Scheduling Aperiodic & Sporadic Jobs in Priority-Driven Systems

Chapter 7

Recap of terminology

- *Aperiodic* jobs
  - Arrival times & execution times unknown
  - Soft or no deadlines
- *Sporadic* jobs
  - Arrival times unknown
  - Execution time and deadline known upon arrival
  - Hard deadlines
- *Periodic* tasks
  - Arrival pattern & execution time known a-priori
  - Hard deadlines

Assumptions & Goals

- Periodic tasks *schedulable* by themselves
  - I.e., in the absence of aperiodic & sporadic jobs
- All jobs are *independent* and fully *preemptable*

- Accept/reject *sporadic* jobs upon arrival
  - Accepted sporadic jobs complete within deadline
- Aim to complete *aperiodic* jobs as soon as possible
  - Minimizing response time of first job in aperiodic queue

OR

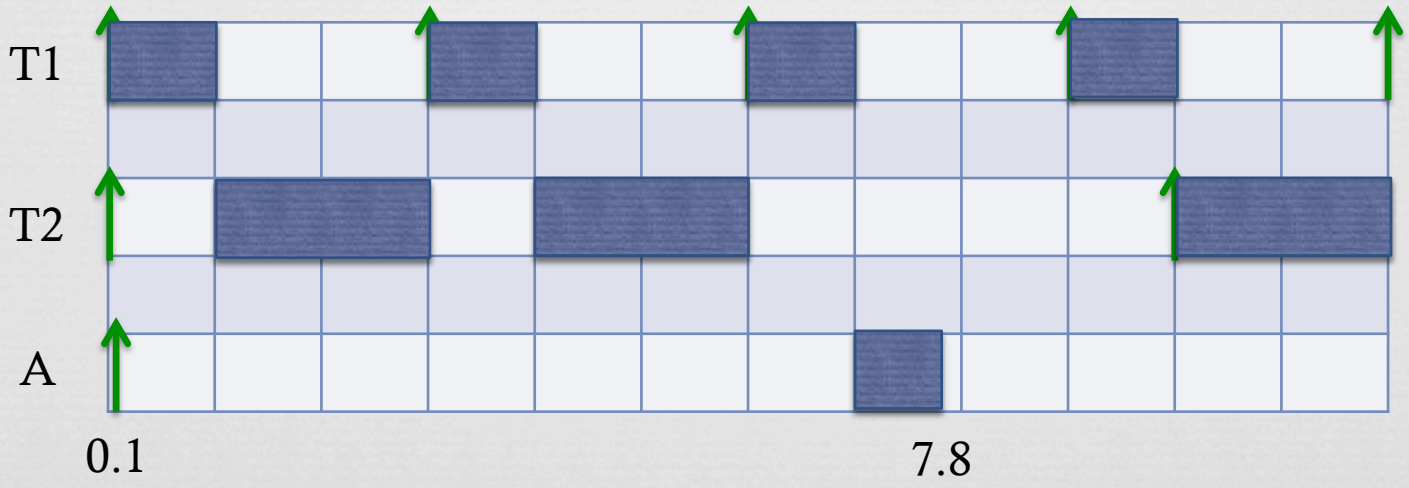
  - Minimizing average response time of all aperiodic jobs

Scheduling aperiodic jobs

- Let's start with only periodic tasks & aperiodic jobs
  - I.e., no sporadic jobs

Background scheduling

- Schedule aperiodic job *only* when no *active* periodic job
- E.g., consider the following system
  - Periodic tasks T1 (3, 1), T2 (10, 4), scheduled with RM
  - Aperiodic job A arrives at 0.1 & has exec time 0.8



Discussion

- Advantages
  - Produces *correct* schedule
  - Very *simple*
- Disadvantages
  - Not optimal for aperiodic jobs – *long* response times
- How can we improve things for aperiodic jobs?
  - *Interrupt* periodic job whenever aperiodic job arrives…
  - …could be *incorrect* – periodic job deadline miss!
  - Should only interrupt if existence of *slack* guaranteed
    - Computing slack is difficult in priority-driven systems

