Real Time Operating Systems and Middleware

Other Scheduling Algorithms

Luca Abeni

abeni@dit.unitn.it

Credits: Luigi Palopoli, Giuseppe Lipari, and Marco Di Natale

Scuola Superiore Sant'Anna

Pisa -Italy

Dynamic Priorities - EDF

- RM and DM are optimal *fixed priority* assignments
- Maybe we can improve schedulability by using *dynamic priorities*?
  - Fixed priority scheduling: a task τ always has the same priority
  - Dynamic priority scheduling: τ's priority can change during time...
  - Let's assume that the priority changes from job to job (a job $J_{i,j}$ always has the same priority $p_{h,k}$)
- Simplest idea: give priority to tasks with the earliest absolute deadline: $d_{i,j} < d_{h,k} \Rightarrow p_{i,j} > p_{h,k}$
  - Earliest Deadline First (EDF)
  - DM $\rightarrow$ *relative* deadlines; EDF $\rightarrow$ *absolute* deadlines

Can We Do any Better than RM?

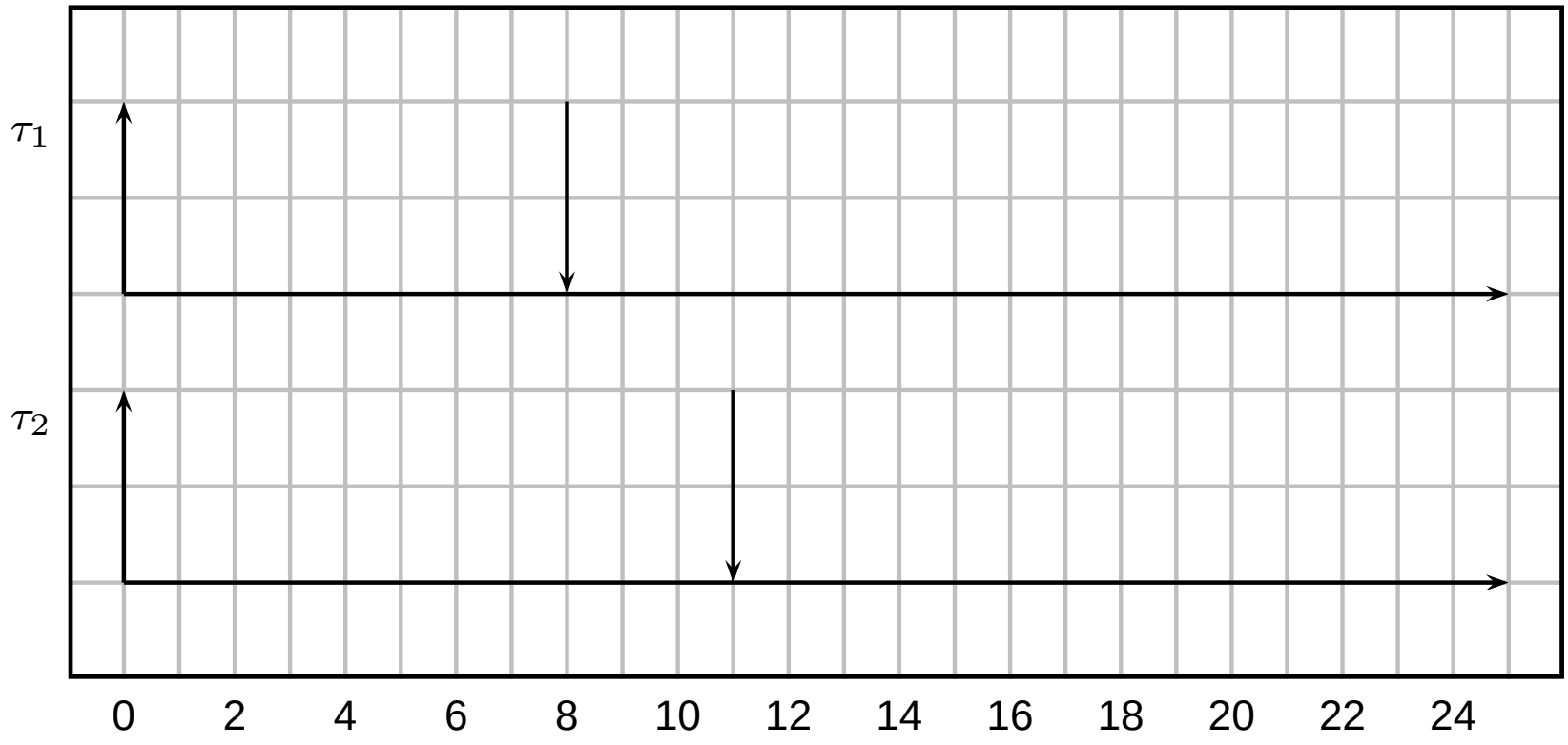
- Yes (of course!): EDF can get full processor utilisation
- Consider a system of periodic tasks with relative deadline equal to the period.
- The system is schedulable with EDF if and only if

$$\sum_i \frac{C_i}{T_i} \leq 1$$

  - $U_{lub} = 1$!!!
- If $D_i \neq T_i$:
  - Processor demand approach or response time analysis can be applied to EDF too
  - But it is not obvious!

