Simulating a Real-Time Scheduler using OMNet++

Chang-Gun Lee (cglee@snu.ac.kr)

Assistant Professor

The School of Computer Science and Engineering

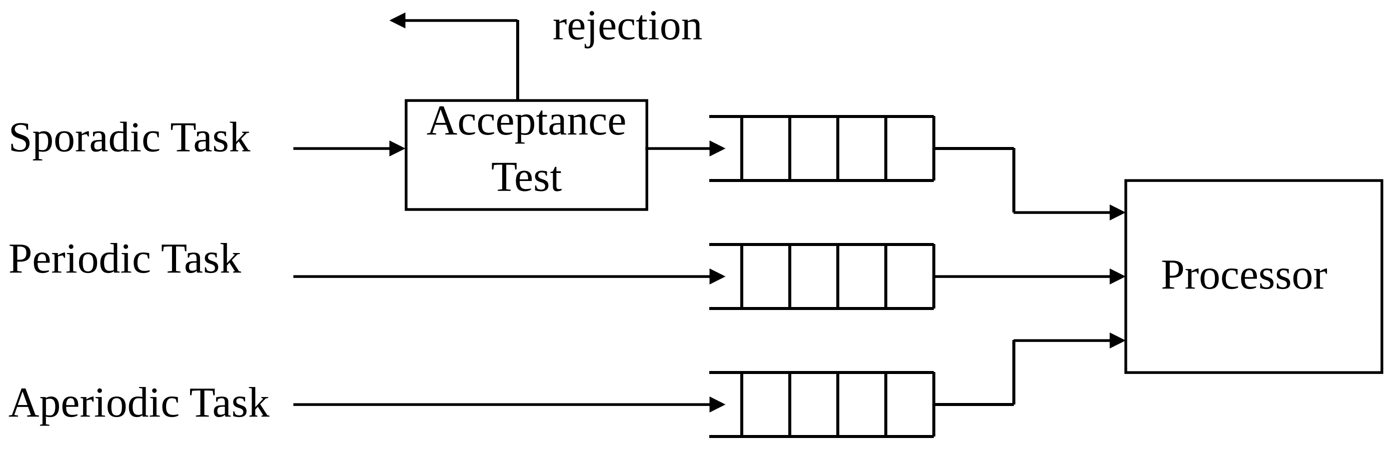
Seoul National University

Real-Time Scheduling Overview

- Tasks that need to be completed by specific deadlines are real-time tasks
  - Cell phones, PDAs
  - Digital cameras
  - Microwave ovens
  - Network adaptor box (e.g., ISDN adaptor)
  - Multimedia systems such as DVR, VOD server, etc
  - Factory process control
  - Radar systems
  - Avionics

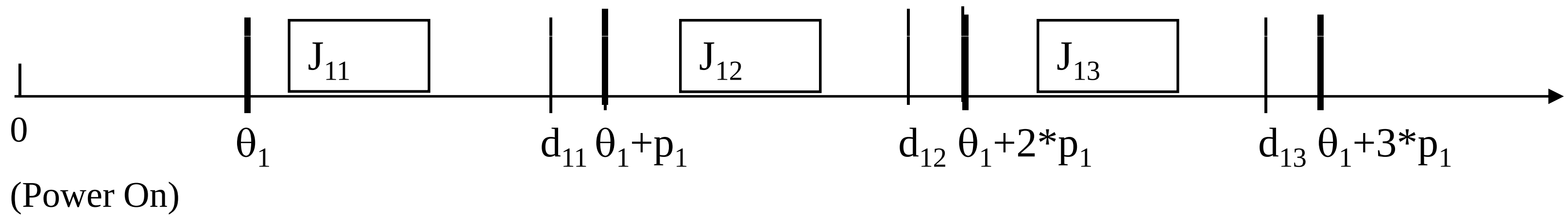
Types of Real-Time Tasks

- Periodic tasks
  - A Task that invokes the same job periodically
  - Usually have hard deadlines (equal to period)
- Non-Periodic Tasks
  - Soft aperiodic tasks:
    - random arrivals such as a Poisson distribution:
    - the execution time can also be random such as exponential distribution
    - typically it models users' requests.
  - Firm aperiodic tasks (Sporadic tasks):
    - there is a minimal separation between 2 consecutive arrivals
    - there is a worst-case execution time bound
    - models emergency requests such as the warning of engine overheat