Server-based scheduling

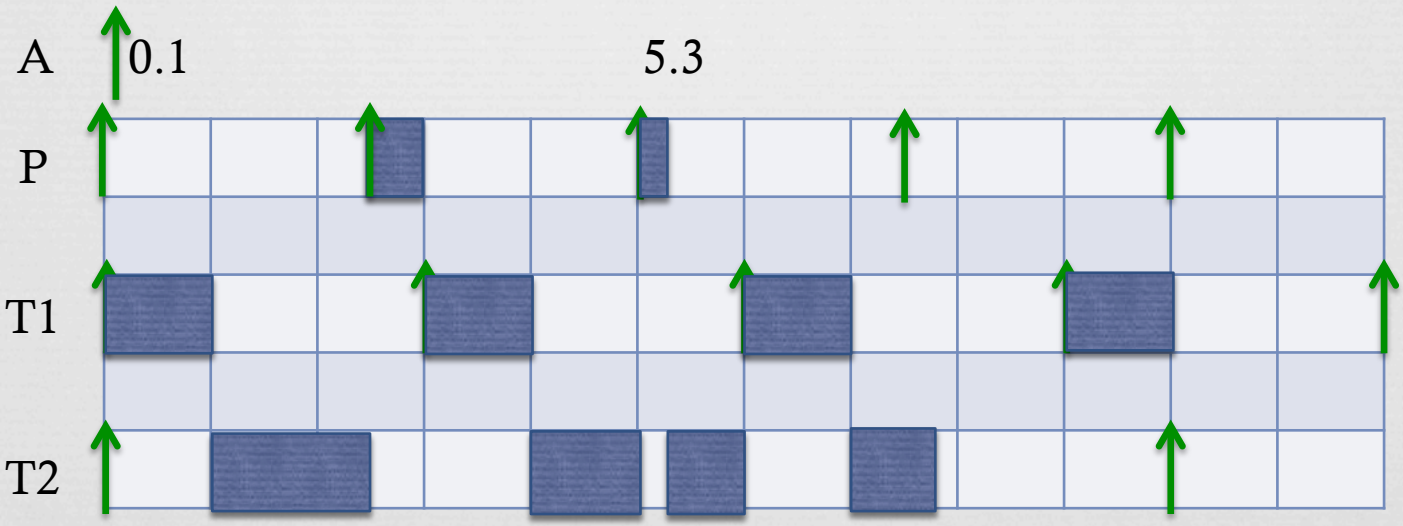
- Create special task called *periodic server*
  - Behaves more or less like periodic task
  - Assigned priority, period & execution *quota*/*budget*
  - When scheduled, server executes *aperiodic* jobs in queue
  - *Consumes* budget based on rules
  - If no budget or no aperiodic jobs, server *suspends* itself
  - Budget *replenished* based on rules
- Terminology
  - Server is *backlogged* when aperiodic job present in queue
  - Server is *idle* when no aperiodic jobs in queue
- Several types of periodic servers have been proposed…

Polling server

- Has period p_s and execution budget e_s
- When server is scheduled
  - It checks aperiodic job queue
  - If jobs found, they are executed for up to server *budget* e_s
  - If no jobs found, server suspends & *gives up* its budget
- Server budget *replenished* only at end of period

Example

- Consider the following system scheduled using RM
  - Periodic tasks T1 (3, 1), T2 (10, 4)
  - Polling server P (2.5, 0.5)
- Aperiodic job A arrives at 0.1 & has exec time 0.8



Discussion

- Advantages
  - Simple to prove *correctness*
- Disadvantages
  - If aperiodic job arrives after polling server suspends
    - Job must wait for *next* period
    - Server budget *wasted*

Bandwidth preserving server

- Type of periodic server
- Tries to *preserve* budget when no aperiodic jobs found
- Goal: improve response times of aperiodic jobs

- Several bandwidth preserving servers have been proposed
  - Differ in budget consumption & replenishment rules

