EEE499 - Real-Time Embedded System Design

New tasks, precedence, multiprocessor scheduling and anomalies

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER
ENGINEERING

GÉNIE ÉLECTRIQUE
ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA

RMC CMR

Simple Task Model

- Assumptions:
  1. ~~Tasks are periodic~~ and the period is constant
  2. Completion-time < period
  3. ~~Tasks are independent~~
  4. Runtime is known and deterministic
  5. all system overheads are negligible or deemed to be included in task computation times
  6. **Critical instant - defined as the maximum load condition when all tasks release together**

- Constraints
  1. ~~Deadline = period~~
  2. **fixed set of tasks**
  3. Preemptive

Tasks Arrival

The algorithms we have seen so far do not allow new tasks to be added.

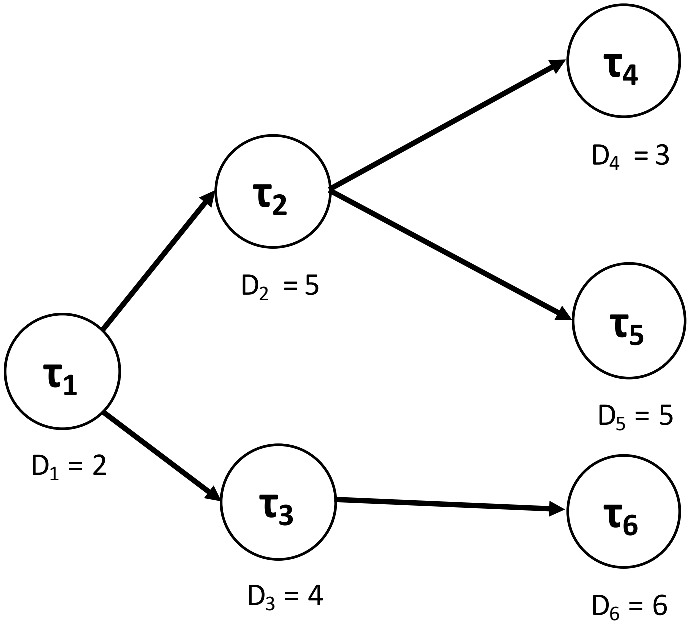
Horn’s algorithm (*Earliest Deadline First* or *EDF*) allows for the arrival of new tasks.

Earliest Deadline First (EDF)

- Dynamic priority ordering
- A task will not have the same priority at each execution.
- More complex implementation, but less preemption.

Precedence

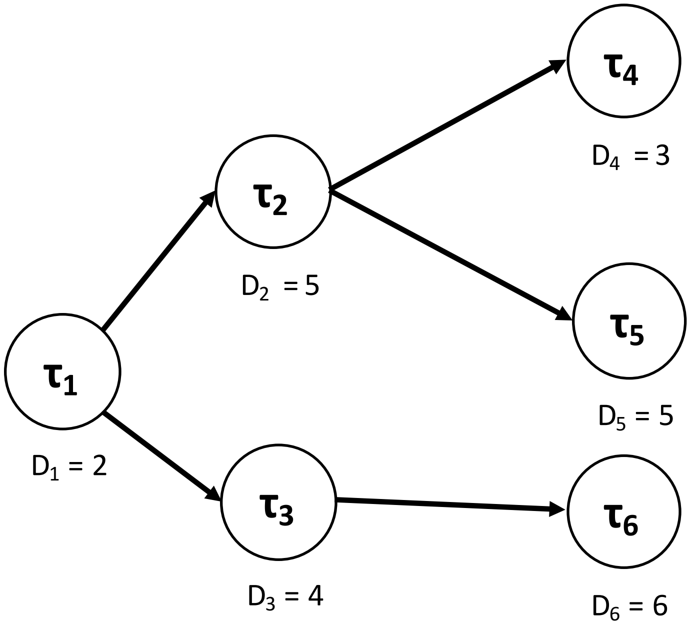
The EDF algorithm is not optimal when there are precedencies



τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	
4	1	3	
5	1	5	
6	1	6	

Precedence

The EDF algorithm is not optimal when there are precedences

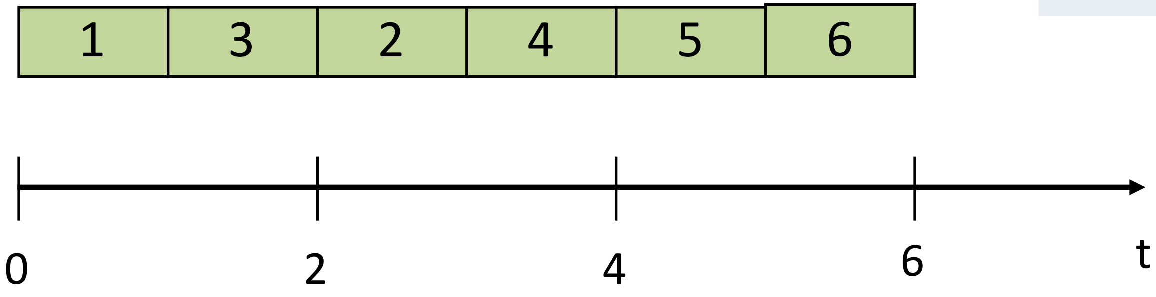


τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	3
3	1	4	2
4	1	3	4
5	1	5	5
6	1	6	6

Precedence

The EDF algorithm is not optimal when there are precedences

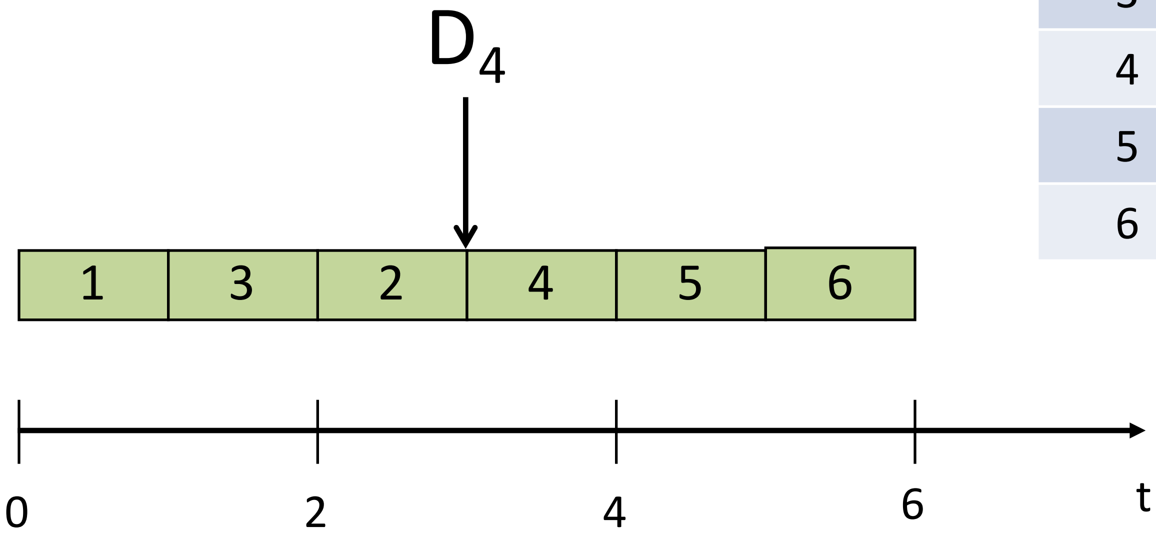
τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	3
3	1	4	2
4	1	3	4
5	1	5	5
6	1	6	6



Precedence

The EDF algorithm is not optimal when there are precedences

τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	3
3	1	4	2
4	1	3	4
5	1	5	5
6	1	6	6

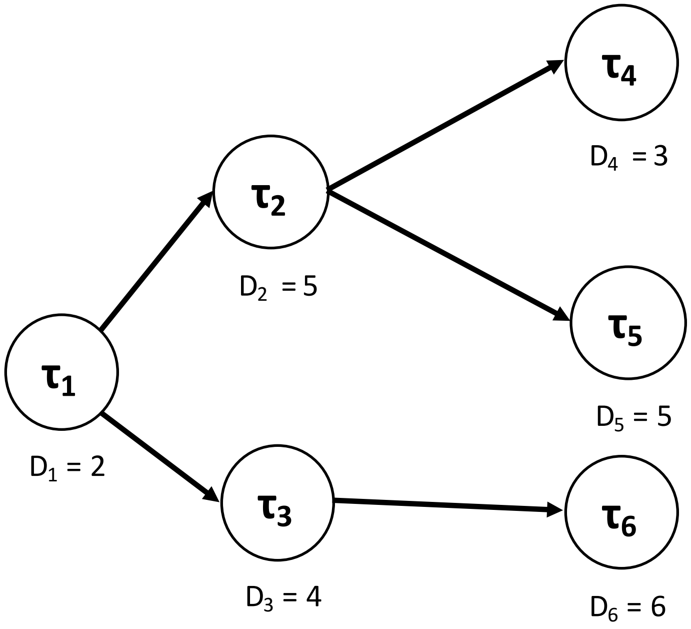


Precedence

The Lawler algorithm, called *Latest Deadline First* (LDF) allows to take precedence into account more effectively.

