Lecture 18: Real-Time Process Scheduling

EECS 388 – Fall 2022

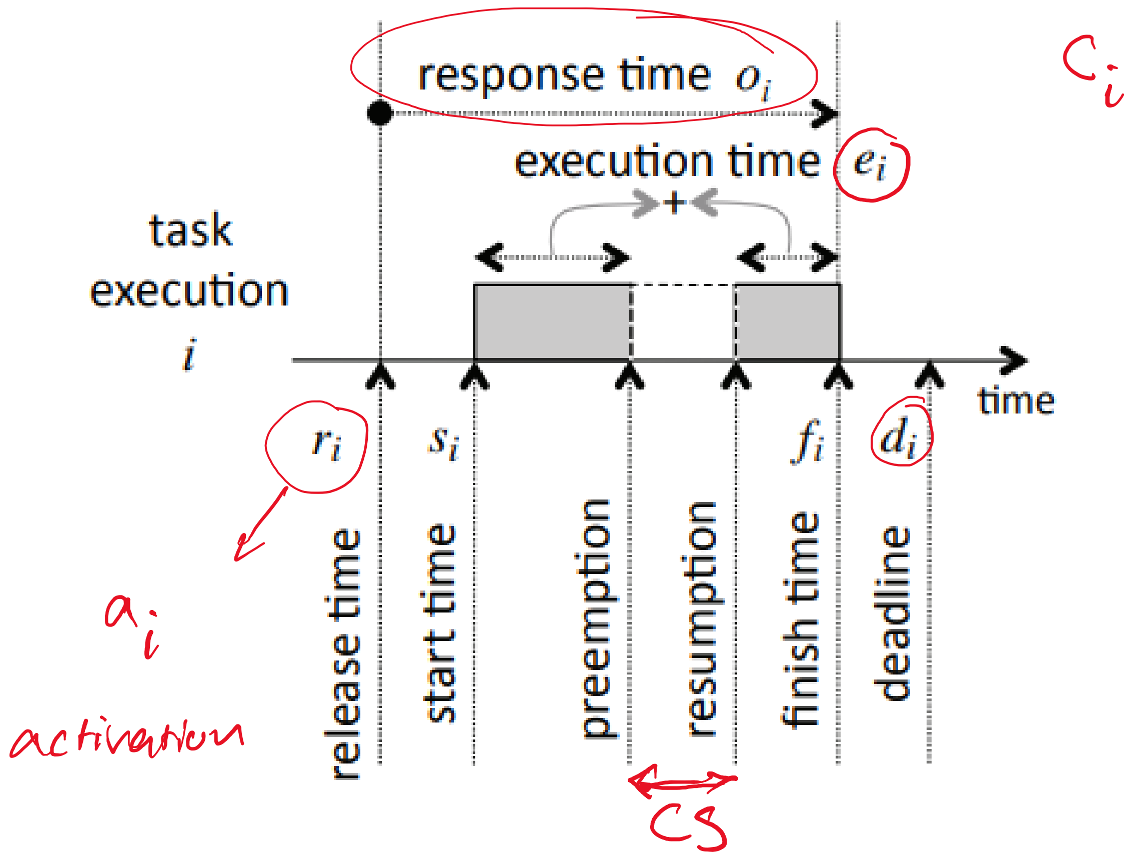
Lecture notes are based on slides created by Prof. Heechul Yun

Recap

- Job
  - A computation instance
- Task
  - A sequence of jobs
- Periodic task model
  - ti = (Ci, Ti) or (Ci, Ti, Di)

Recommended Reading: Chapter 12
Introduction to Embedded Systems - A Cyber-Physical Systems Approach, by E Lee and S. Seshia.