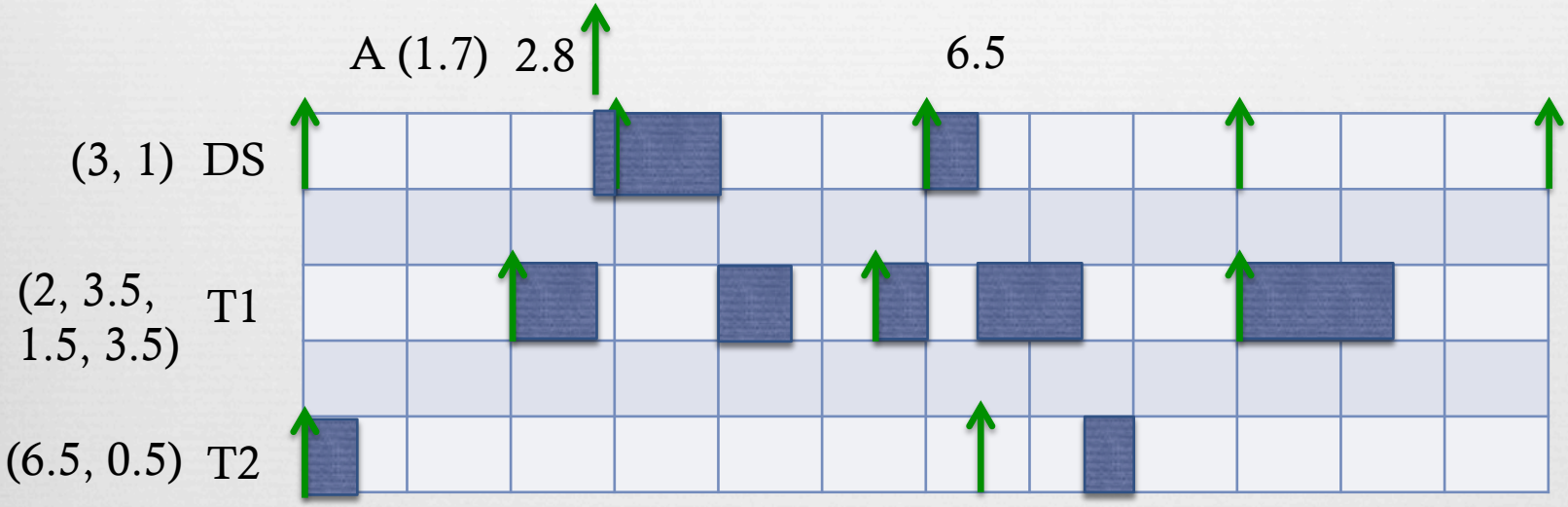
Deferrable server

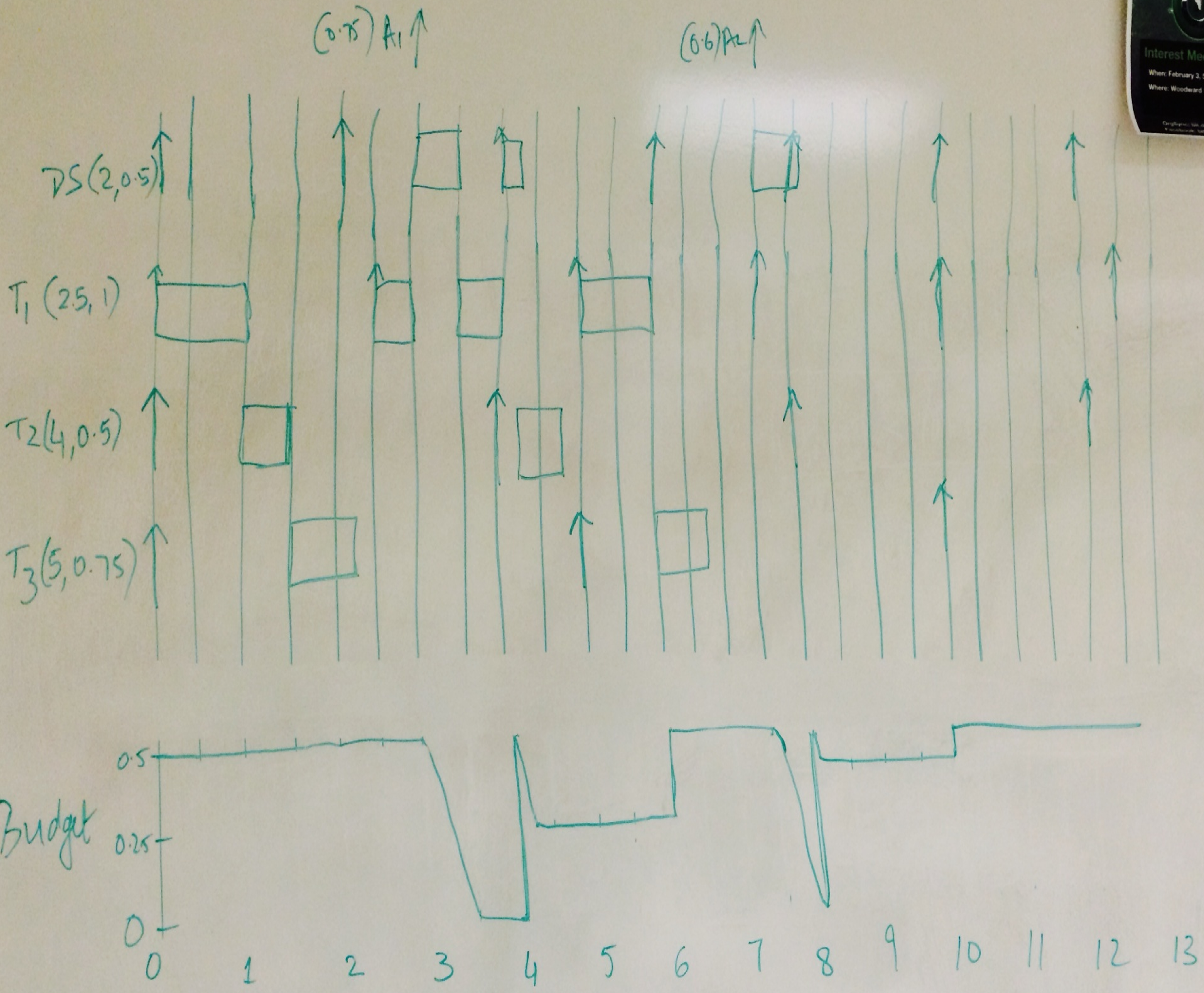
- Has period p_s and execution budget e_s
- Budget *consumption* rule
  - Consumes budget @ 1 per unit time whenever executed
- Budget *replenishment* rule
  - Budget replenished (to e_s) at multiples of period
- Budget *preservation*
  - Preserves budget till end of period in wait of aperiodic jobs
  - Budget cannot be carried forward to next period