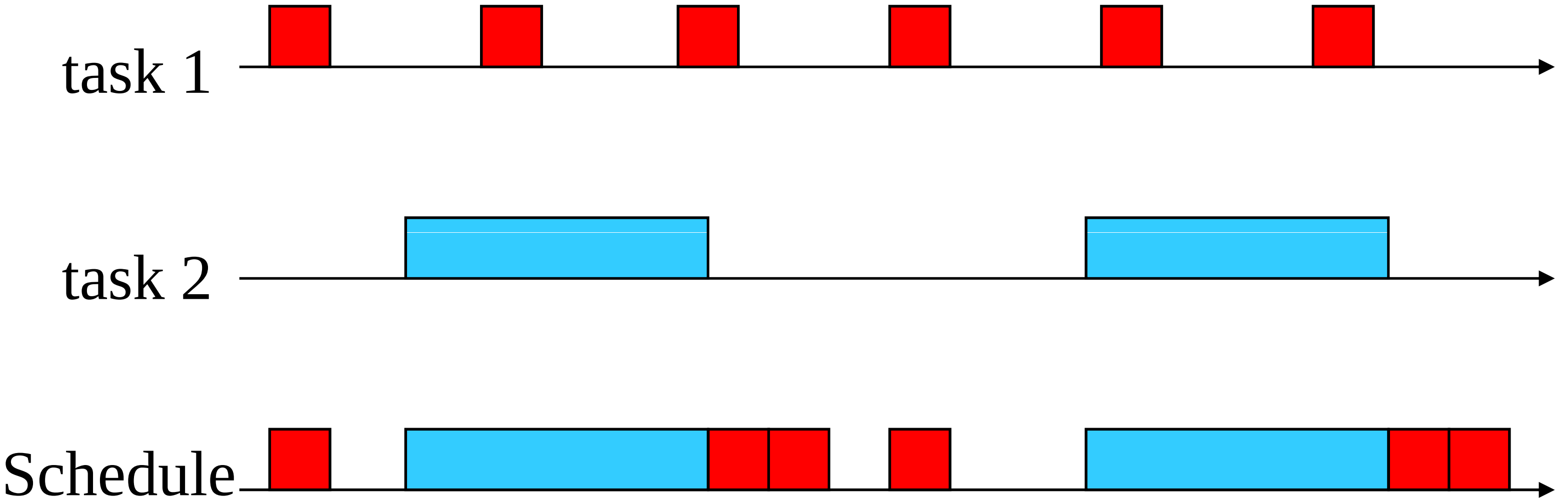
Periodic Task Model

- A periodic task T_i is characterized by
  - phase: θ_i
  - Period: p_i
  - Execution time : e_i
  - Relative deadline: D_i from the beginning of the period.



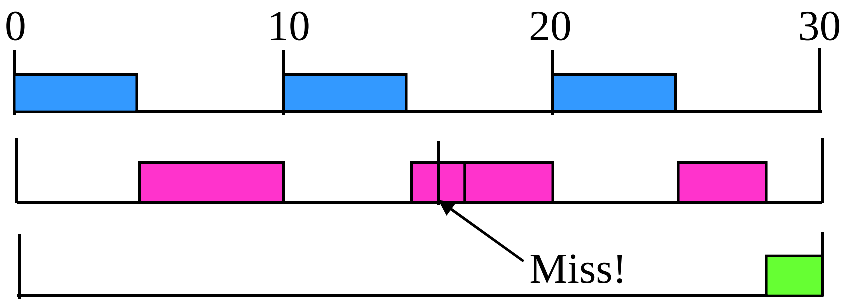
- Default assumption: $D_i = p_i$. That is, a periodic task deadline is located at the end of the period

FIFO Scheduler?

