Schedulability Analysis for Directed Acyclic Graphs on Multiprocessor Systems at a Subtask Level

ADA-Europe, Paris France

June 26, 2014

Manar Qamhieh Serge Midonnet

University of Paris-Est - LIGM, France

Outline

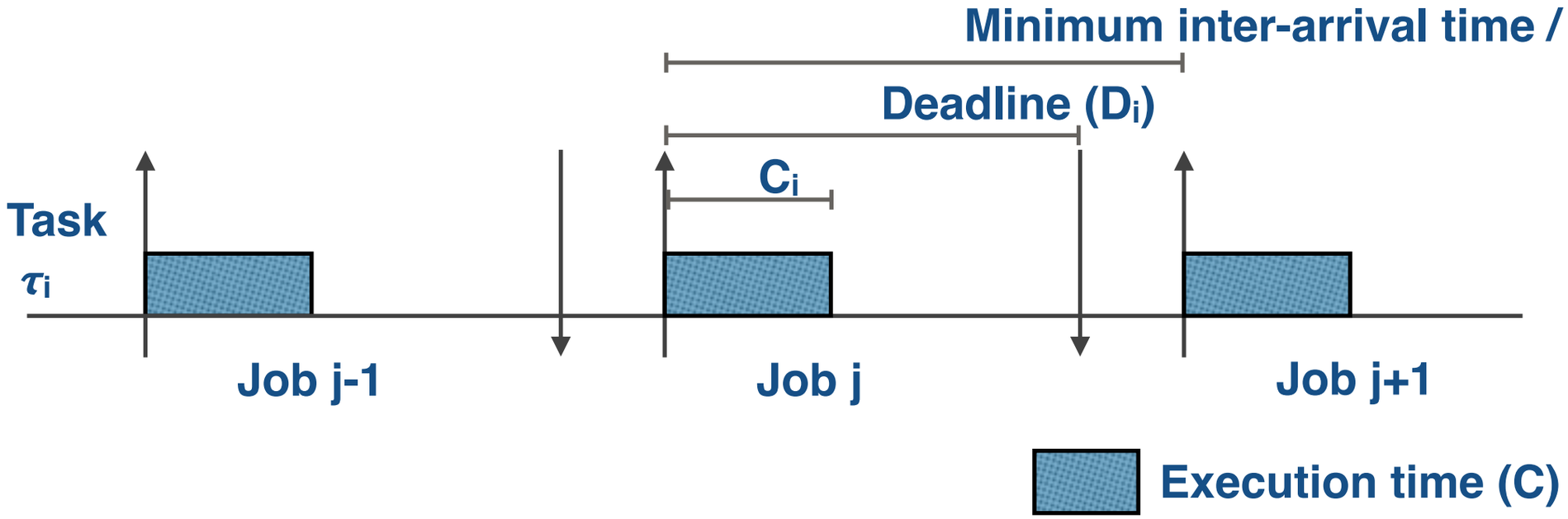
- Introduction
- Task Model and Motivation
- Scheduling on DAG vs. Subtask Levels
- Advantage of Subtask-Level Scheduling
- Interference & Workload Analysis
- Experimental Results
- Conclusion & Perspective

Introduction

- **A real-time System** is defined as “the system whose correctness depends not only on the correctness of the results, but also on respecting certain timing parameters”.
- They are divided into two types: **hard** & **soft**.
  - **Hard**: missing a deadline causes a system failure.
  - **Soft**: missing a deadline degrades the system’s quality of service.

Sequential Task Model

- A real-time periodic sequential task generates an infinite sequence of jobs.



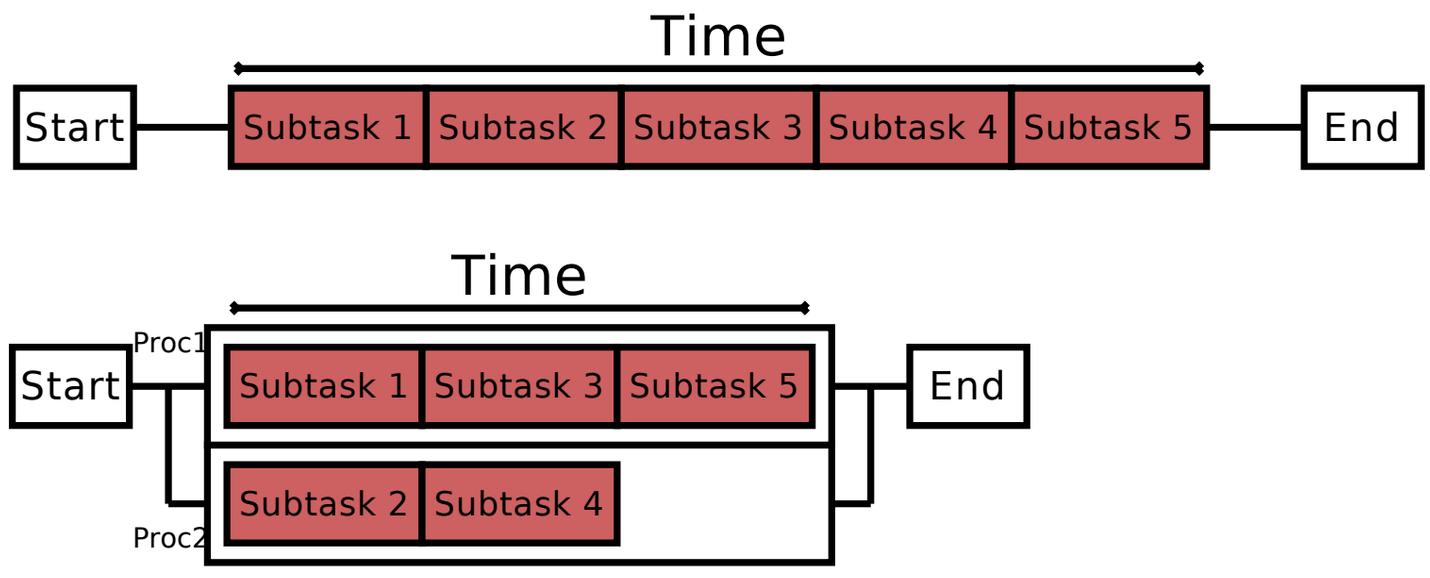
Parallelism in Real-time Multiprocessor Systems

Multiprocessor systems:

- Increase the performance of systems using **multiple processors** to overcome physical constraints of uniprocessors.

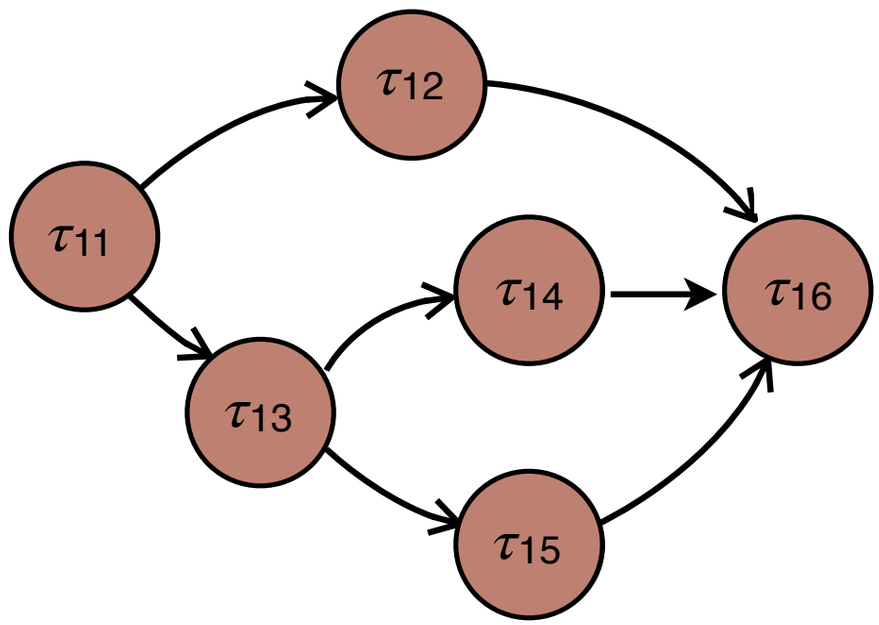
Software parallelism:

- The improvement in performance is gained by the use of multiple processors depends on the **software**.



Task Model: Directed Acyclic Graph (DAG)

- General Model of parallel tasks (other common models are special cases of the DAG model, such as the fork-join model and the multi-threaded segment model.)
- Intra-subtask parallelism is determined using precedence constraints between the subtasks presented as a graph.



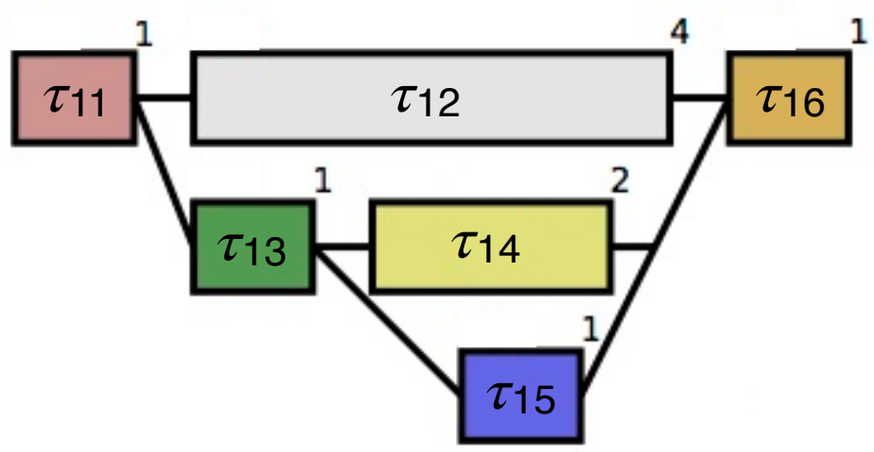
Task Model: Directed Acyclic Graph (DAG)

DAG set

- A set τ of n **sporadic, constrained deadline**, parallel real-time DAG tasks, scheduled on a system of m identical processors.

DAG task

- Each parallel task τ_i, where $1 \leq i \leq n$, is a DAG task.
- DAG task τ_i is characterized by $(n_i, 1 \leq j \leq n_i \mid \tau_{i,j}, G_i, D_i, T_i)$.



