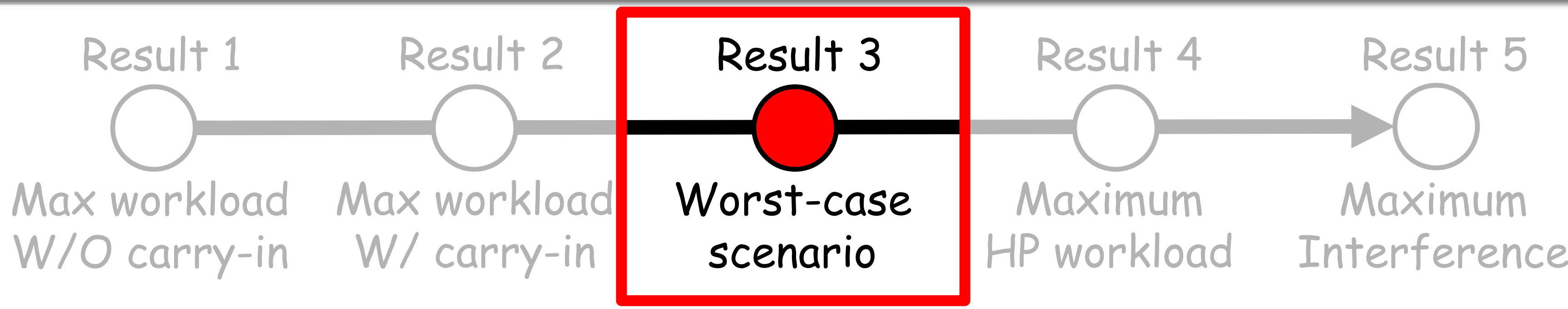
# Let's focus on ADS




**Result 3: Worst-case interference scenario**



# Let's focus on ADS

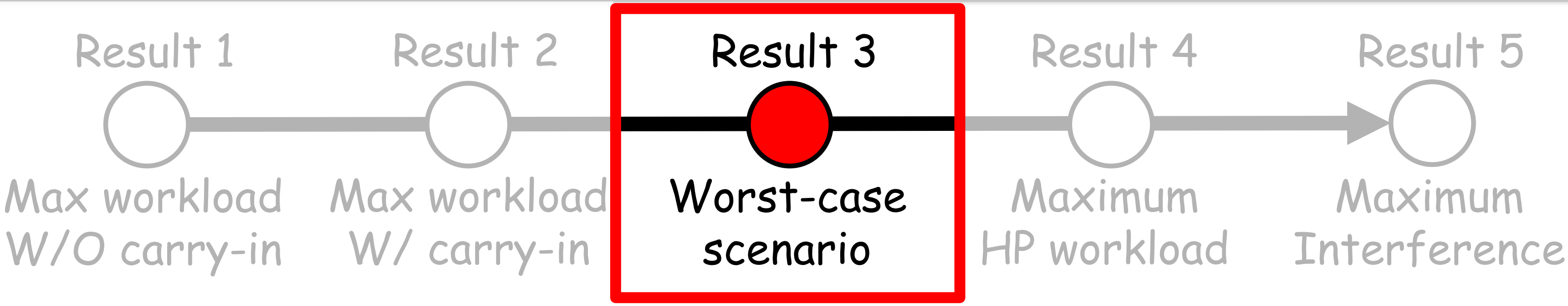



**Result 3: Worst-case interference scenario**

If m cores and HP tasks:

