Tardiness Bounds for Fixed-Priority Global Scheduling without Intra-Task Precedence Constraints

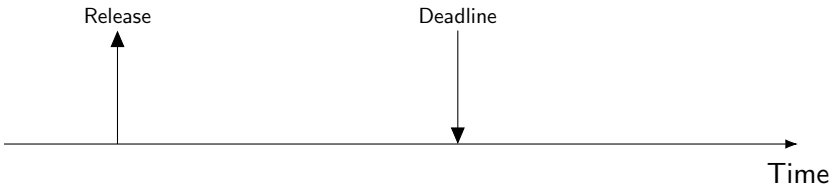
Sergey Voronov, James H. Anderson, Kecheng Yang

The University of North Carolina at Chapel Hill
Department of Computer Science

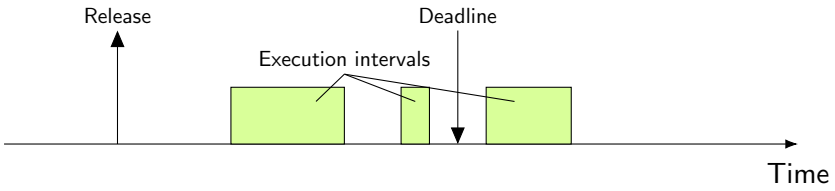
October 10, 2018

Soft Real-Time (SRT)

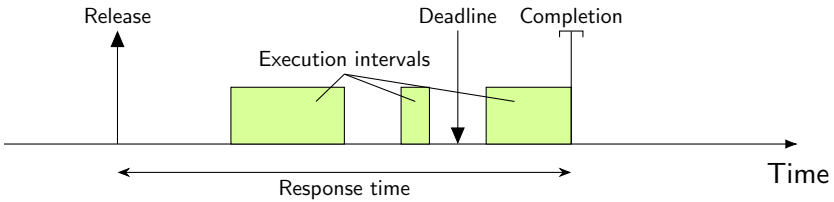
Soft Real-Time (SRT)



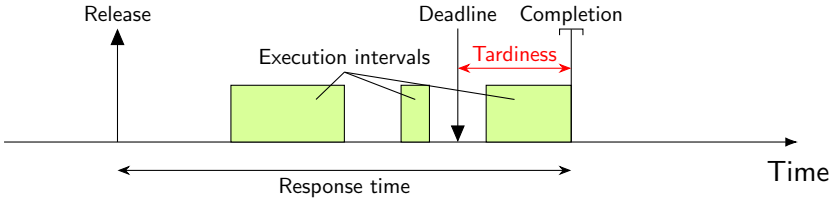
Soft Real-Time (SRT)



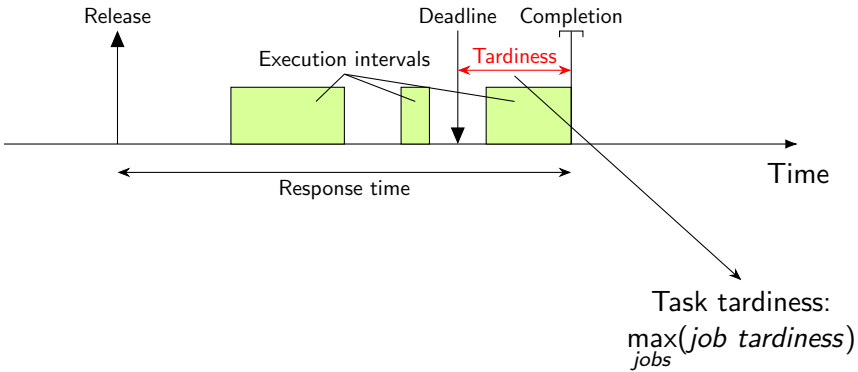
Soft Real-Time (SRT)



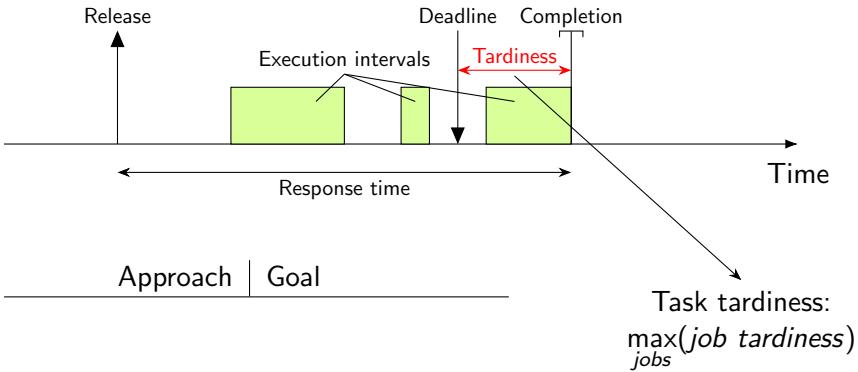
Soft Real-Time (SRT)



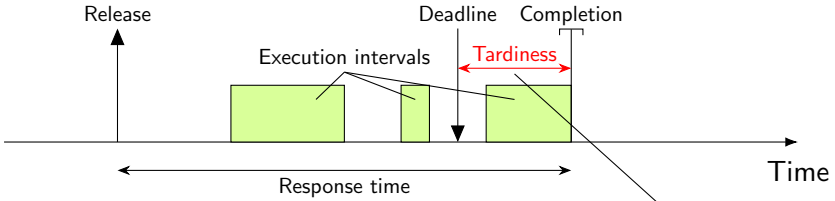
Soft Real-Time (SRT)



Soft Real-Time (SRT)



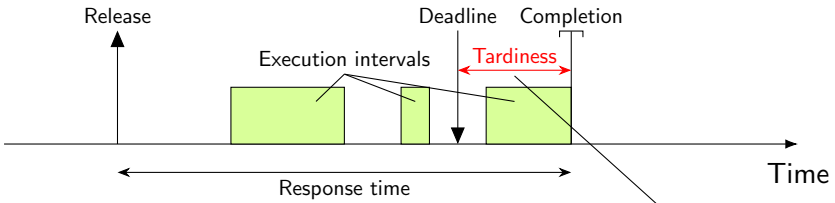
Soft Real-Time (SRT)



Approach	Goal
Hard Real-Time	no tardiness at all

Task tardiness:
$\max_{jobs}(job\ tardiness)$

Soft Real-Time (SRT)



Approach	Goal
Hard Real-Time	no tardiness at all
Soft Real-Time	bounded tardiness for every task

Task tardiness:
$\max_{jobs}(job\ tardiness)$

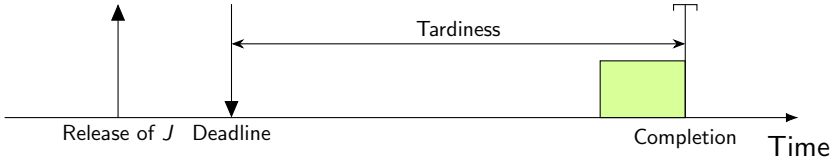
Outline

- Review of G-EDF SRT-optimality proof
- G-FP and SRT-optimality
- Sporadic task model generalization: NPC-sporadic
- Key idea of G-FP-optimality proof under the generalized model

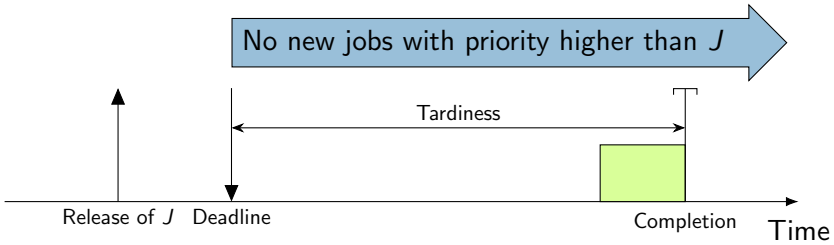
Main result:

Global Fixed Priority scheduling is SRT-optimal under the NPC-sporadic task model.