Deadline $D_1 = 8$
Period $T_1 = 8$
Total WCET $C_1 = 10$

DAG Scheduling: DAG vs. Subtask Level

Scheduling on a DAG Level **(State-of-the-art)**

- Subtasks inherit the global parameters of their DAG
- Use the global timing parameters of each DAG in the scheduling process.

Scheduling on a Subtask Level **(Our contribution)**

- Assign extra local timing parameters to subtasks based on their precedence constraints.
- Use local subtask parameters in the scheduling process.

DAG Scheduling: DAG vs. Subtask Level

- Example : set of 2 periodic implicit-deadline DAGs execute on 2 identical processors using global EDF.

DAG Parameters

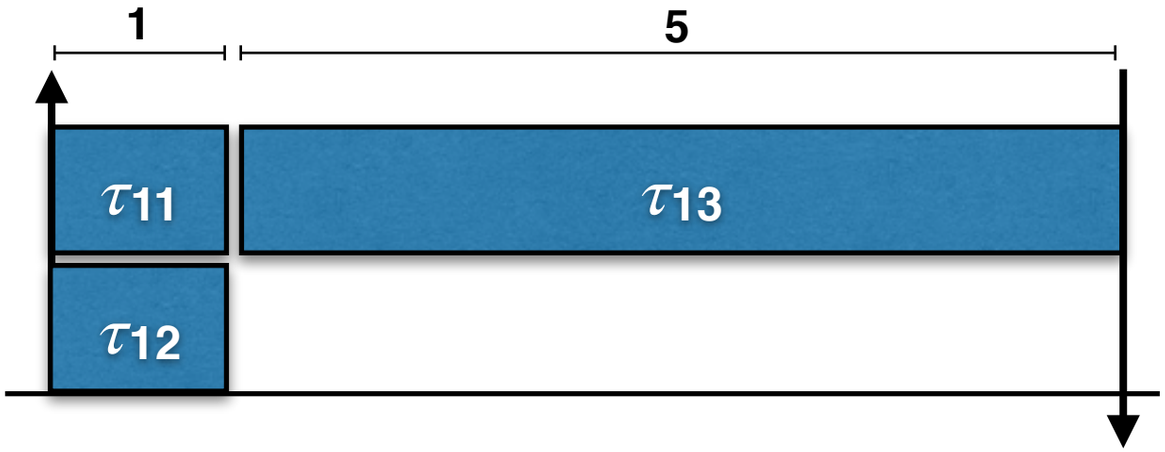
τ_{11} = (0,1,6,6)

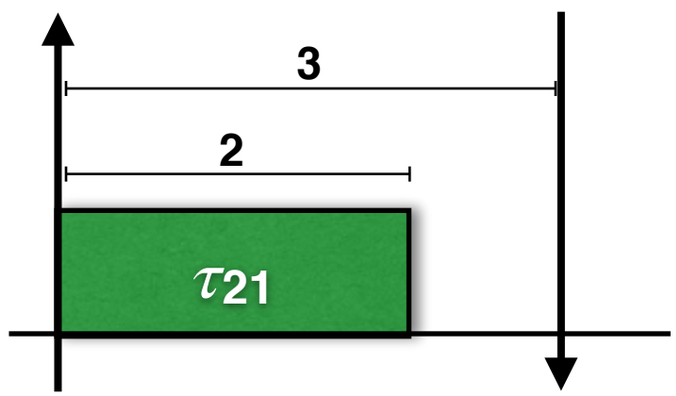
τ_{12} = (0,1,6,6)

τ_{13} = (0,5,6,6)

τ_{21} = (0, 2, 3, 3)

Offset
Execution time
Deadline
Period





Subtask Parameters

τ_{11} = (0,1,1,6)

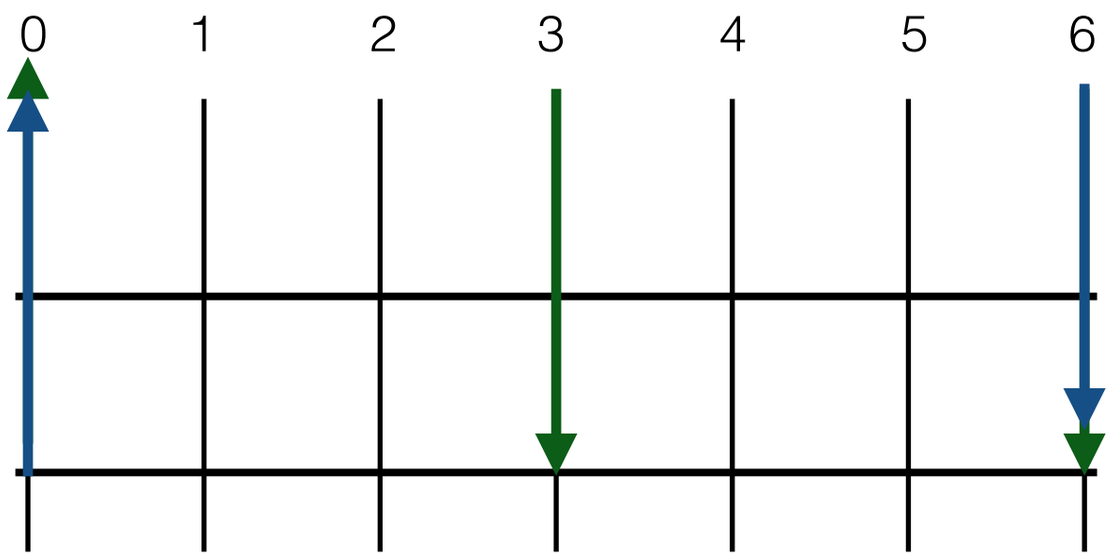
τ_{12} = (0,1,1,6)

τ_{13} = (1,5,5,6)

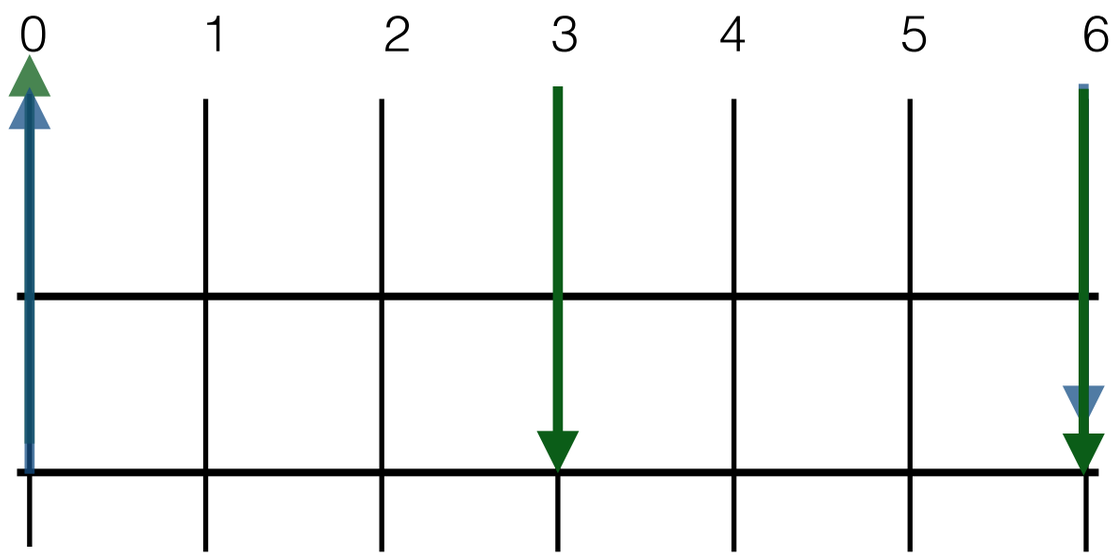
τ_{21} = (0,2,3,3)

DAG Scheduling: DAG vs. Subtask Level

DAG-Level Scheduling



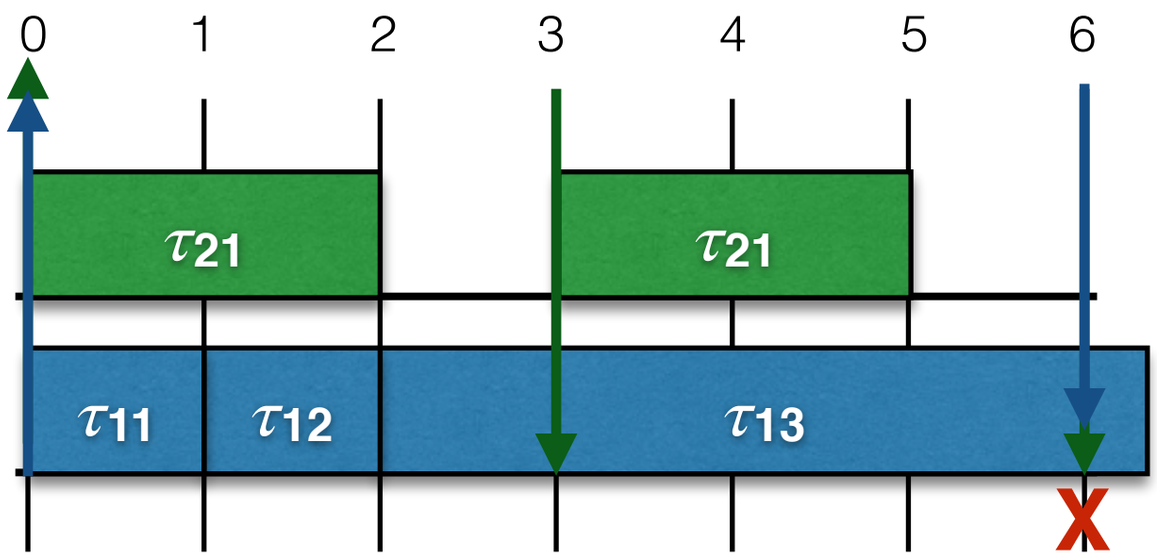
Subtask-Level Scheduling



DAG scheduling using **global Earliest Deadline First (EDF)** algorithm on a system of 2 identical processors.

