EEE499 - Real-Time Embedded System Design

Modeling Real-Time Tasks

Reference Model for Real-time Systems

- A reference model and consistent terminology let us reason about real-time systems
- A reference model is characterized by:
  - A **workload model** that describes the applications supported by the system
  - A **resource model** that describes the system resources available to the applications
  - **Algorithms** that define how the application system uses the resources at all times

Timing Model

- We need to characterize jobs in order to schedule, manage and **reason about them with respect to time**

Periodic Tasks Timing Model

A set of jobs that are executed at regular time intervals can be modelled as a periodic task.

$$T_i = (\phi_i, p_i, e_i, D_i)$$

Task

Phase

Period

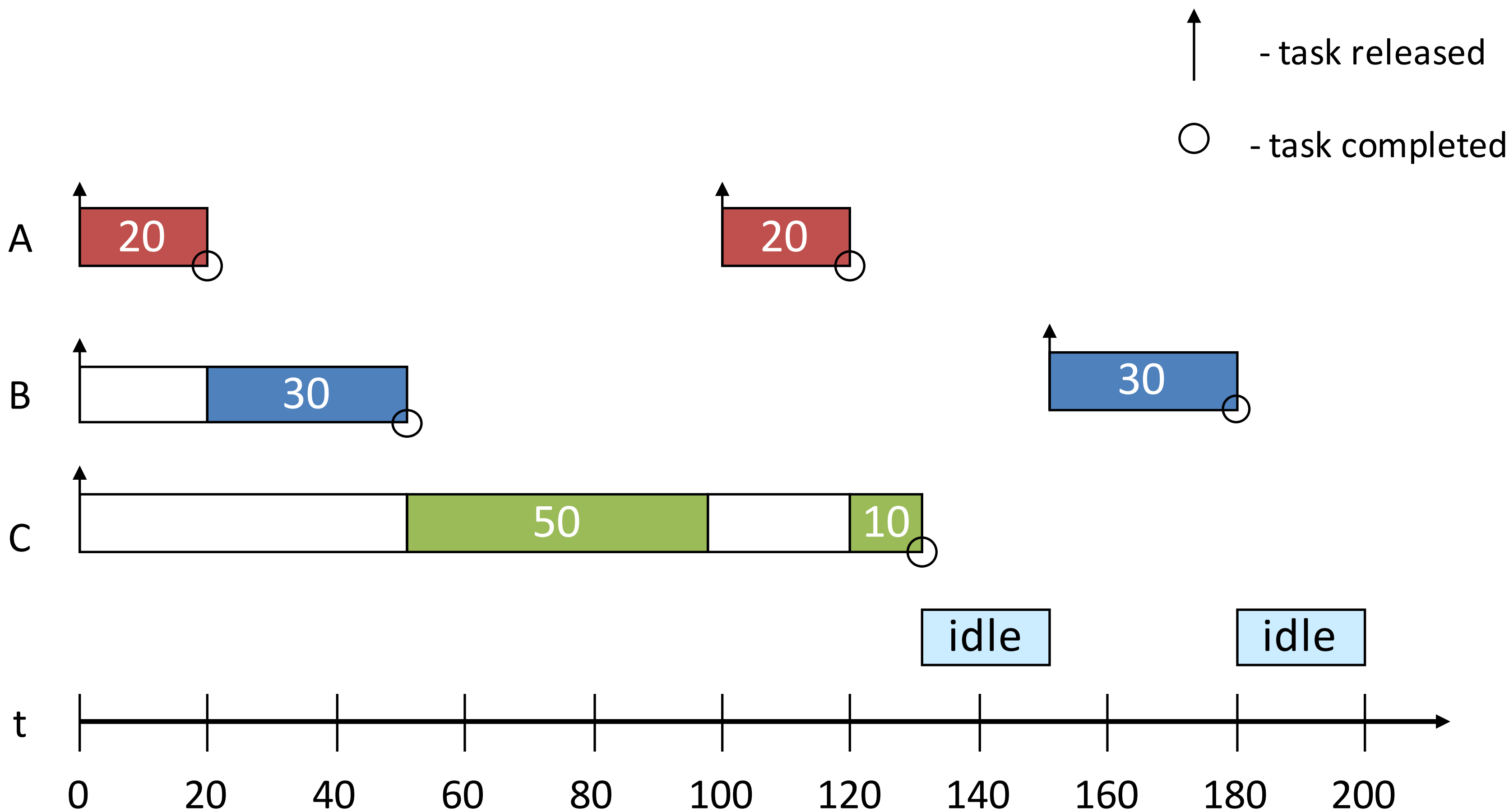
Execution Time

Relative Deadline

Periodic Tasks Timing Model

- The 4-tuple $T_i = (\phi_i, p_i, e_i, D_i)$ refers to a periodic task T_i with
  - **phase ϕ_i**, is the release time $r_{i,1}$ of the first job $J_{i,1}$ in the task.
  - **period p_i**, is the minimum length of all time intervals between release times of consecutive jobs
  - **execution time e_i**, is the maximum execution time of all jobs in the periodic task
  - and **relative deadline D_i**
  - Default phase of T_i is $\phi_i = 0$,
  - Default relative deadline is the period $D_i = p_i$