Times Associated with a Task Execution – Text Book's Terminology



Source: Introduction to Embedded Systems Lee & Seshia

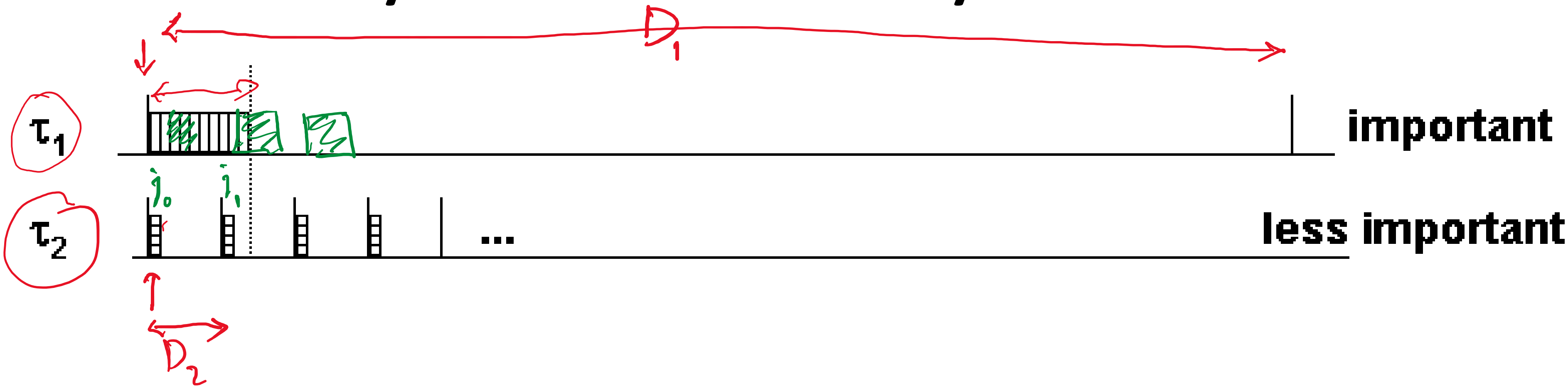
Real-Time Scheduling

- Scheduling
  - Pick which task to run next
- Priority-based scheduling
  - Pick the highest priority task among ready tasks

Priority and Criticality

- Priority
  - priority is the order we execute ready jobs.
- Criticality (Importance)
  - represents the penalty if a task misses a deadline (one of its jobs misses a deadline).
- Question
  - Which task should have higher priority?
  - Task 1: The most import task in the system: if it does not get done, serious consequences will occur
  - Task 2: A mp3 player: if it does not get done in time, the played song will have a glitch
  - If it is feasible, we would like to meet the real-time deadlines of both tasks

Priority and Criticality

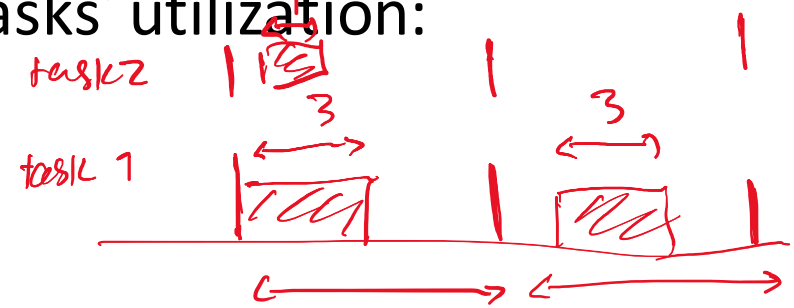


- If priorities are assigned according to importance, you can miss the deadline even if the system is mostly idle
- If p2 (period of t2) < C1 (execution time of t1), t2 will miss the deadline

Utilization

- A task’s utilization (of the CPU) is the fraction of time for which it uses the CPU (over a long interval of time).
- A periodic task’s utilization U_i (of CPU) is the ratio between its execution time and period: $U_i = C_i/p_i$
- Given a set of periodic tasks, the total CPU’s utilization is equal to the sum of periodic tasks’ utilization:

$$U = \sum_i \frac{C_i}{p_i}$$



single CPU

- What is the limit on U?

$\frac{4}{10}$

P_i 10 10

$U_1 = \frac{3}{10} + U_2 = \frac{1}{10}$

Real-Time Scheduling Algorithms

- Fixed (static) -priority scheduling
  - All jobs of a task have the same priority
  - Rate Monotonic (RM)
- Dynamic priority scheduling
  - Different jobs of the same task may have different priorities
  - Earliest Deadline First (EDF)

