Finding the Critical Instant

- A critical instant for a job is the worst-case release time for that job, taking into account all jobs that have higher priority
  - i.e. a job released at the same instant as all jobs with higher priority are released, and must wait for all those jobs to complete before it executes
  - The response time of a job in T_i released at a critical instant is called the *maximum (possible) response time*, and is denoted by W_i

- The schedulability test involves checking each task in turn, to verify that it can be scheduled when started at a critical instant
  - If schedulable at all critical instants, will work at other times
  - More work than the test for maximum schedulable utilization, but less than an exhaustive simulation

Finding the Critical Instant

- A critical instant of a task T_i is a time instant such that:

  If $w_{i,k} \leq D_{i,k}$ for every $J_{i,k}$ in T_i then

  The job released at that instant has the maximum response time of all jobs in T_i

  and $W_i = w_{i,k}$

  else if $\exists\, J_{i,k} : w_{i,k} > D_{i,k}$ then

  The job released at that instant has response time > D

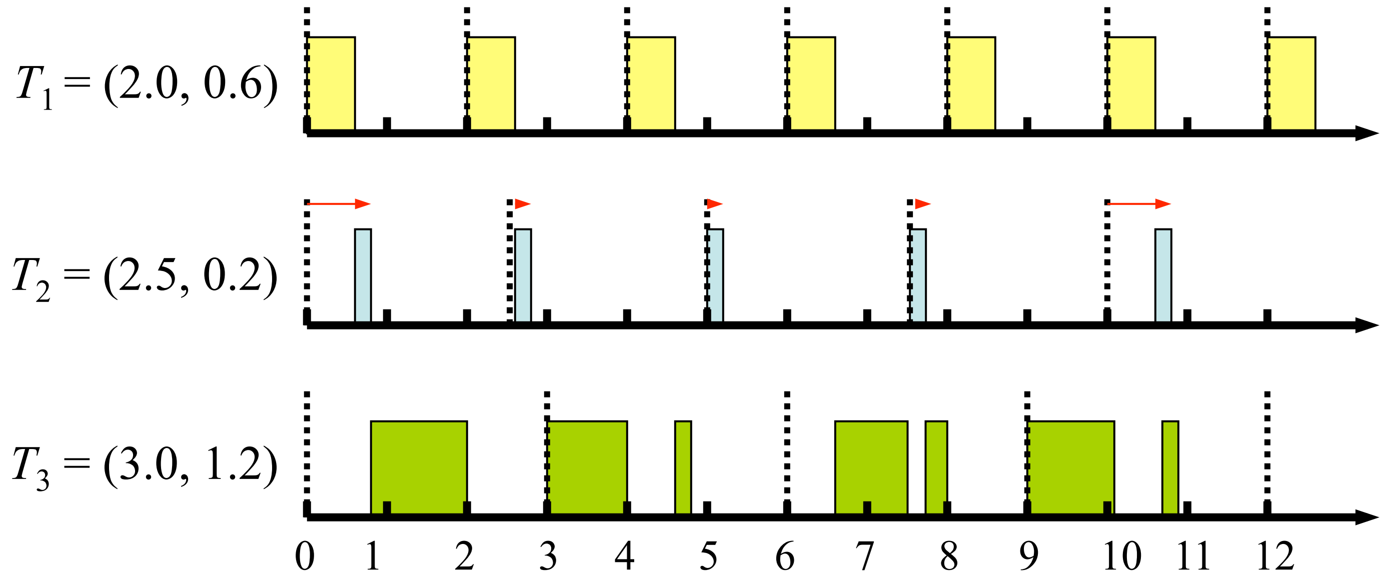
  where $w_{i,k}$ is the response time of the job

All jobs meet deadlines, but this instant is when the job with the slowest response is started

If some jobs don't meet deadlines, this is one of those jobs

- Theorem: In a fixed-priority system where every job completes before the next job in the same task is released, a critical instant occurs when one of its jobs $J_{i,c}$ is released at the same time with a job from every higher-priority task.
  - Intuitively obvious, but proved in the book

Finding the Critical Instant: Example



- 3 tasks scheduled using rate-monotonic
- Response times of jobs in T_2 are:

  $r_{2,1} = 0.8$, $r_{2,3} = 0.3$, $r_{2,3} = 0.2$, $r_{2,4} = 0.3$, $r_{2,5} = 0.8$, …