Example

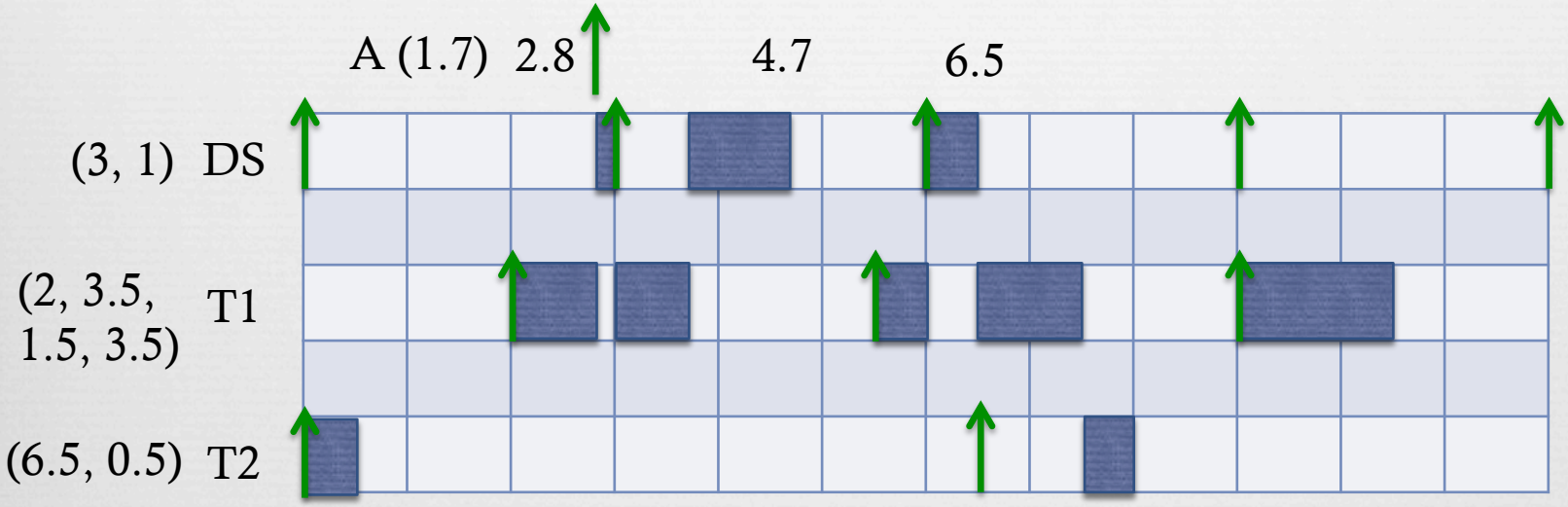
- Consider the following system
  - Periodic tasks T1 (2, 3.5, 1.5, 3.5), T2 (6.5, 0.5)
  - Deferrable server DS (3, 1)
  - Aperiodic job arrives at 2.8 & has exec time 1.7

Schedule using RM




(0.75) A1
(0.6) A2
DS(2,0.5)
T1 (2.5, 1)
T2(4,0.5)
T3(5,0.75)
Budget
0.5
0.25
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13

Schedule using EDF



Discussion

- Deferrable server may be scheduled for longer than its execution *budget* in a time interval equal to its *period*
- Lower priority task may be delayed *longer* than it would be by *periodic* task with same period & execution time
- I.e., deferrable server *does not* behave as periodic task

Schedulability analysis

- Time demand analysis may be used
- Assumes deferrable server is *highest priority* task

Theorem: In a fixed-priority system T in which $D_i \leq p_i$ for all i , with a deferrable server (p_s, e_s) with the highest priority among all tasks, a critical instant of every periodic tasks Ti occurs at a time t_0 when all of the following are true:

- One of its jobs $J_{i,c}$ is released at t_0
- A job in every higher-priority periodic task is released at t_0
- The budget of the server is e_s at t_0, one or more aperiodic jobs are released at t_0, and they keep the server backlogged hereafter
- The next replenishment time of the server is $t_0 + e_s$

$$w_i(t) = e_i + b_i + e_s + \left\lceil \frac{t - e_s}{p_s} \right\rceil \times e_s + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil \times e_k$$