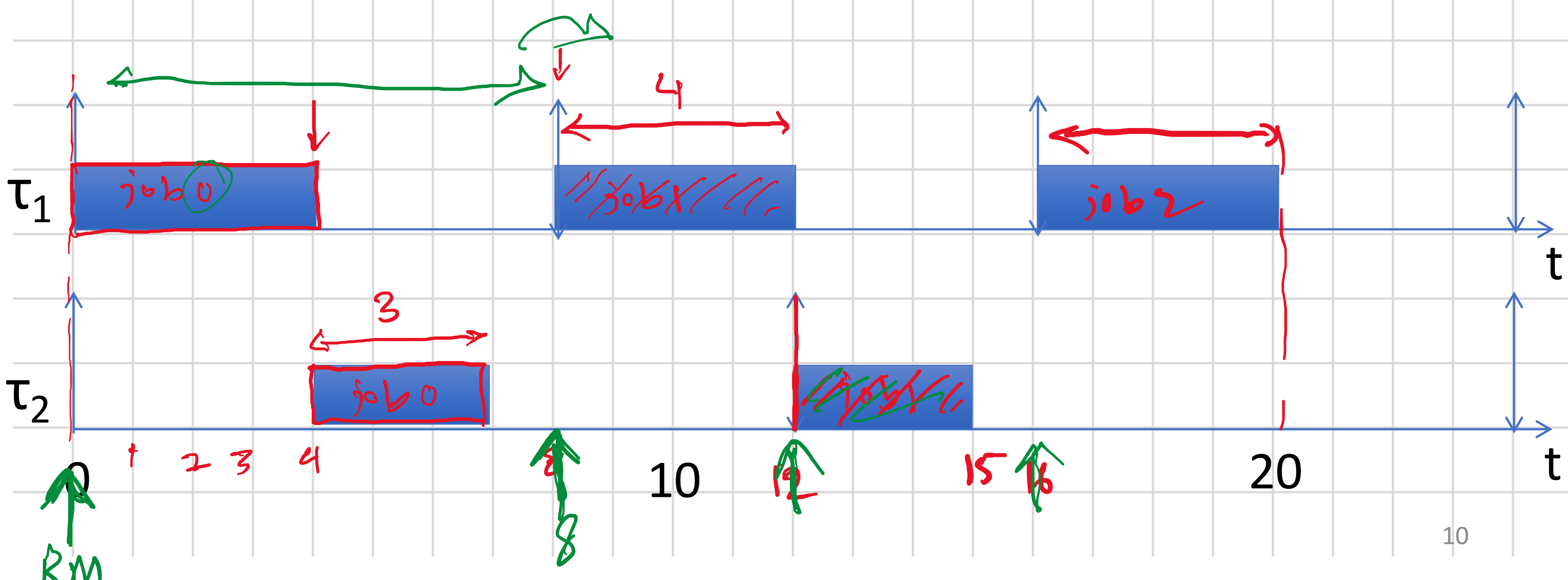
Rate Monotonic (RM)

- Fixed-priority scheduling algorithm
- Assigns priorities to tasks on the basis of their periods
- **Shorter period = higher priority.**

RM Example

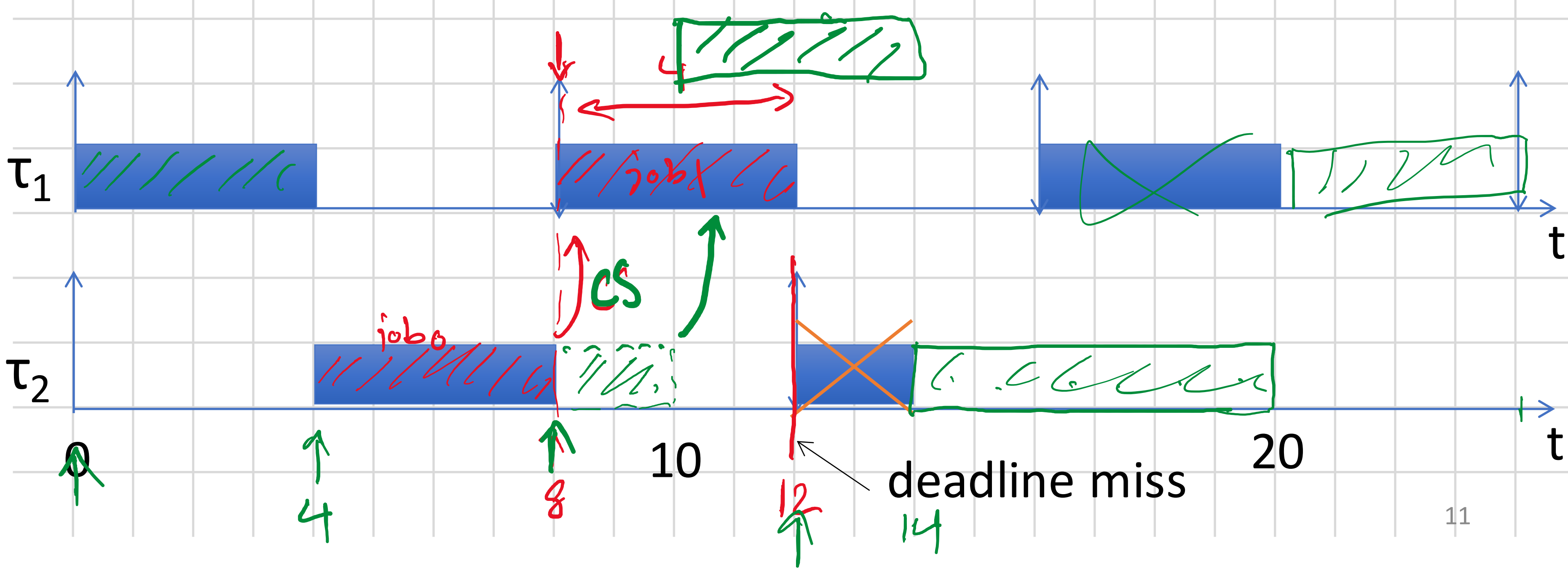
- Case#1
  - τ_1 (C1 = 4, T1 = 8), high prio
  - τ_2 (C2 = 3, T2 = 12), low prio
  - Utilization: U = 4/8 + 3/12 = 0.75

