EECS 223
Real-Time Computer Systems

Week 2

Round Robin Scheduler

- Given a set of tasks (jobs), all jobs are entered in a queue
  - Jobs are executed for a fixed quantum then returned to the queue
  - Some jobs may be given more than one quantum (weighted round robin)
- It guarantees fairness, not response time
  - Admission control can be used to bound the response time

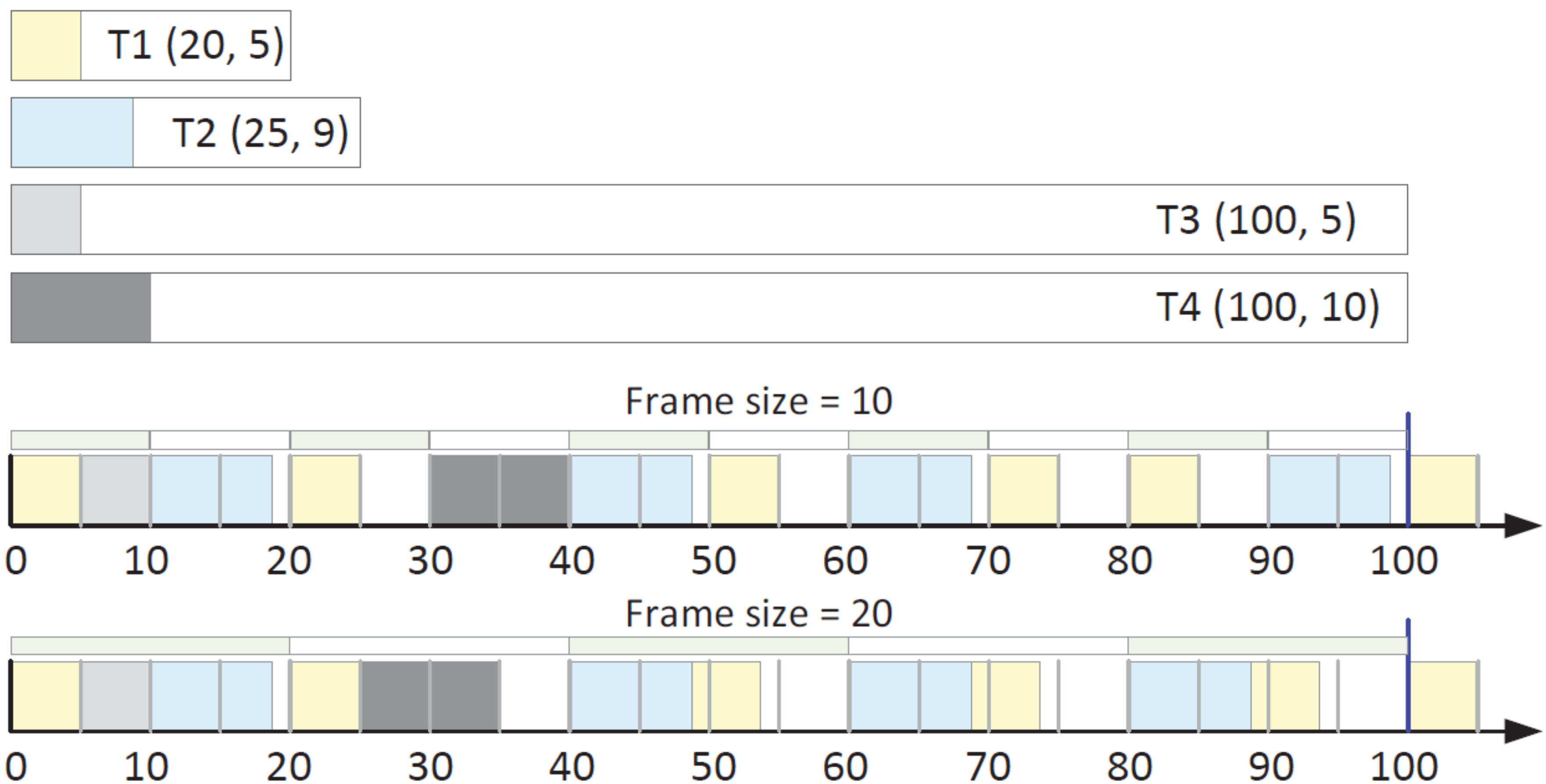
Cyclic Scheduling

- *Cyclic scheduling* is an approach to scheduling periodic tasks according to a *static schedule table* pre-computed off-line. The schedule table specifies exactly when each job of a periodic
- When there is no periodic job ready, the processor can take the time to execute aperiodic jobs.