  Therefore critical instants of T_2 are $t = 0$ and $t = 10$

Using the Critical Instant

- Having determined the critical instants, show that for each job $J_{i,c}$ released at a critical instant, that job and all higher priority tasks complete executing before their relative deadlines
- If so, the entire system be schedulable…

- That is: don't simulate the entire system, simply show that it has correct characteristics following a critical instant
  - This process is called *time demand analysis*

Time-Demand Analysis

- Compute the total demand for processor time by a job released at a critical instant of a task, and by all the higher-priority tasks, as a function of time from the critical instant
- Check if this demand can be met before the deadline of the job:
  - Consider one task, T_i, at a time, starting highest priority and working down to lowest priority
  - Focus on a job, J_i, in T_i, where the release time, t_0, of that job is a critical instant of T_i
  - At time $t_0 + t$ for $t \geq 0$, the processor time demand $w_i(t)$ for this job and all higher-priority jobs released in $[t_0, t]$ is:

$$\underbrace{w_i(t) = e_i}_{\text{Execution time of job } J_i} + \underbrace{\sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k}_{\text{Execution time of higher priority jobs started during this interval}} \quad \text{for } 0 < t \leq p_i$$

$w_i(t)$ = the time-demand function

Time-Demand Analysis

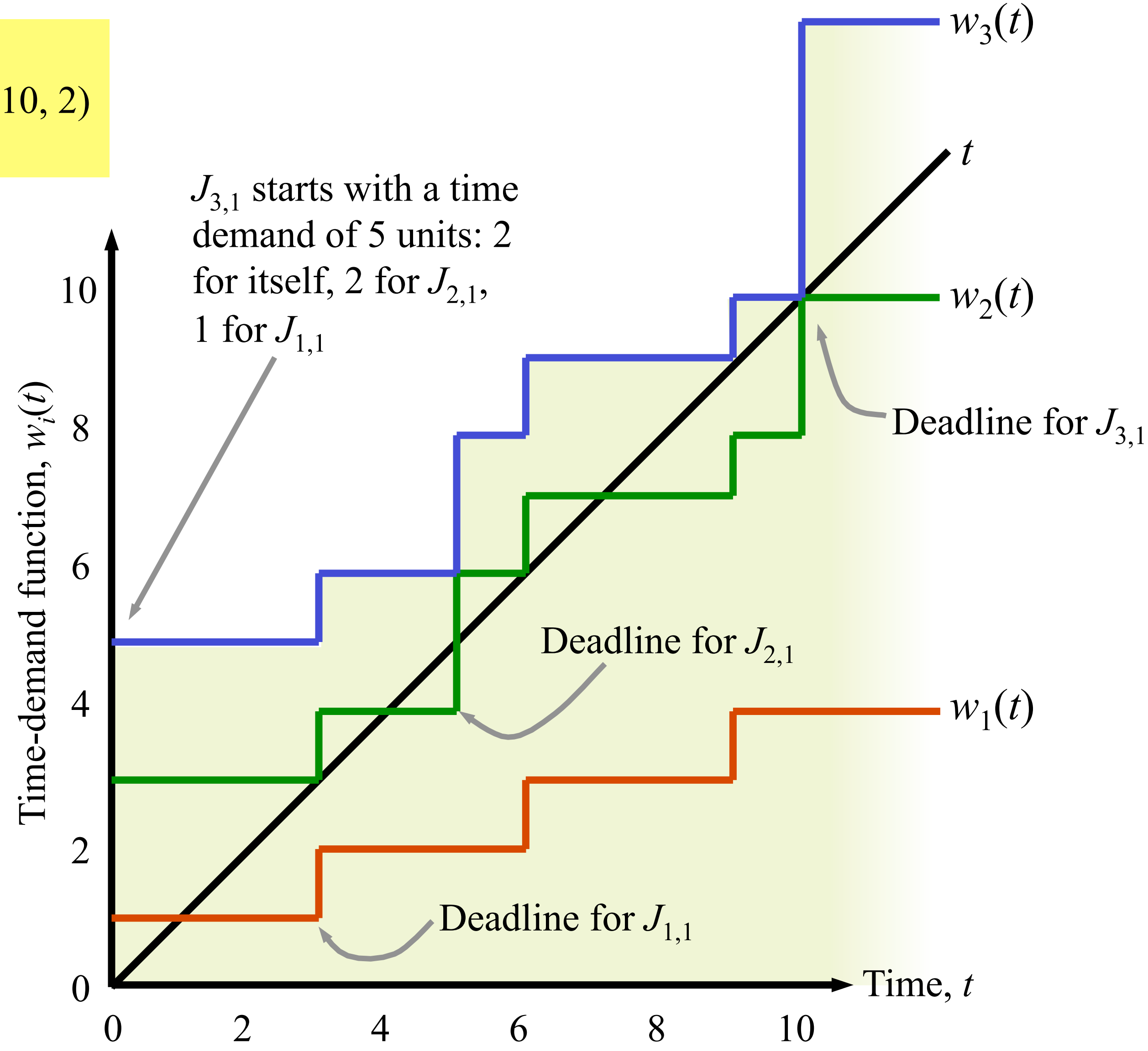
- Compare the time demand, $w_i(t)$, with the available time, t:
  - If $w_i(t) \leq t$ for some $t \leq D_i$, the job, J_i, meets its deadline, $t_0 + D_i$
  - If $w_i(t) > t$ for all $0 < t \leq D_i$ then the task probably cannot complete by its deadline; and the system likely cannot be scheduled using a fixed priority algorithm
    - Note that this is a sufficient condition, but not a necessary condition. Simulation may show that the critical instant never occurs in practice, so the system could be feasible…
- Use this method to check that all tasks are schedulable if released at their critical instants; if so conclude the entire system can be scheduled