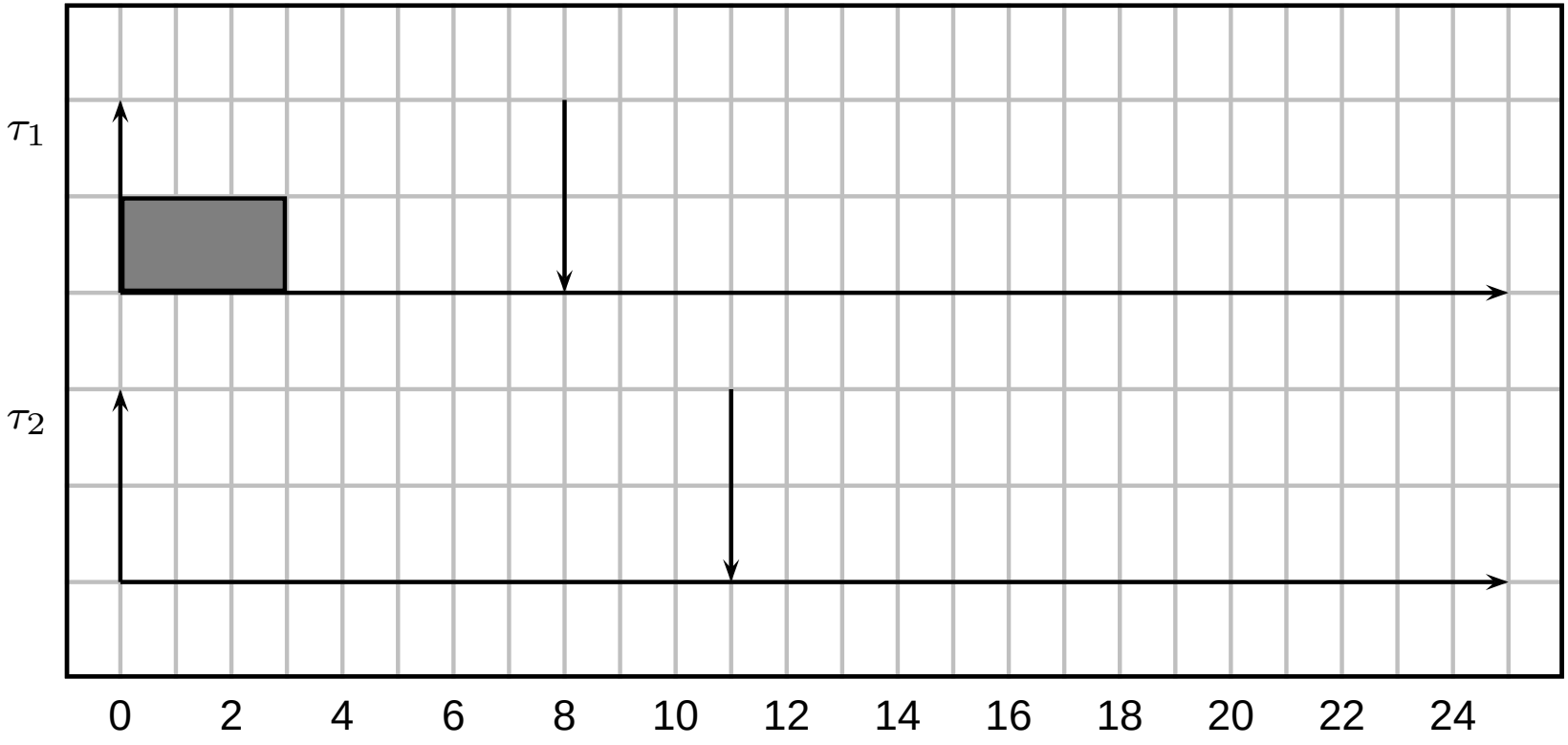
An Example – RM

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



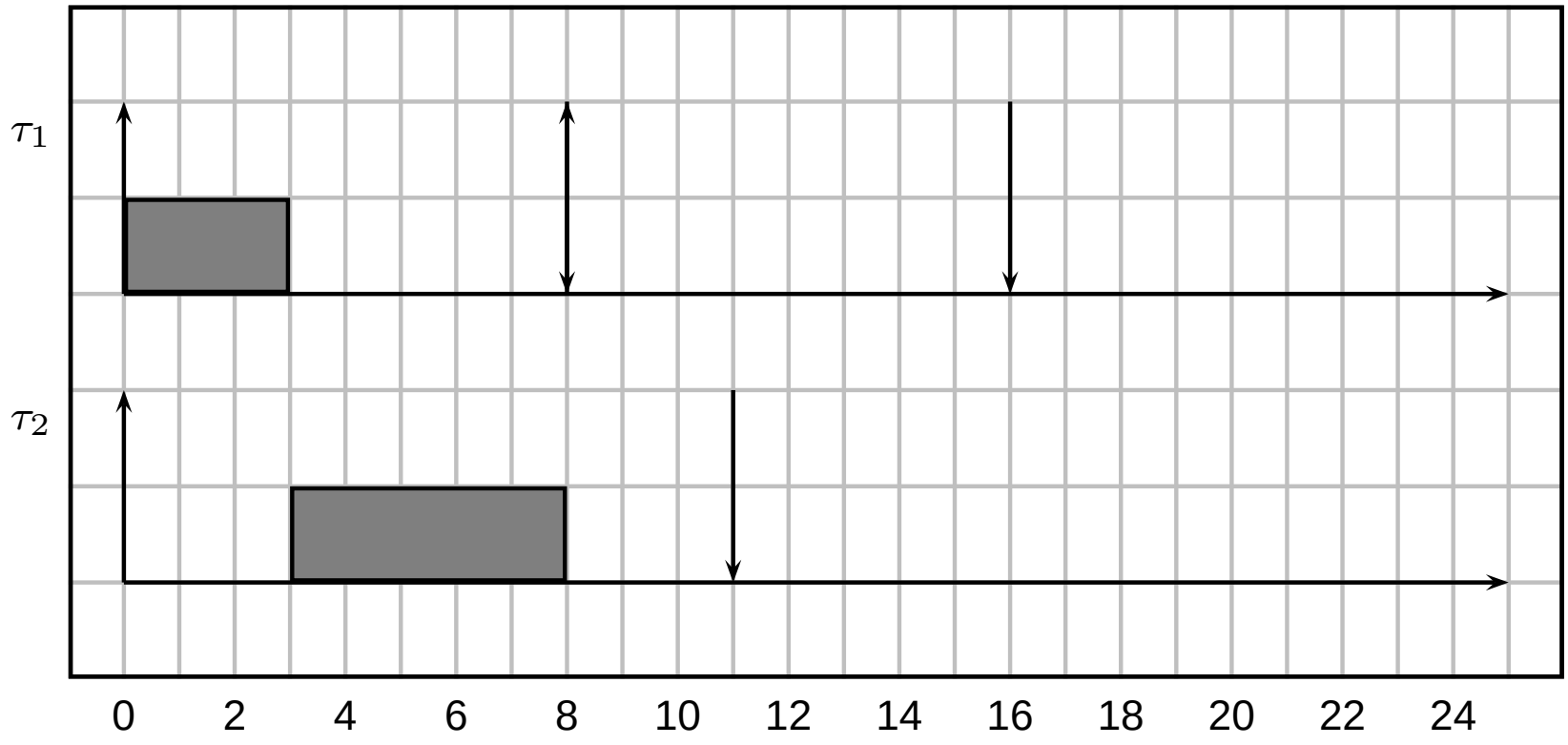
An Example – RM

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