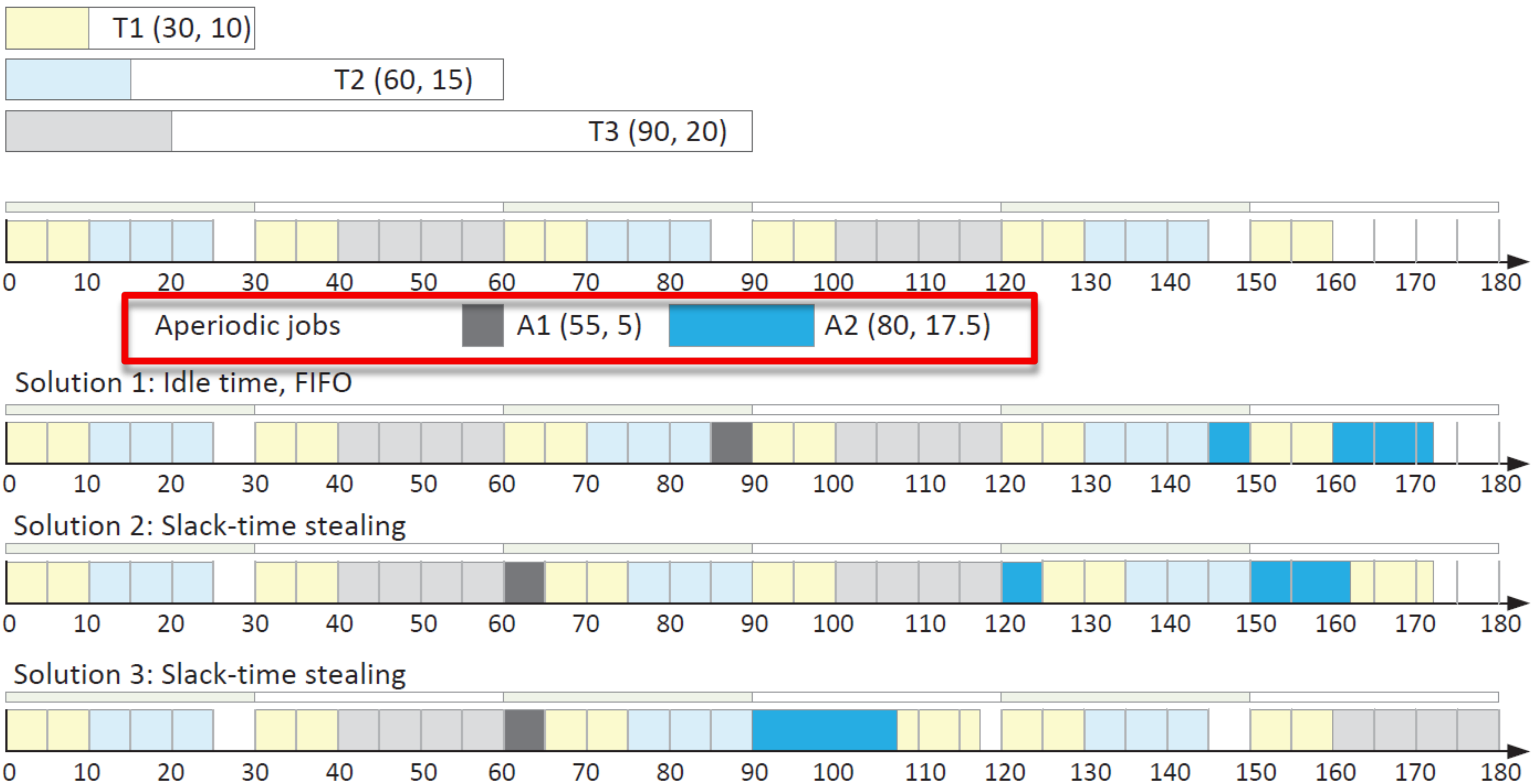
Cyclic Scheduling

- Given a set of tasks (jobs), produce the schedule table offline or before execution.
  - Exhaustive search may be used to find optimal schedules offline
  - Online table construction is possible
- During run time, the schedule table is followed to start and stop jobs
  - The table length is the major cycle, which is divided into frames
- Job executions are guaranteed
- Systems may be left idle if jobs finish earlier