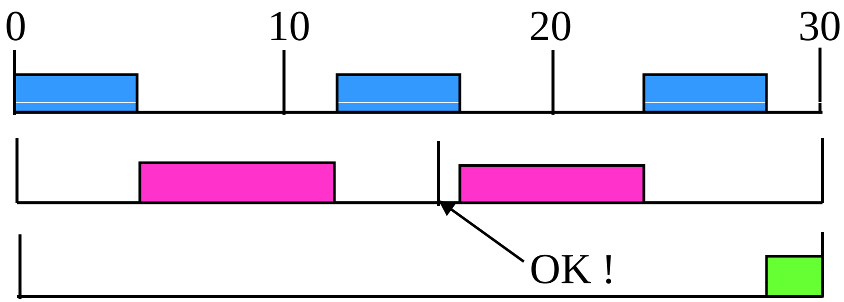


Priority-Driven Scheduler

- $\{T_1=(p_1=10, e_1=4), T_2=(p_2=15, e_2=8), T_3=(p_3=30, e_3=2)\}$



Fixed Priority Schedule (RM)



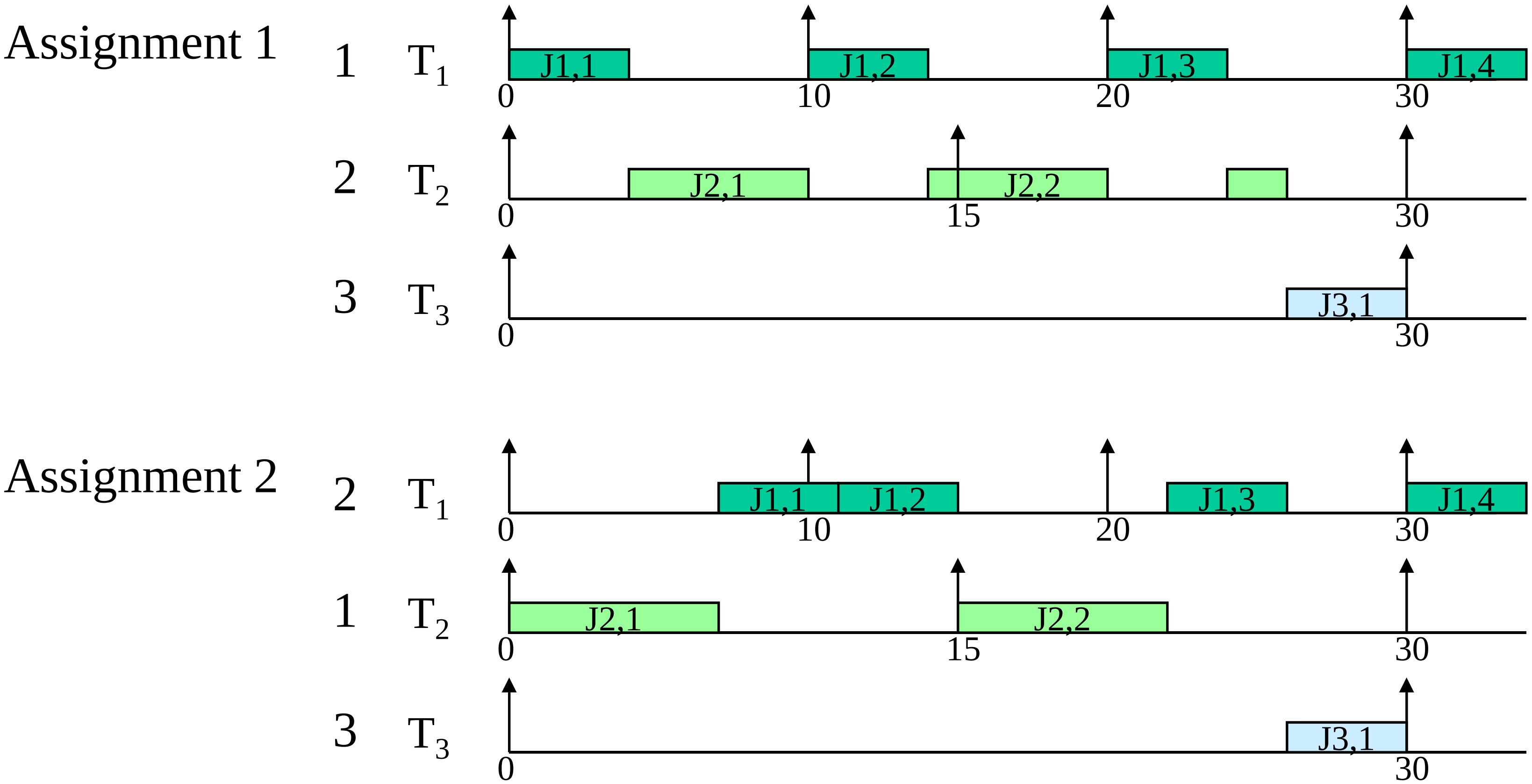
Dynamic Priority Schedule (EDF)

Fixed-Priority Scheduling

- How to assign Priorities?
- How to check the schedulability?