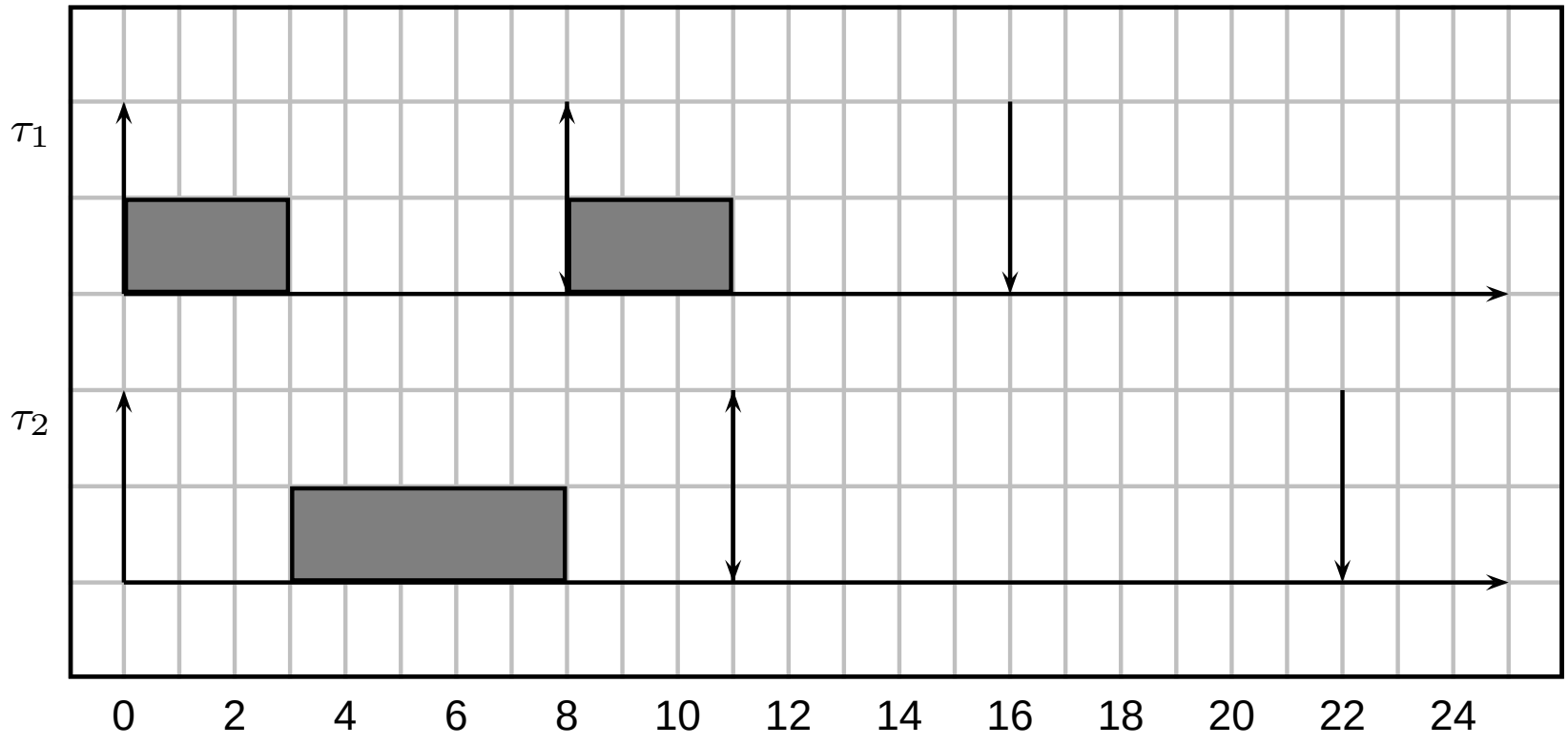
An Example – RM

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



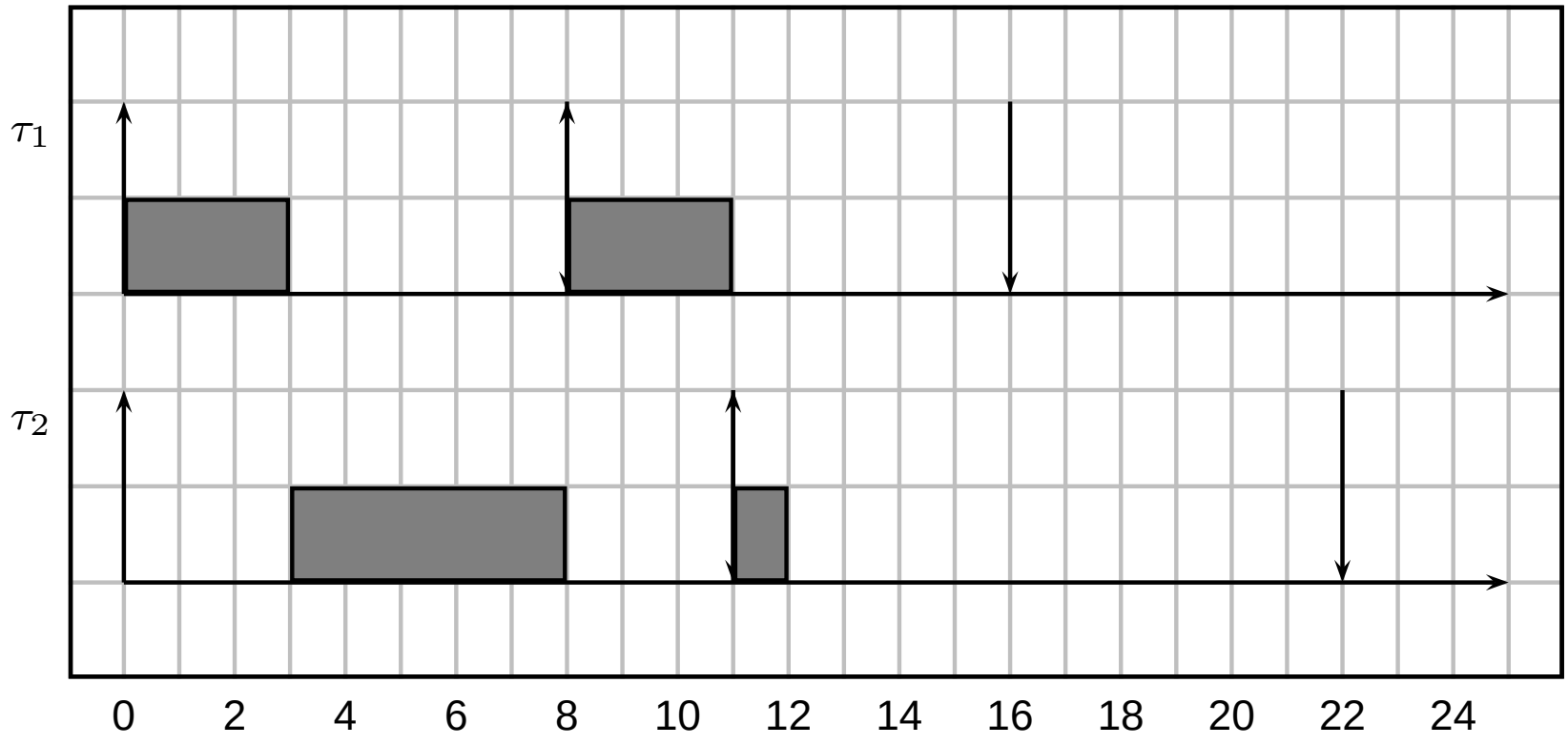
An Example – RM

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