Time-Demand Analysis: Example

Rate Monotonic:
$T_1 = (3, 1)$, $T_2 = (5, 2)$, $T_3 = (10, 2)$
$U = 0.933$

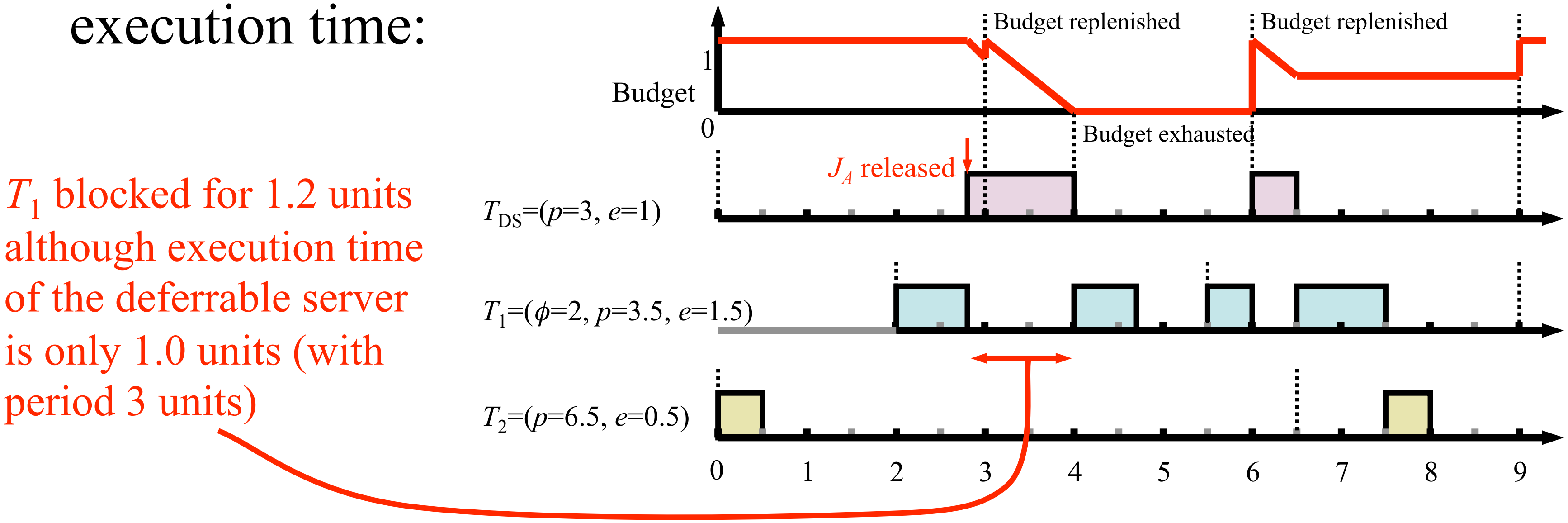
The time-demand functions $w_1(t)$, $w_2(t)$ and $w_3(t)$ are not above t at their deadline ⇒ system can be scheduled

Exercise: simulate the system to check this!