task 1 task 2



RM Example

- Case#2
  - τ_1 (C1 = 4, T1 = 8), high prio
  - τ_2 (C2 = 6, T2 = 12), low prio
  - Utilization: U = 4/8 + 6/12 = 1

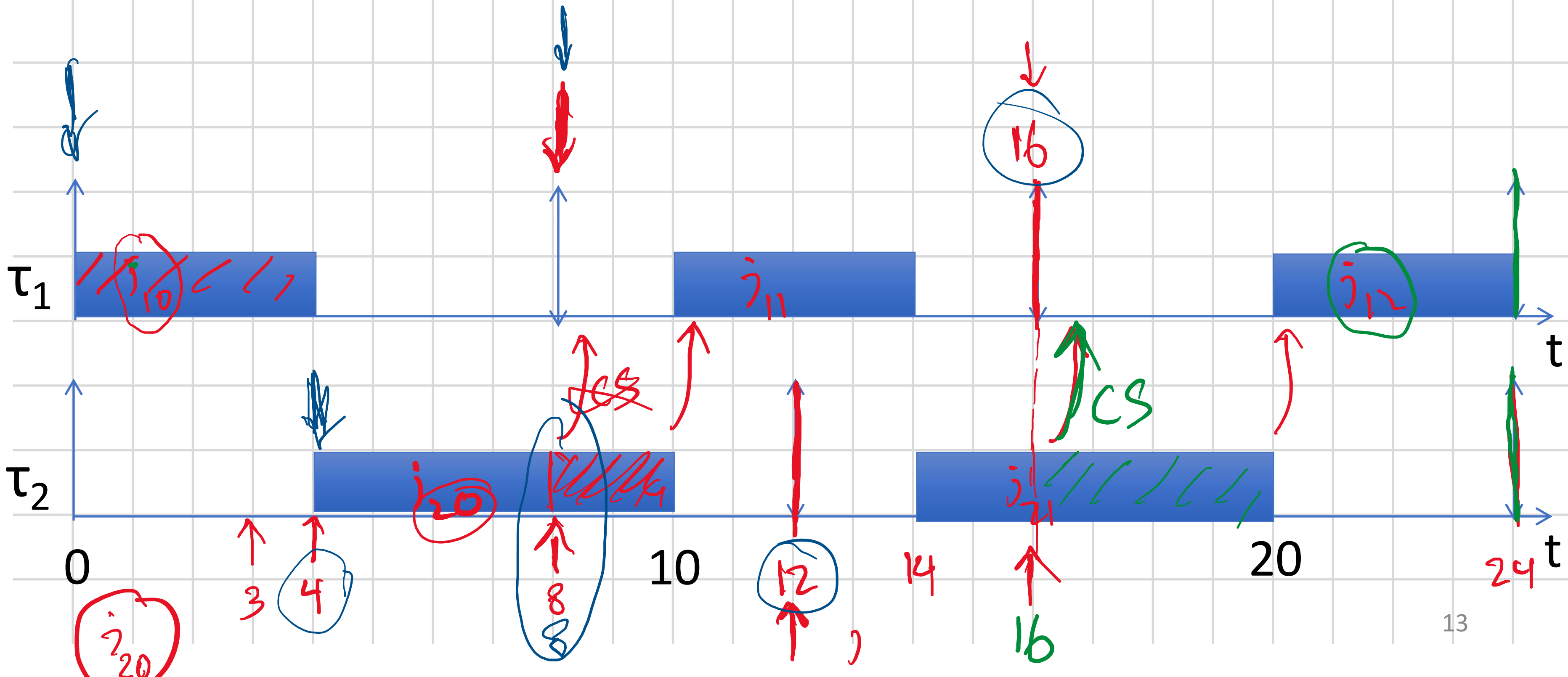


Earliest Deadline First (EDF)

- Dynamic-Priority Scheduling Algorithm
- Task priority is inversely proportional to its current absolute deadline
- **Earlier deadline = higher priority**
- Each job of a task has a different deadline, hence a different priority.