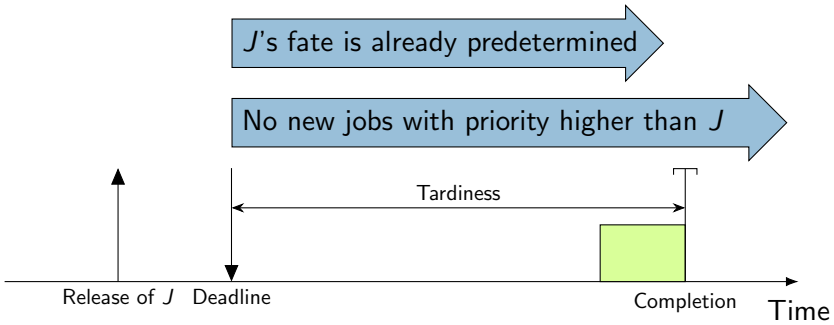
G-EDF Soft Real-Time Optimality



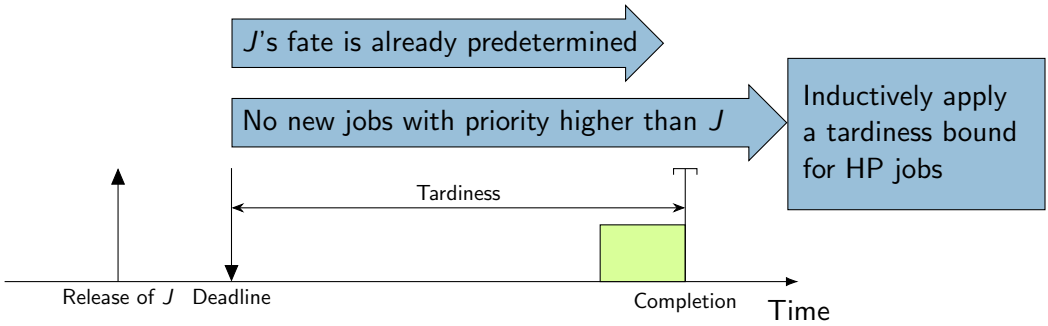
G-EDF Soft Real-Time Optimality



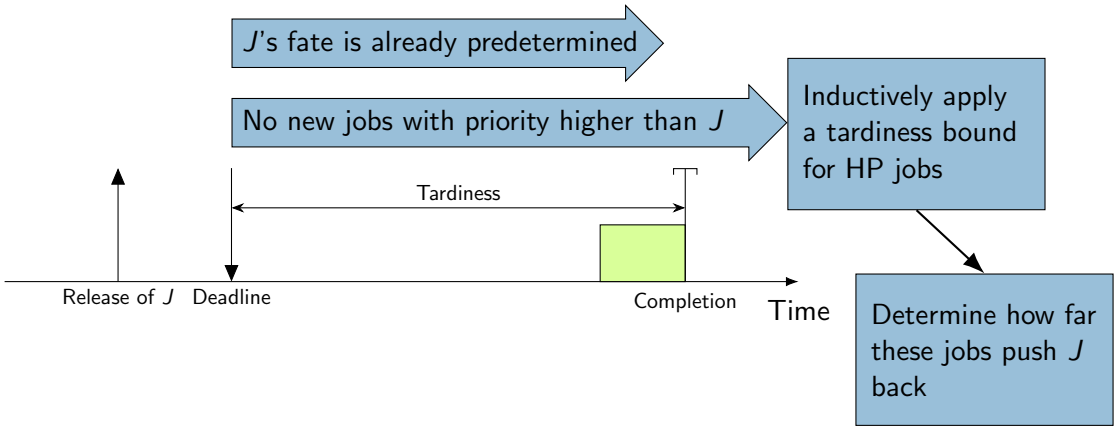
G-EDF Soft Real-Time Optimality



G-EDF Soft Real-Time Optimality



G-EDF Soft Real-Time Optimality



G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization

G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads

G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

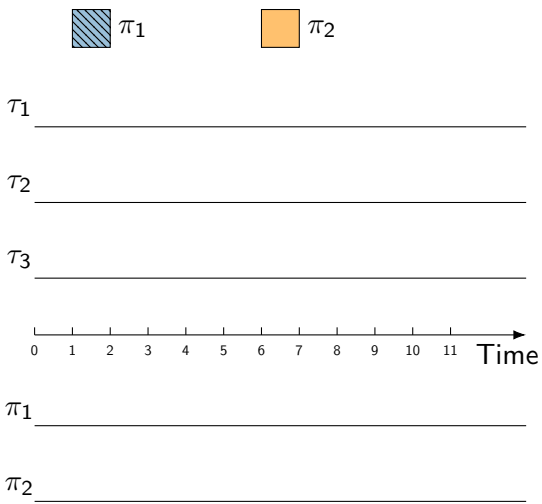
- prioritization
- smaller overheads
- easier to implement and debug

G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads
- easier to implement and debug

3 tasks with different priorities: $C = 2,\ T = 3$
multiprocessor with two unit-speed cores (π_1, π_2)

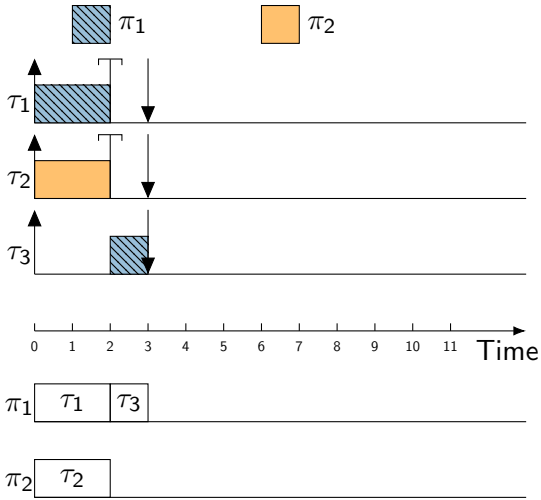


G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads
- easier to implement and debug

3 tasks with different priorities: $C = 2,\ T = 3$
multiprocessor with two unit-speed cores (π_1, π_2)

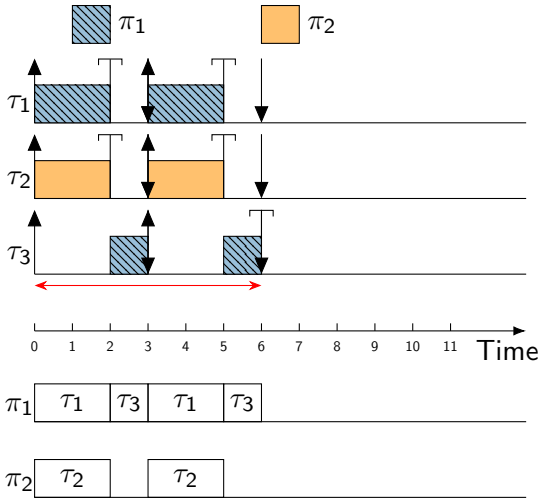


G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads
- easier to implement and debug

3 tasks with different priorities: $C = 2,\ T = 3$
multiprocessor with two unit-speed cores (π_1, π_2)

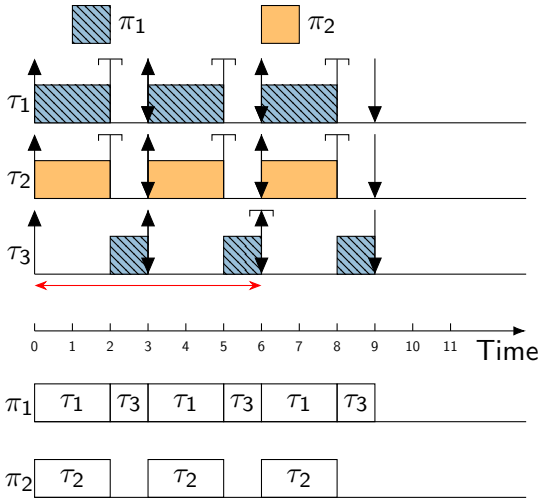


G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads
- easier to implement and debug