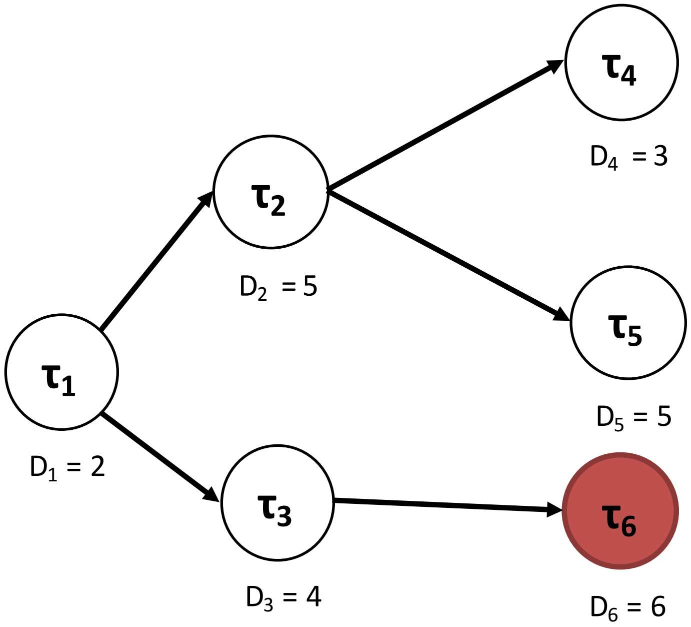
- – Builds the schedule from the end
- – Finite and well-known task set
- – Chooses the last task to execute (no dependents and latest deadline).
- – Continues until all tasks have been scheduled.

Latest Deadline First (LDF)



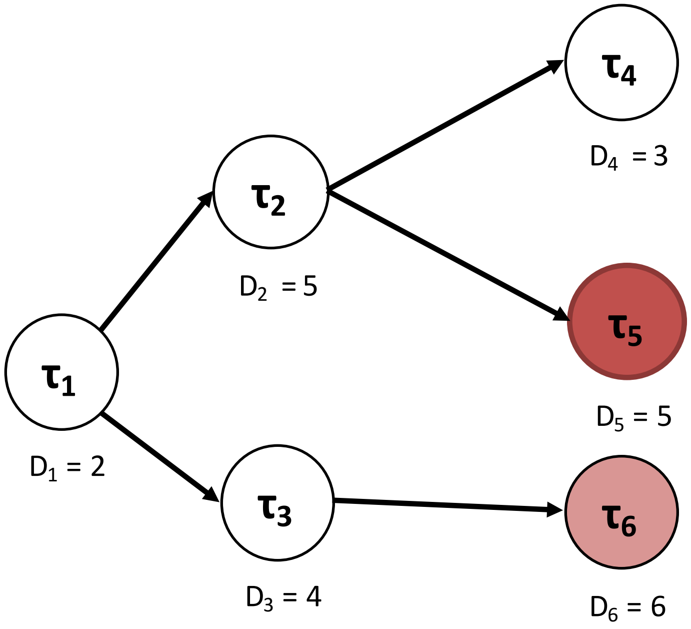
τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	
4	1	3	
5	1	5	
6	1	6	

Latest Deadline First (LDF)



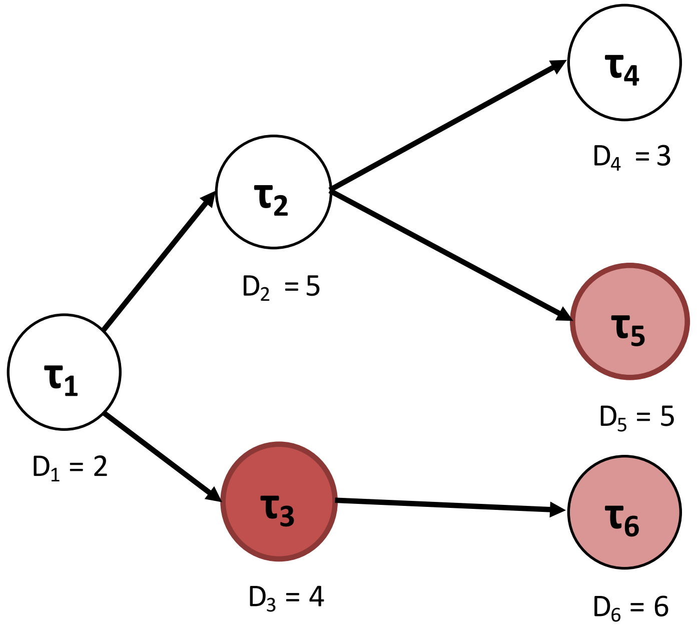
τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	
4	1	3	
5	1	5	
6	1	6	6

Latest Deadline First (LDF)



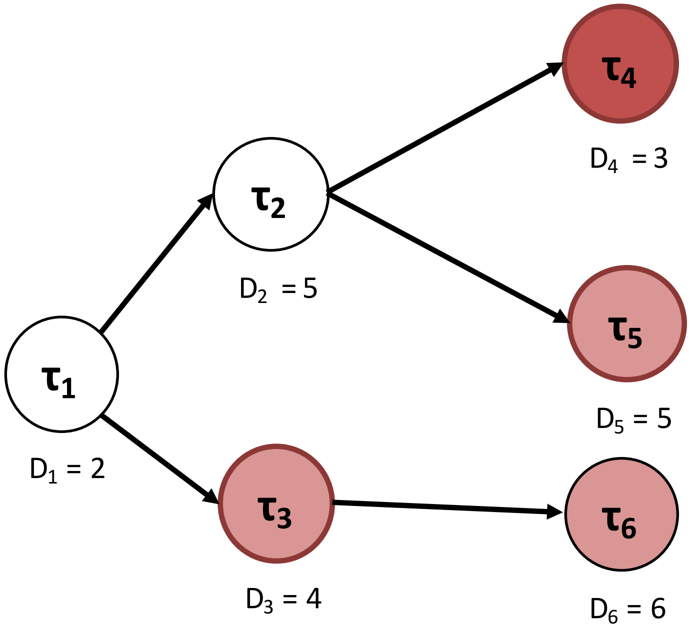
τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	
4	1	3	
5	1	5	5
6	1	6	6

Latest Deadline First (LDF)



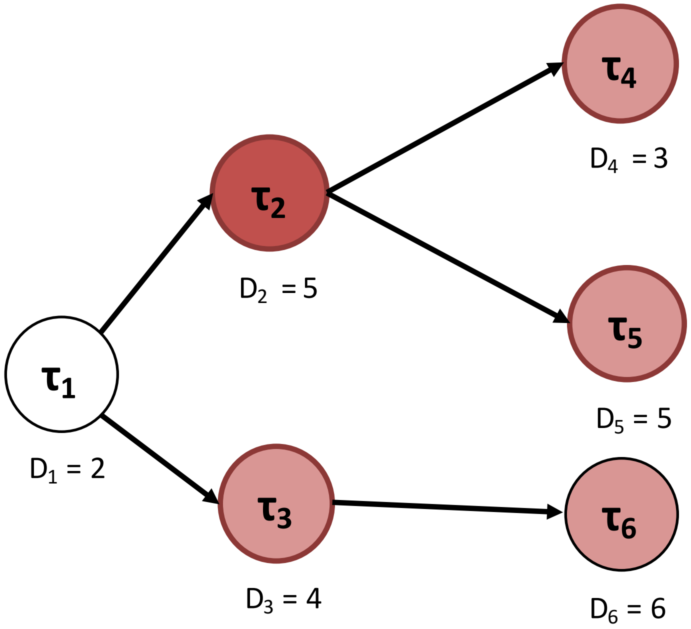
τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	4
4	1	3	
5	1	5	5
6	1	6	6

Latest Deadline First (LDF)