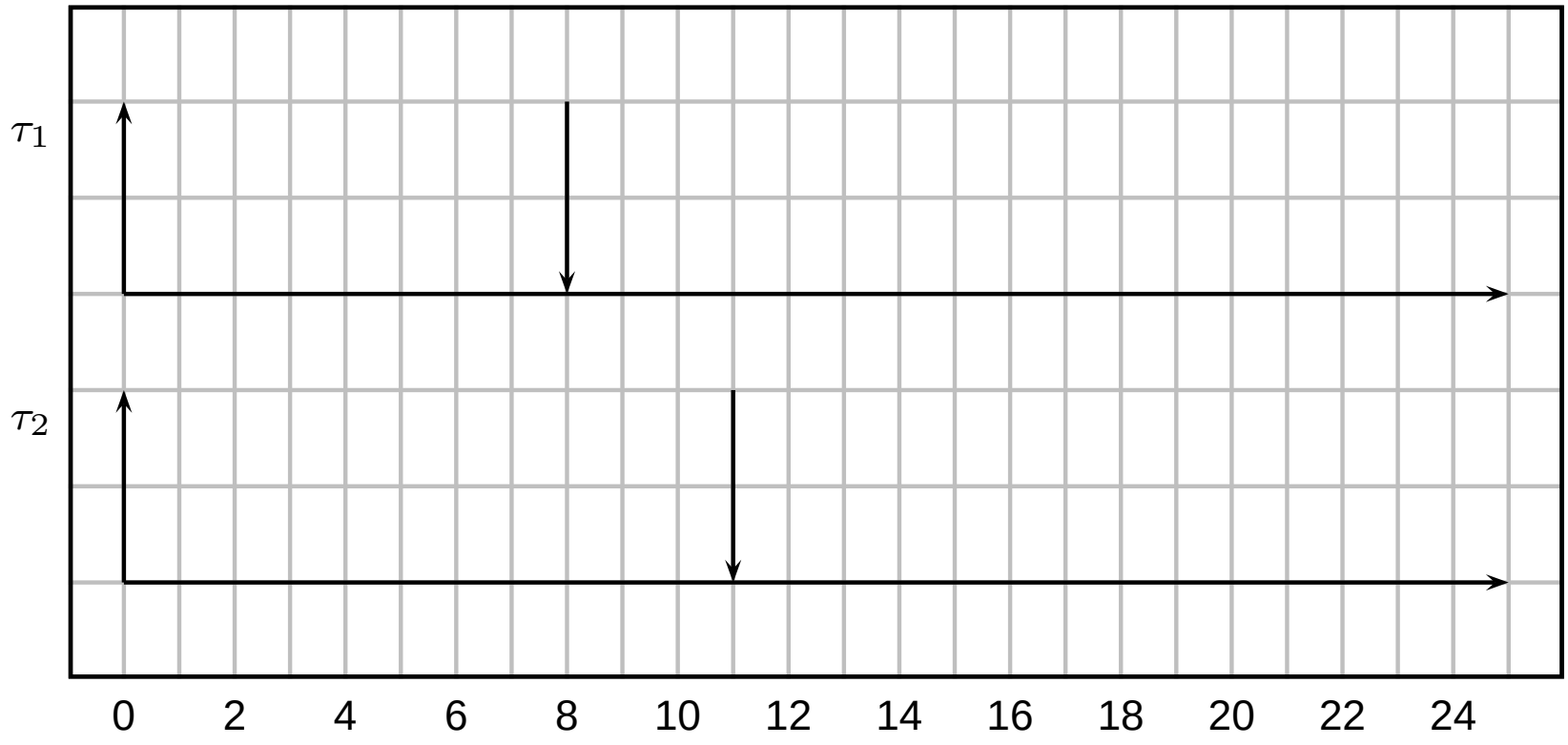
An Example – RM

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



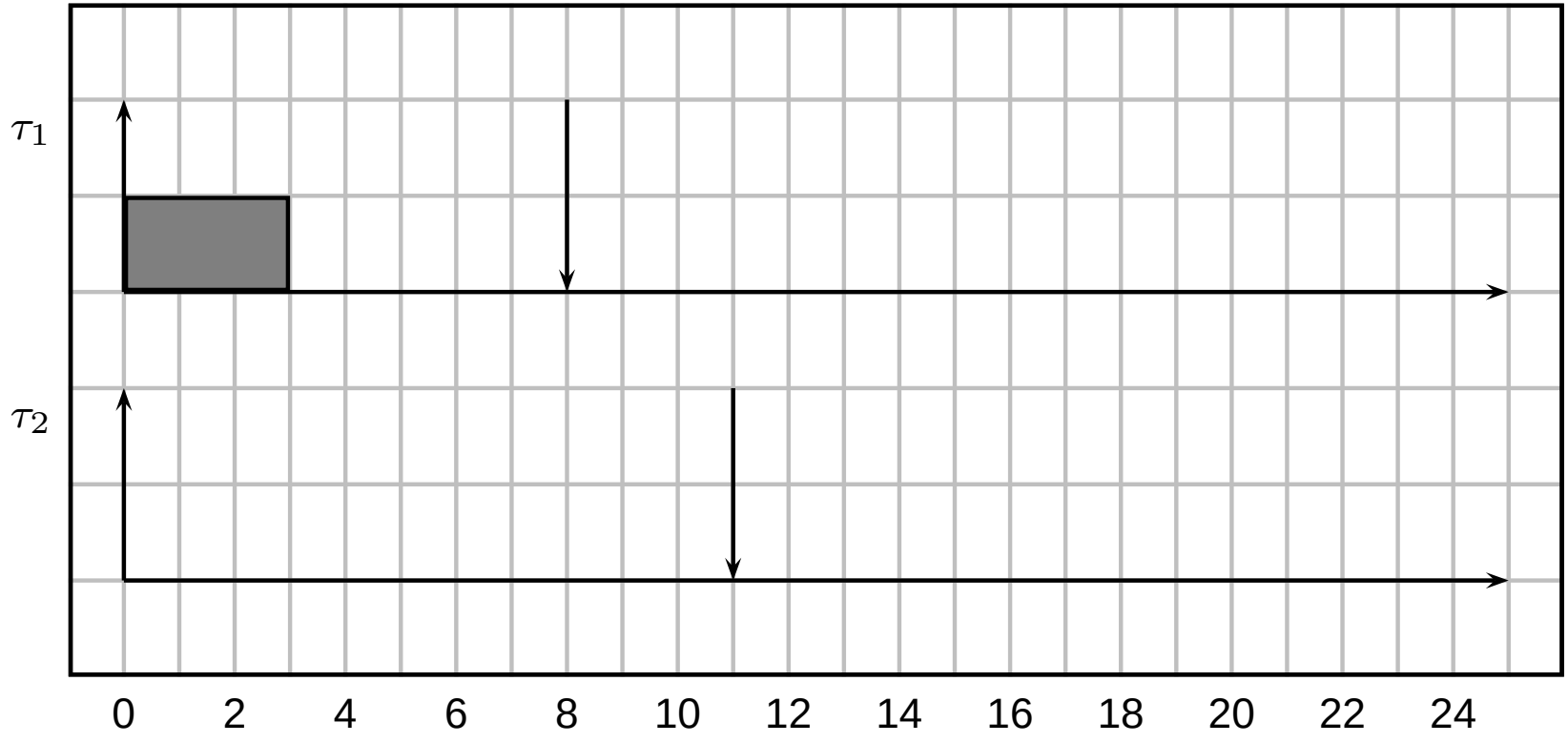
The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