3 tasks with different priorities: $C = 2,\ T = 3$
multiprocessor with two unit-speed cores (π_1, π_2)

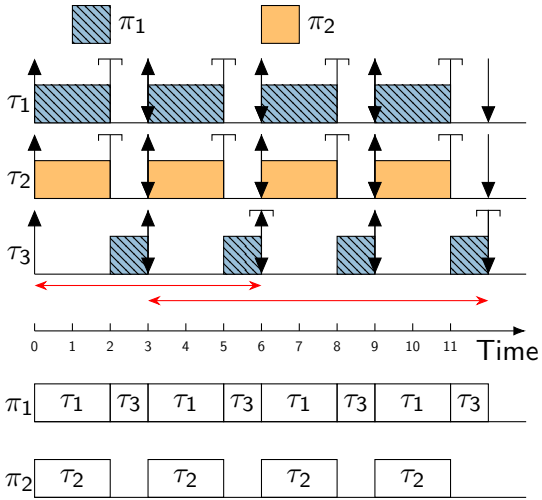


G-FP Soft Real-Time Non-Optimality

Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads
- easier to implement and debug

3 tasks with different priorities: $C = 2,\ T = 3$
multiprocessor with two unit-speed cores (π_1, π_2)

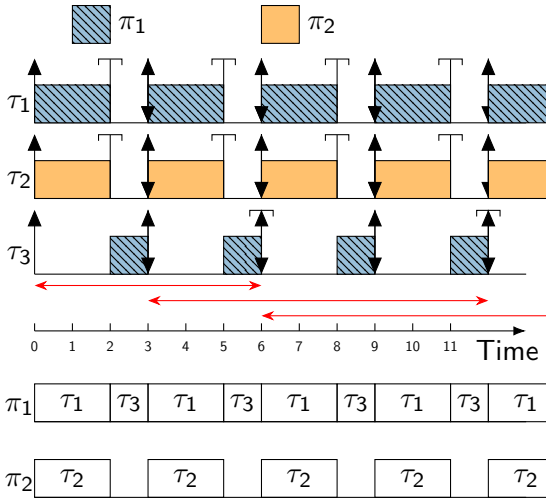


G-FP Soft Real-Time Non-Optimality

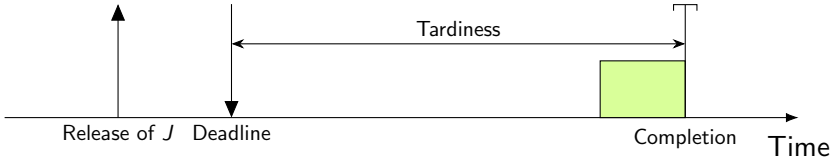
Reasons to use Fixed Priority scheduler?

- prioritization
- smaller overheads
- easier to implement and debug

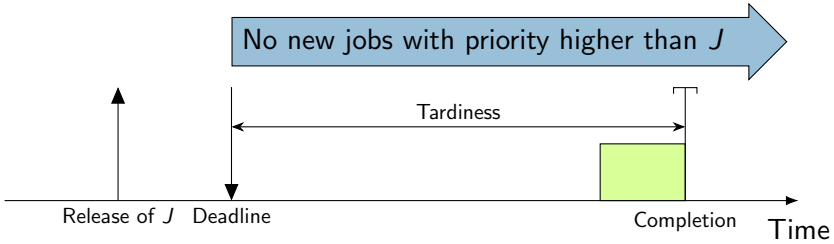
3 tasks with different priorities: $C = 2,\ T = 3$
multiprocessor with two unit-speed cores (π_1, π_2)



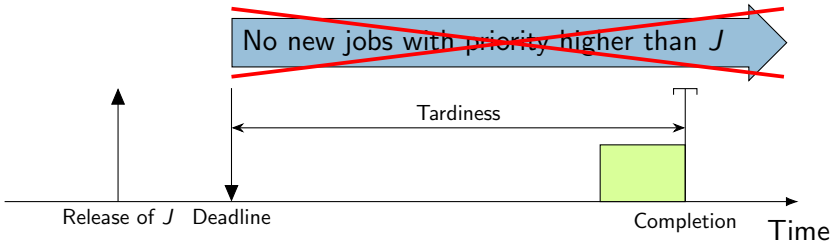
G-FP Soft Real-Time Non-Optimality



G-FP Soft Real-Time Non-Optimality



G-FP Soft Real-Time Non-Optimality



Sporadic Tasks With **N**o Job **P**recedence **C**onstraints (NPC)

- Any two jobs of an NPC-sporadic task can be scheduled in parallel with each other.

Sporadic Tasks With **N**o Job **P**recedence **C**onstraints (NPC)

- Any two jobs of an NPC-sporadic task can be scheduled in parallel with each other.
- Jobs of an NPC-sporadic task are prioritized according to FIFO scheduling principle.

Sporadic Tasks With **N**o Job **P**recedence **C**onstraints (NPC)

- Any two jobs of an NPC-sporadic task can be scheduled in parallel with each other.
- Jobs of an NPC-sporadic task are prioritized according to FIFO scheduling principle.

Applications:

Sporadic Tasks With **N**o Job **P**recedence **C**onstraints (NPC)

- Any two jobs of an NPC-sporadic task can be scheduled in parallel with each other.
- Jobs of an NPC-sporadic task are prioritized according to FIFO scheduling principle.

Applications:

- Video and Image processing (e.g, object detection from camera)

Sporadic Tasks With **N**o Job **P**recedence **C**onstraints (NPC)

- Any two jobs of an NPC-sporadic task can be scheduled in parallel with each other.
- Jobs of an NPC-sporadic task are prioritized according to FIFO scheduling principle.

Applications:

- Video and Image processing (e.g, object detection from camera)
  Sporadic to NPC-sporadic task model switch significantly decreases response time[#]

[#] Yang et al., Making OpenVX Really "Real Time", RTSS'18, to appear.

Sporadic Tasks With **N**o Job **P**recedence **C**onstraints (NPC)

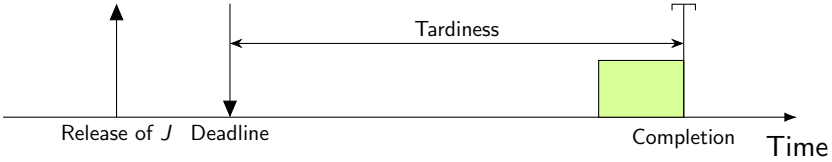
- Any two jobs of an NPC-sporadic task can be scheduled in parallel with each other.
- Jobs of an NPC-sporadic task are prioritized according to FIFO scheduling principle.

Applications:

- Video and Image processing (e.g, object detection from camera)
  Sporadic to NPC-sporadic task model switch significantly decreases response time[#]
- Industry interest: Huawei cell base stations

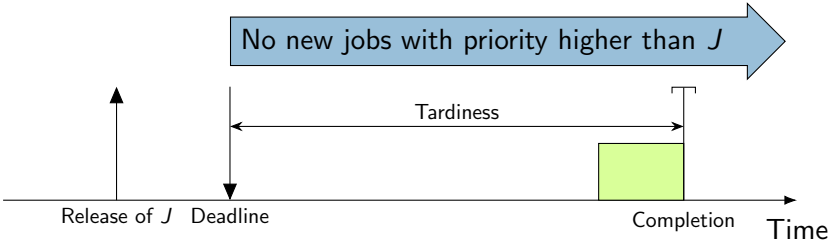
[#] Yang et al., Making OpenVX Really "Real Time", RTSS'18, to appear.