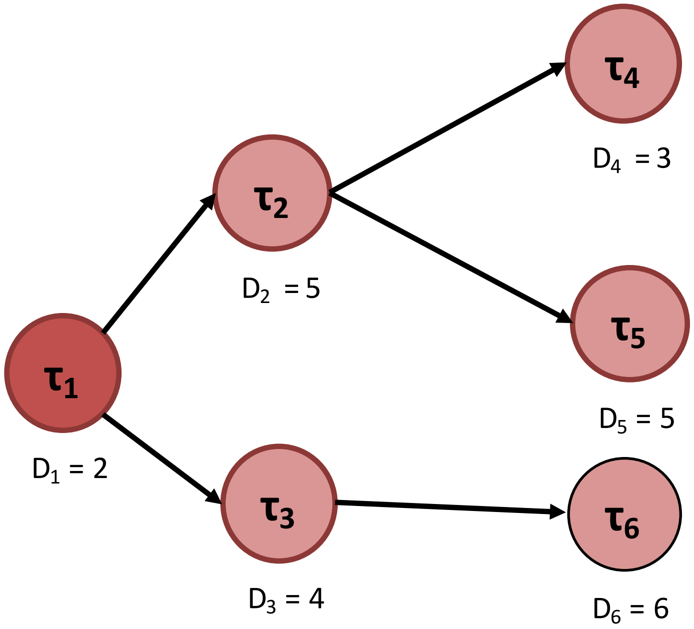
τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	4
4	1	3	3
5	1	5	5
6	1	6	6

Latest Deadline First (LDF)



τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	2
3	1	4	4
4	1	3	3
5	1	5	5
6	1	6	6

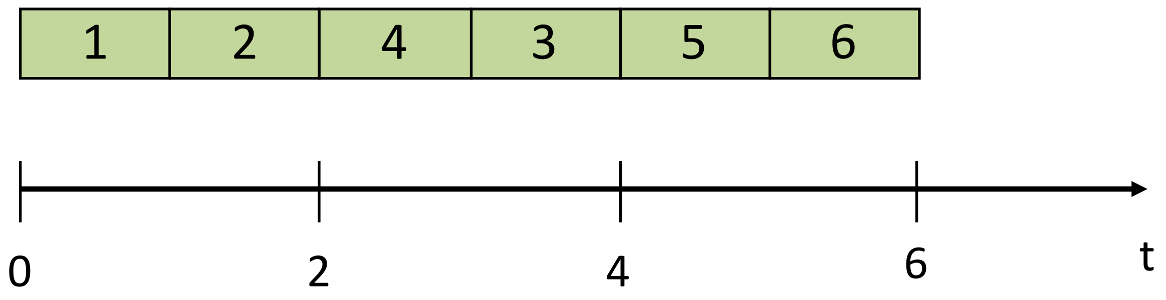
Latest Deadline First (LDF)



τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	2
3	1	4	4
4	1	3	3
5	1	5	5
6	1	6	6

Latest Deadline First (LDF)

τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	2
3	1	4	4
4	1	3	3
5	1	5	5
6	1	6	6

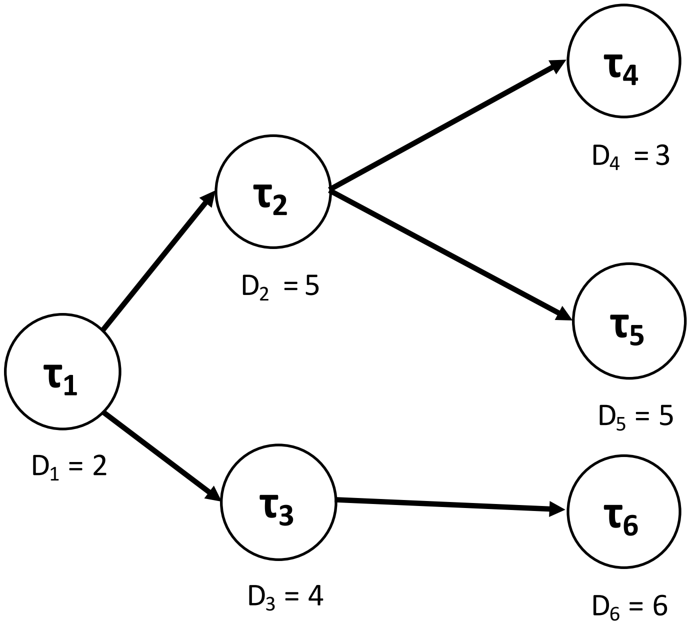


Earliest Deadline First with precedence (EDF*)

- A modification of LDF
- Based on a dynamic change of deadlines based on the dependencies.
- For a set of tasks T where $\tau_i \in \mathrm{T}$ and all the dependents of τ_i are in the subset $Q(i) \subset \mathrm{T}$
- For all execution of τ_i, a new deadline is computed

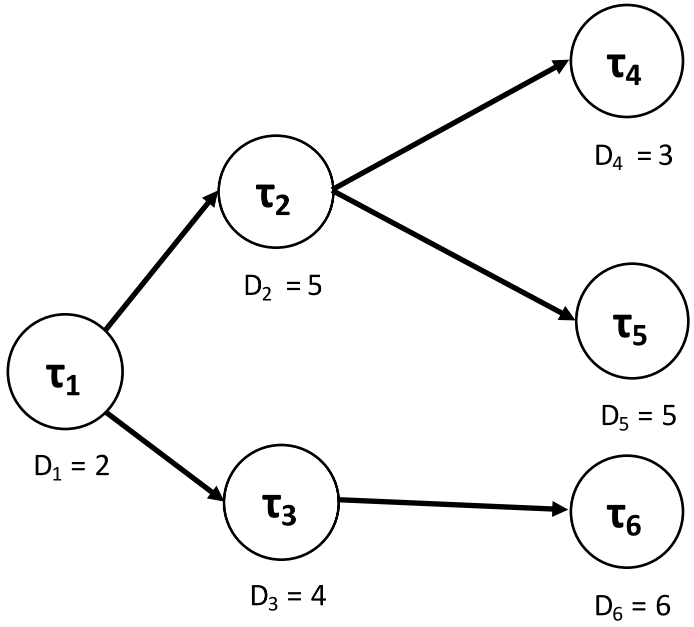
$$D_i^{'} = \min\left(d_i, \min_{j \in Q(i)}\left(D_j^{'} - e_j\right)\right)$$

Earliest Deadline First with precedence (EDF*)



τ_i	e_i	D_i	P_i
1	1	2	
2	1	5	
3	1	4	
4	1	3	
5	1	5	
6	1	6	

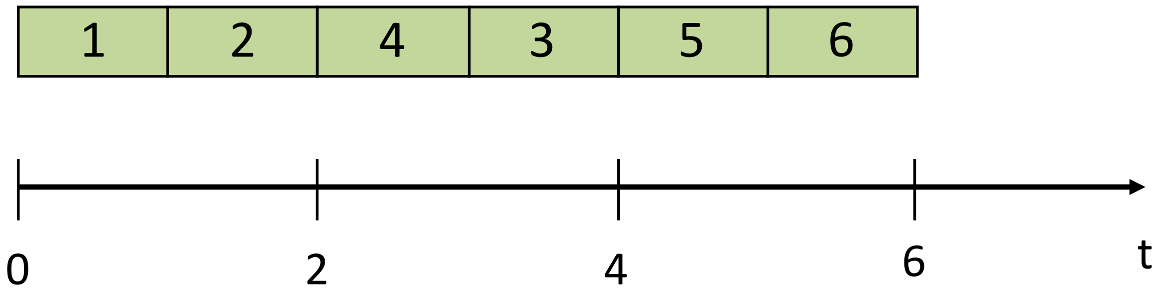
Earliest Deadline First with precedence (EDF*)



τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	2
3	1	4	4
4	1	3	3
5	1	5	5
6	1	6	6

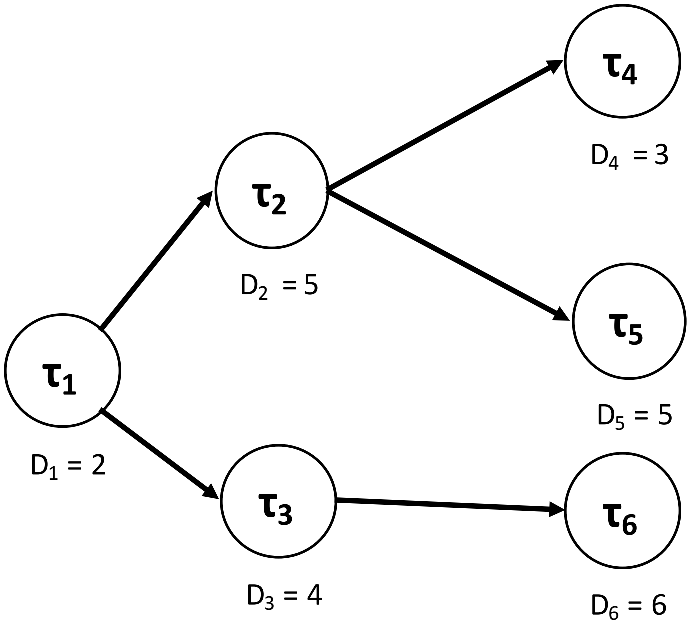
Earliest Deadline First with precedence (EDF*)