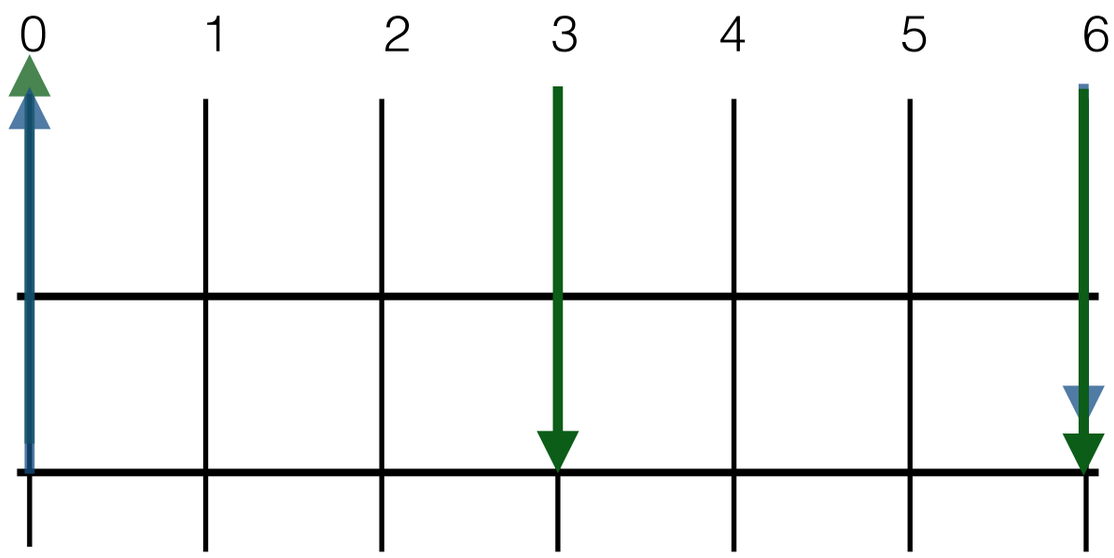
DAG Scheduling: DAG vs. Subtask Level

DAG-Level Scheduling



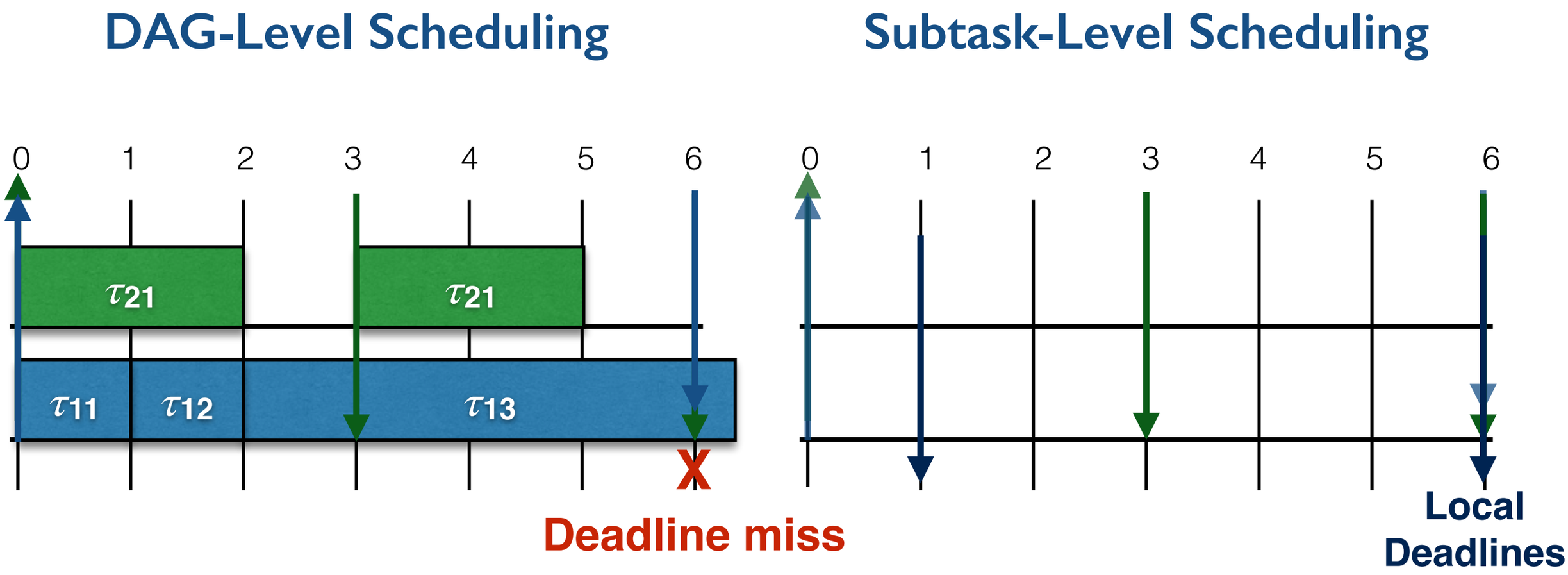
Deadline miss

Subtask-Level Scheduling



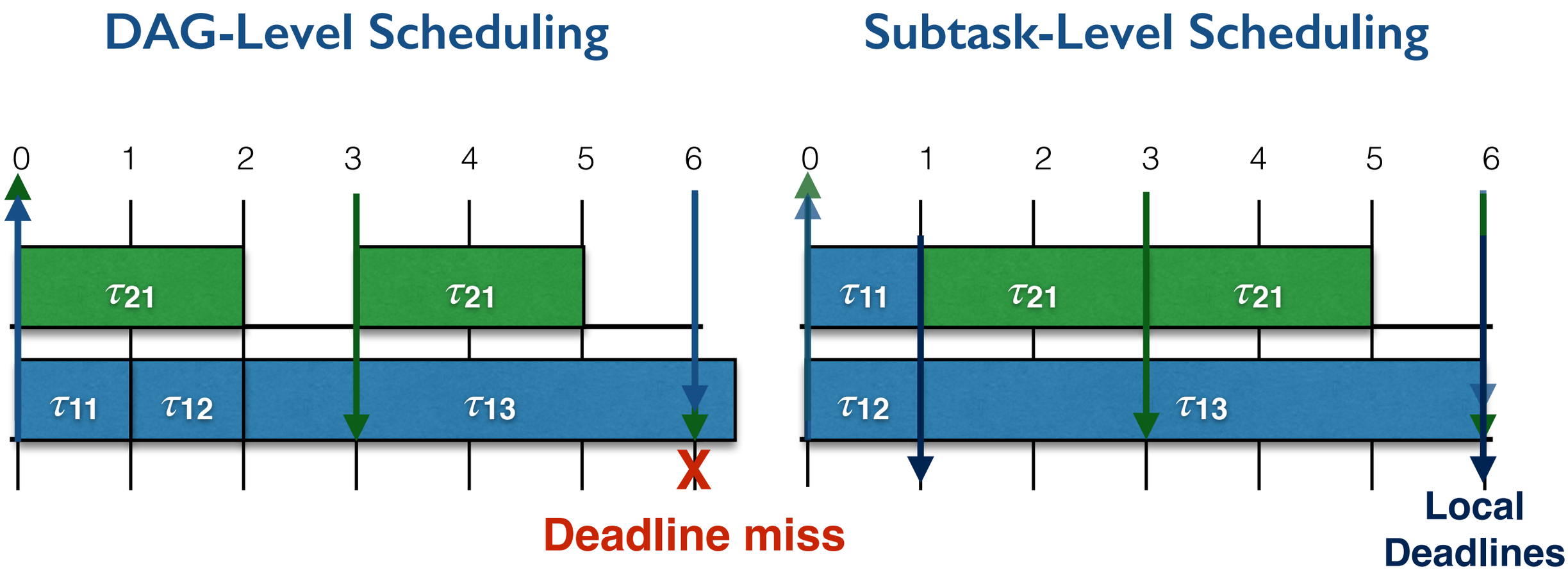
DAG scheduling using **global Earliest Deadline First (EDF)** algorithm on a system of 2 identical processors.

DAG Scheduling: DAG vs. Subtask Level



DAG scheduling using **global Earliest Deadline First (EDF)** algorithm on a system of 2 identical processors.