Periodic Tasks Timing Model



Precedence Constraints and Dependencies

- Jobs in a task may be constrained to execute in a particular order
  - Known as a precedence constraint
  - j_i is a predecessor to another job j_k if j_k cannot begin execution until j_i completes its execution
    - Denoted by $j_i < j_k$
    - j_i is an immediate predecessor of j_k if there are no job such that $j_i < j_j < j_k$
    - j_i and j_k are independent when neither $j_i < j_k$ nor $j_k < j_i$
- A job with a precedence constraint **becomes ready** for execution when its release time has passed and all predecessors have completed

Release Times

- The release time of a task is the time at which a task becomes available for execution
- Why *becomes* available?
- What if all tasks are always available for execution?

Release Time

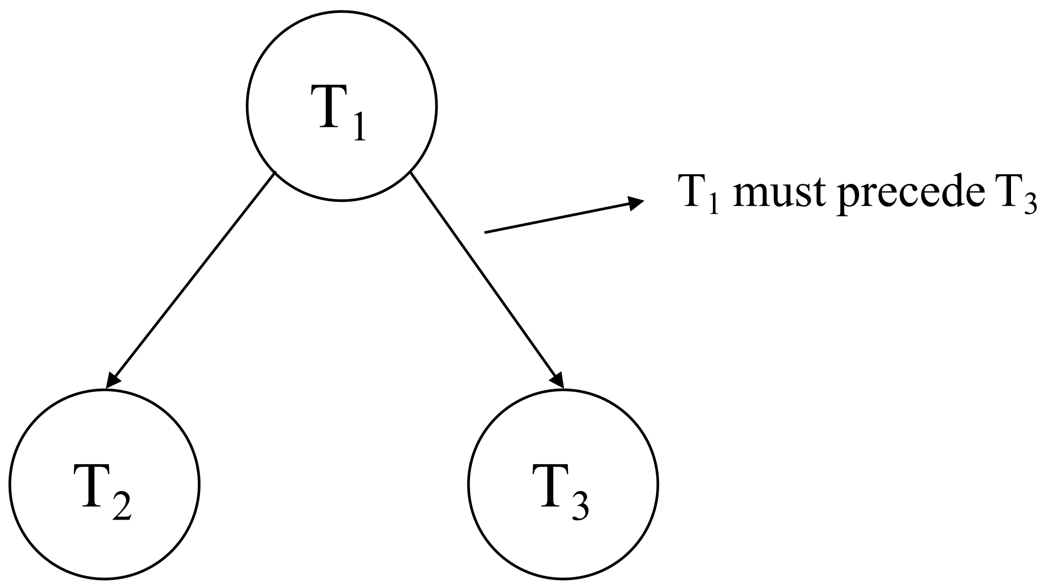
- We do not know exactly when the job will be released
  - Release time jitter due to
    - the granularity of clock tick (RTOS)
    - ISR time (depends on number of tasks and resources… why?)
- Even after it is ready to be executed, a task may still suffer **interference** from higher priority tasks
- And other interrupts…

Release Time

- So we will have a release time variation
- $[r_i^-, r_i^+]$
- r_i^- is the "earliest release time"
- r_i^+ is the "latest release time"
- For periodic tasks, r_i is the phase ϕ_i *of the task*

Release Time

- But what happens if we specify a release time that is not **effective**?