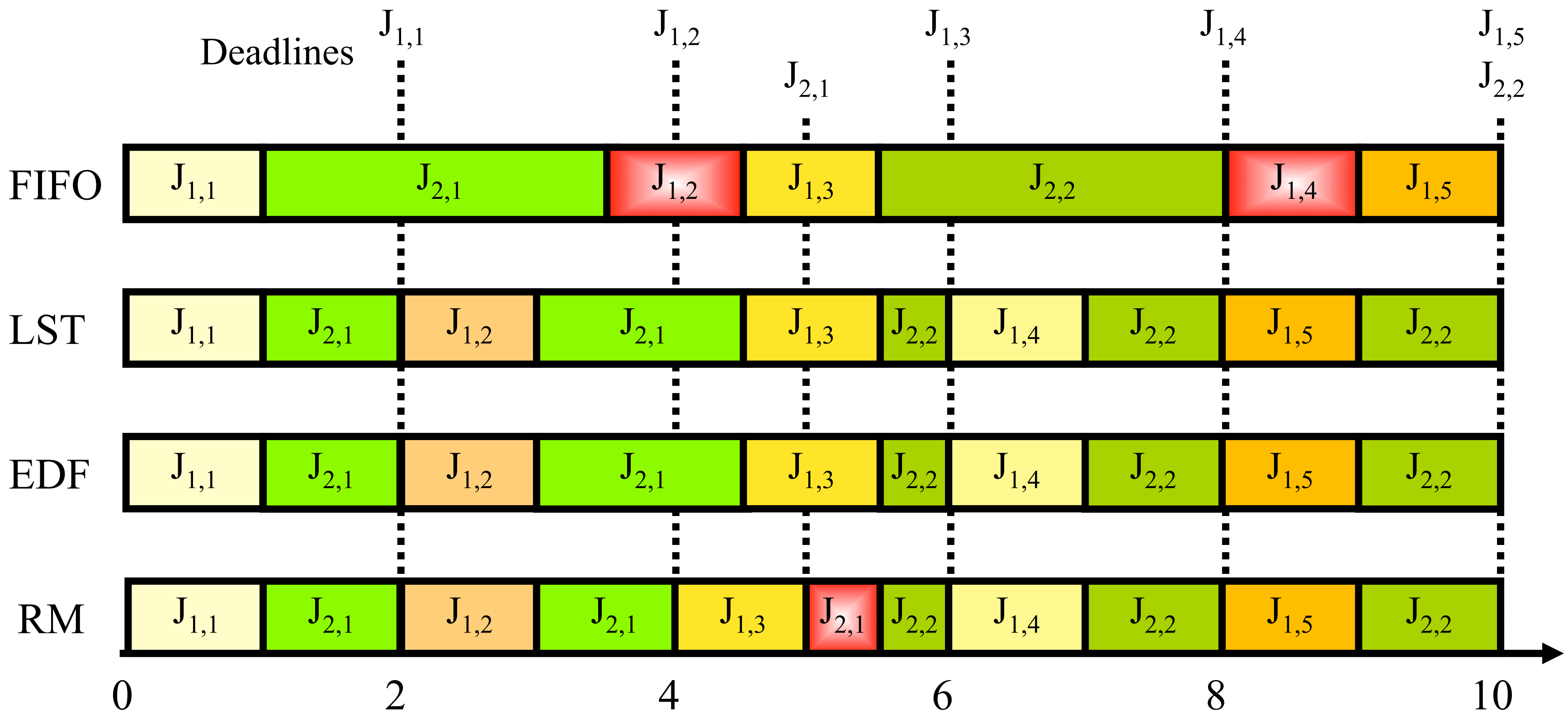
Limitations of Deferrable Servers

- Limitation of deferrable servers – they may delay lower-priority tasks for more time than a periodic task with the same period and execution time:



- A *sporadic server* is designed to eliminate this limitation
  - A different type of bandwidth preserving server
  - More complex consumption and replenishment rules ensure that a sporadic server with period p_S and budget e_S never demands more processor time than a periodic task with the same parameters

Example: RM, EDF, LST and FIFO



- Demonstrate by exhaustive simulation that LST and EDF meet deadlines, but FIFO and RM don’t