Frame-based scheduling: An example



Scheduling Aperiodic Jobs

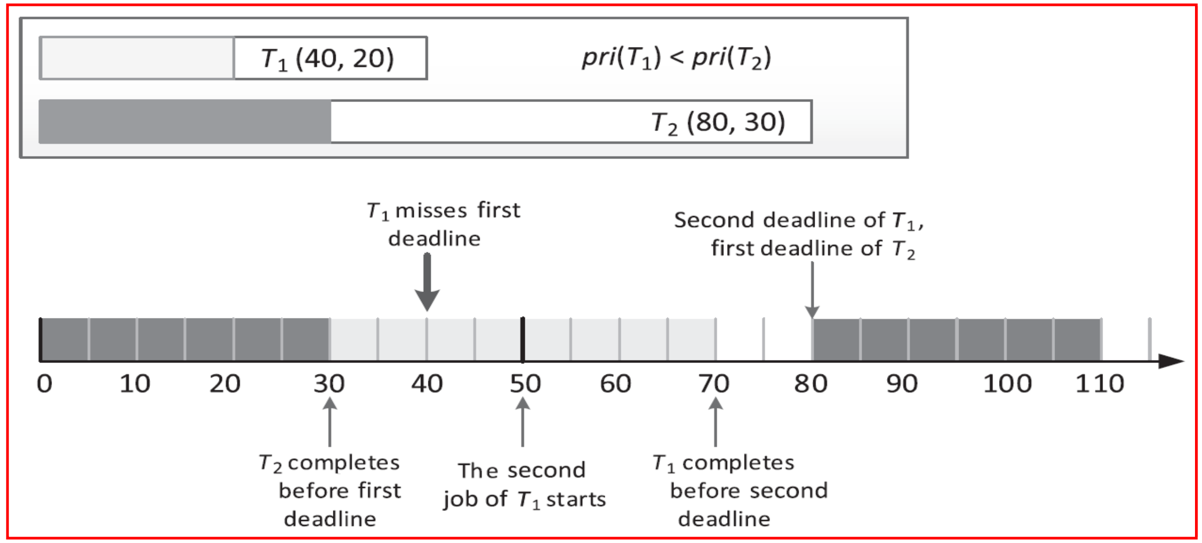


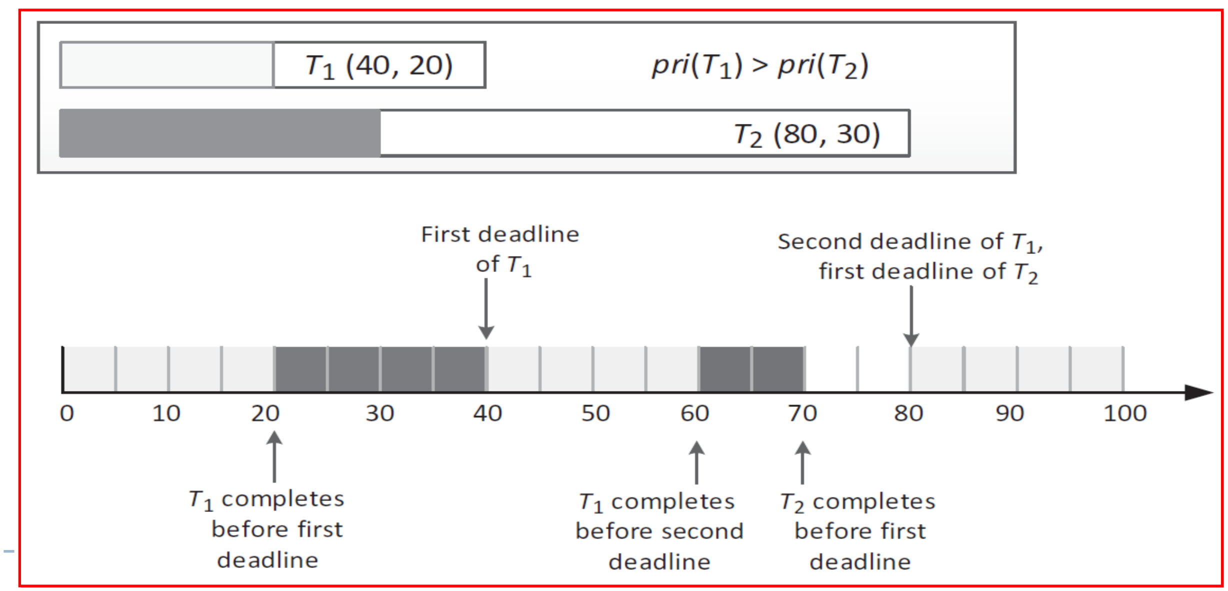
Priority Driven for Periodic Tasks: RM & EDF

- Preemptive Rate Monotonic: we compare the periods at run time
  - Priority $\propto$ rate of a task = 1/period
  - (1,3) > (2, 5) > (4, 20)
  - RM has static priorities