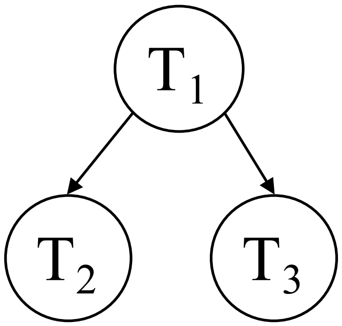
$\underline{T_i(\varphi_i, p_i, e_i, D_i)}$

$T_1(8,10,2,10)$

$T_2(7,15,3,15)$

$T_3(5,12,1,12)$

Release Time

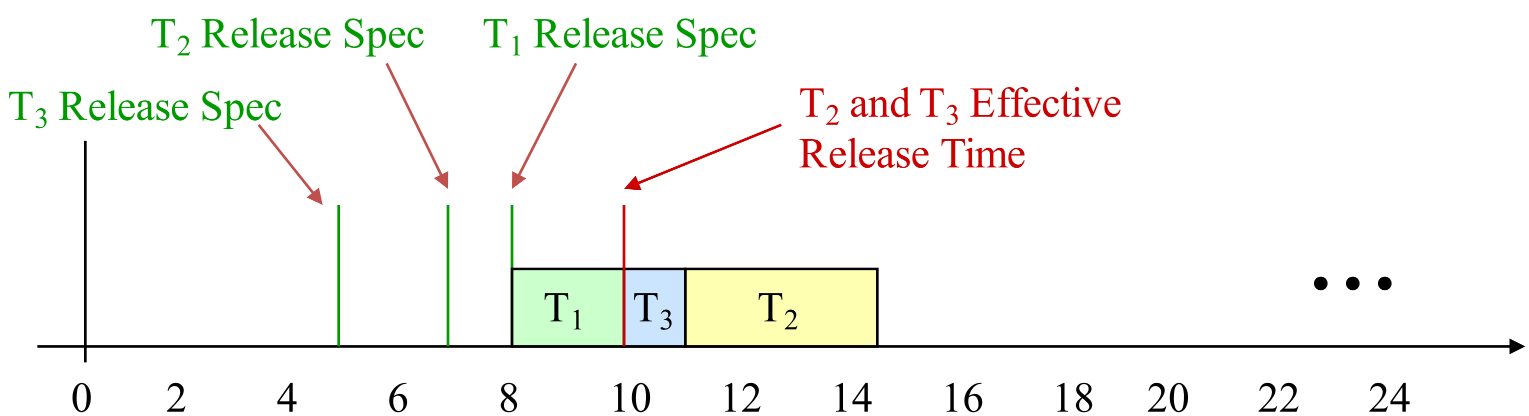


$\underline{T_i(\varphi_i, p_i, e_i, D_i)}$

$T_1(8,10,2,16)$

$T_2(7,15,3,15)$

$T_3(5,12,1,12)$



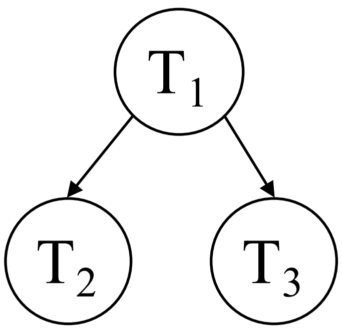
Effective Release Time

- So the effective release time for task T_i is the maximum of its specified release time and the maximum effective release time of any of its predecessors + their execution times
  - Obviously this is recursive
  - We have not yet considered mutual exclusion constraints, just precedence due to data and control requirements

Deadline

- The deadline of a task is the time at which a job must complete execution (within the clock tic)
- A deadline can be specified in two ways:
  – **Absolute Deadline**: Release Time plus a relative deadline
    - Limits our ability to reason about schedulability
  – **Relative Deadline**: Maximum Allowable Response-Time
    - Includes interference and blocking from other tasks… More to come.