- m-1 HP tasks **with** carry-in

# Let's focus on ADS




**Result 3: Worst-case interference scenario**

If m core
- **m-1** H
- 1 LP c

If m cor
- **m-2**
- 2 LP

If m co
- **m-3**
- 3 LP

If m cores:
- **0** HP tasks **with** carry-in
- **m LP carry-in jobs**

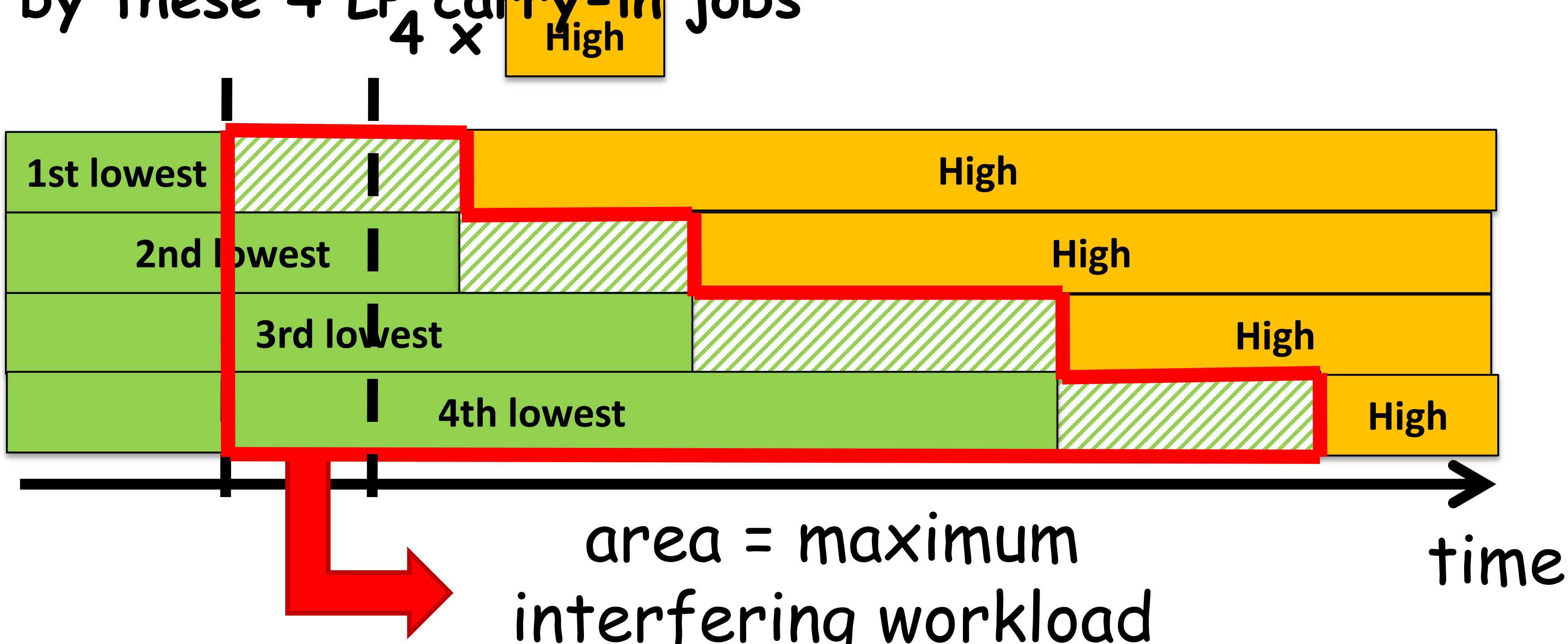
And so on...