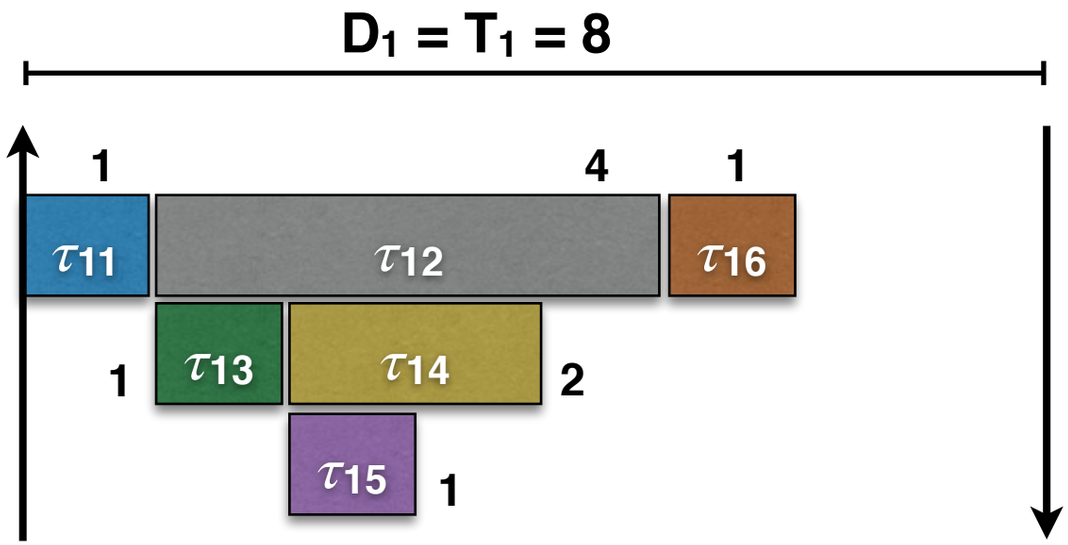
DAG Scheduling: DAG vs. Subtask Level



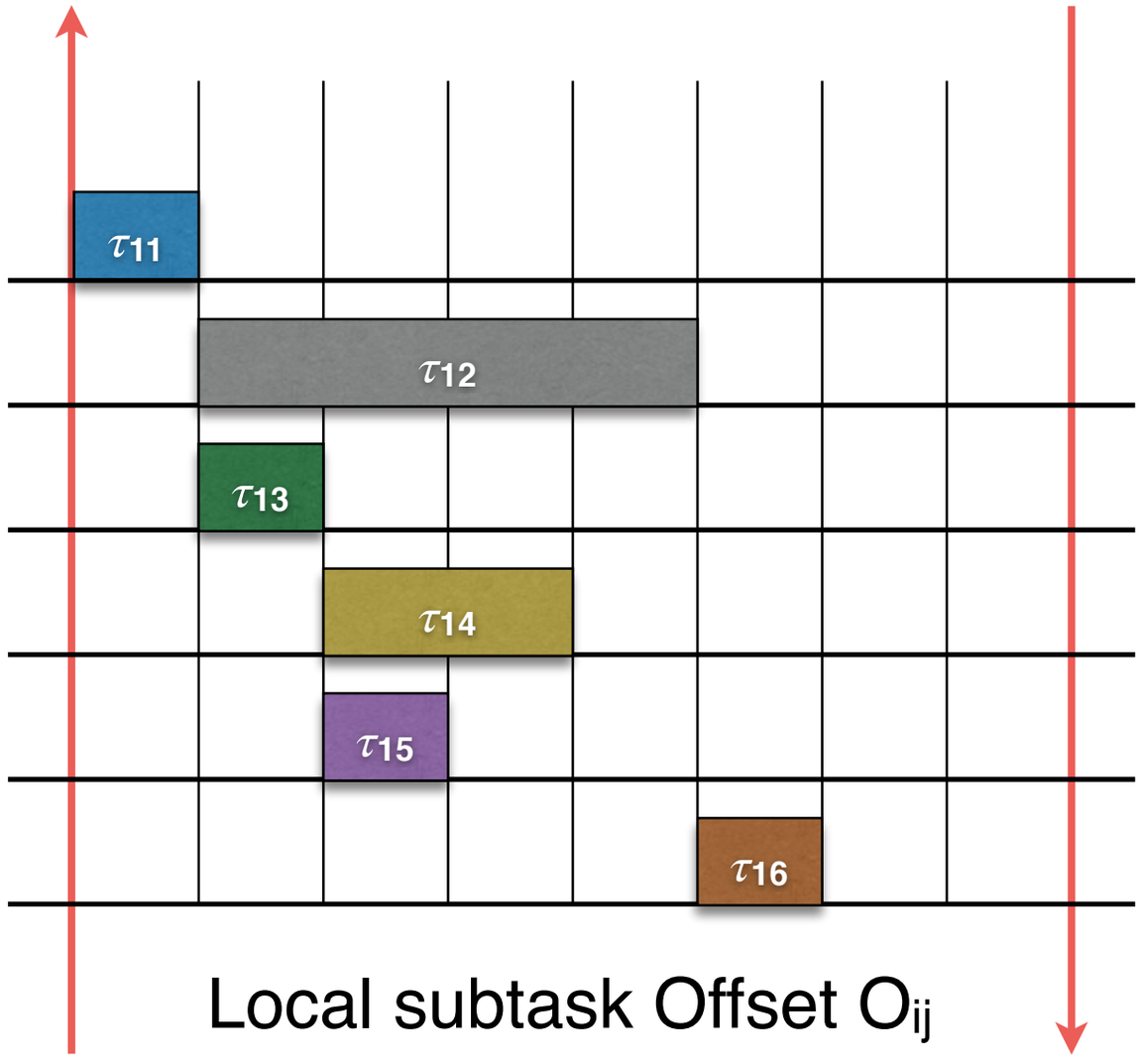
DAG scheduling using **global Earliest Deadline First (EDF)** algorithm on a system of 2 identical processors.

DAG Scheduling on a Subtask Level

Local Parameters

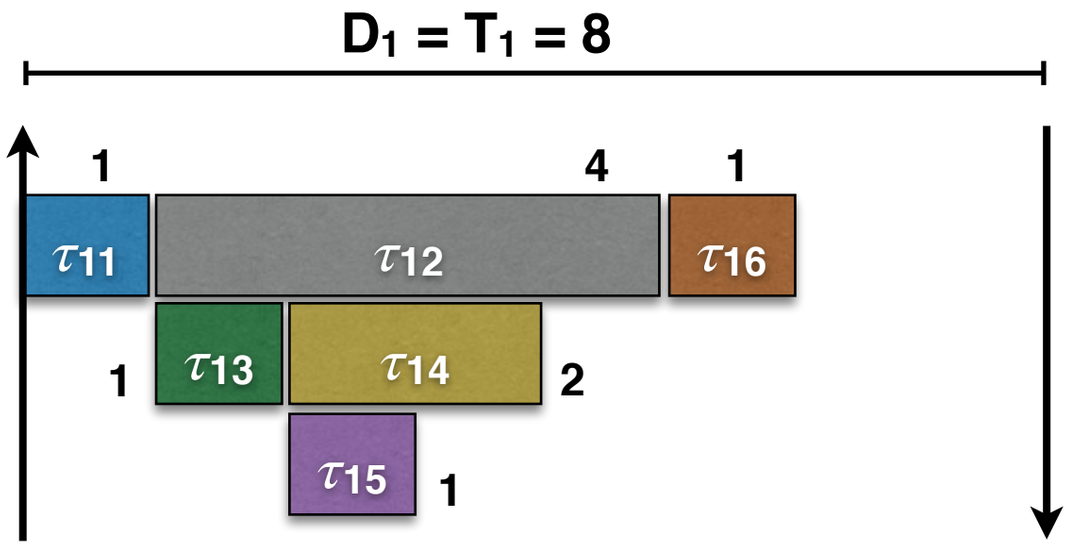


Based on the timing parameters of a DAG task, we identify the **local offset** of each subtask which is its **earliest possible activation** because of its predecessors.