Deadline

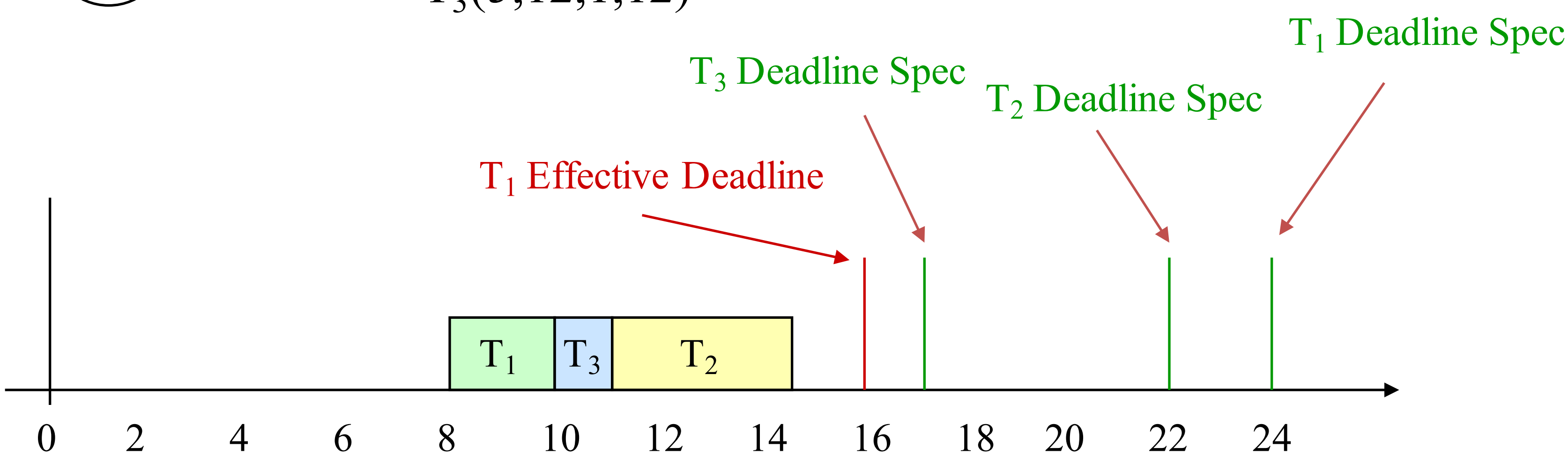


$\underline{T_i(\varphi_i, p_i, e_i, D_i)}$

$T_1(8,16,2,16)$

$T_2(7,15,3,15)$

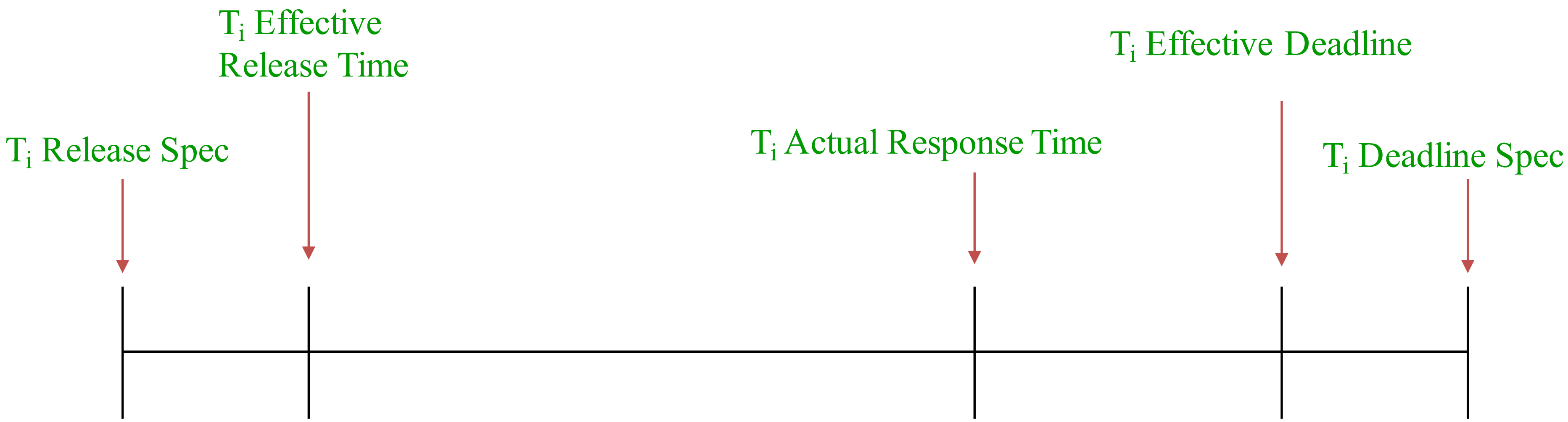
$T_3(5,12,1,12)$



Effective Deadline

- In the same sense as for release times, the effective deadline is calculated from the precedence constraints.
  - More precisely from the successor constraints
- Effective deadline of a task is the minimum of the specified deadline for the task and the minimum effective deadline of all its successors minus their execution time

Life of a task



Processors and Resources

- Recall that a job executes on a special resource which we call a **processor**
- The job may also depend on some **resources**
- A processor, *P,* is an active component on which jobs are scheduled: i.e.
  - Threads scheduled on a CPU
  - Data scheduled on a transmission link
  - Read/write requests scheduled to a disk
  - Transactions scheduled on a database server

Processors and Resources

- Processor (continued)
  - Each processor has a speed attribute that determines the rate of progress of a job
  - Two processors are of the same **type** if they are functionally identical and can be used interchangeably.
    - What would make two processors **heterogeneous**?