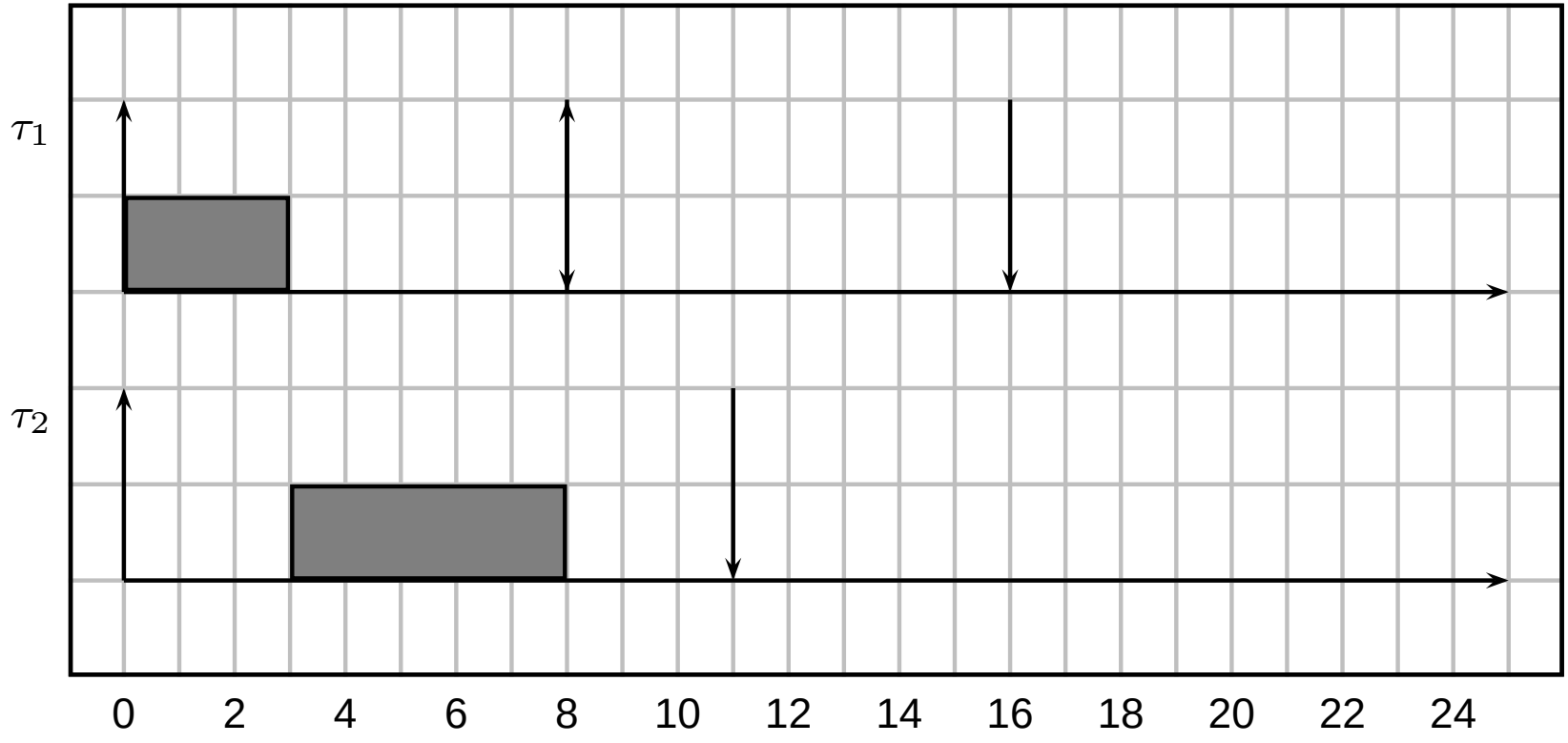
The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