# m - 4 HP carry-in tasks
# 4 LP carry-in jobs

Assume 4 carry-in jobs from LP tasks

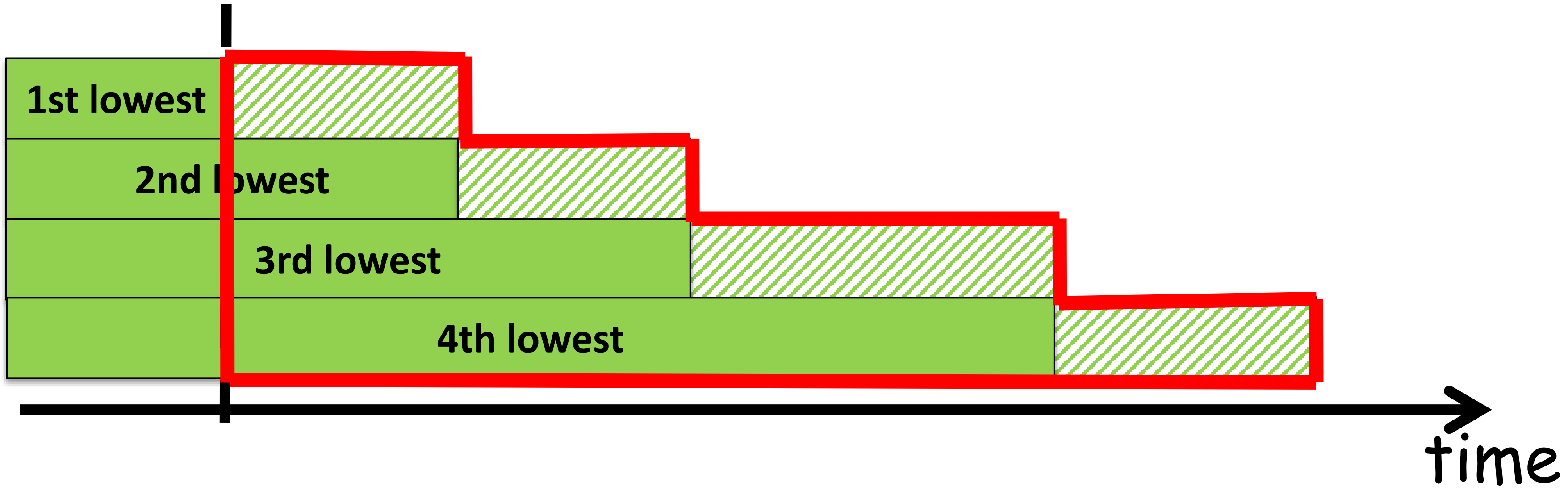
The worst-case “carry-in pattern”:

We must compute the maximum workload generated by these 4 LP carry-in jobs




To compute the exact area requires to enumerate all possible subsets of 4 tasks out of the set of LP tasks

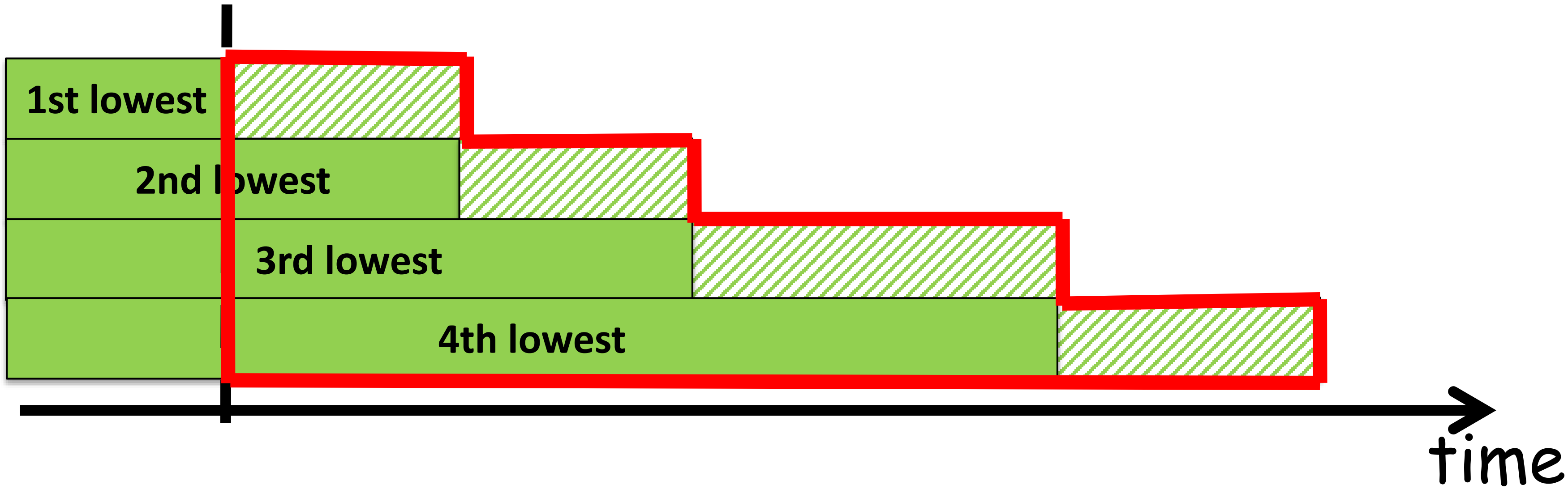
# Our solution




We propose three different techniques to upper-bound this area:

- Two of them are fairly straigthforward
- The third one is more complex

# Our solution




Upper-bound on the area

→ Maximum interfering workload from LP tasks

→ Maximum interference from HP and LP tasks

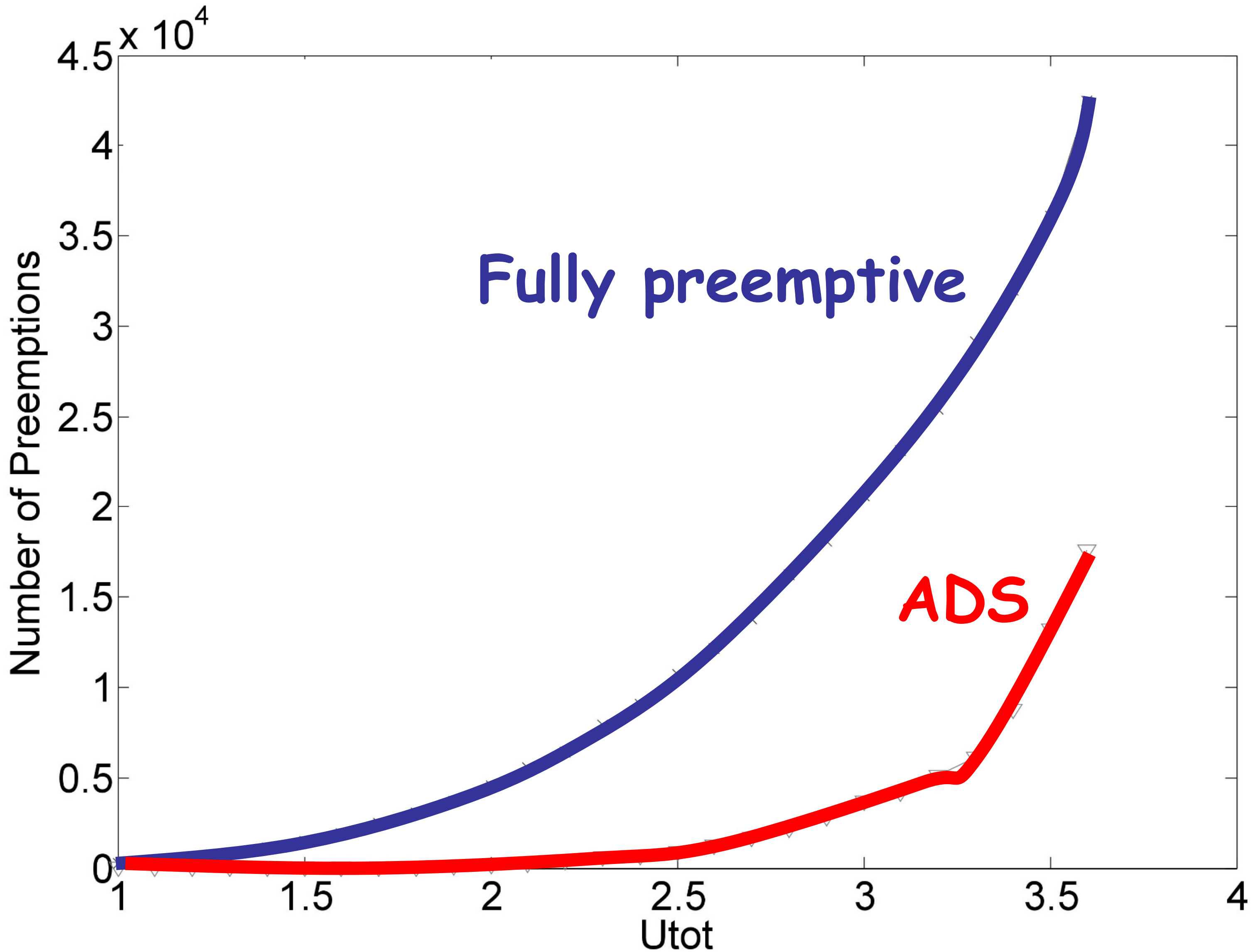
→ Schedulability test

# Second contribution

We designed a top-down approach (starting with the highest priority task) to assign to each task the maximum non-preemptive region length such that all the HP tasks are still schedulable.

→ Reduce the number of preemption

# Some preliminary results



# Questions?

**Thank you for your attention ☺**