Processors and Resources

- A resource, *R,* is a passive entity upon which jobs may depend: i.e.
  - Memory, sequence numbers, mutexes, database locks, mailboxes,…
  - Resources have different **types** and **sizes**, but do not have a speed attribute
  - Resources are usually **reusable**, and are not consumed by use

Use of Resources

- If a system contains *n* types of resources it means:
  - There are *n* different types of **serially reusable** resources
  - There are one or more units of each type of resources, only one job can use each unit at once (Mutually exclusive access)
  - A job must obtain a unit of a needed resources, use it and release it

Execution Time

- Perhaps one of the most difficult number to estimate is that of execution time.
- The execution time for job J_i varies in the interval $[e_i^-, e_i^+]$ the interval depends on:
  - Conditional branches
  - Iterations in unbounded loops
  - Caches
- Without loss in generality we can have $\boldsymbol{e_i = e_i^+}$
  - The execution time for the job is therefore the **maximum execution time** ignoring the interval and lower bound

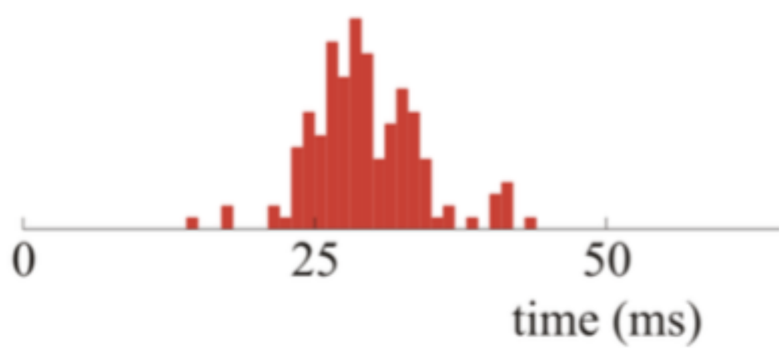
Execution Time

- Needed to determine if deadlines can be met.
- What time do we choose?
  - The one from the last execution?
  - The one from the longest execution?
  - The average?
- How do we determine the execution time?

Execution Time

How do we determine the execution time?

1. Analysis of the source code
2. Estimation from empirical evidence



[2]

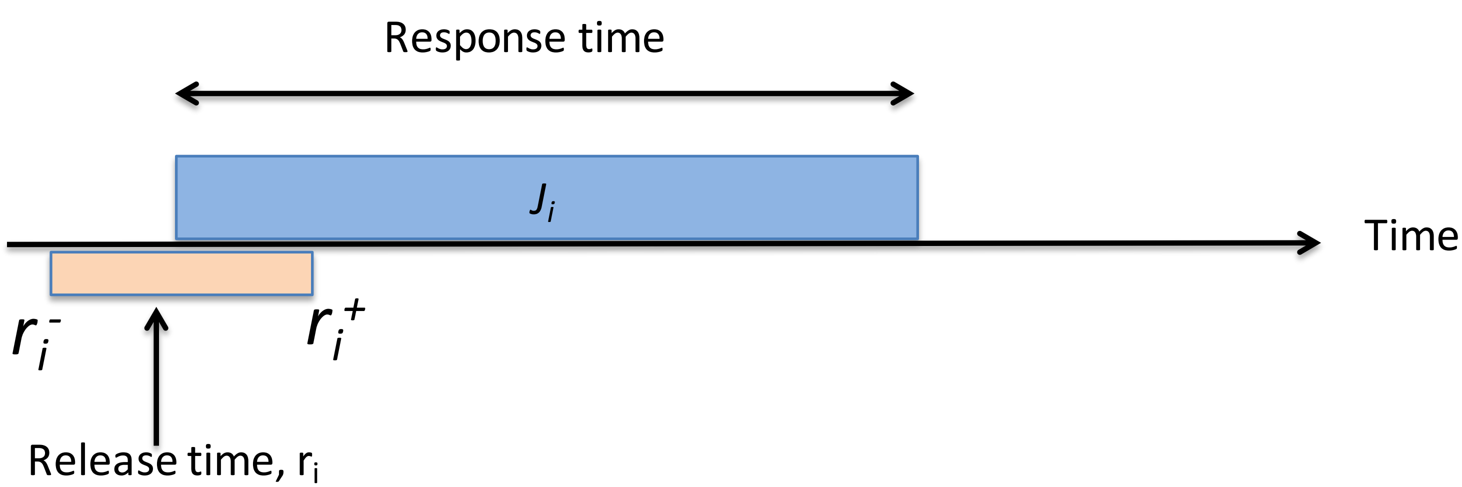
Release Time

- So we will have a release time variation
- $[r_i^-, r_i^+]$
- r_i^- is the "earliest release time"
- r_i^+ is the "latest release time"
- For periodic tasks, r_i is the phase φ_i *of the task*

Release and Response Time

- Release Time – The instant in time when a job becomes available for execution
  - May not be exact: Release time jitter in the interval $[r_i^-, r_i^+]$
  - A job can be scheduled and executed at any time at or after its release time provided its resource dependency conditions are met
    - Including precedence constraints
- Response Time – the length of time from the release time of the job to the time instant when it completes
    - Not the same as execution time!