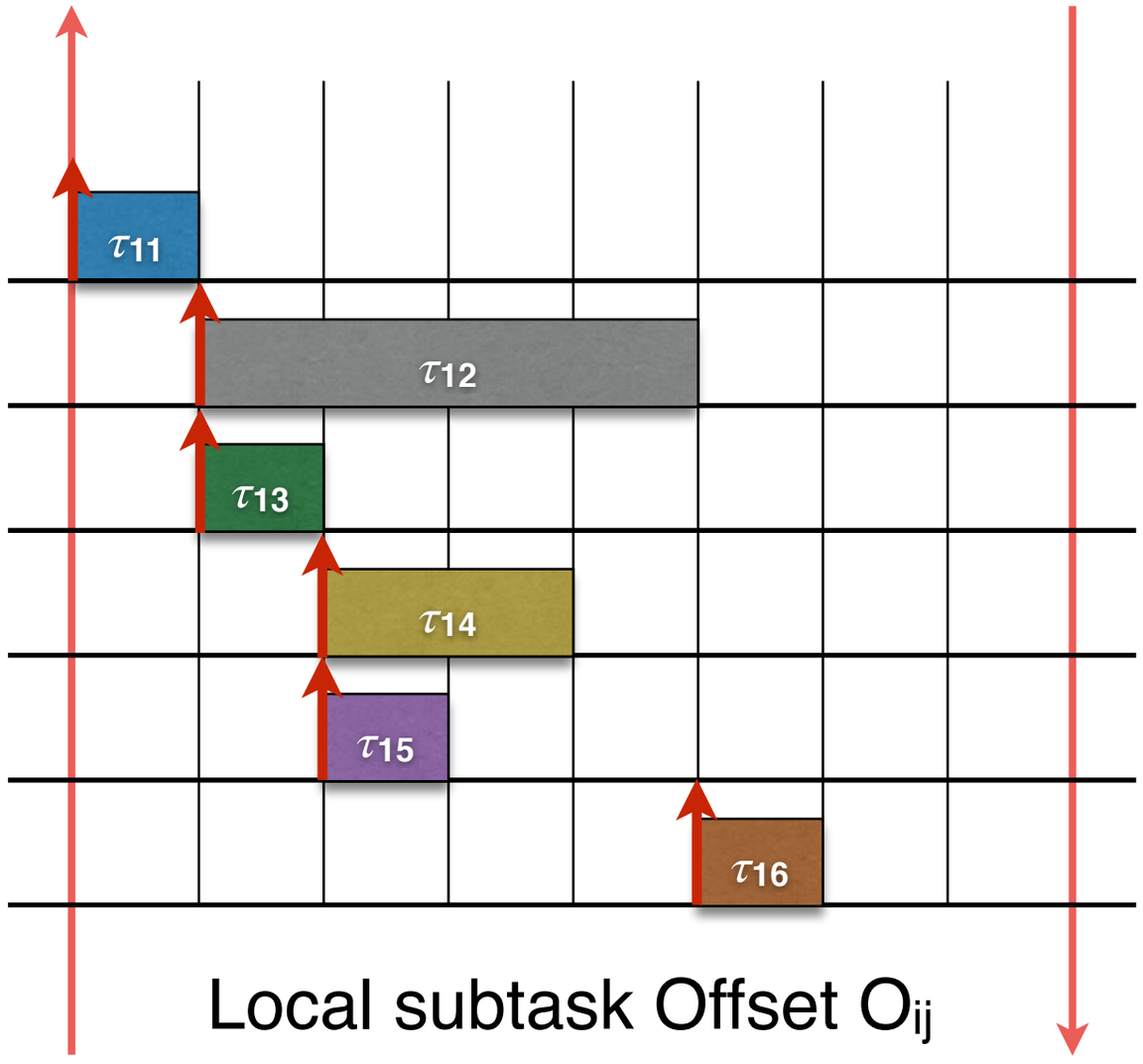


DAG Scheduling on a Subtask Level

Local Parameters



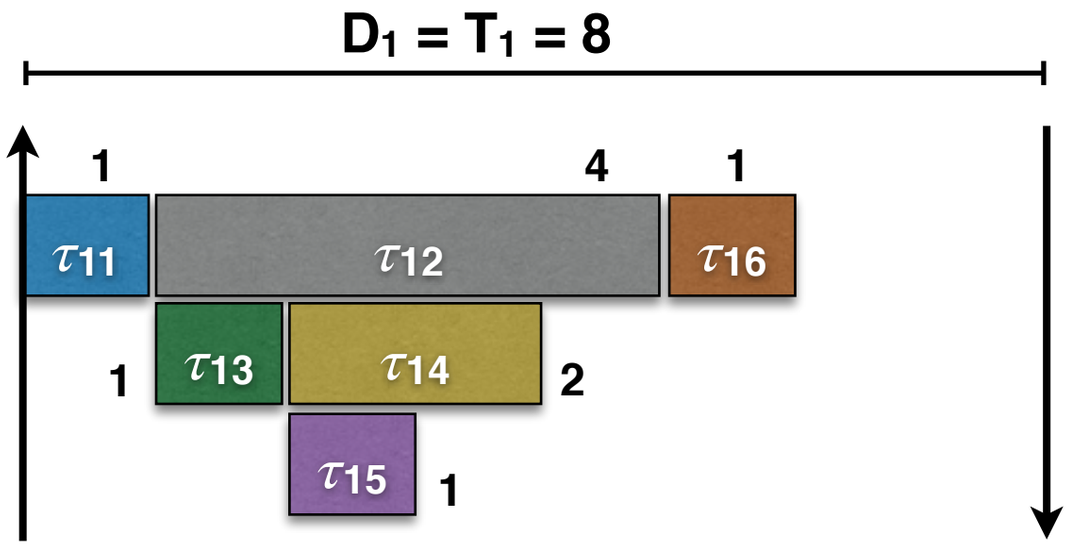
Based on the timing parameters of a DAG task, we identify the **local offset** of each subtask which is its **earliest possible activation** because of its predecessors.



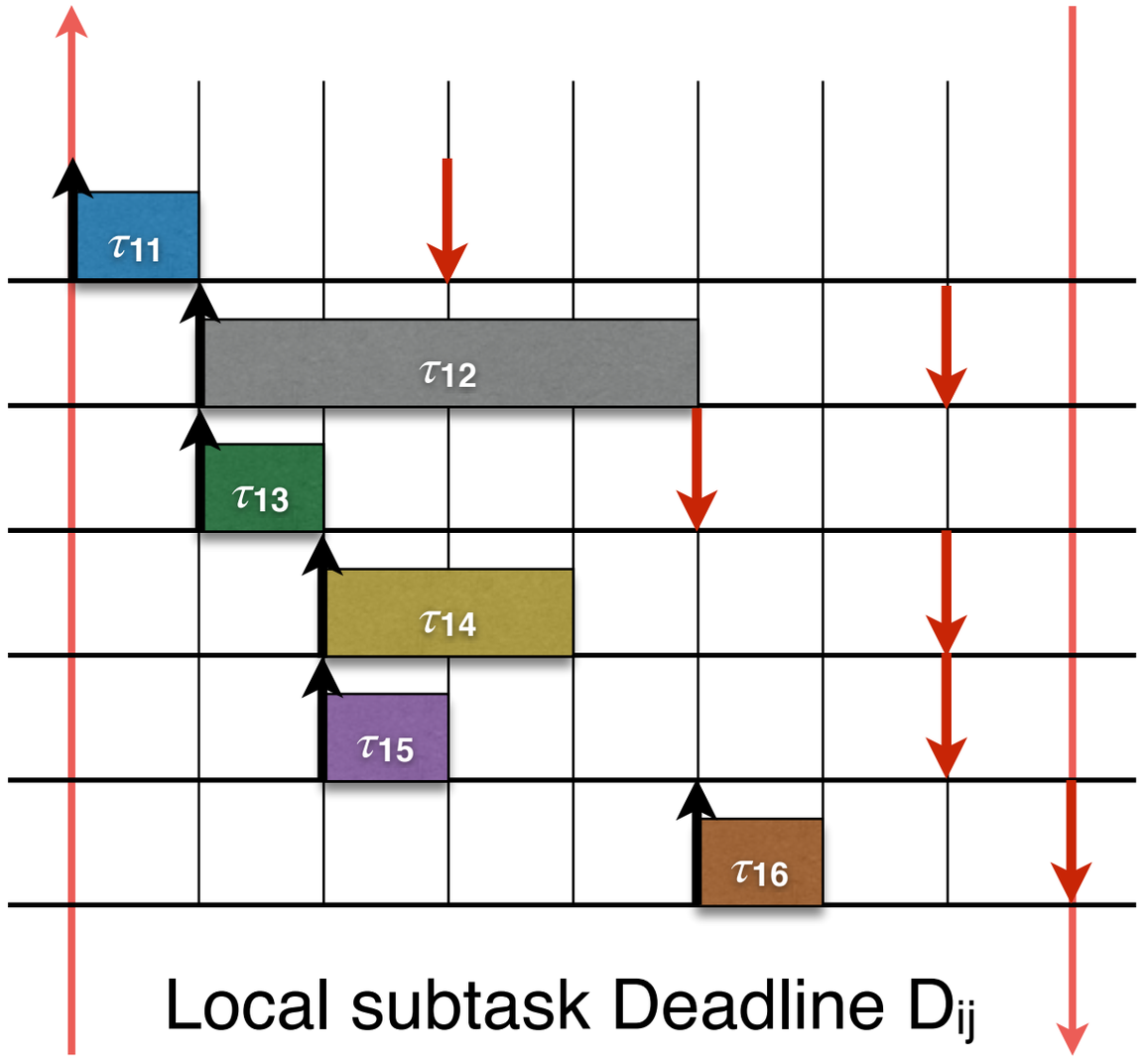
Local subtask Offset O_{ij}

DAG Scheduling on a Subtask Level

Local Parameters



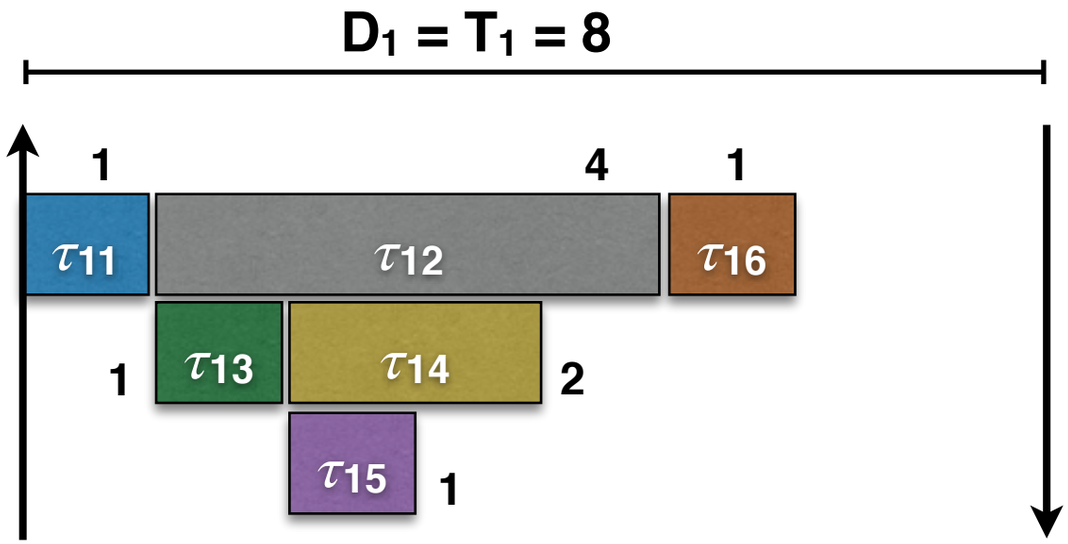
Based on the timing parameters of a DAG task, we identify the **local deadline** of each subtask which is its **latest possible finish time** because of its successors.