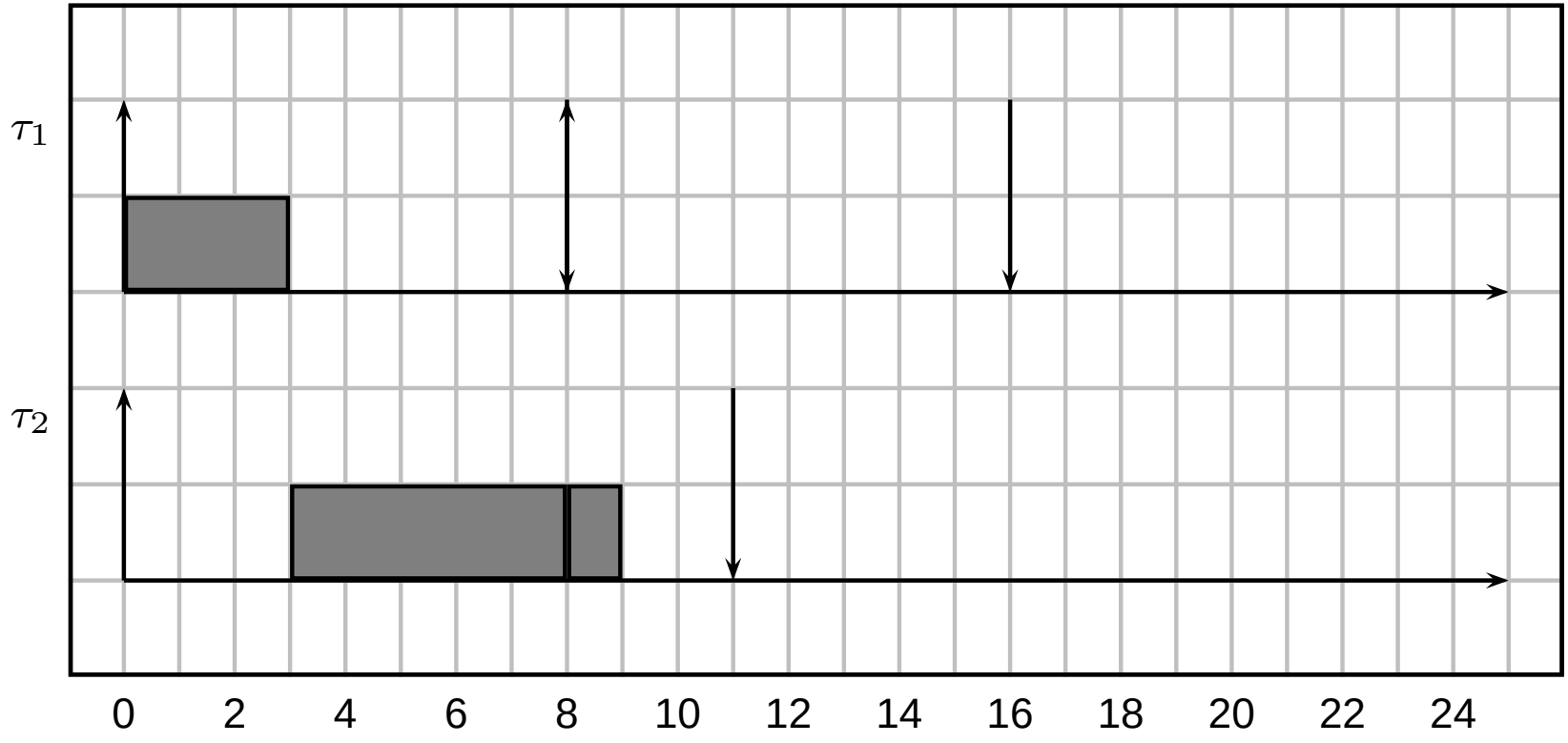
The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



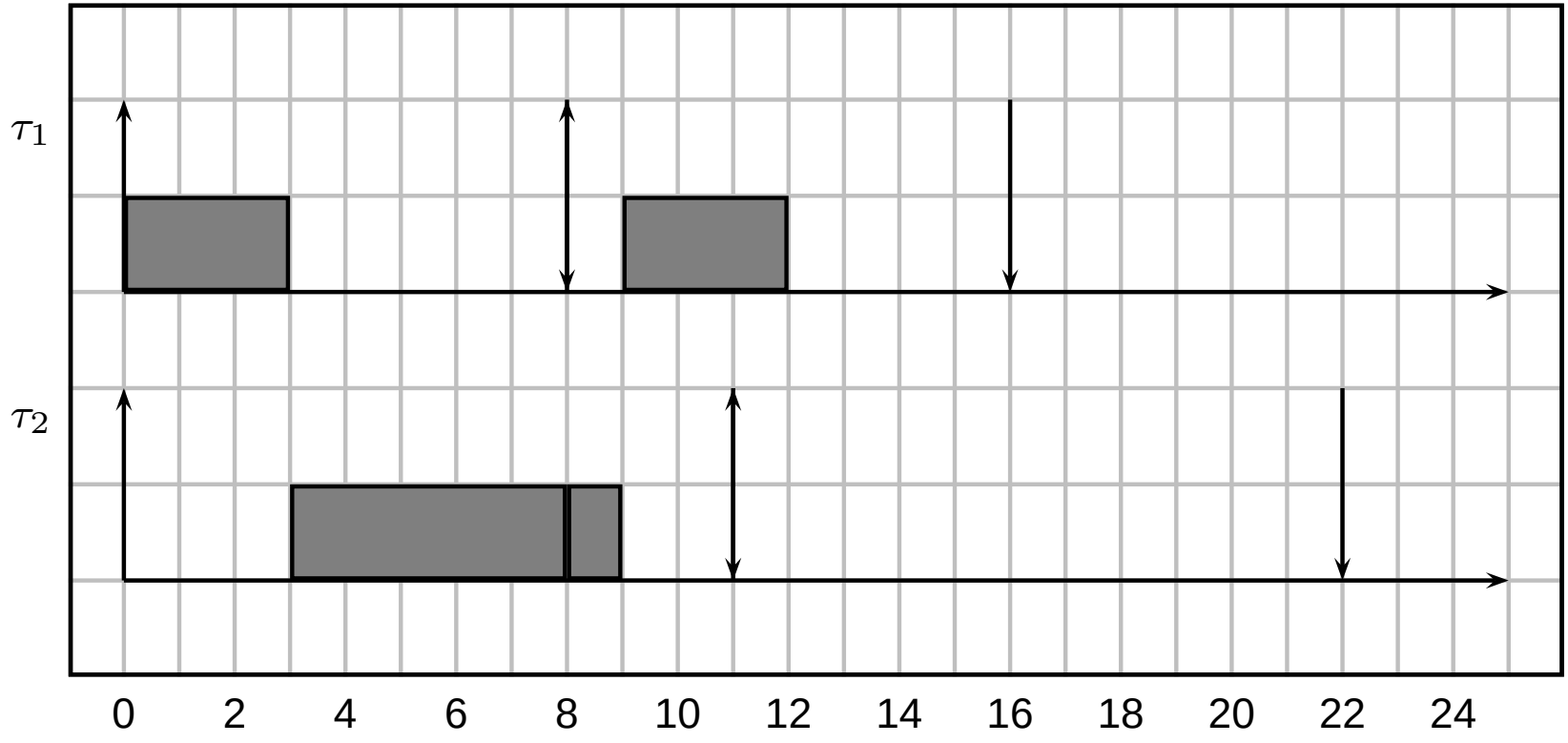
The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