Priority Assignment

- $\{T_1=(p_1=10, e_1=4), T_2=(p_2=15, e_2=7), T_3=(p_3=30, e_3=4)\}$



Intuitive priority assignments

- Random – mostly perform poorly
- Functional Criticality (Semantic importance)
  - T_1 is a video display task
  - T_2 is a task monitoring and controlling patient's blood pressure
- Urgency
  - If all tasks are feasibly schedulable, the critical task doesn't have to be the highest priority task
  - RM and DM are examples

Optimal Fixed Priority Algorithm

- ***RM (Rate Monotonic)*** is an optimal static priority assignment for periodic tasks with <u>deadlines at the end of the period</u>.
  - Higher priority is assigned to a task with higher rate (inverse of period)

- ***DM (Rate Monotonic)*** is an optimal static priority assignment for periodic tasks with <u>arbitrary relative deadlines</u>.
  - Higher priority is assigned to a task with shorter relative deadline

Schedulability Check!

- Important for
  - Offline design phase
    - period selection
    - algorithm selection
    - identifying modules to be optimized
  - Online admission phase (in dynamic real-time systems)
    - periodic tasks are dynamically created by external events
      - In case that the system becomes unschedulable by adding the new task, we cannot admit it. Instead, we have to ring a warning alarm ASAP for alternative action.
    - control frequency and algorithm negotiation
    - frame rate and QoS parameter negotiation in multimedia

Formulation (Exact Analysis)

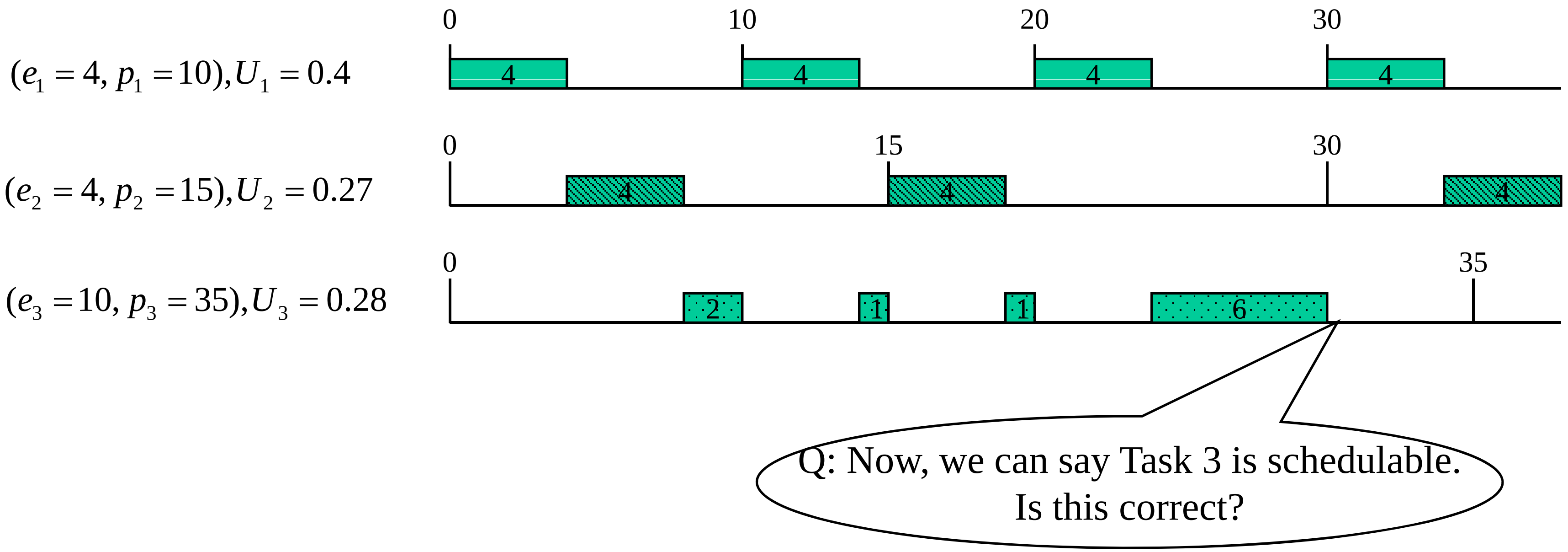
$$r_i^{k+1} = e_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^{k}}{p_j} \right\rceil e_j, \quad \text{where } r_i^{0} = \sum_{j=1}^{i} e_j$$