G-EDF under NPC-Sporadic Task Model



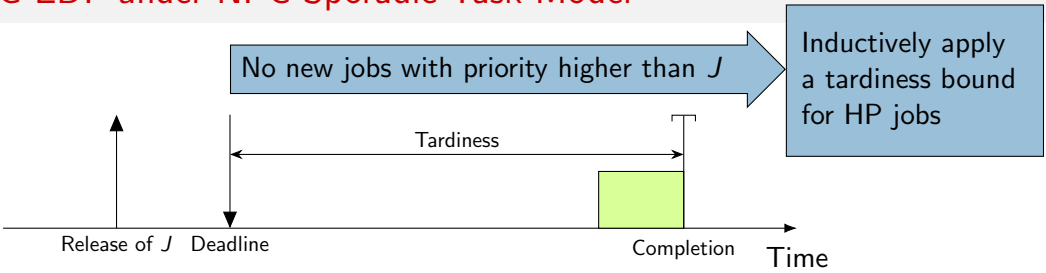
J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

G-EDF under NPC-Sporadic Task Model



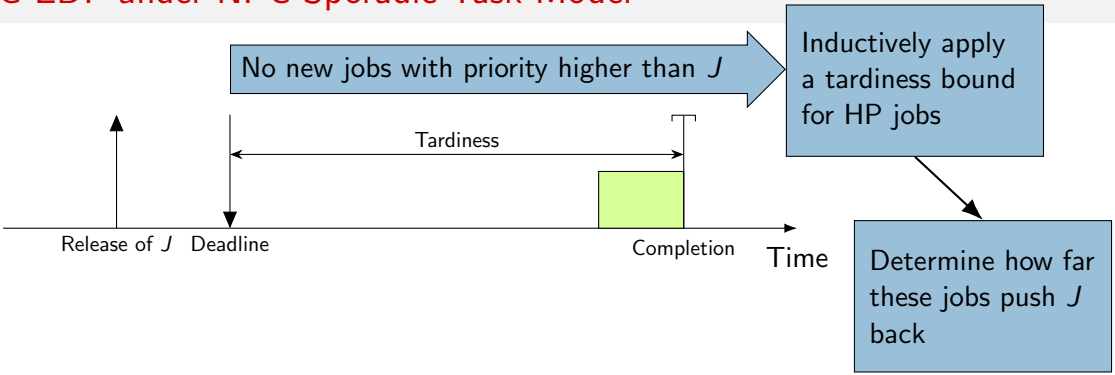
J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

G-EDF under NPC-Sporadic Task Model



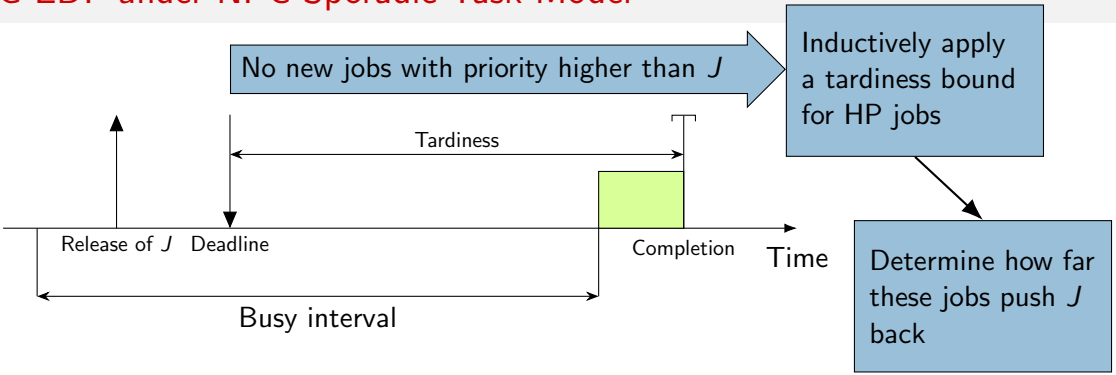
J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

G-EDF under NPC-Sporadic Task Model



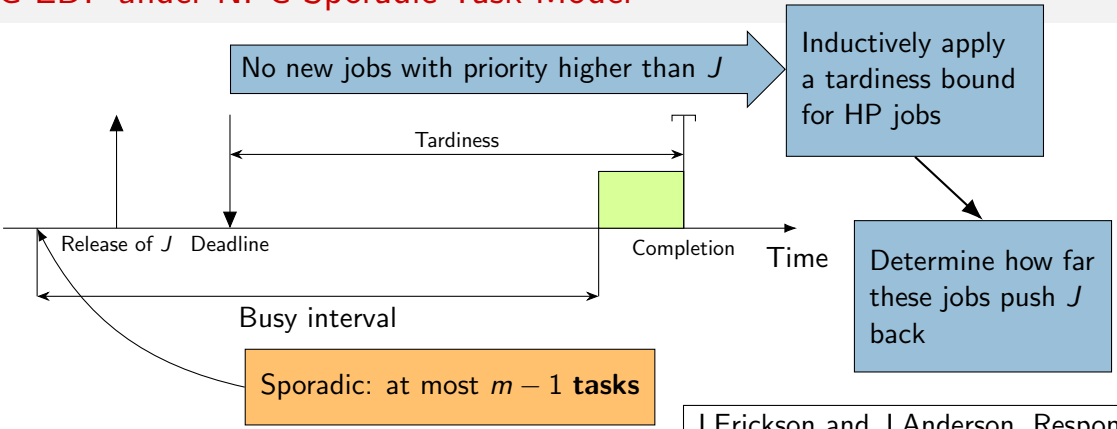
J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

G-EDF under NPC-Sporadic Task Model



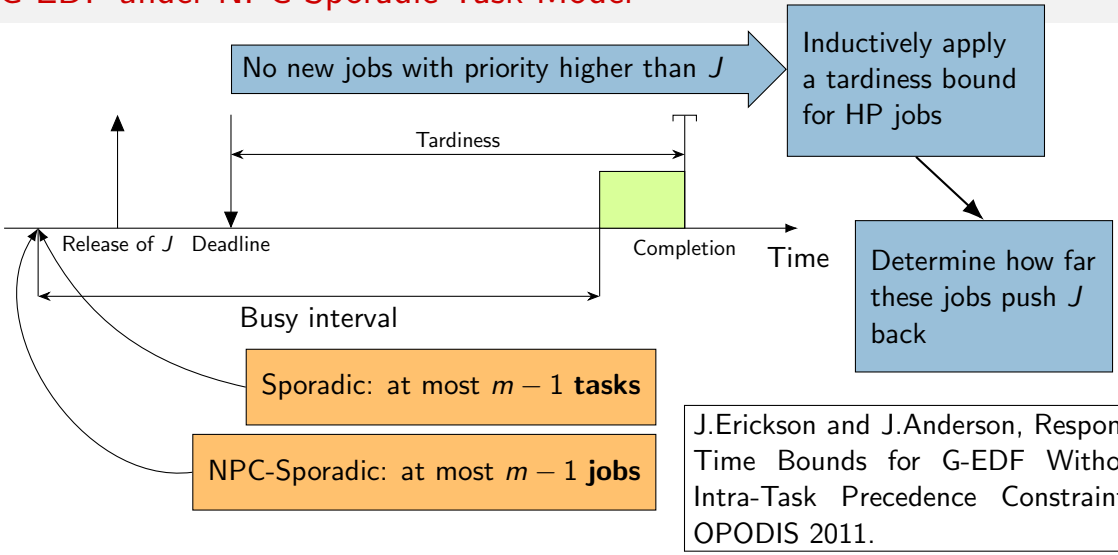
J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