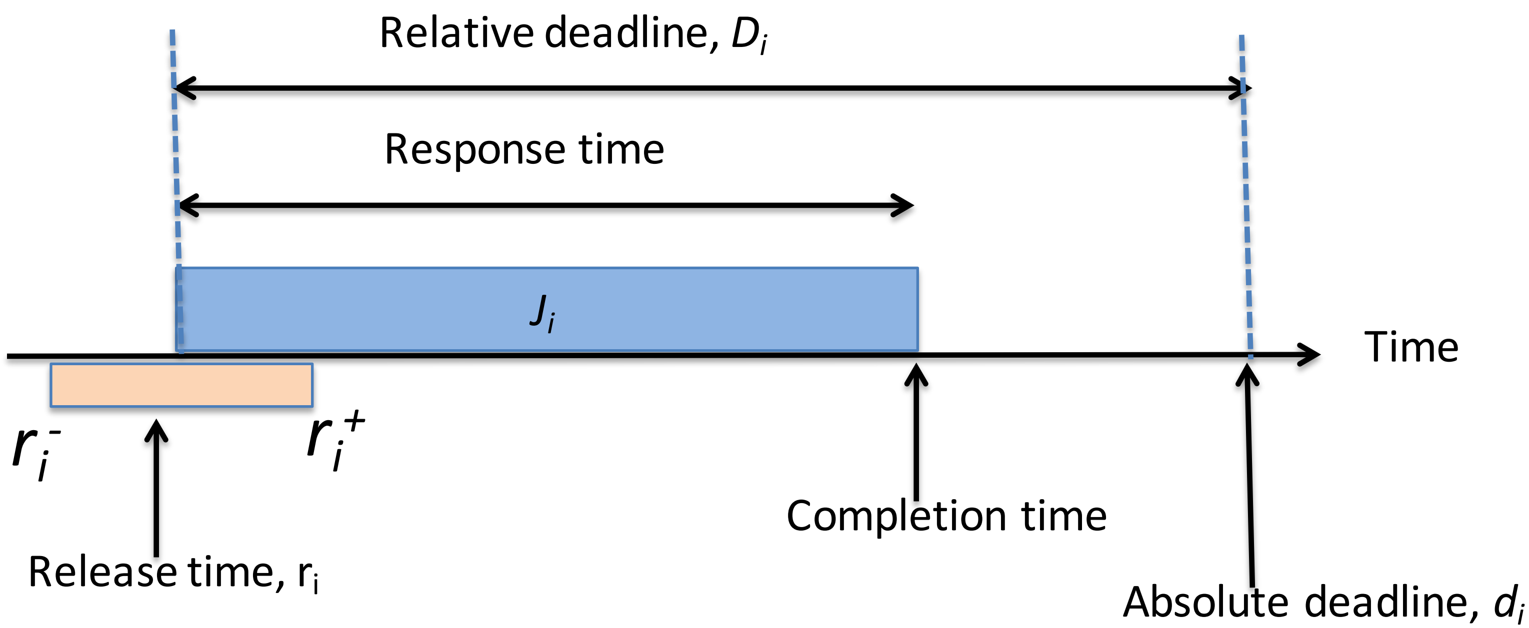
Response Time



Deadlines and Timing Constraints

- Completion Time – the instant at which a job completes execution
- Relative deadline – the maximum allowable job response time (Schedulability analysis)
- Absolute deadline – the instant of time by which a job is required to be completed
  - Absolute deadline = release time + relative deadline
  - Feasible interval for job J_i is the interval $(r_i, d_i]$