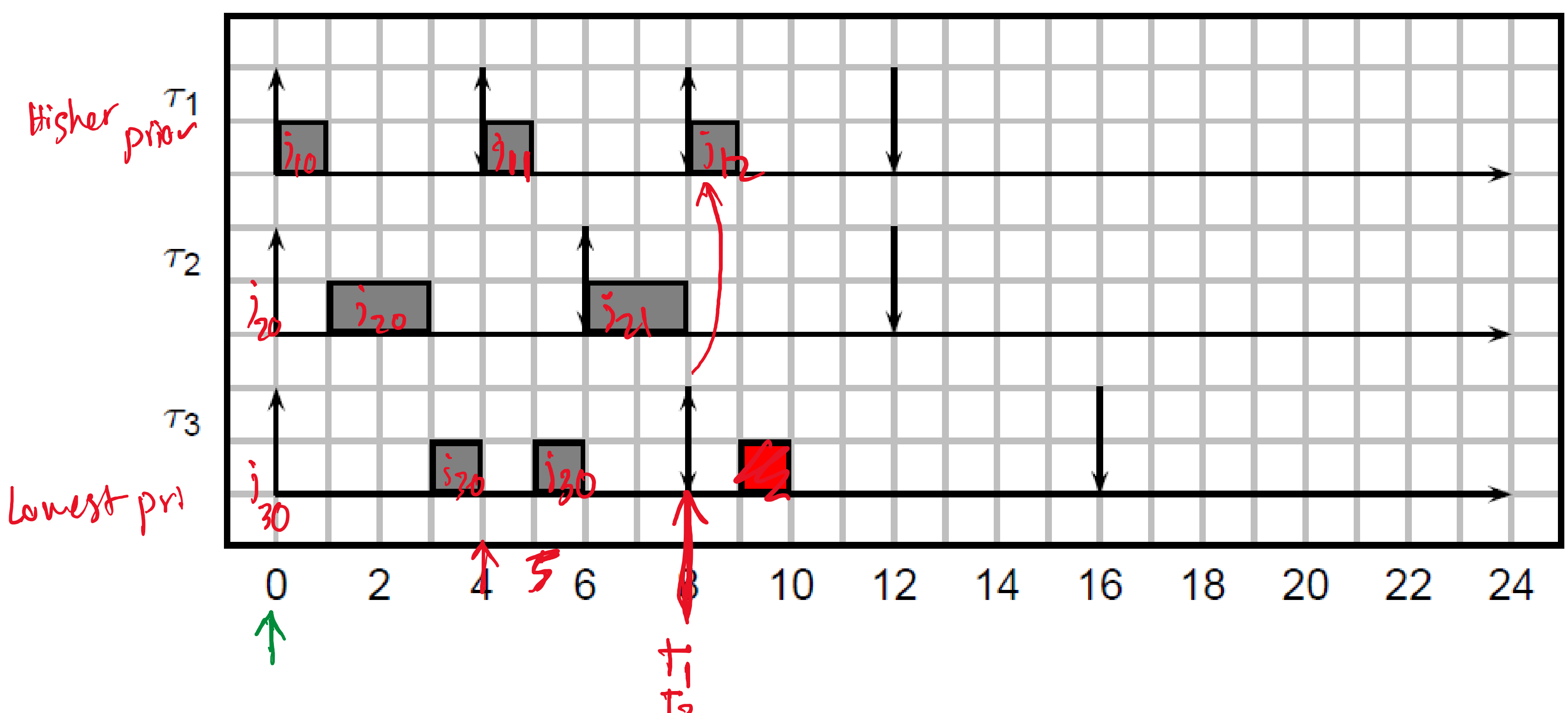
EDF Example

- Case#2:
  - τ_1 (C1 = 4, T1 = 8), τ_2 (C2 = 6, T1 = 12)
  - Utilization: U = 4/8 + 6/12 = U_b = 1