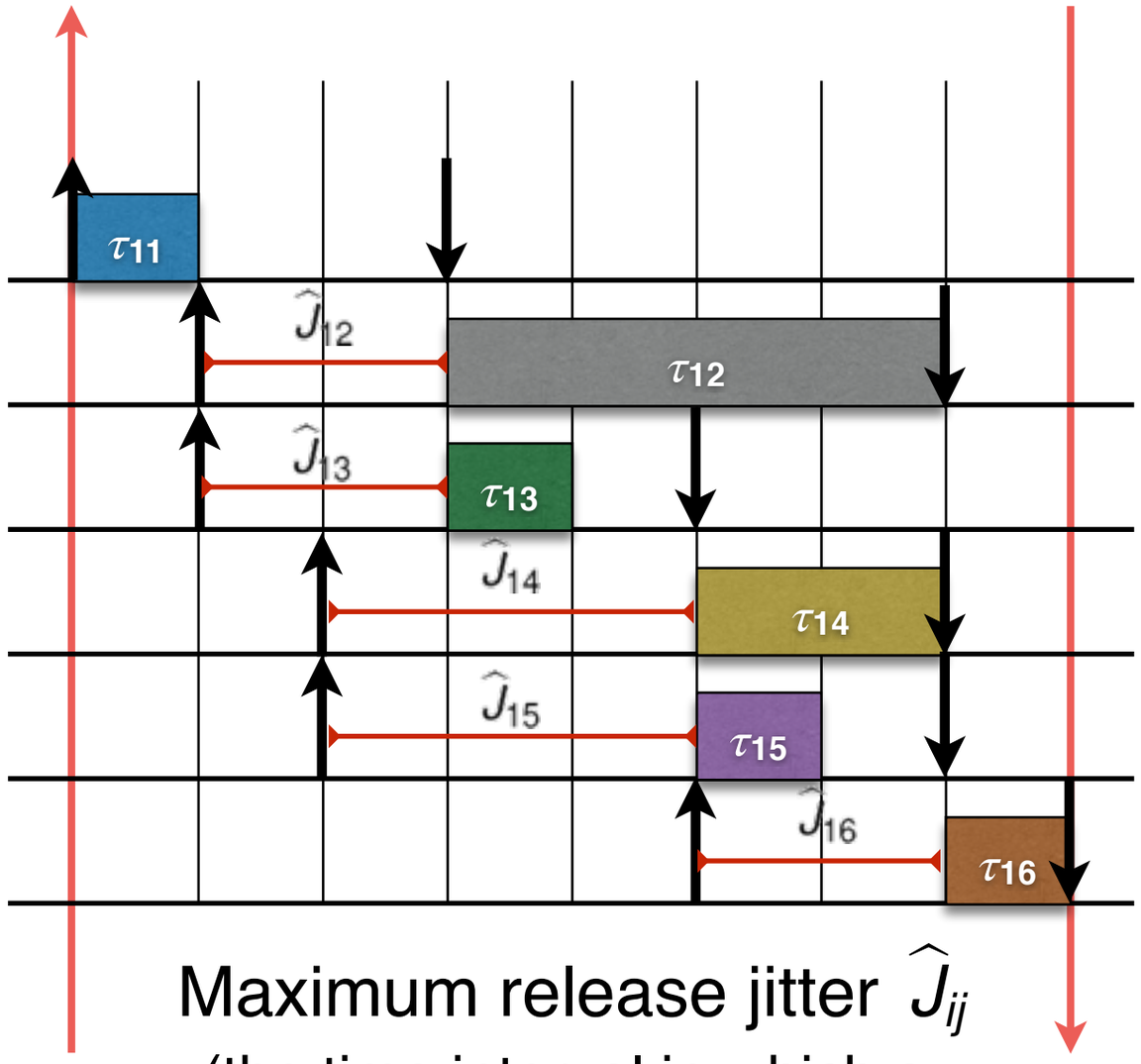
Local subtask Deadline D_{ij}

DAG Scheduling on a Subtask Level

Local Parameters



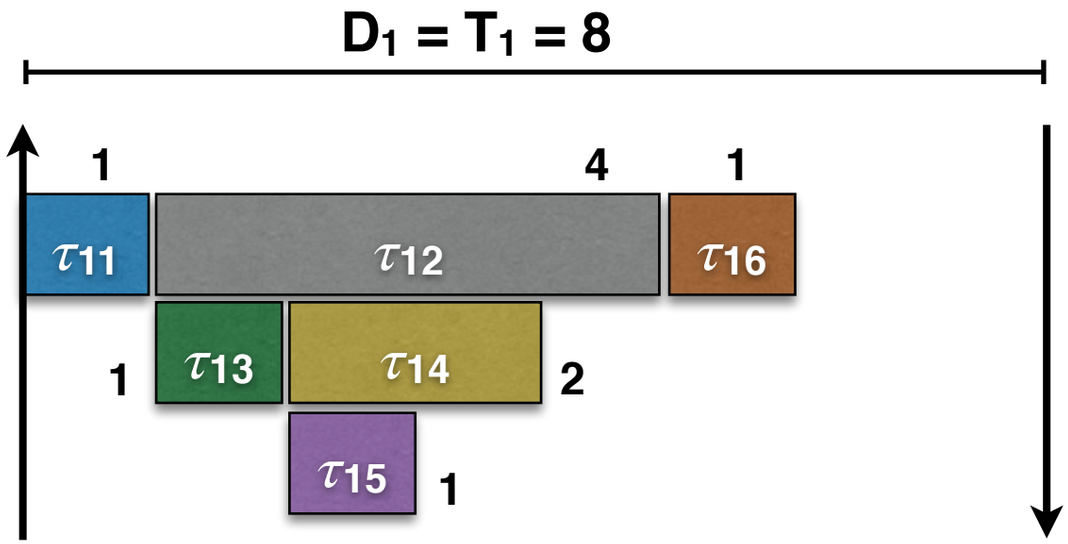
The **maximum release jitter** of each subtask which is its **maximum slack time** due to internal interference only.



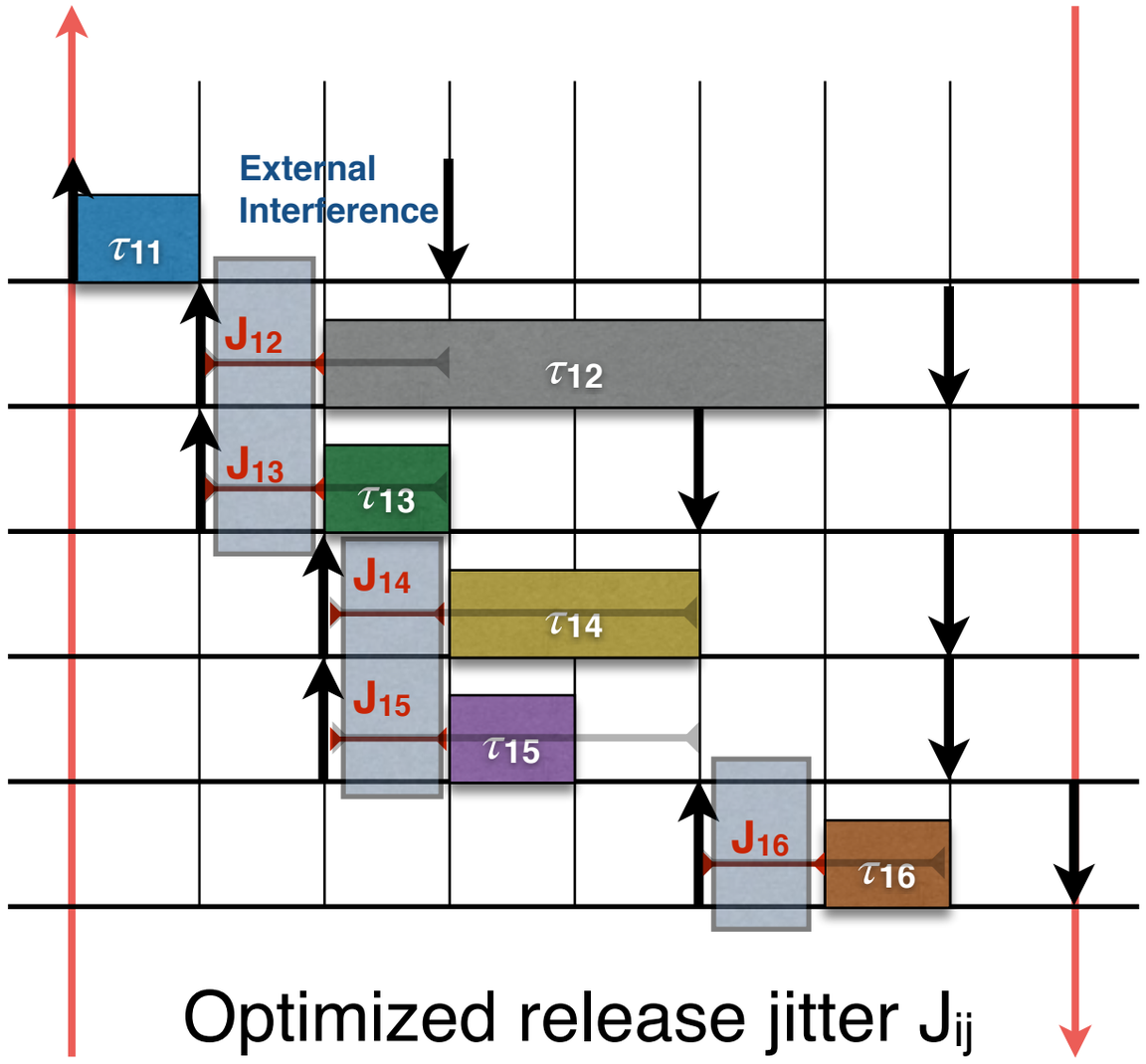
Maximum release jitter $\widehat{J}_{ij}$
(the time interval in which a subtask can be activated)

DAG Scheduling on a Subtask Level

Local Parameters



Using both internal and external interference analyses, we can **optimize** the **release jitter** of each subtask based on their **latest finish time**.



Optimized release jitter J_{ij}

Advantage of Subtask-level Scheduling

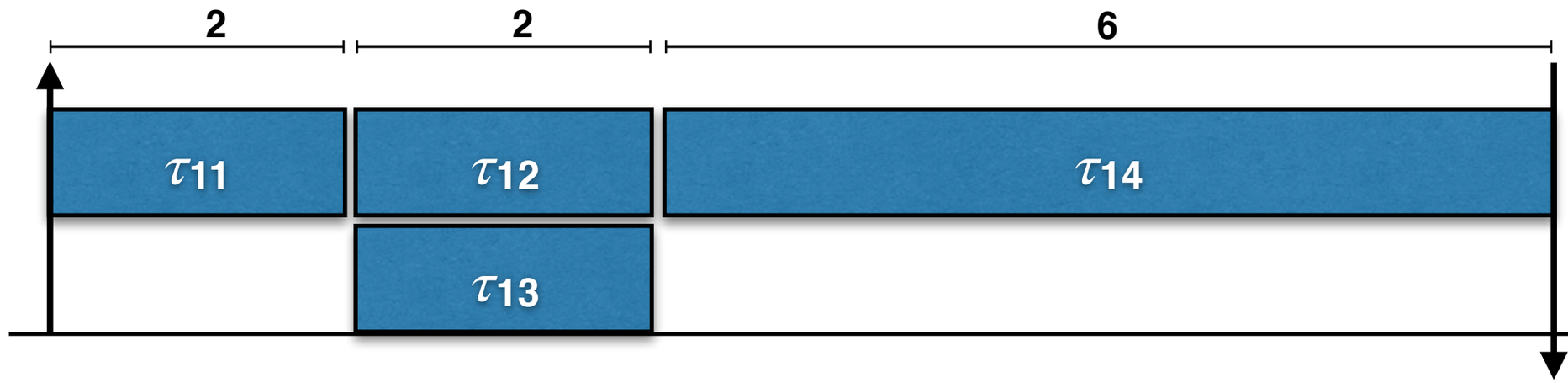
- **Demand bound function *h(t)*** is the maximum amount of task execution that can be released in an interval [0,t) and has to complete in this interval.
  - DAG-level : $$h(t) = \sum_{i=1}^{n} max\left(0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) C_i$$
  - Subtask-level : $$h(t) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} max\left(0, \left\lfloor \frac{t - D_{i,j} - O_{i,j}}{T_{i,j}} \right\rfloor + 1\right)$$

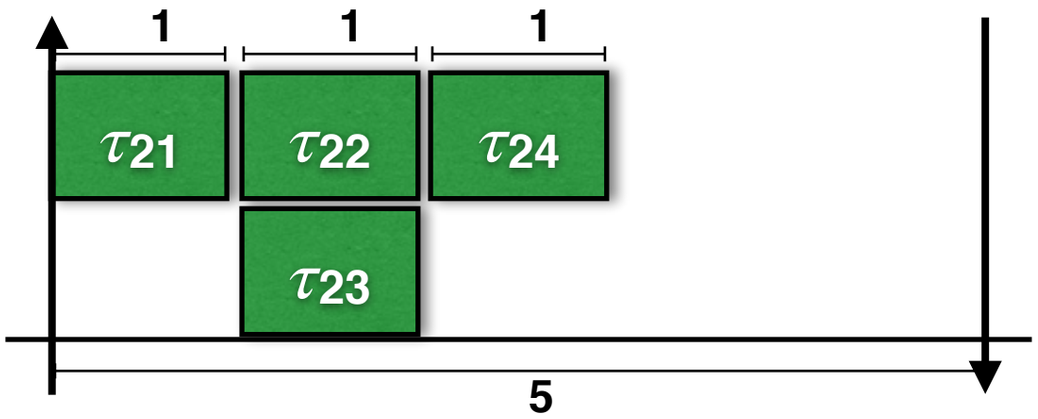
- A necessary feasibility condition of multiprocessor task sets :

$$\mathrm{load}(\tau) = max_{\forall t}\left(\frac{h(t)}{t}\right) \leq m$$

Advantage of Subtask-level Scheduling

- Example : set of 2 periodic implicit-deadline DAGs execute on 2 identical processors using global EDF.