τ_i	e_i	D_i	P_i
1	1	2	1
2	1	5	2
3	1	4	4
4	1	3	3
5	1	5	5
6	1	6	6



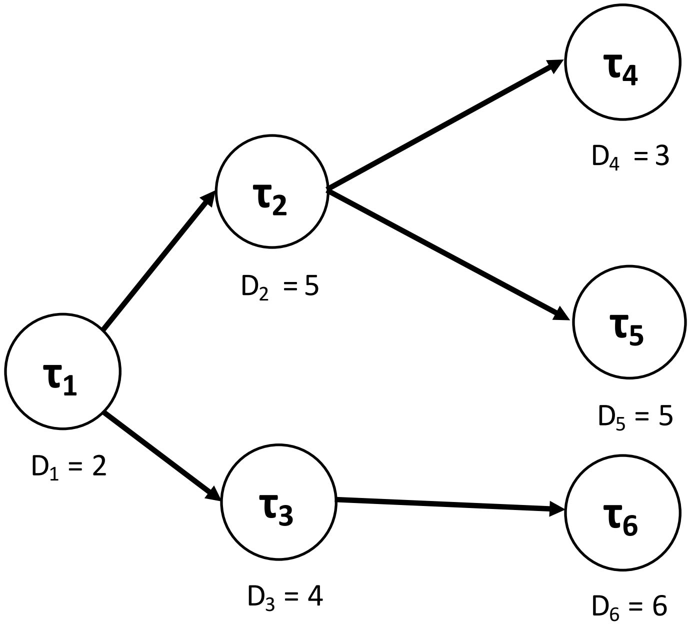
Multiprocessor scheduling

A difficult problem when combined with precedence.



Multiprocessor scheduling

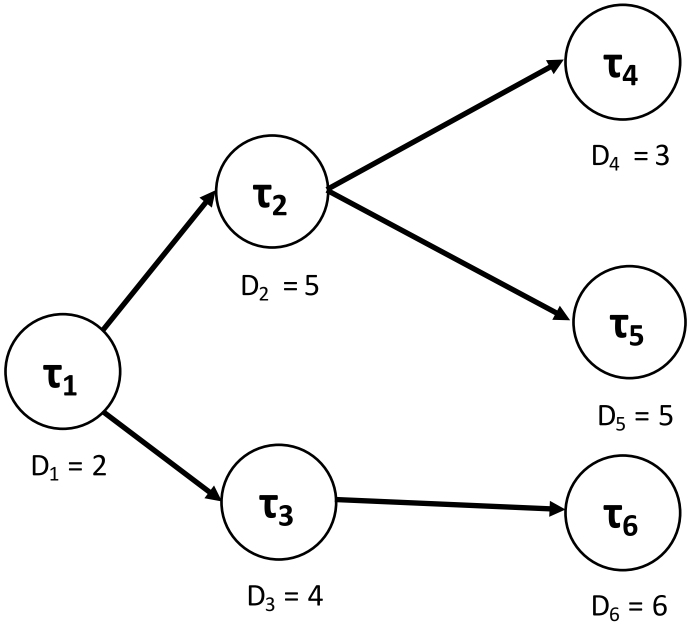
The priority can be assigned by Hu's level algorithm.



τ_i	e_i	D_i	L_i	P_i
1	1	2	3	
2	1	5	2	
3	1	4	2	
4	1	3	1	
5	1	5	1	
6	1	6	1	

Multiprocessor scheduling

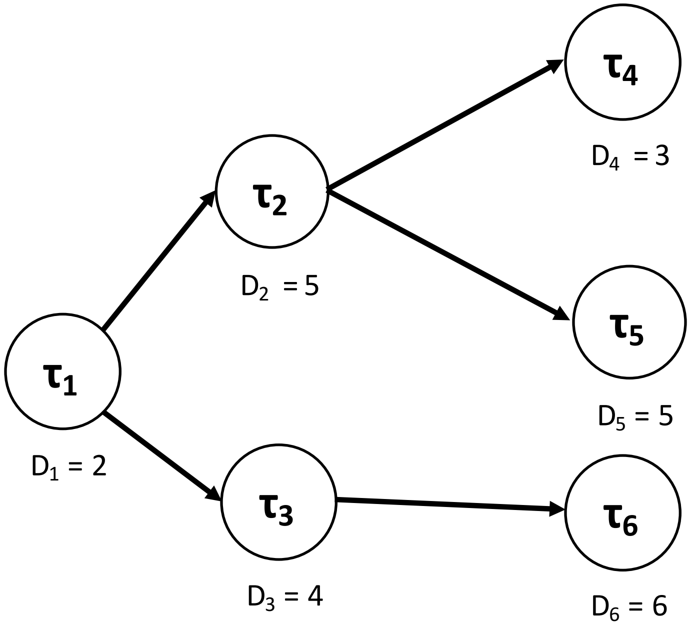
The priority can be assigned by Hu's level algorithm.



τ_i	e_i	D_i	L_i	P_i
1	1	2	3	1
2	1	5	2	2
3	1	4	2	2
4	1	3	1	3
5	1	5	1	3
6	1	6	1	3

Multiprocessor scheduling

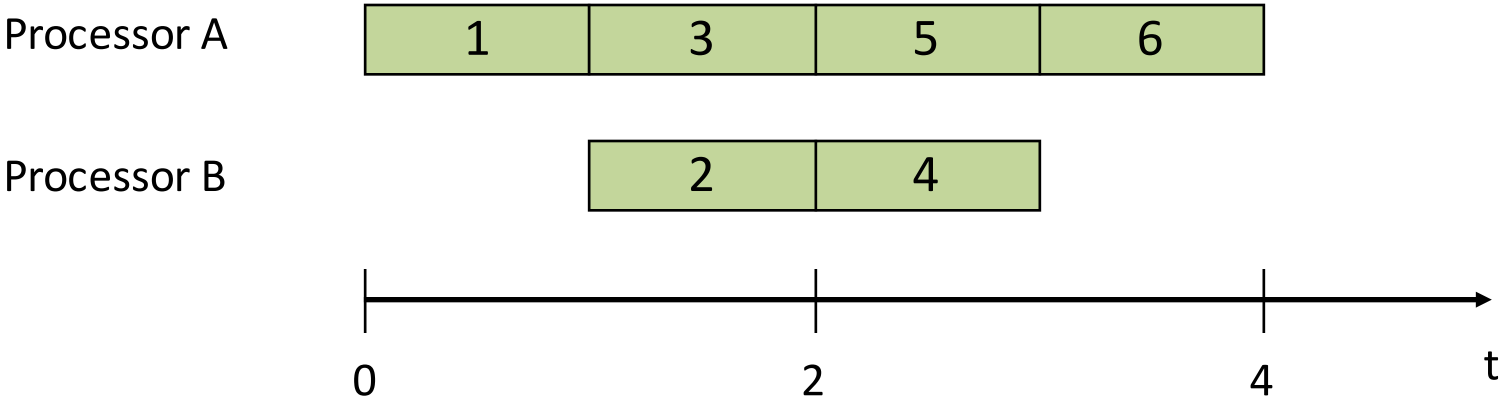
The priority can be assigned by Hu's level algorithm.



τ_i	e_i	D_i	L_i	P_i
1	1	2	3	1
2	1	5	2	3
3	1	4	2	2
4	1	3	1	4
5	1	5	1	5
6	1	6	1	6

Multiprocessor scheduling

τ_i	e_i	D_i	L_i	P_i
1	1	2	3	1
2	1	5	2	3
3	1	4	2	2
4	1	3	1	4
5	1	5	1	5
6	1	6	1	6



Scheduling Anomalies

- Priority inversion
- Deadlocks
- Richard’s Anomalies
- Mutual exclusion lock

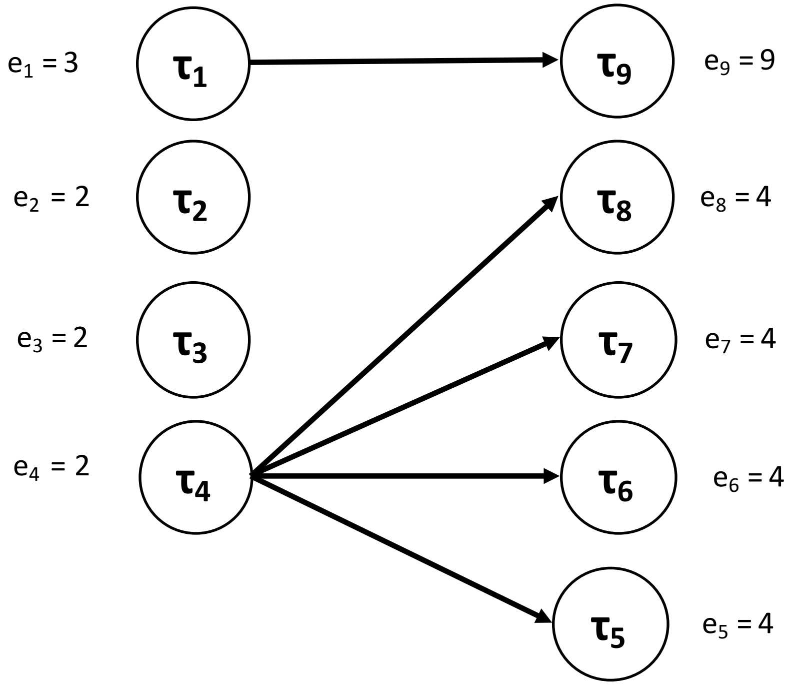
Scheduling Anomalies

Richard’s Anomalies

“If a task set with fixed priorities, execution times, and precedence constraints is scheduled on a fixed number of processors in accordance with the priorities, then increasing the number of processors, reducing execution times, or weakening precedence constraints can increase the schedule length.” [1]