G-EDF under NPC-Sporadic Task Model



J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

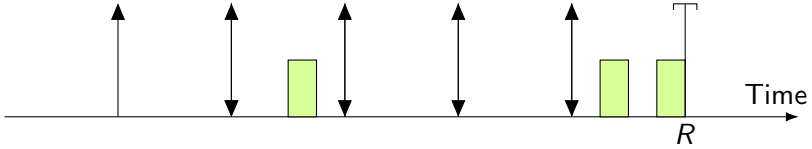
G-EDF under NPC-Sporadic Task Model



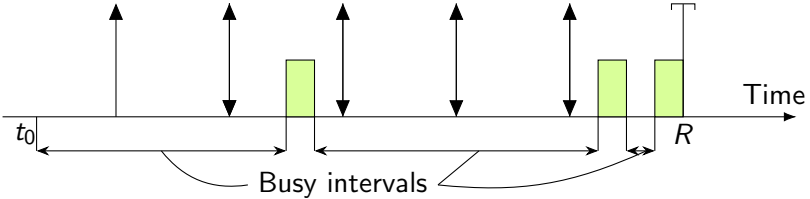
J.Erickson and J.Anderson, Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints, OPODIS 2011.

G-FP under NPC-Sporadic Task Model

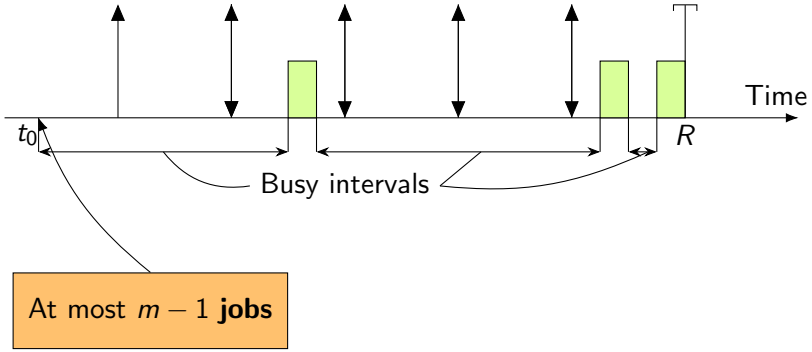
G-FP under NPC-Sporadic Task Model



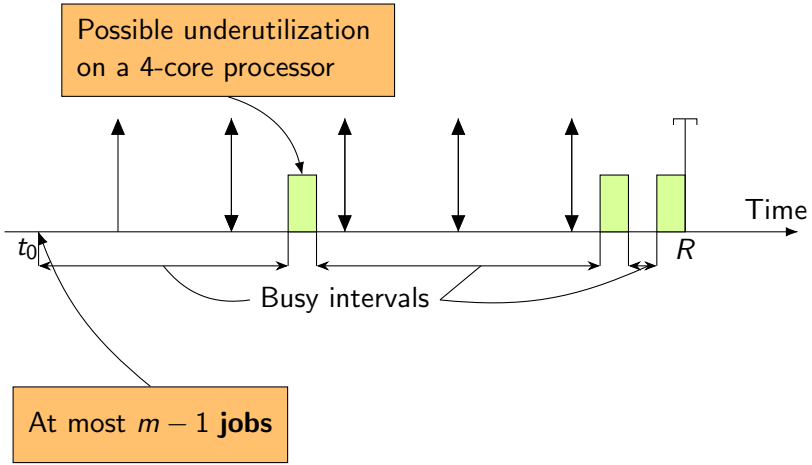
G-FP under NPC-Sporadic Task Model



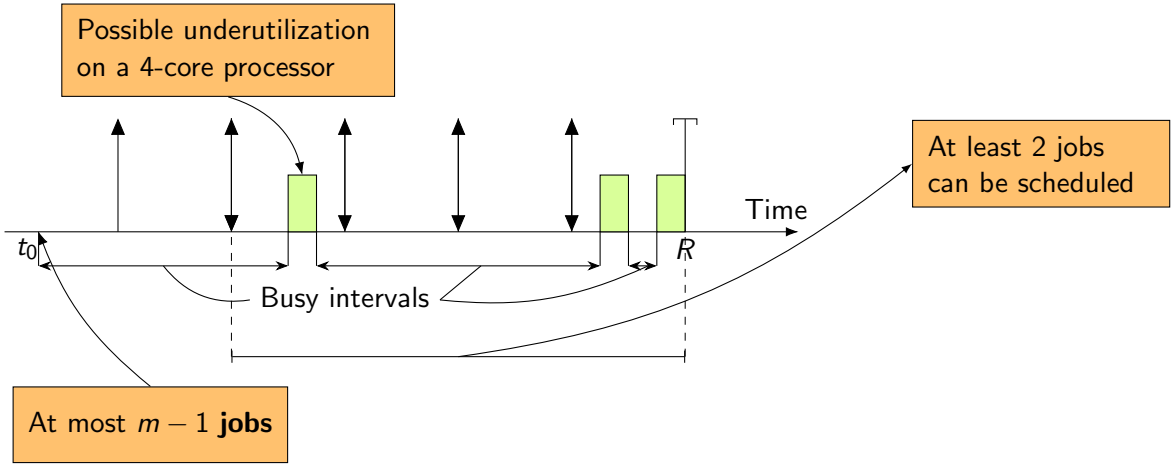
G-FP under NPC-Sporadic Task Model



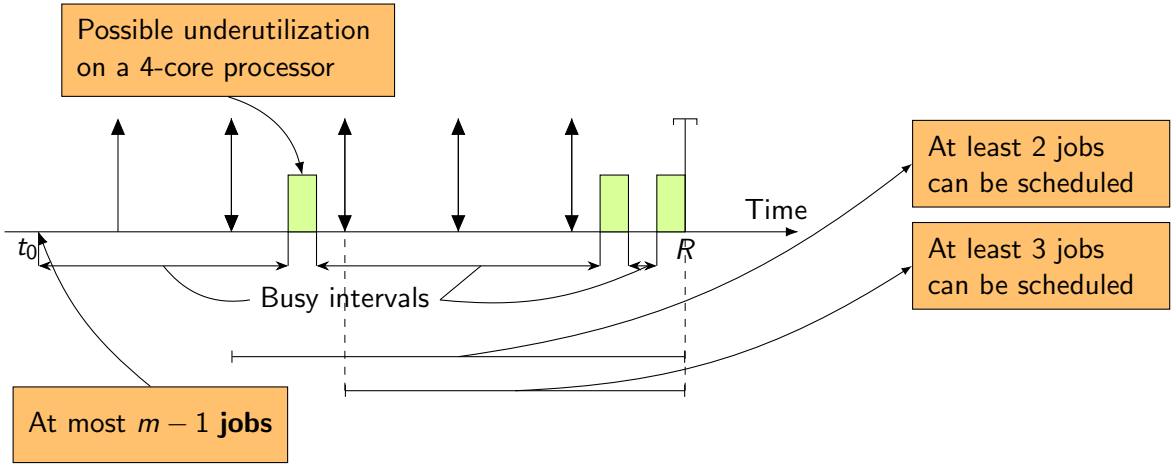
G-FP under NPC-Sporadic Task Model



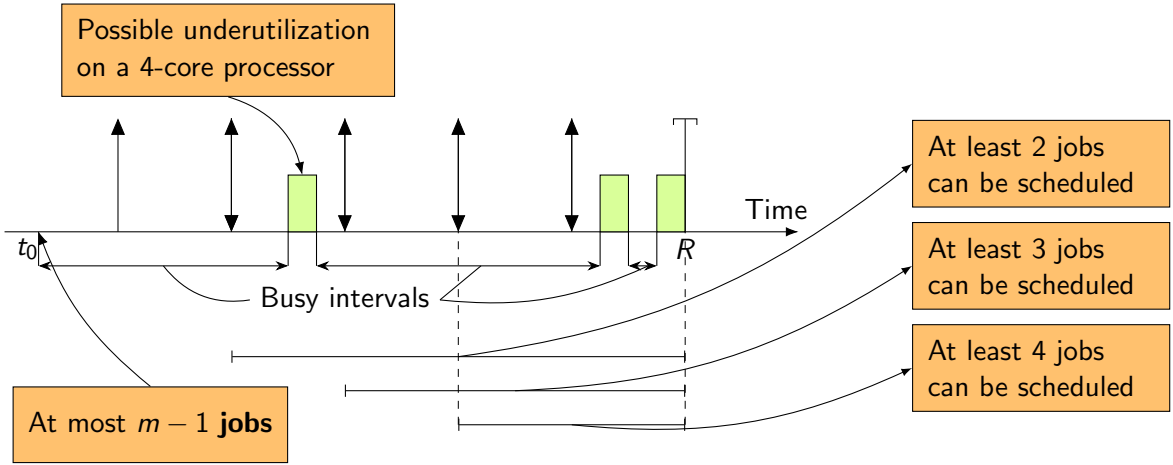
G-FP under NPC-Sporadic Task Model



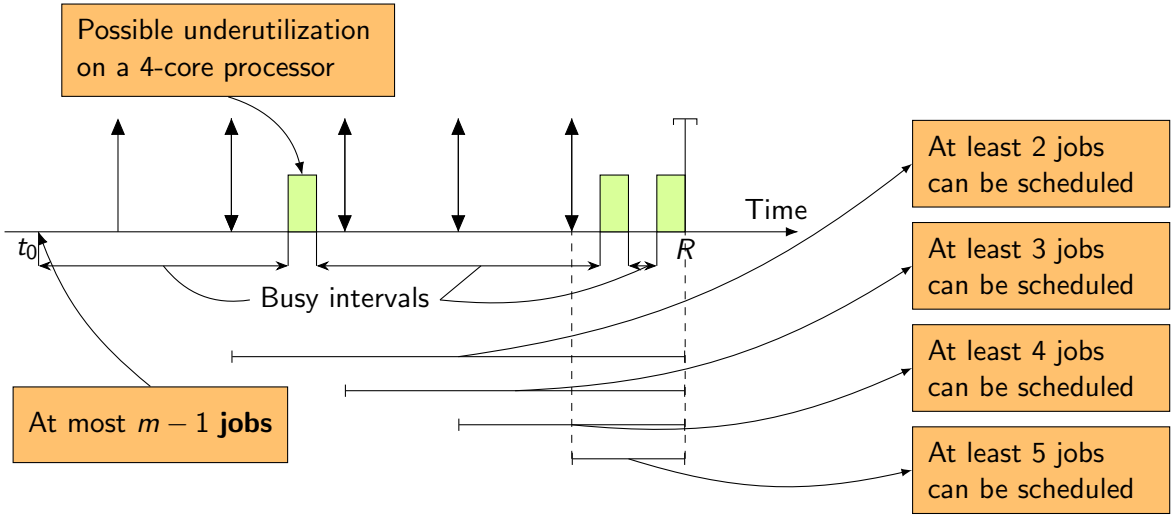
G-FP under NPC-Sporadic Task Model



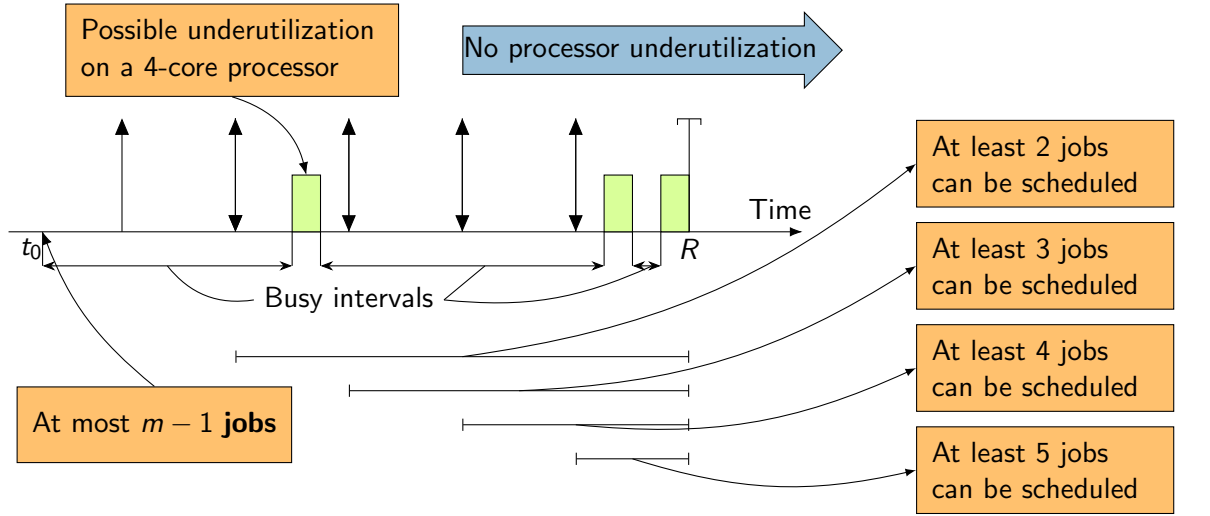
G-FP under NPC-Sporadic Task Model



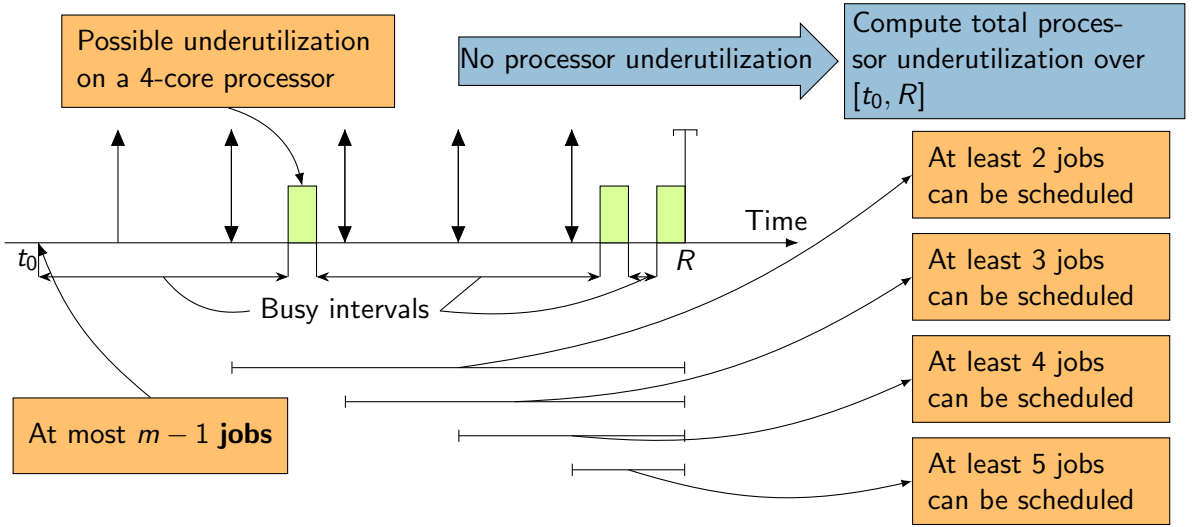
G-FP under NPC-Sporadic Task Model



G-FP under NPC-Sporadic Task Model



G-FP under NPC-Sporadic Task Model



Result Discussion

Applicability:

- Any NPC-sporadic task, even if its utilization exceeds one ($u_i \leq m$ instead of $u_i \leq 1$)

Result Discussion

Applicability:

- Any NPC-sporadic task, even if its utilization exceeds one ($u_i \leq m$ instead of $u_i \leq 1$)
- Any NPC-sporadic task, regardless of other tasks in system

Result Discussion

Applicability:

- Any NPC-sporadic task, even if its utilization exceeds one ($u_i \leq m$ instead of $u_i \leq 1$)
- Any NPC-sporadic task, regardless of other tasks in system
- Any work-conserving scheduler (preemptive or not)

Result Discussion

Applicability:

- Any NPC-sporadic task, even if its utilization exceeds one ($u_i \leq m$ instead of $u_i \leq 1$)
- Any NPC-sporadic task, regardless of other tasks in system
- Any work-conserving scheduler (preemptive or not)
- Asymptotically optimal

Prior work: J.Erickson and J.Anderson,
Response Time Bounds for G-EDF Without Intra-Task Precedence Constraints,
OPODIS 2011.