- Preemptive Earliest Deadline First: we compare the deadlines at run time
  - Deadline of a job makes its priority
  - The earlier the deadline, the higher the priority
  - EDF has dynamic priorities

Fixed Priority Assignment

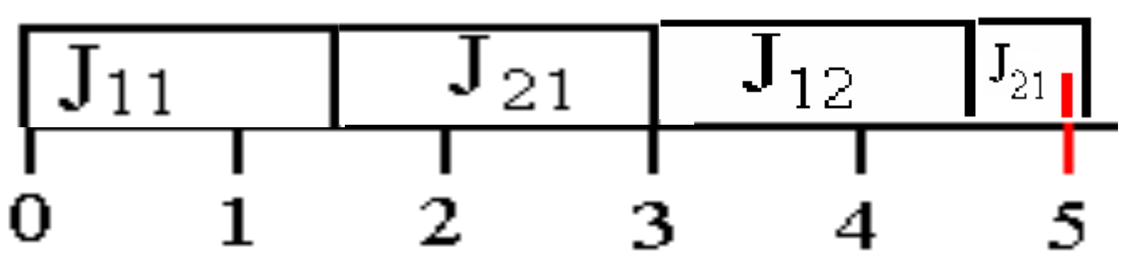
$$\mathbb{T}_1 = \begin{cases} T_1 = (40, 20), \\ T_2 = (80, 30). \end{cases}$$