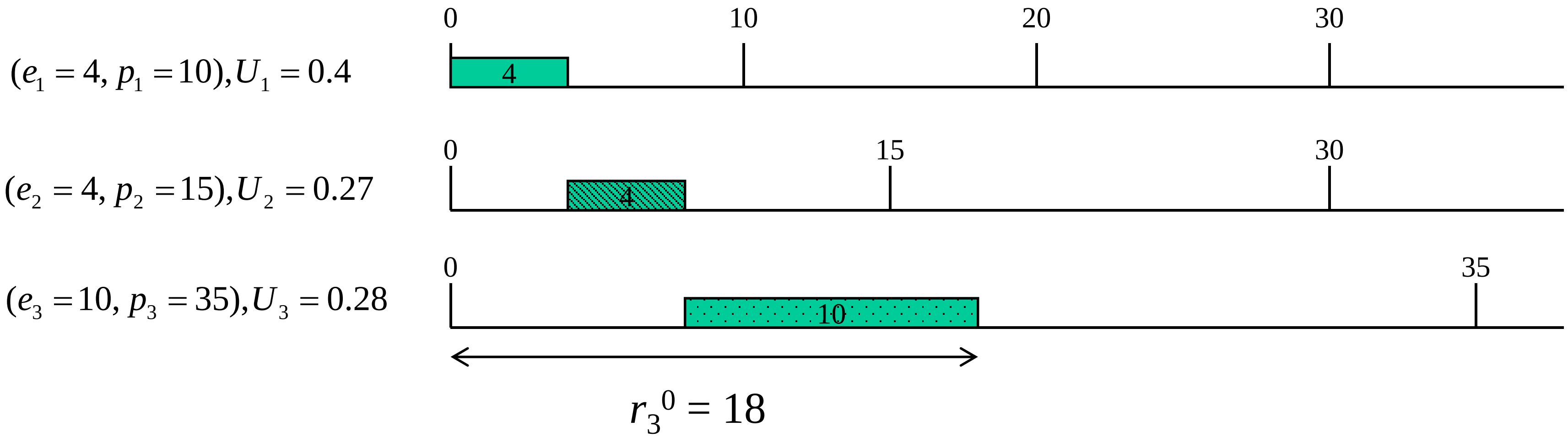
Test terminates when $r_i^{k+1} > p_i$ (not schedulable)
or when $r_i^{k+1} = r_i^{k} \leq p_i$ (schedulable).

- Tasks are ordered according to their priority: T_1 is the highest priority task.