Experiments

Problem:

- Large obtained bound for low priority tasks
- Increase of effect with the higher tasks number

Experiments

Problem:

- Large obtained bound for low priority tasks
- Increase of effect with the higher tasks number

Strategy:

- Apply partitioned scheduling

Experiments

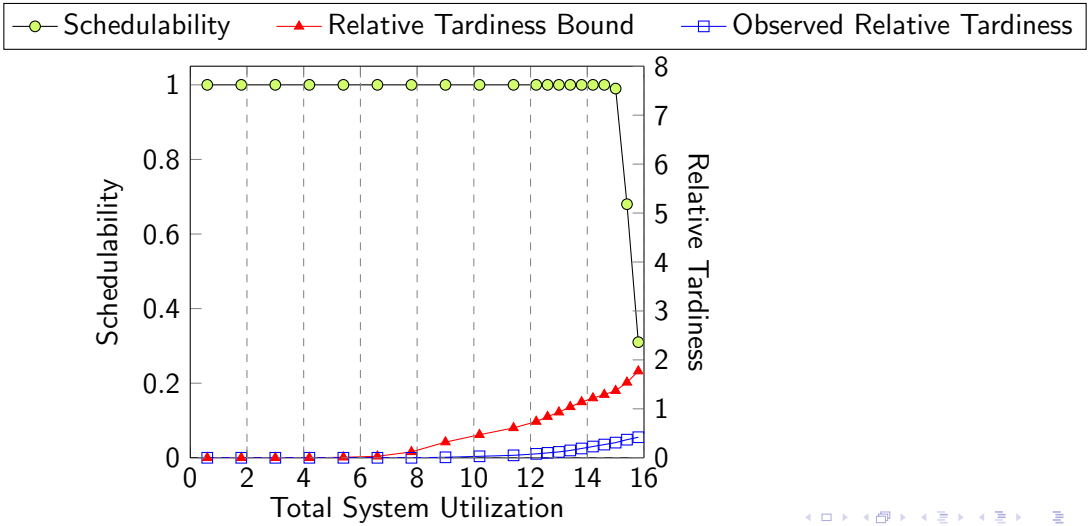
Problem:

- Large obtained bound for low priority tasks
- Increase of effect with the higher tasks number

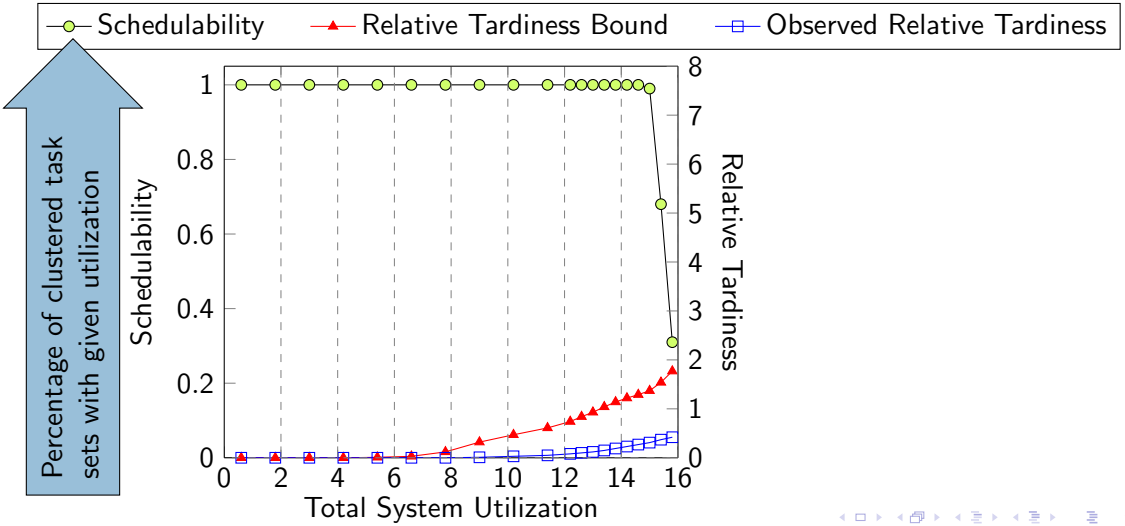
Strategy:

- Apply partitioned scheduling
- Use only bin packing heuristics
- Get smaller tardiness bounds