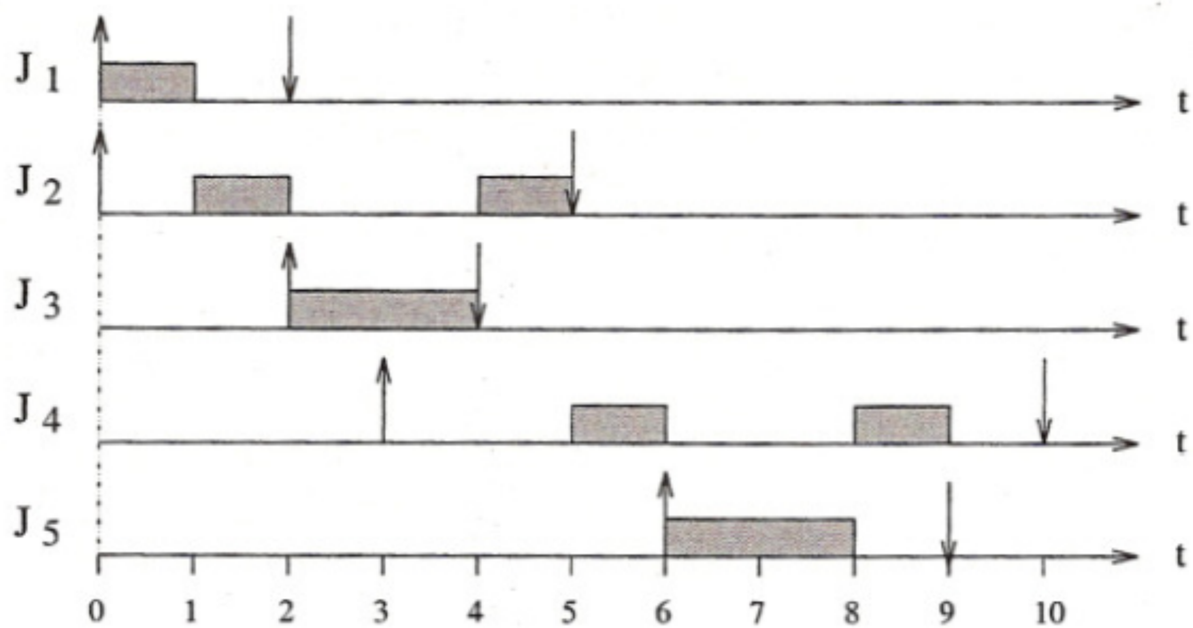
Schedulability by Utilization Analysis

- Can Rate Monotonic guarantee a schedule if the total utilization is ≤ 1?
- See the following example
  - Let $\tau_1 = 1.51 / 3$, $\tau_2 = 2 / 5$, total utilization is 0.903
  - τ_1 has a higher priority than τ_2
  - τ_2 misses deadline at t=5



(Aperiodic) Earliest Deadline First

	J_1	J_2	J_3	J_4	J_5
a_i	0	0	2	3	6
C_i	1	2	2	2	2
d_i	2	5	4	10	9



EDF Optimality

- Given a set of tasks $G = \{T_l\}$, if *any* A can meet all deadlines, EDF can do it as well.
  - Consider no context switch time or system overload, with infinity priority labels.