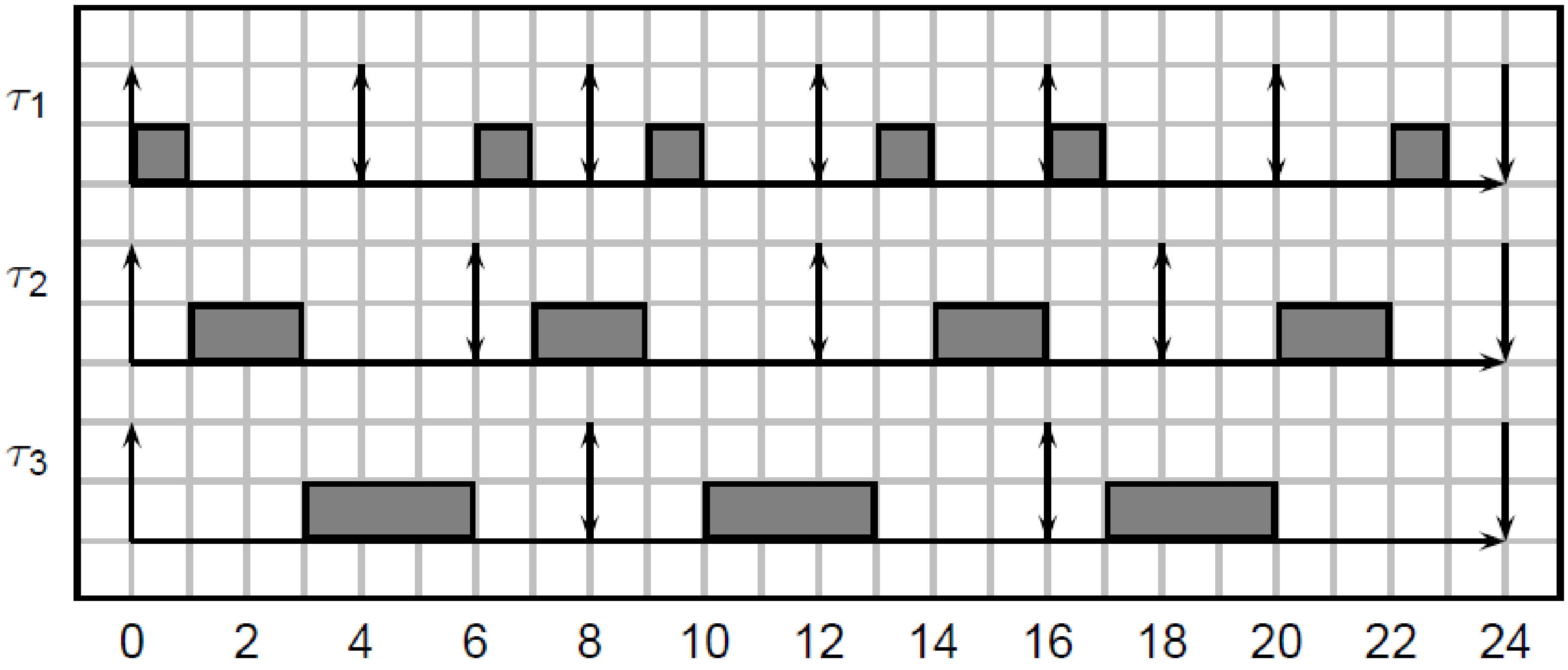
RM vs. EDF

- τ_1 (C1 = 1, T1 = 4), τ_2 (C2 = 2, T1 = 6), τ_3 (C3 = 3, T3 = 8)
- Utilization: U = 1/4 + 2/6 + 3/8 = 23/24
- RM scheduling:



RM vs. EDF

- τ_1 (C1 = 1, T1 = 4), τ_2 (C2 = 2, T1 = 6), τ_3 (C3 = 3, T3 = 8)
- Utilization: U = 1/4 + 2/6 + 3/8 = 23/24
- EDF scheduling?



Key Results

For periodic tasks with relative deadlines equal to their periods:

- Rate monotonic scheduling is the optimal fixed-priority priority policy
  - No other static priority assignment can do better
  - Yet, it might not achieve 100% CPU utilization

- Earliest deadline first scheduling is the optimal dynamic priority policy
  - EDF can achieve 100% CPU utilization

RM vs. EDF

- In practice, industrial systems heavily favor RM over EDF. Why?

RM vs. EDF

- In practice, industrial systems heavily favor RM over EDF. Why?