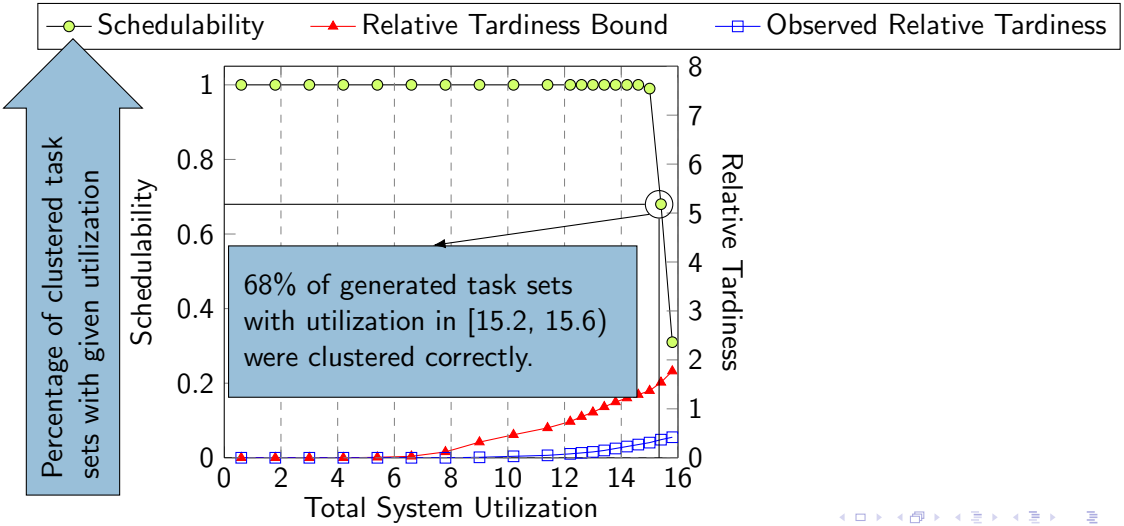
Experiments: Sample Plot



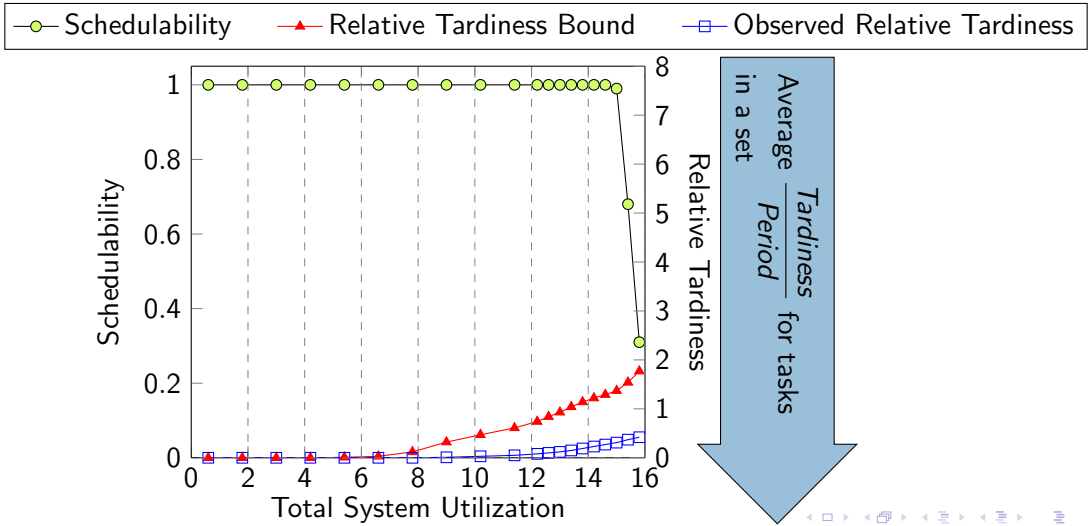
Experiments: Sample Plot



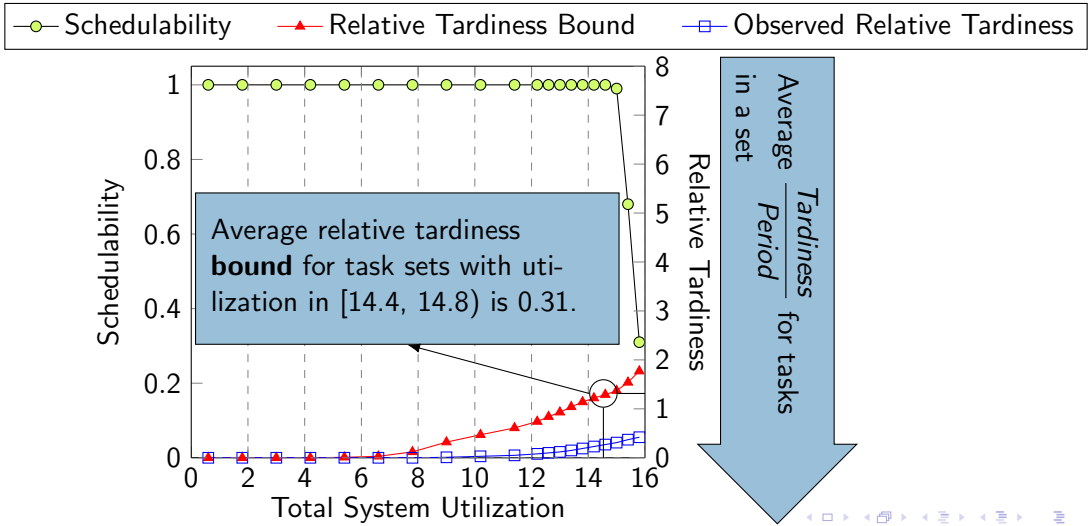
Experiments: Sample Plot



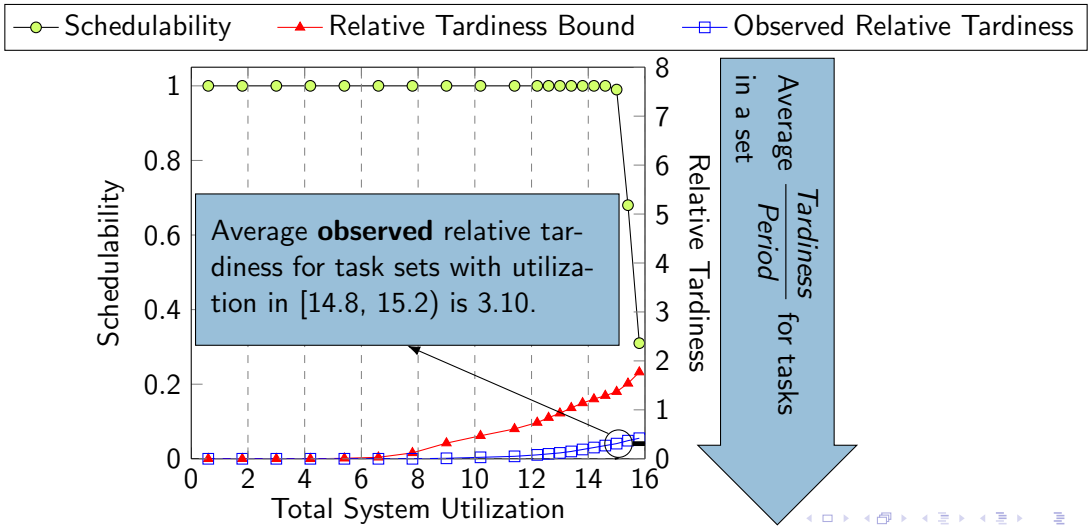
Experiments: Sample Plot



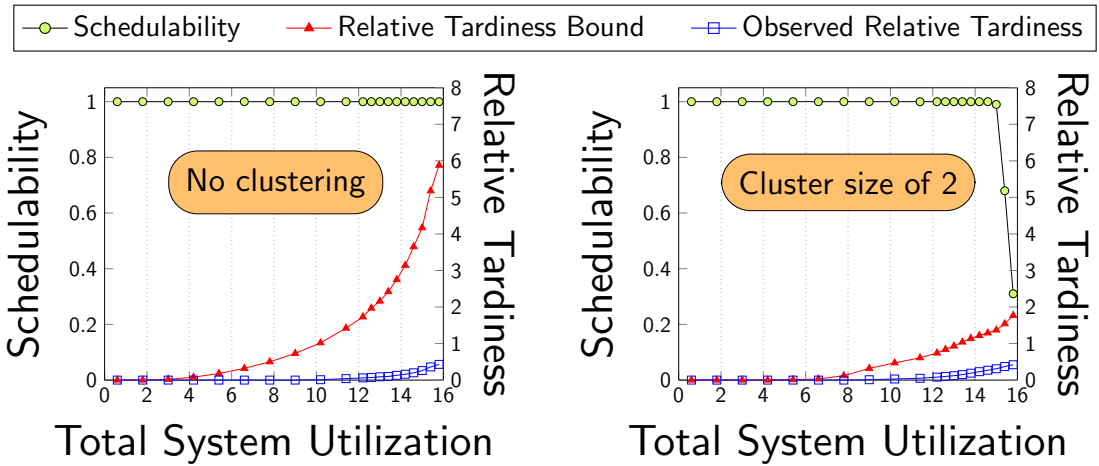
Experiments: Sample Plot



Experiments: Sample Plot



Experiments



Tardiness Bounds for NPC G-FP Scheduling

Global Fixed Priority scheduling is SRT-optimal under the NPC-sporadic task model.

Tardiness Bounds for NPC G-FP Scheduling

Global Fixed Priority scheduling is SRT-optimal under the NPC-sporadic task model.

Future work:

- Extend proposed analysis to DAG-based systems under Fixed-Priority scheduling

Tardiness Bounds for NPC G-FP Scheduling

Global Fixed Priority scheduling is SRT-optimal under the NPC-sporadic task model.

Future work:

- Extend proposed analysis to DAG-based systems under Fixed-Priority scheduling
- Consider more careful analysis of High Priority workload for other schedulers

Tardiness Bounds for NPC G-FP Scheduling

Global Fixed Priority scheduling is SRT-optimal under the NPC-sporadic task model.

Future work:

- Extend proposed analysis to DAG-based systems under Fixed-Priority scheduling
- Consider more careful analysis of High Priority workload for other schedulers

Thank you for listening.