Scheduling Anomalies

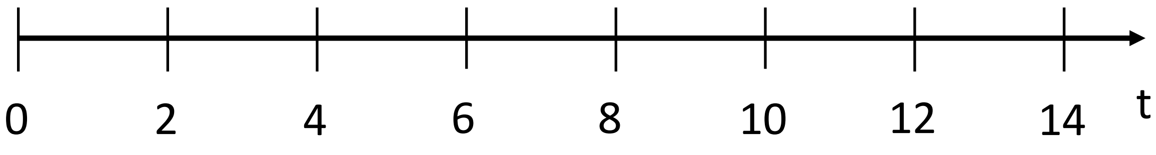
Richard's Anomalies



Proc 1

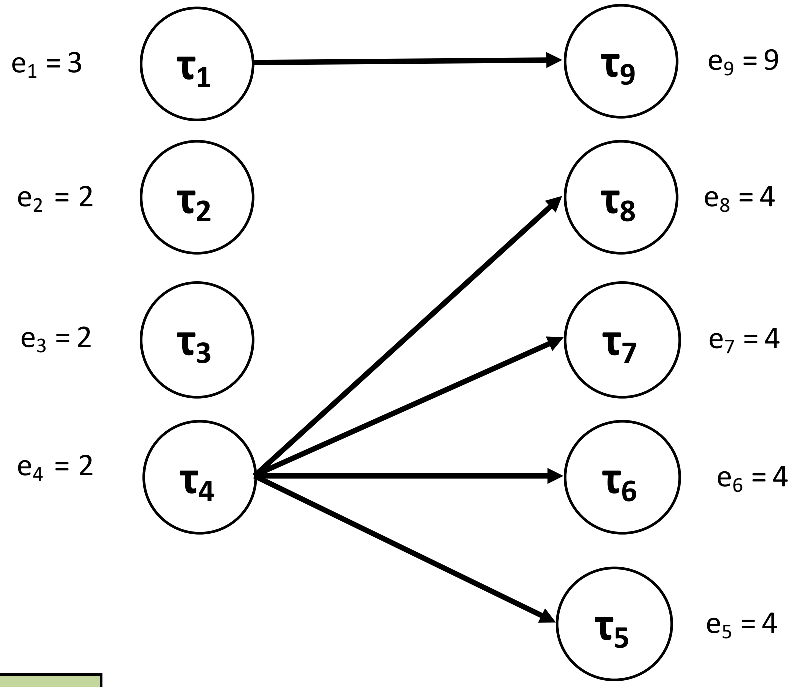
Proc 2

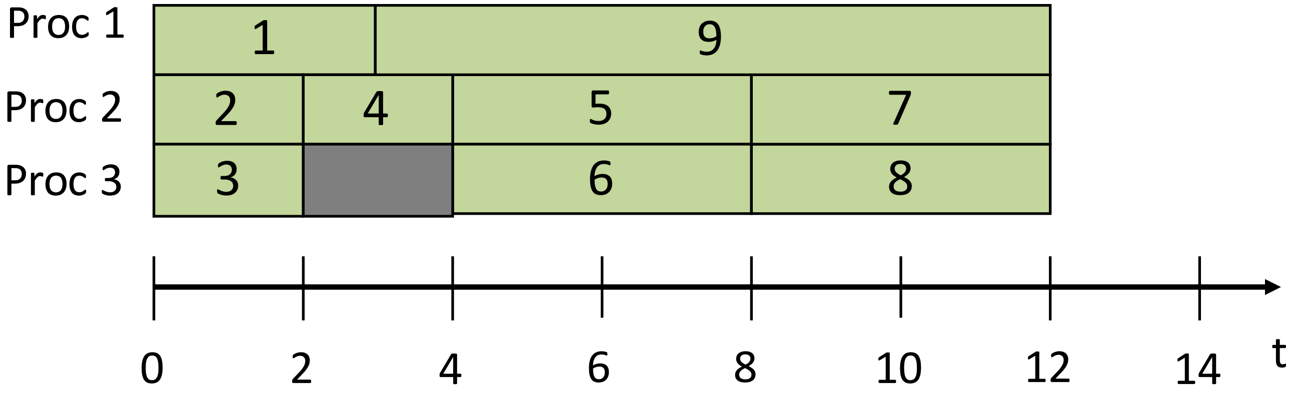
Proc 3



Scheduling Anomalies

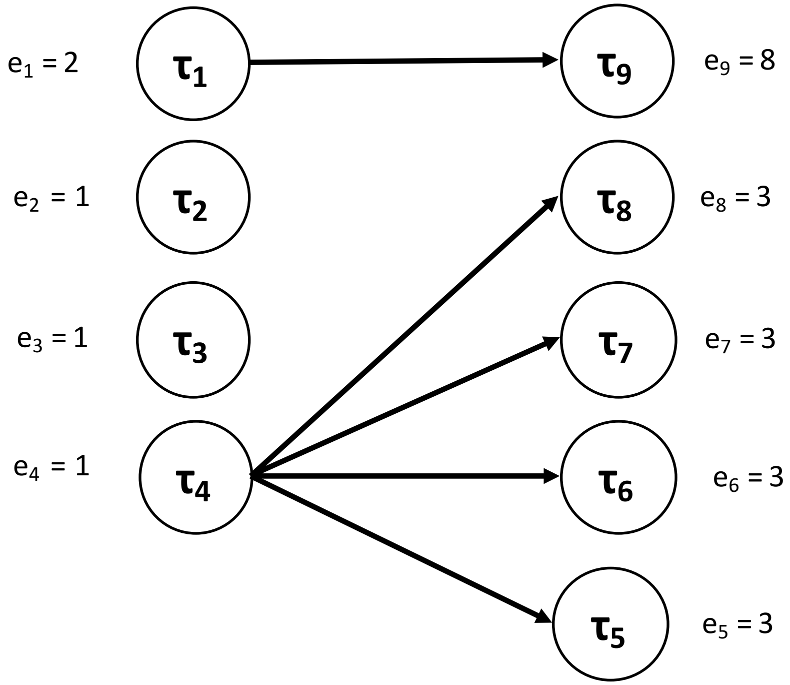
Richard's Anomalies

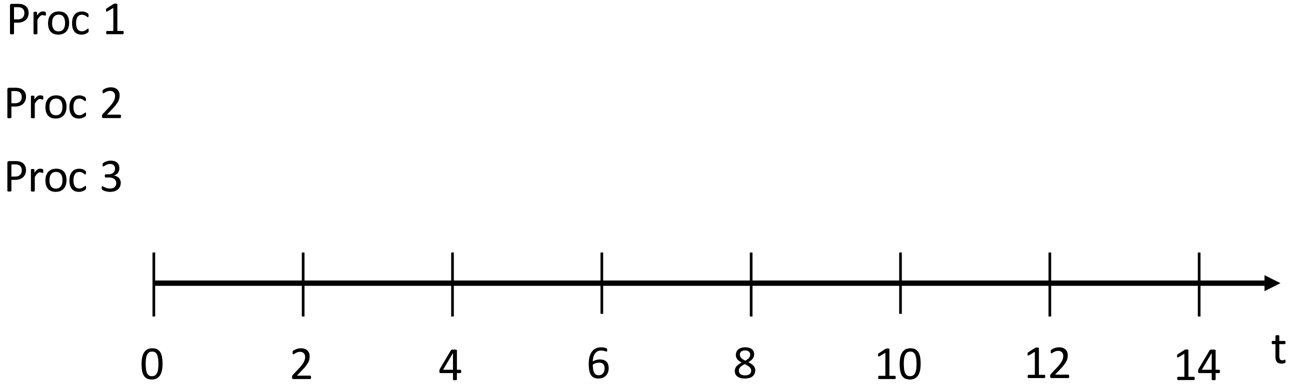




Scheduling Anomalies

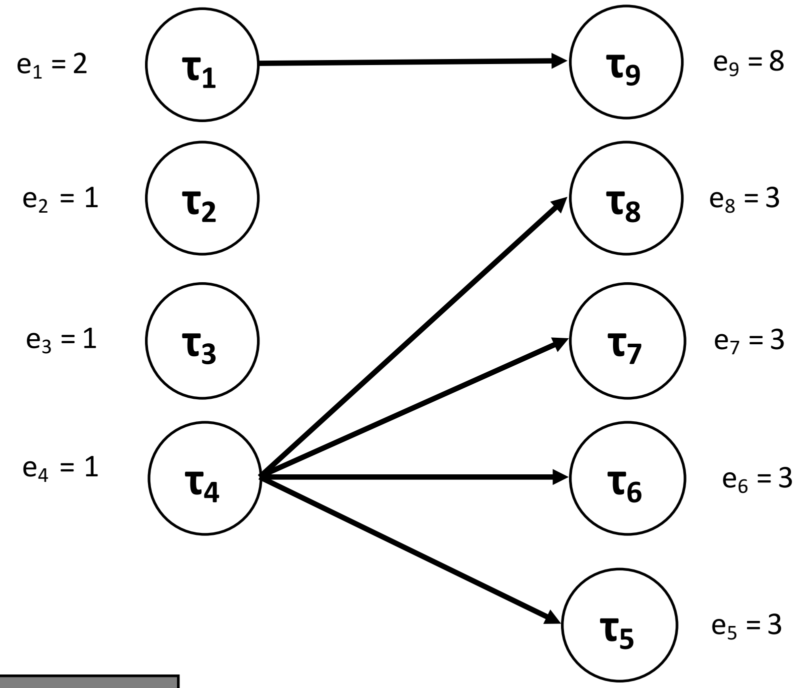
Richard's Anomalies