The Exact Schedulability Test

- Basically, “*Enumerate*” the schedule
- “Task by Task” schedulability test



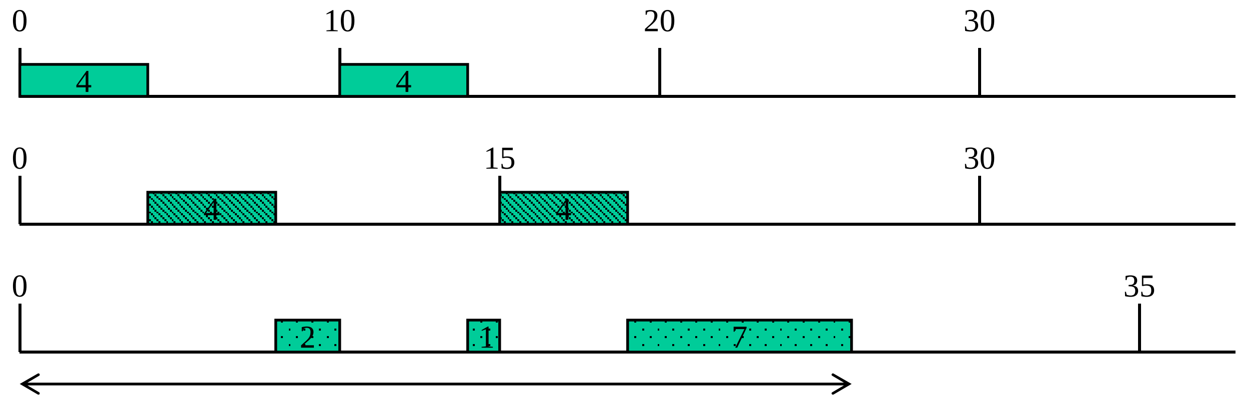


$(e_1 = 4, p_1 = 10), U_1 = 0.4$
$(e_2 = 4, p_2 = 15), U_2 = 0.27$
$(e_3 = 10, p_3 = 35), U_3 = 0.28$
0
10
20
30
4
0
15
30
4
0
35
10
$r_3^0 = 18$

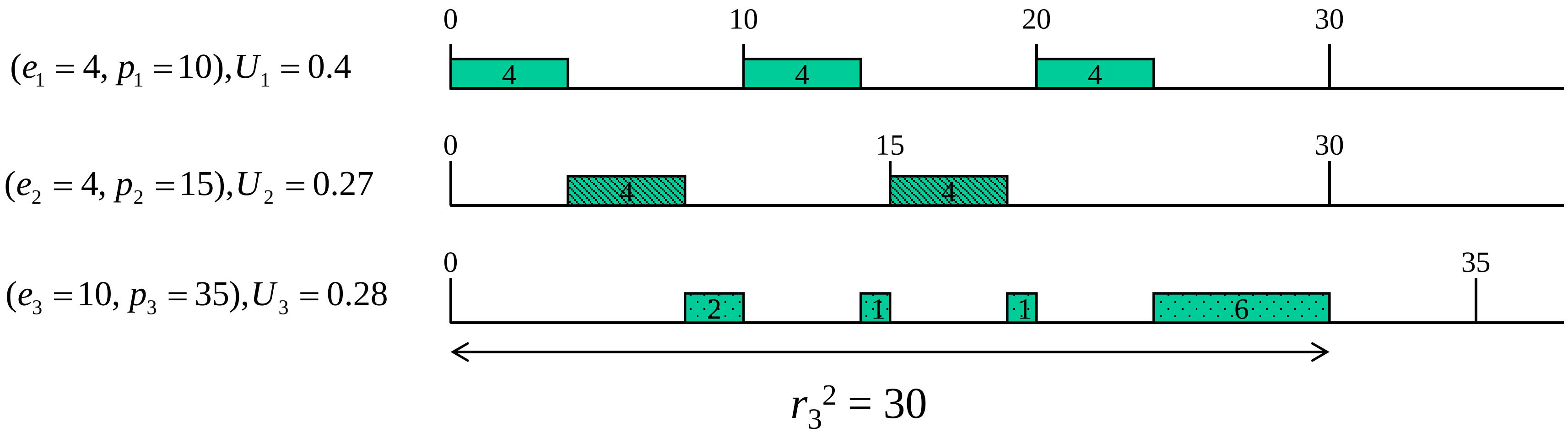
$(e_1 = 4, p_1 = 10), U_1 = 0.4$

$(e_2 = 4, p_2 = 15), U_2 = 0.27$

$(e_3 = 10, p_3 = 35), U_3 = 0.28$



$r_3^1 = 26$



$(e_1 = 4, p_1 = 10), U_1 = 0.4$
$(e_2 = 4, p_2 = 15), U_2 = 0.27$
$(e_3 = 10, p_3 = 35), U_3 = 0.28$
0
10
20
30
4
4
4
0
15
30
4
4
0
35
2
1
1
6
$r_3^{\,2} = 30$