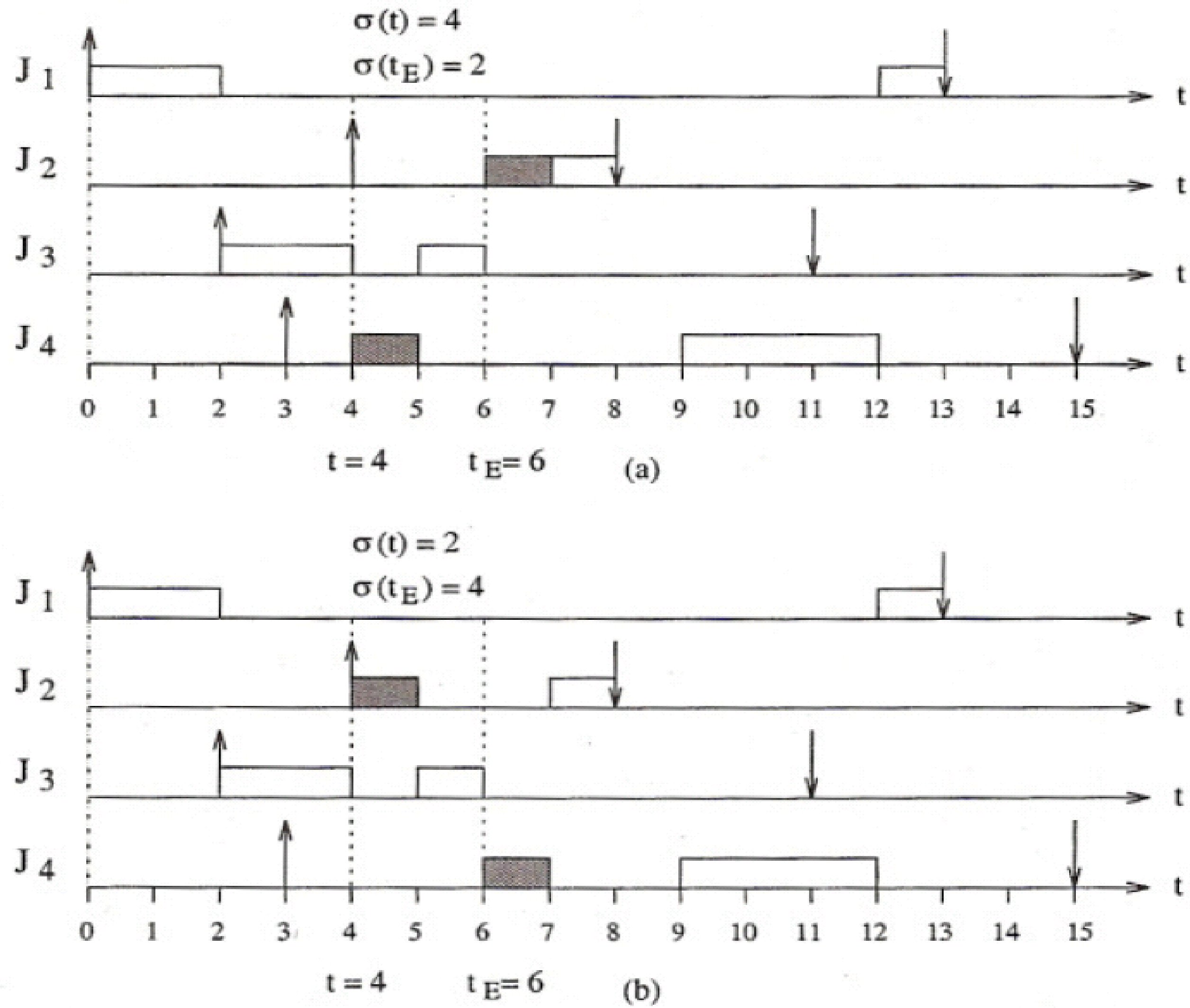


Figure 3.4 Proof of the optimality of the EDF algorithm. **a.** schedule σ at time $t = 4$. **b.** new schedule obtained after a transposition.

Utilization of Periodic Task

- The utilization of a periodic task is defined as $U_i = e_i / P_i$
  - A task with 1 / 4 uses 0.25 CPU time
  - A system with 4 tasks of $U_i = 0.25$ has a total utilization of 1
- No system can meet all tasks' deadline if the total utilization is >1

EDF Schedulability Bound

- An EDF schedule S has a job misses its deadline at t if and only if $\sum u_i > 1$
  - Don't need to worry about tasks with ready times after T_i and deadlines after T_i, since in EDF they won't compete for CPU time.

EDF Proof

- How much execution time is needed, total for T_i ?

$$\frac{t-\varphi_i}{P_i}e_i$$

- For the rest of the tasks?

$$\sum e_j \left\lfloor \frac{t-\varphi_j}{P_j} \right\rfloor$$