$\tau_{11} = (0,2,2,10)$ $\tau_{21} = (0,1,3,5)$

$\tau_{12} = (2,2,2,10)$ $\tau_{22} = (1,1,3,5)$

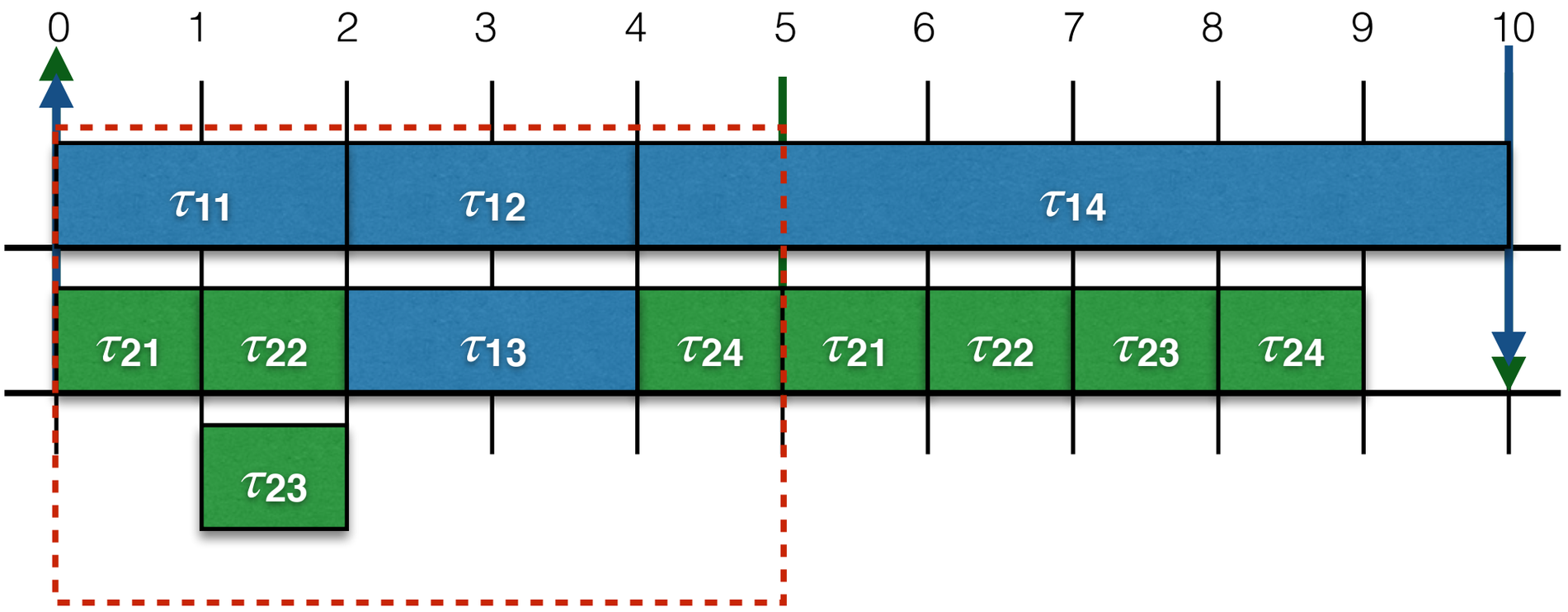
$\tau_{13} = (2,2,2,10)$ $\tau_{23} = (1,1,3,5)$

$\tau_{14} = (4,6,6,10)$ $\tau_{24} = (2,1,3,5)$

Necessary feasibility condition is OK. $\sum_{\tau_i \in \tau} u_i \leq m$

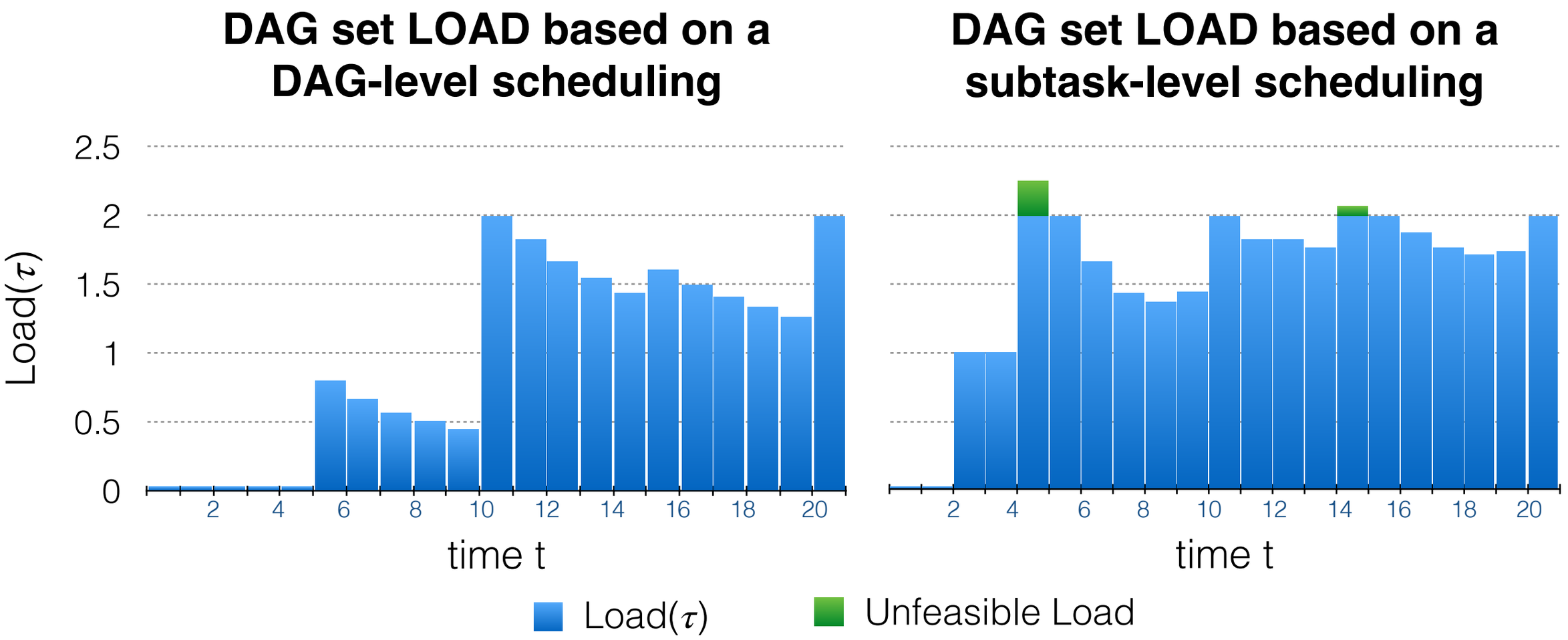
Advantage of Subtask-level Scheduling

- DAG set is **not feasible** on a system of 2 processors.



Both DAG tasks need **11 execution time units** in a **time interval equal to 5**

Advantage of Subtask-level Scheduling



The load analysis is **more accurate** on a subtask-level than on a DAG-level.

Interference Analysis of DAG tasks

- Scheduling interference on a subtask in a DAG task is due to:
  - Internal structure: because of interference from predecessor and sibling subtasks.
  - External structure: because of interference from other higher priority subtasks in the set.

Corollary 1. *A taskset τ of DAG tasks is schedulable on m identical processors, using any work conserving algorithm, if:*

$$\forall \tau_{k,h} \in \tau_k \in \tau$$

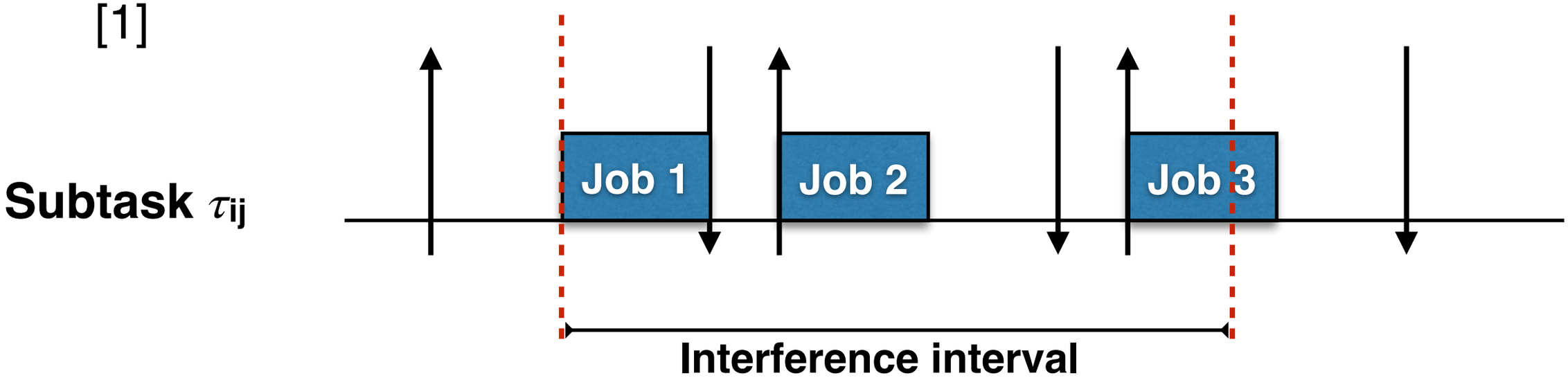
$$\widehat{I}e_{k,h} + \widehat{I}i_{k,h} \leq (D_{k,h} - C_{k,h} - j'_{k,h}) \quad (8)$$

where

$$\widehat{I}i_{k,h} = \frac{1}{m} * \sum_{\forall \tau_{k,i} \in sibling(\tau_{k,h})} I^{k,i}_{k,h}(a,b) \quad (9)$$

Workload Analysis for DAG tasks

- The **interference** of a task on another in an interval **cannot be greater than its maximum workload** within the same interval.
- For any work conserving algorithm, the maximum possible workload happens due to the worst case activation scenario of interfering jobs. [1]



- In the case of subtask-level scheduling of DAGs, this scenario is applied on each subtask to calculate maximum workload.