Intuitions of Exact Schedulability Test

- Obviously, the response time of task 3 should larger than or equal to $e_1 + e_2 + e_3$

$$r_3^0 = \sum_{j=1}^{3} e_j = e_1 + e_2 + e_3 = 4 + 4 + 10 = 18$$

Intuitions of Exact Schedulability Test

- Obviously, the response time of task 3 should larger than or equal to $e_1+e_2+e_3$

$$r_3^0 = \sum_{j=1}^{3} e_j = e_1 + e_2 + e_3 = 4 + 4 + 10 = 18$$

- The high priority jobs released in r_3^0, should lengthen the response time of task 3

$$r_3^1 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^0}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{18}{10} \right\rceil 4 + \left\lceil \frac{18}{15} \right\rceil 4 = 26$$

Intuitions of Exact Schedulability Test

- Keep doing this until either r_3^k no longer increases or $r_3^k > p_3$

$$r_3^2 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^1}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{26}{10} \right\rceil 4 + \left\lceil \frac{26}{15} \right\rceil 4 = 30$$

$$r_3^3 = e_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{p_j} \right\rceil e_j = 10 + \left\lceil \frac{30}{10} \right\rceil 4 + \left\lceil \frac{30}{15} \right\rceil 4 = 30$$ **Done!**

Class Exercise 1

Suppose that we have two tasks

- $e_1 = 3$, $p_1 = 5$
- $e_2 = 5$, $p_2 = 14$

- Use exact test to check the schedulability of task 2. Draw the schedule timeline to confirm that

- $r_2^0 = e_1 + e_2 = 3 + 5 = 8$

$$r_2^1 = e_2 + \left\lceil \frac{r_2^0}{p_1} \right\rceil e_1 = 5 + \left\lceil \frac{8}{5} \right\rceil 3 = 11$$

$$r_2^2 = e_2 + \left\lceil \frac{r_2^1}{p_1} \right\rceil e_1 = 5 + \left\lceil \frac{11}{5} \right\rceil 3 = 14$$

$$r_2^3 = e_2 + \left\lceil \frac{r_2^2}{p_1} \right\rceil e_1 = 5 + \left\lceil \frac{14}{5} \right\rceil 3 = 14$$

Done! ➔ the task set is schedulable

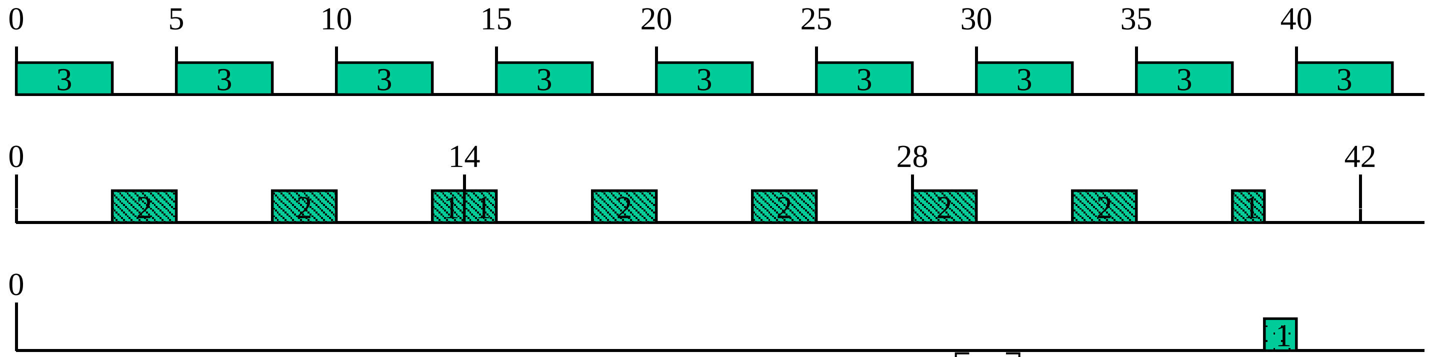
Class Exercise 1

Suppose that we have two tasks

- $e_1 = 3$, $p_1 = 5$
- $e_2 = 5$, $p_2 = 14$

- Can we add a task 3 with $e_3 = 1$ and $p_3 = 50$? What would be the shortest period of p_3 that it can still meet its deadlines? Apply the exact test formulation to confirm that.