Response Time

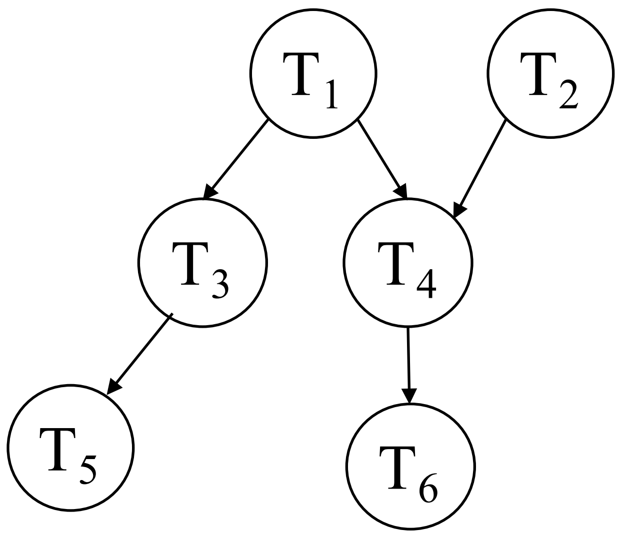


Aperiodic and Sporadic Jobs

- Many real-time systems are required to respond to unpredictable events.
- These are modelled as aperiodic or sporadic jobs
  - An aperiodic job has unpredictable release times
  - A sporadic job is a aperiodic job that has a hard deadline.
- Aperiodic jobs are always accepted.
- Sporadic jobs make the design of a hard real-time system impossible,
  - unless some bounds can be placed on their inter-arrival times. Note that without a minimum inter-arrival time restriction, it is impossible to guarantee that a deadline of a sporadic task would always be met.
  - Based on the execution time and deadline of each newly arrived sporadic job, decide whether to accept or reject the job.

Exercise 1

- Calculate the effective release time for the following tasks.



$T_i(\varphi_i, p_i, e_i, D_i)$

$T_1(8,10,2,16)$

$T_2(10,15,3,15)$

$T_3(5,12,1,12)$

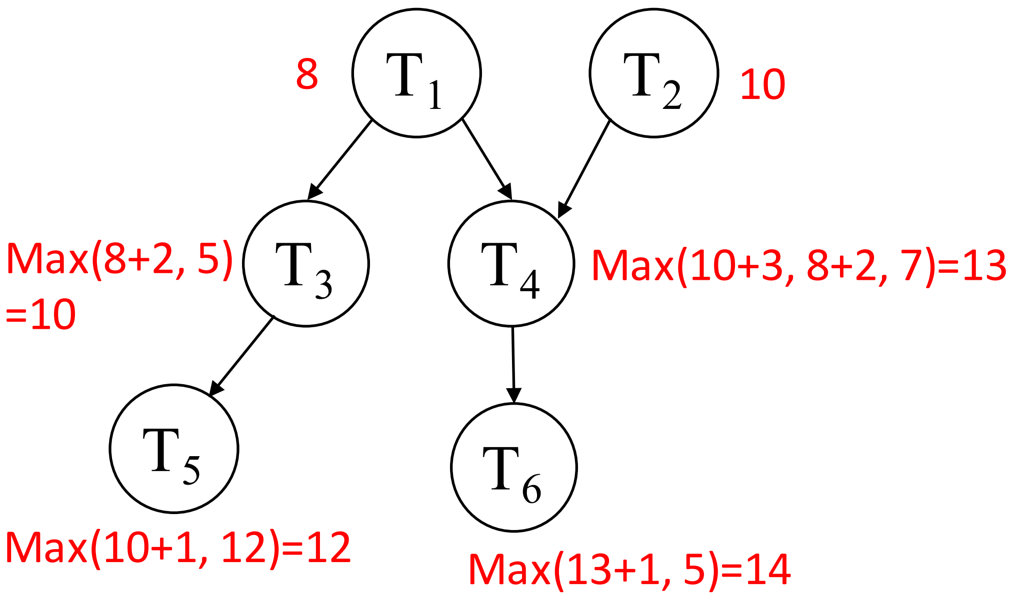
$T_4(7,12,1,12)$

$T_5(12,12,1,12)$

$T_6(5,12,1,12)$

Exercise 1

- Calculate the effective release time for the following tasks.