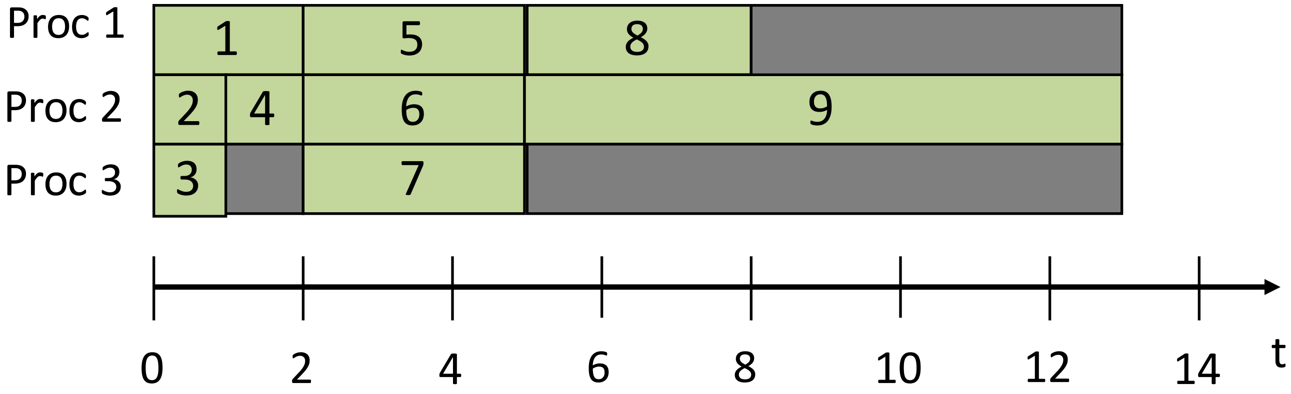




Scheduling Anomalies

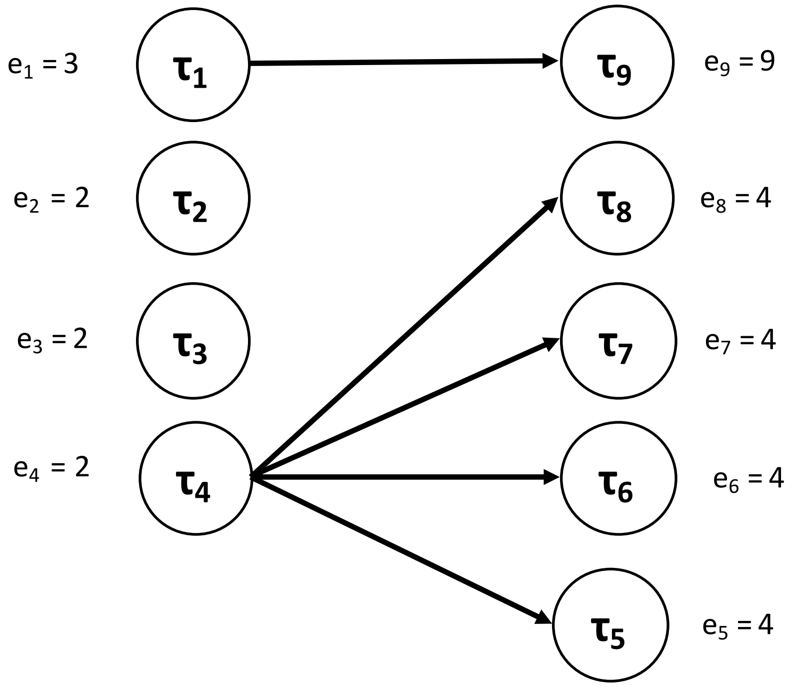
Richard's Anomalies





Scheduling Anomalies

Richard's Anomalies



$e_1 = 3$ τ_1 → τ_9 $e_9 = 9$

$e_2 = 2$ τ_2

$e_3 = 2$ τ_3

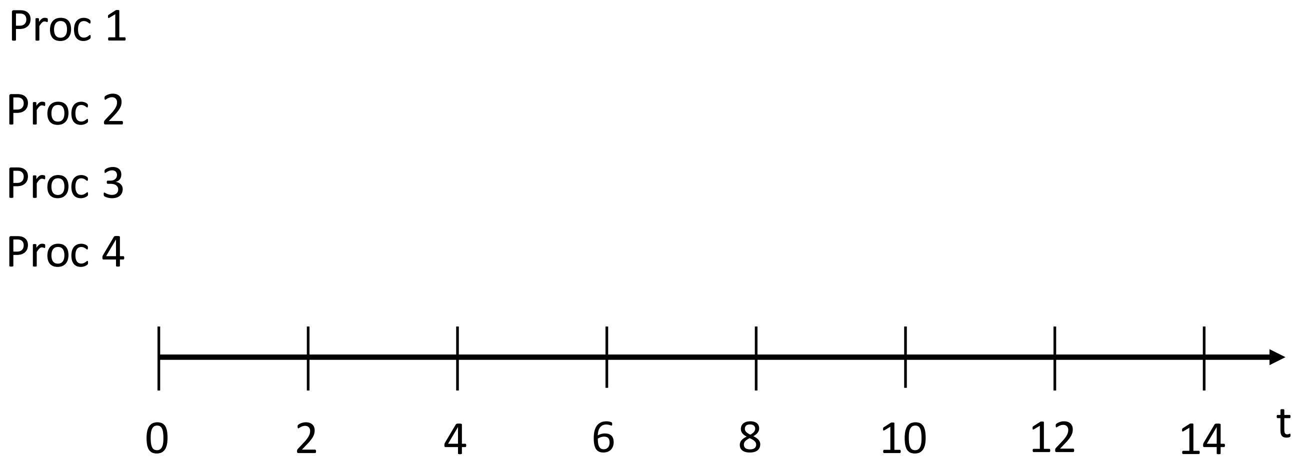
$e_4 = 2$ τ_4 → τ_8, τ_7, τ_6, τ_5

$e_8 = 4$ τ_8

$e_7 = 4$ τ_7

$e_6 = 4$ τ_6

$e_5 = 4$ τ_5



Proc 1

Proc 2

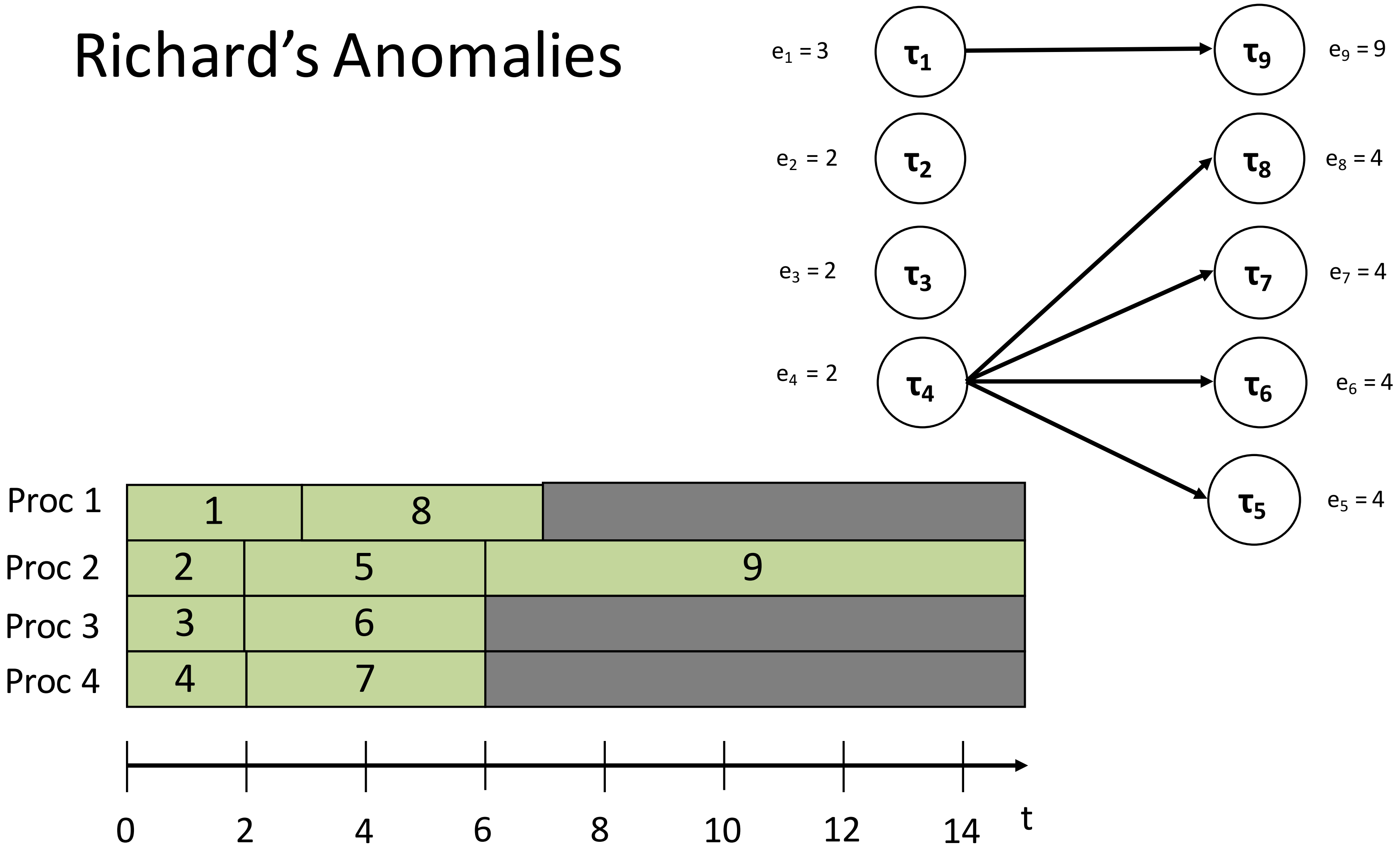
Proc 3