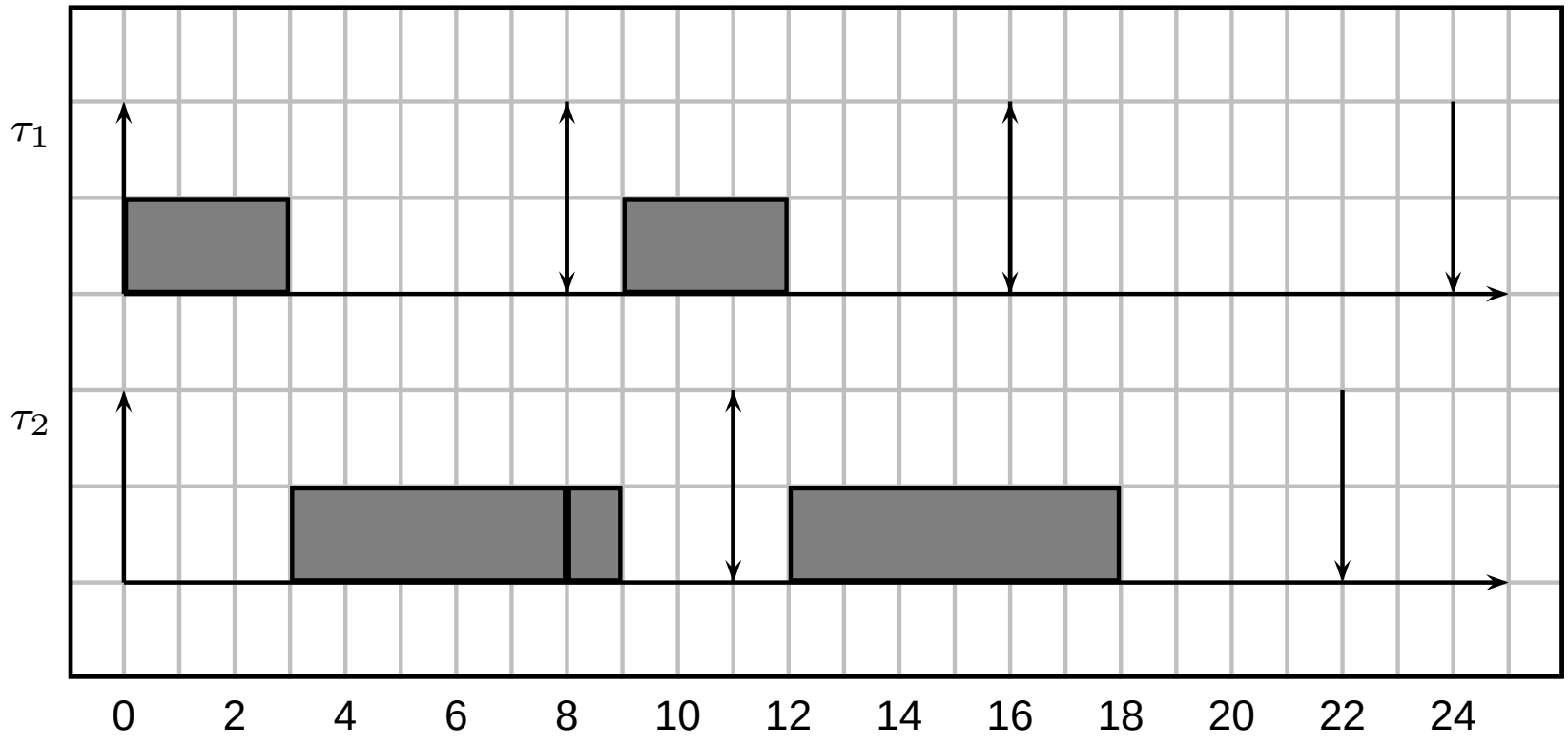
The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



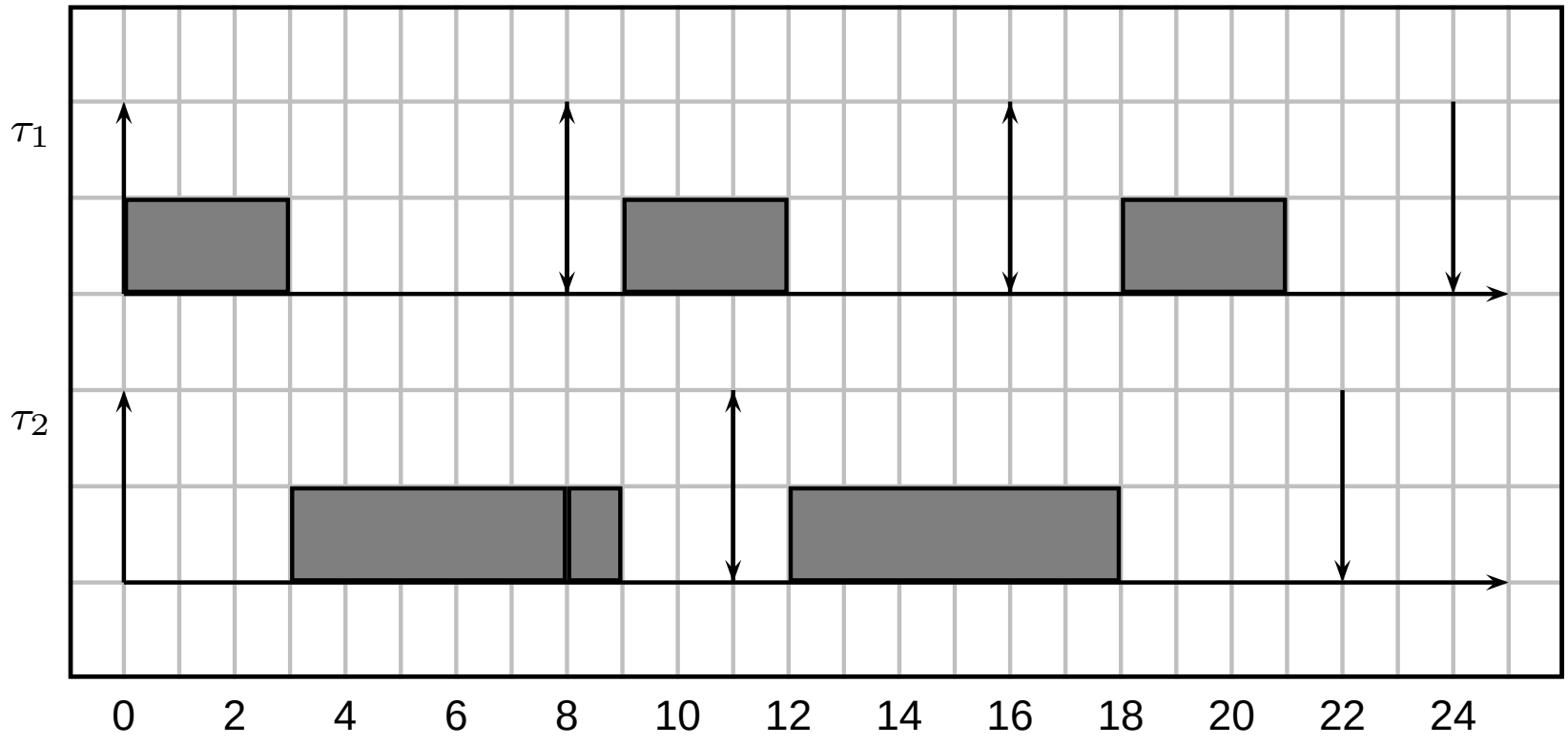
The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