- RM is easier to implement
  - Task priority never changes.
- RM is more transparent and robust
  - easier to understand what is going on if something goes wrong (ex: overload).
  - if a task executes for longer than its prescribed worst-case time, higher priority tasks will be left untouched.

Example

Consider the following real-time taskset:

Task	C	P	U
t1	2	10	0.200
t2	4	15	0.267
t3	10	35	0.286

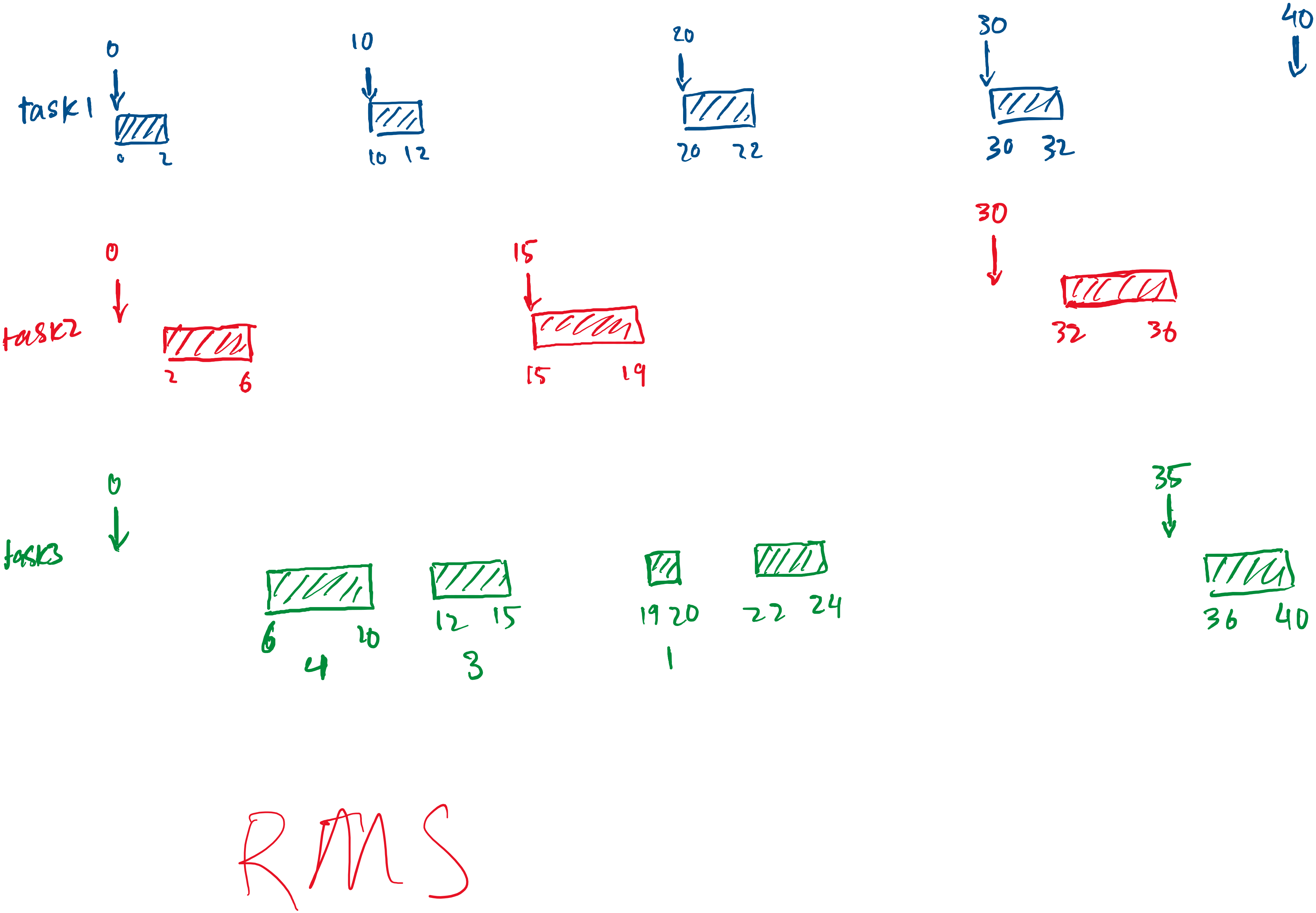
Note1: C = worst-case execution time; P = period; U = utilization.
Assume a single core processor.