$\underline{T_i(\varphi_i, p_i, e_i, D_i)}$

$T_1(8,10,2,16)$

$T_2(10,15,3,15)$

$T_3(5,12,1,12)$

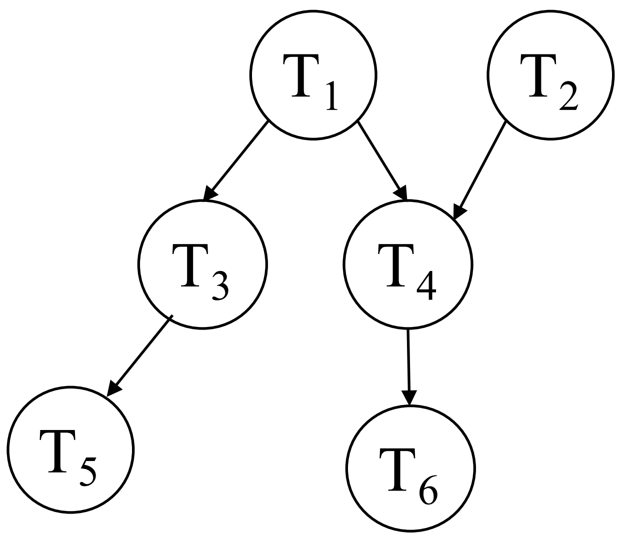
$T_4(7,12,1,12)$

$T_5(12,12,1,12)$

$T_6(5,12,1,12)$

Exercise 2

- Calculate the effective deadline for the following tasks.



$\underline{T_i(\varphi_i, p_i, e_i, D_i)}$

$T_1(8,10,2,16)$

$T_2(10,15,3,15)$

$T_3(5,12,1,13)$

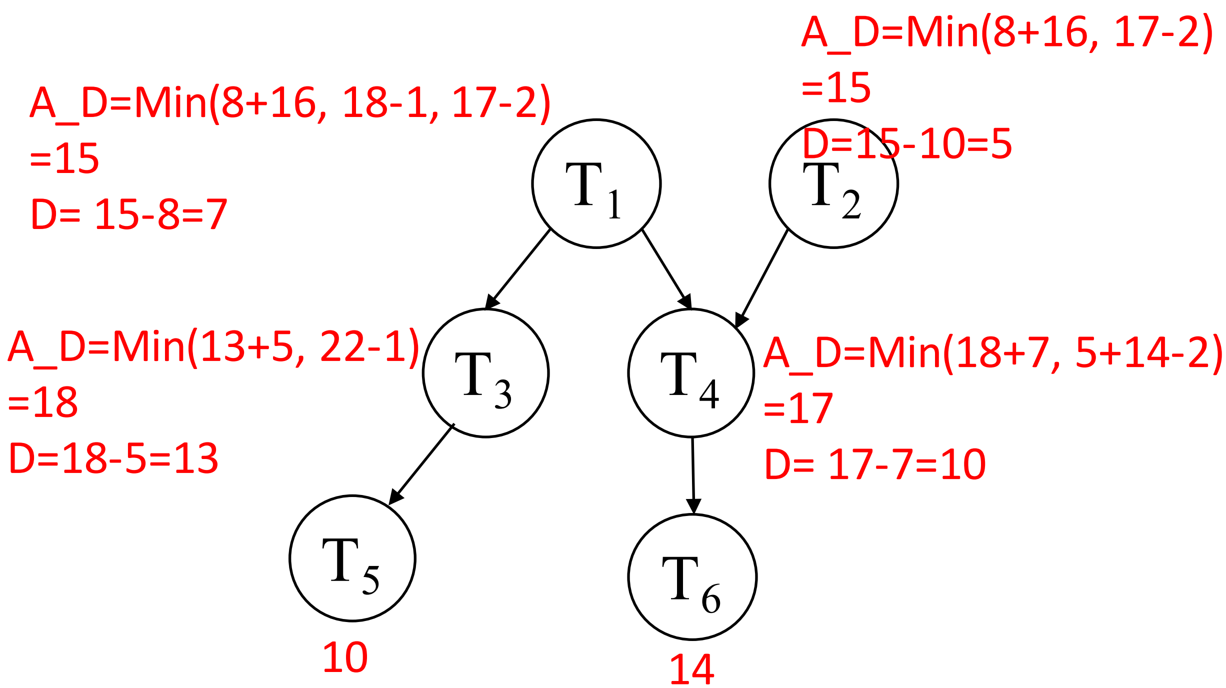
$T_4(7,12,2,18)$

$T_5(12,12,1,10)$

$T_6(5,12,2,14)$

Exercise 2

- Calculate the effective deadline for the following tasks.



$\underline{T_i(\varphi_i, p_i, e_i, D_i)}$

$T_1(8,10,2,16)$

$T_2(10,15,3,15)$

$T_3(5,12,1,13)$

$T_4(7,12,2,18)$

$T_5(12,12,1,10)$

$T_6(5,12,2,14)$

References

[1] Liu, J. W. S. Real-Time Systems. Prentice Hall, 2000.

[2] Harder, D. W., Zarnett, J., Montaghami, V., Giannikouris, A. A practical introduction to real-time systems for undergraduate engineering, Version 0.2017.05.01. University of Waterloo, 2017. [Available online]