The Same Example – EDF

- $\tau_1 = (3, 8, 8), \tau_2 = (6, 11, 11) \Rightarrow U = 0.92$



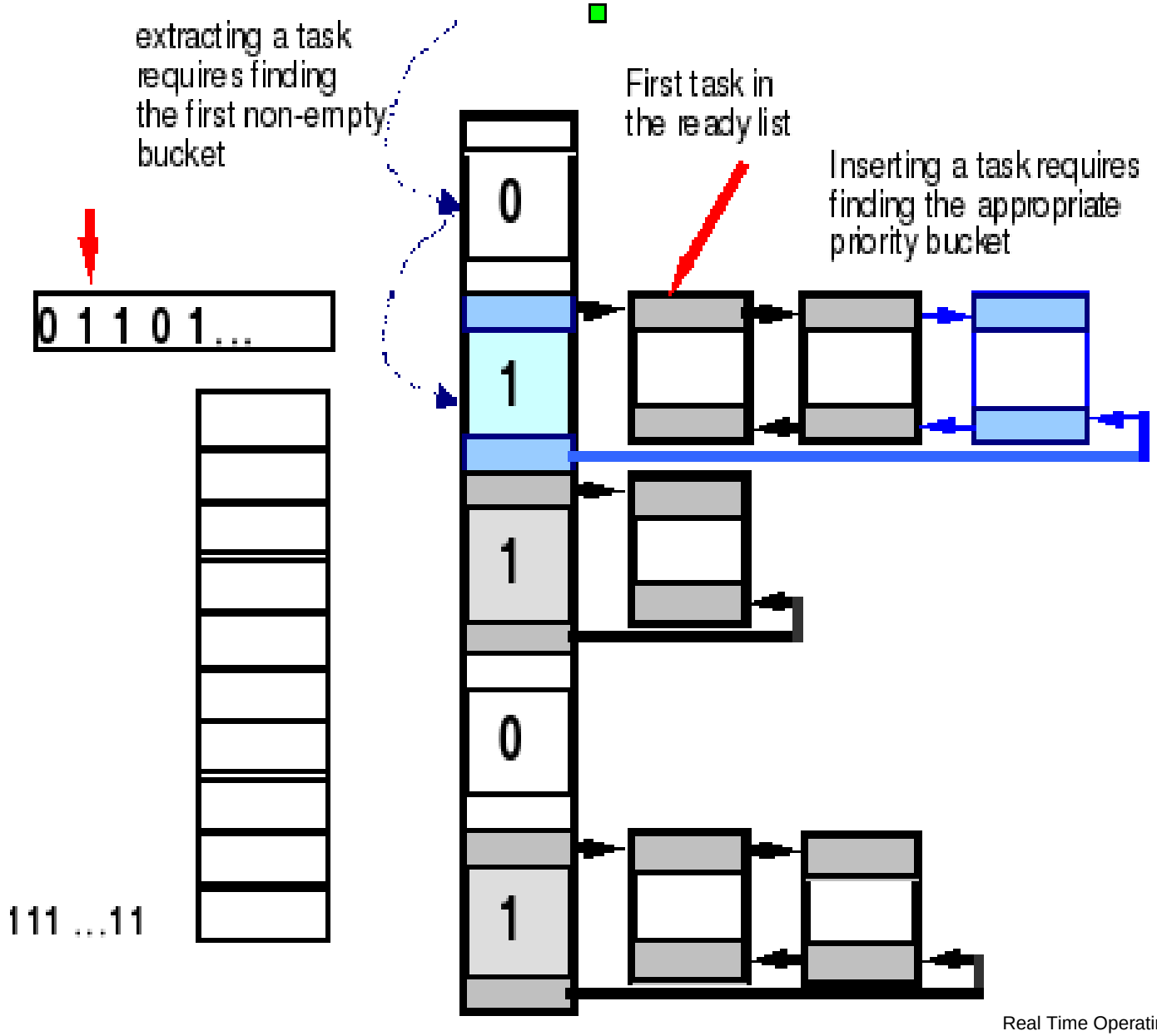
A Legitimate Question

- Why don’t commercial RT kernels use EDF?
- Potential problems:
  - Absolute deadlines change for each new task instance, therefore the priority needs to be updated every time the task moves back to the ready queue
  - Absolute deadlines are always increasing, how can we associate a (finite) priority value to an ever-increasing deadline value
  - Absolute deadlines are impossible to compute a-priori (there are infinitely many). Do we need infinitely many priority levels?
  - Less predictability in overload conditions

Implementation of Fixed Priorities

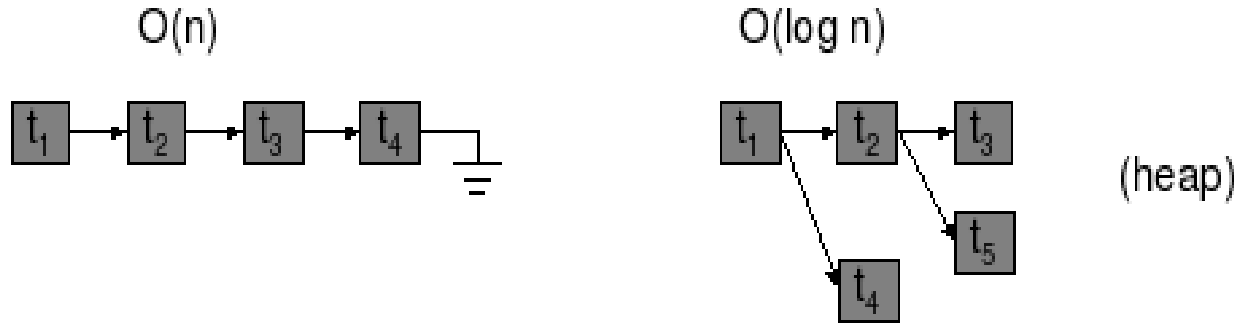
- When implementing fixed priority it is possible to have an array of queues (one for each priority level)
- Insertion into the queue is $O(1)$ operation
- Extracting from the queue would entail $O(n)$ search on the different priority levels to find the first nonempty queue
- However, we can use a bitmap (i.e., an array of bits) to tag the queues that are non-empty
- Extraction becomes $O(1)$ if we have a microinstruction that returns the first 1 bit in a word (CLZ)
- If not we can use a table to implement the operation $\lceil \log w \rceil$, but we need as many entries as the bits in the table

Implementation of fixed priority - I



EDF Queueing

- EDF can only use $O(n)$ or $O(\log(n))$ queueing



- In principle Queueing could be a bottleneck but only for the limited class of applications for which n is very large