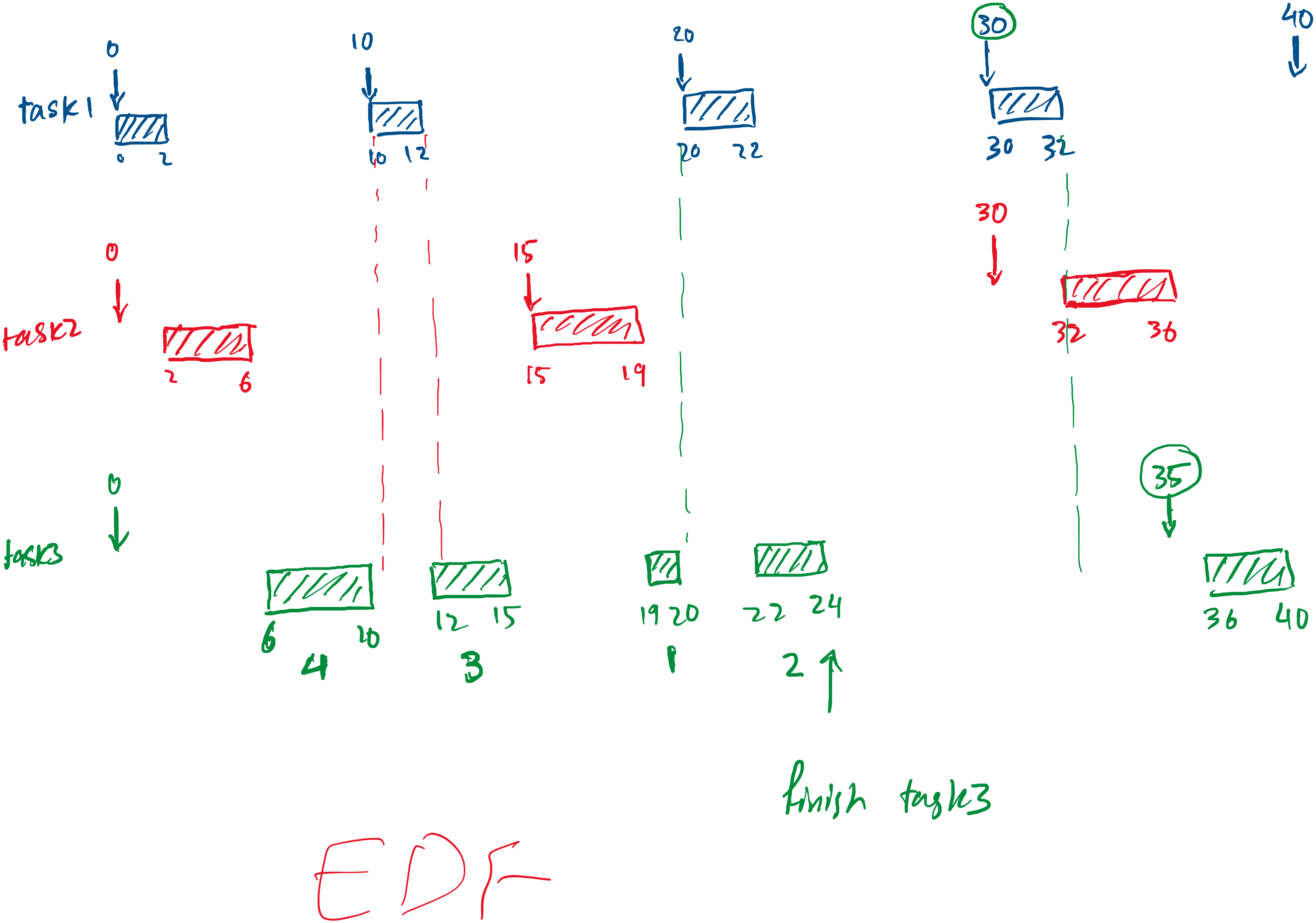
1- Compute the total utilization of the taskset.

2- Assign priorities of the tasks following the rate monotonic policy. You can use numbers, e.g. 1 (lowest priority) to 3 (highest).

3- Draw a timeline (from time 0 to 40) of the taskset under the rate monotonic scheduling (RMS).

EDF


task1
0
0 2
10
10 12
20
20 22
30
30 32
40
task2
0
2 6
15
15 19
30
32 36
task3
0
6 10
4
12 15
3
19 20
1
22 24
35
36 40
RMS


task1
task2
task3
finish task3
EDF

Acknowledgements

- These slides draw on materials developed by
  - Lui Sha and Marco Caccamo (UIUC)
  - Rodolfo Pellizzoni (U. Waterloo)
  - Edward A. Lee and Prabal Dutta (UCB) for EECS149/249A