Proc 4

0 2 4 6 8 10 12 14 t

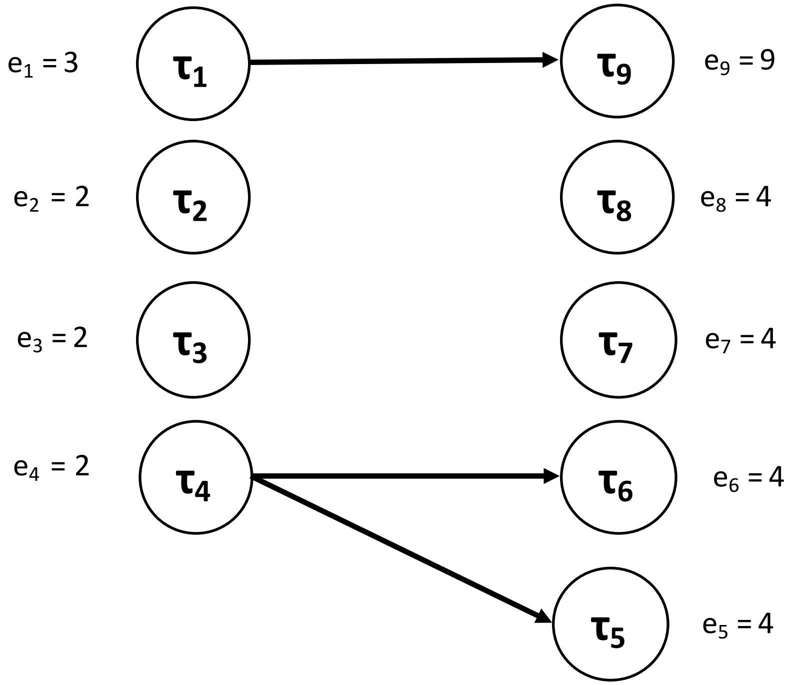
Scheduling Anomalies

Richard’s Anomalies



Scheduling Anomalies

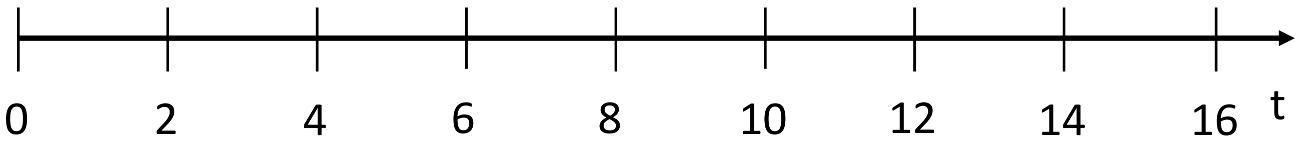
Richard’s Anomalies



Proc 1

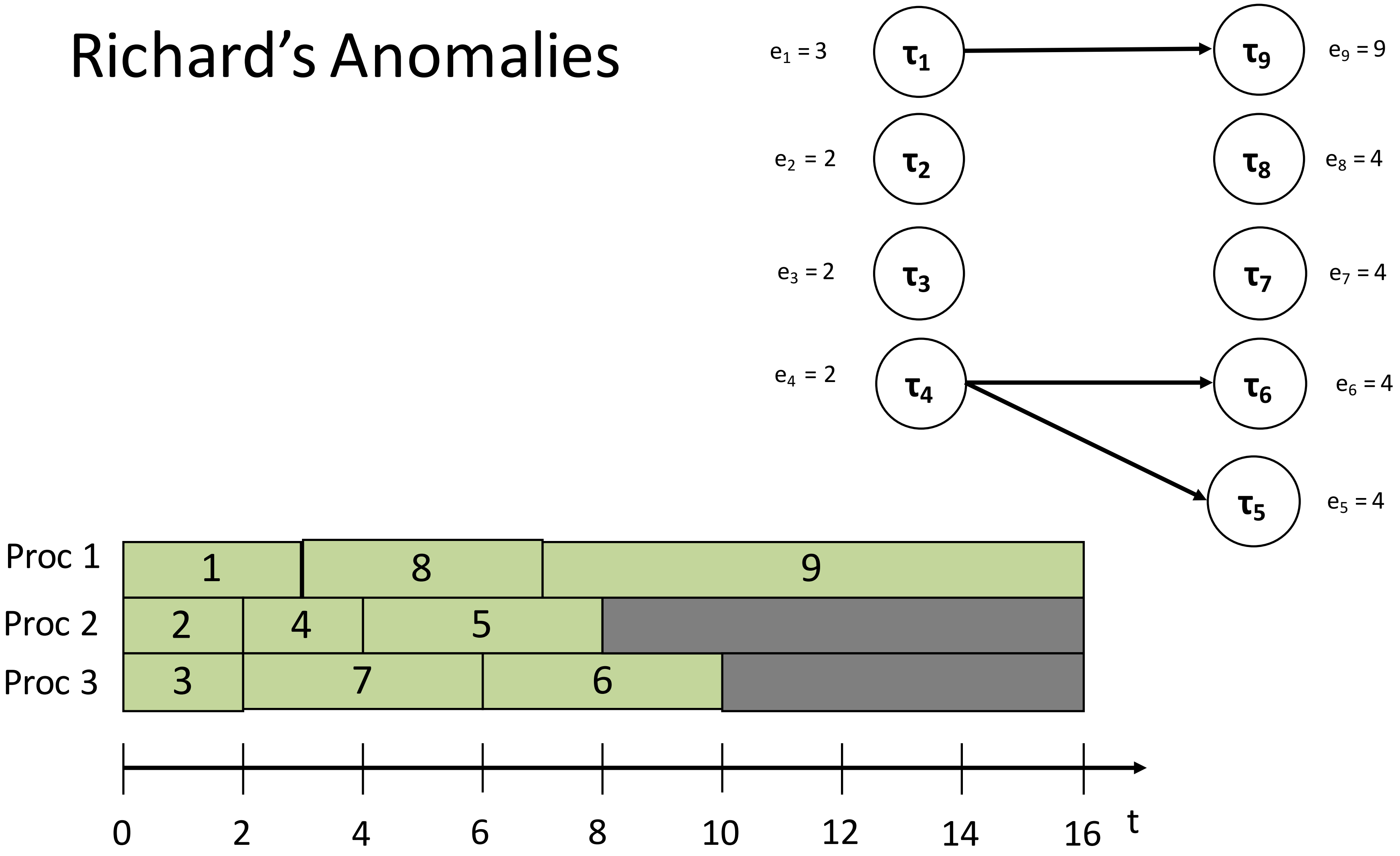
Proc 2

Proc 3



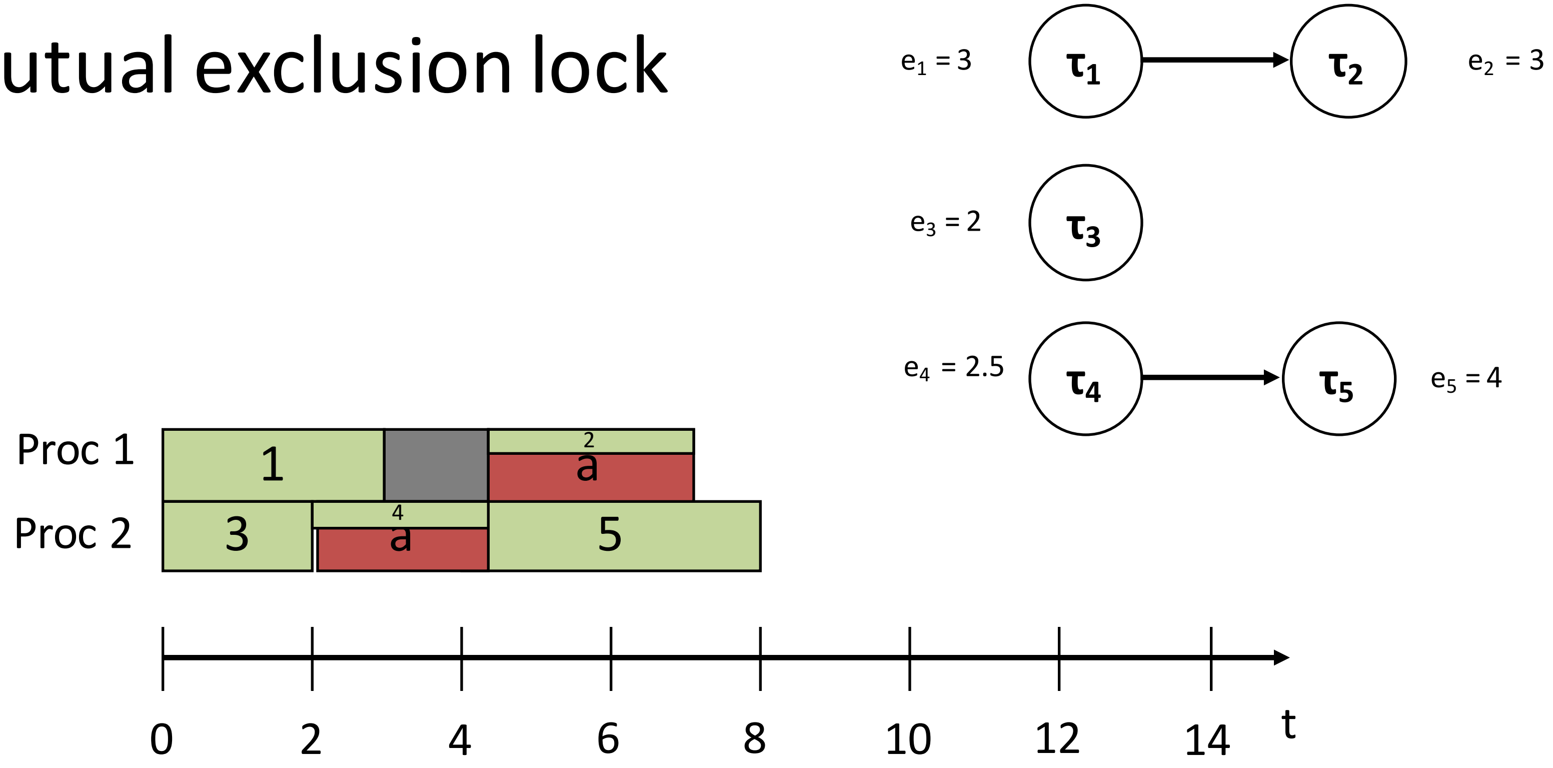
Scheduling Anomalies

Richard’s Anomalies



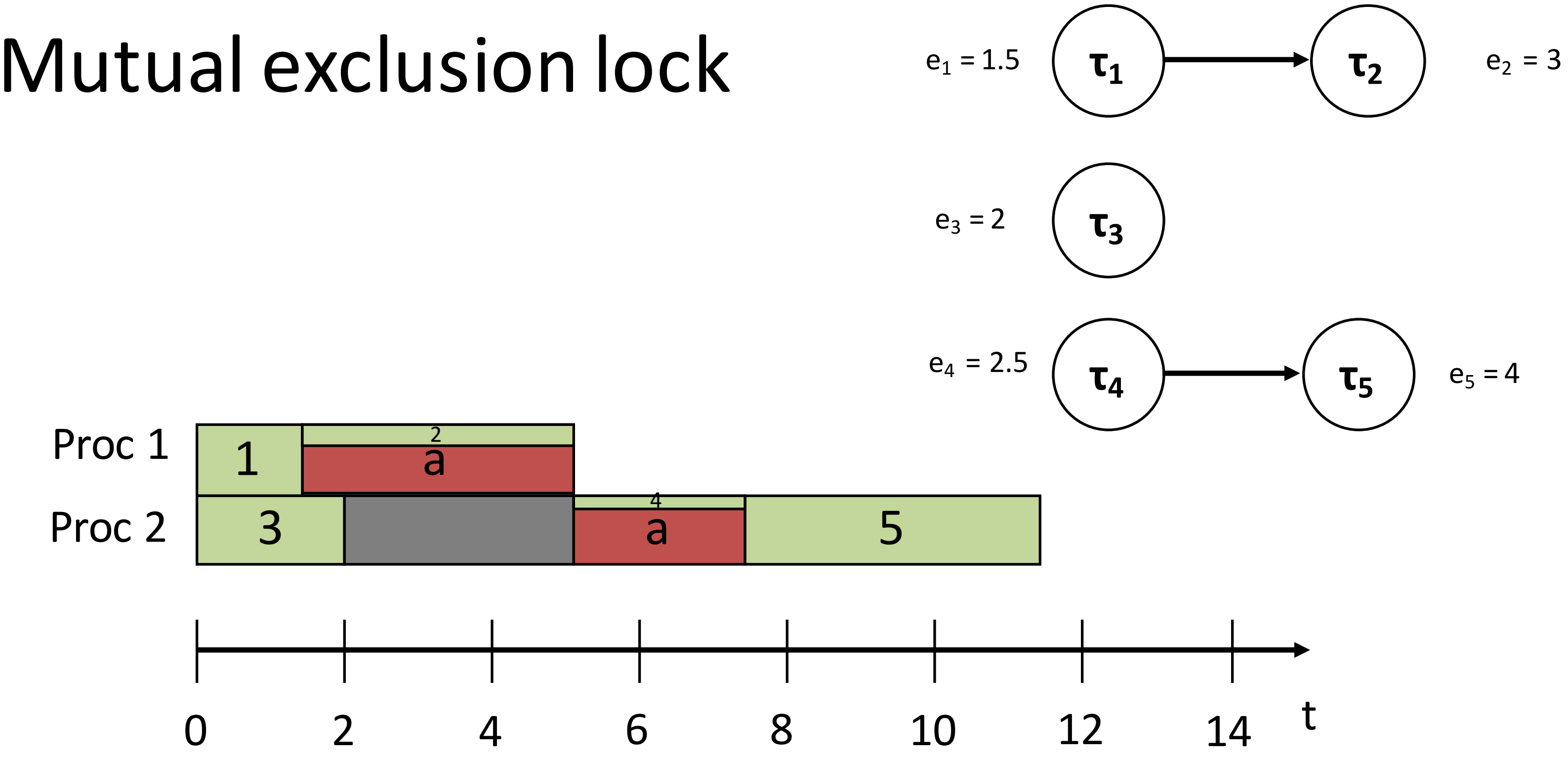
Scheduling Anomalies

Mutual exclusion lock



Scheduling Anomalies

Mutual exclusion lock



References

[1] Lee, E. A., Seshia, S. A. “Introduction to Embedded Systems - A Cyber-Physical Systems Approach”, Second Edition, MIT Press, 2017.

[2] Burns, A. and Wellings, A., “*Real-Time Systems and Programming Languages*”, Chapter 13, Addison Wesley, 1997

[3] Gomaa, H., “*Software Design Methods for Concurrent and Real-Time Systems*”, Addison-Wesley, 1993.