[1] Bertogna et al. (Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms). *TPDS'09*

Workload Analysis for DAG tasks

- **Contribution:**
  We provide the following schedulability test for DAG scheduling on a subtask-level.

Lemma 4. *The external interference* $\widehat{Ie}_{k,h}^{i,j}$ *of subtask* $\tau_{i,j}$ *on subtask* $\tau_{k,h}$ *in an interval, whose length is equal to the absolute deadline* $D_{k,h}$ *of* $\tau_{k,h}$*, is bounded by:*

$$\widehat{Ie}_{k,h}^{i,j} \leq N_{i,j}(D_{k,h})C_{i,j} + min(C_{i,j}, D_{k,h} + D_{i,j} - C_{i,j} - N_{i,j}(D_{k,h})T_{i,j}) \quad (12)$$

where

$$N_{i,j}(D_{k,h}) = \lfloor \frac{D_{k,h} + D_{i,j} - C_{i,j}}{T_{i,j}} \rfloor$$

Experimental Analysis

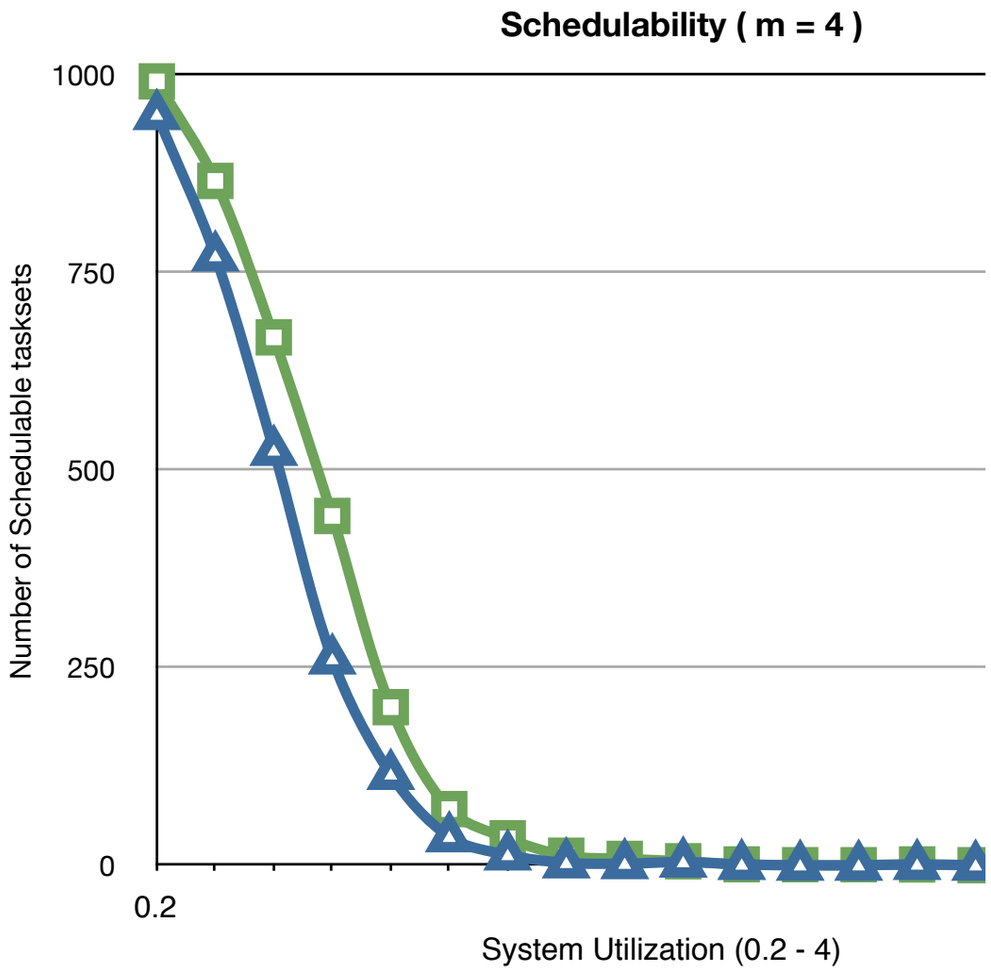
- Our workload schedulability test is compared with the **schedulability test (BMS)** proposed in [1] for DAG scheduling on a DAG-level without considering the internal structure.
- BMS provided a GEDF schedulability test for DAG sets on homogeneous processors.

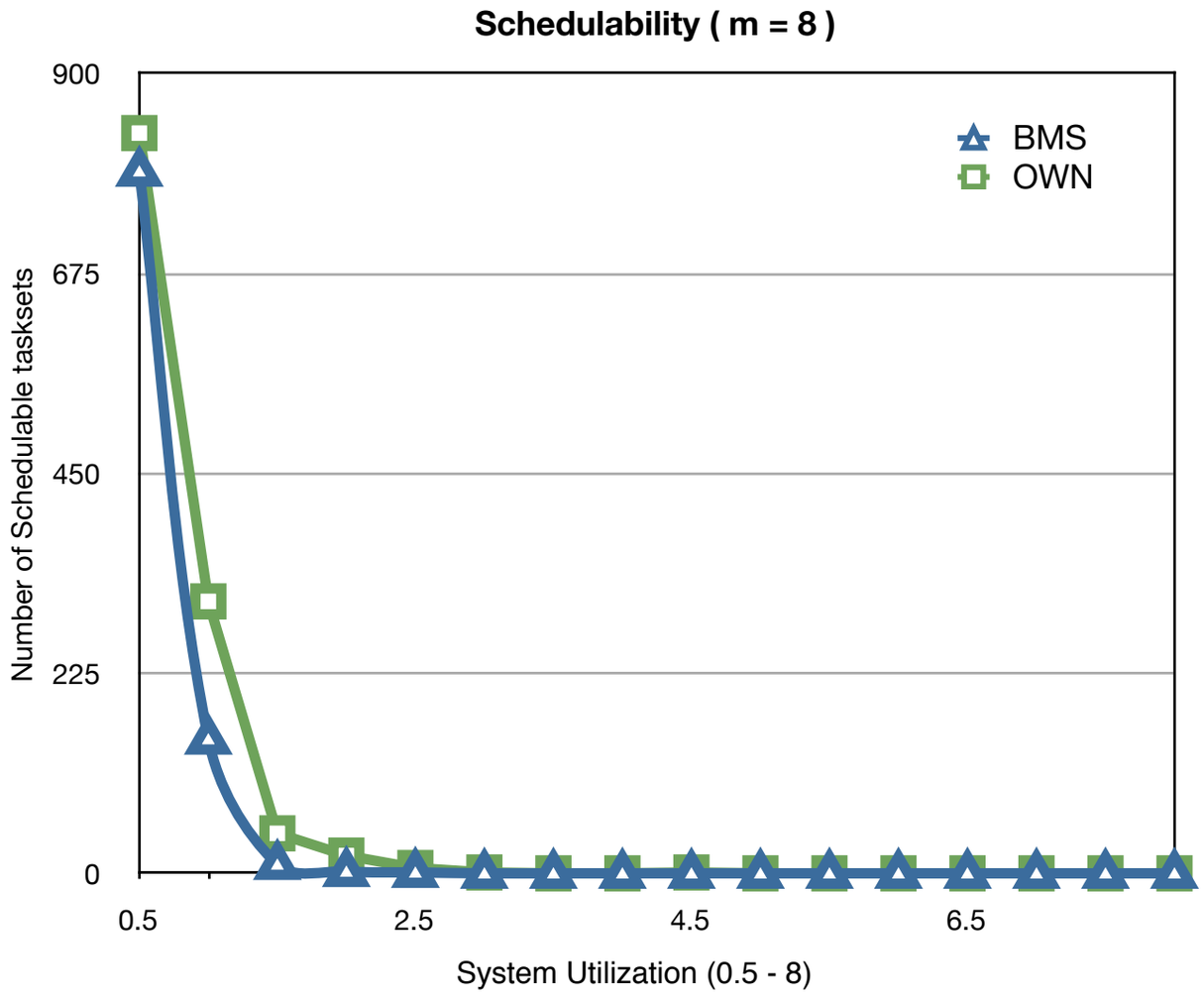
Simulation Environment

- **YARTISS** is a real-time multiprocessor simulator written in Java.
- Random generation of large DAG sets with utilization from 1 to 8.

[1] Bonifaci et al. (Feasibility Analysis in the Sporadic DAG Task Model). *ECRTS'13*

Experimental Analysis





The Our schedulability test (OWN) performs better than the BMS test on 4 and 8 processors.

Conclusion & Future Work

Conclusion

- We studied schedulability of DAG tasks on multiprocessor systems on a subtask level.
- We compared the scheduling of DAG tasks on a DAG and subtask level.
- We calculated Interference and Workload schedulability tests for DAG scheduling on a subtask level for any work conserving algorithm.

Perspective & Future Work

- Extend work to include other common scheduling algorithms.
- Provide performance metrics to evaluate the performance of the subtask level scheduling of DAGs.

Thank You

Questions?

manar.qamhieh@univ-paris-est.fr