Class Exercise 1 (continued)



$$r_3^0 = \sum_{j=1}^{3} C_j = 3+5+1 = 9$$

$$r_3^1 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^0}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{9}{5} \right\rceil 3 + \left\lceil \frac{9}{14} \right\rceil 5 = 12$$

$$r_3^2 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^1}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{12}{5} \right\rceil 3 + \left\lceil \frac{12}{14} \right\rceil 5 = 15$$

$$r_3^3 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{15}{5} \right\rceil 3 + \left\lceil \frac{15}{14} \right\rceil 5 = 20$$

$$r_3^4 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^3}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{20}{5} \right\rceil 3 + \left\lceil \frac{20}{14} \right\rceil 5 = 23$$

$$r_3^5 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^4}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{23}{5} \right\rceil 3 + \left\lceil \frac{23}{14} \right\rceil 5 = 26$$

$$r_3^6 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^5}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{26}{5} \right\rceil 3 + \left\lceil \frac{26}{14} \right\rceil 5 = 29$$

$$r_3^7 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^6}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{29}{5} \right\rceil 3 + \left\lceil \frac{29}{14} \right\rceil 5 = 34$$

$$r_3^8 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^7}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{34}{5} \right\rceil 3 + \left\lceil \frac{34}{14} \right\rceil 5 = 37$$

$$r_3^9 = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^8}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{37}{5} \right\rceil 3 + \left\lceil \frac{37}{14} \right\rceil 5 = 40$$

$$r_3^{10} = C_3 + \sum_{j=1}^{2} \left\lceil \frac{r_3^9}{T_j} \right\rceil C_j = 1 + \left\lceil \frac{40}{5} \right\rceil 3 + \left\lceil \frac{40}{14} \right\rceil 5 = 40$$

Schedulable utilization bound

- Simpler method for the schedulabiity check

The L&L Bound

A set of n periodic task is schedulable if :

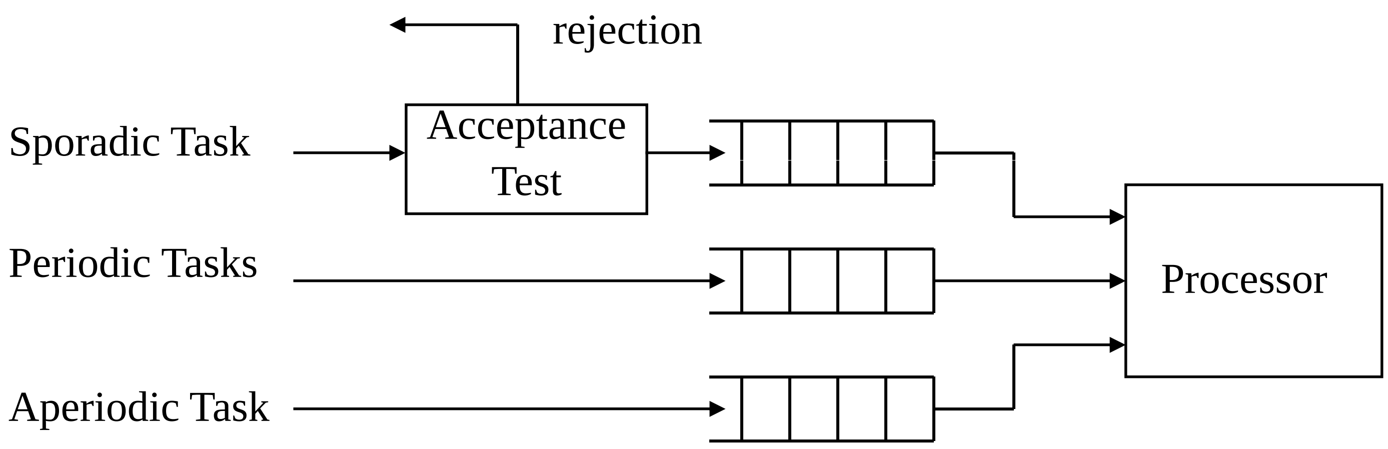
$$\frac{e_1}{p_1} + \frac{e_2}{p_2} + \ldots + \frac{e_n}{p_n} \leq n(2^{1/n} - 1)$$

- U(1) = 1.0 U(4) = 0.756 U(7) = 0.728
- U(2) = 0.828 U(5) = 0.743 U(8) = 0.724
- U(3) = 0.779 U(6) = 0.734 U(9) = 0.720

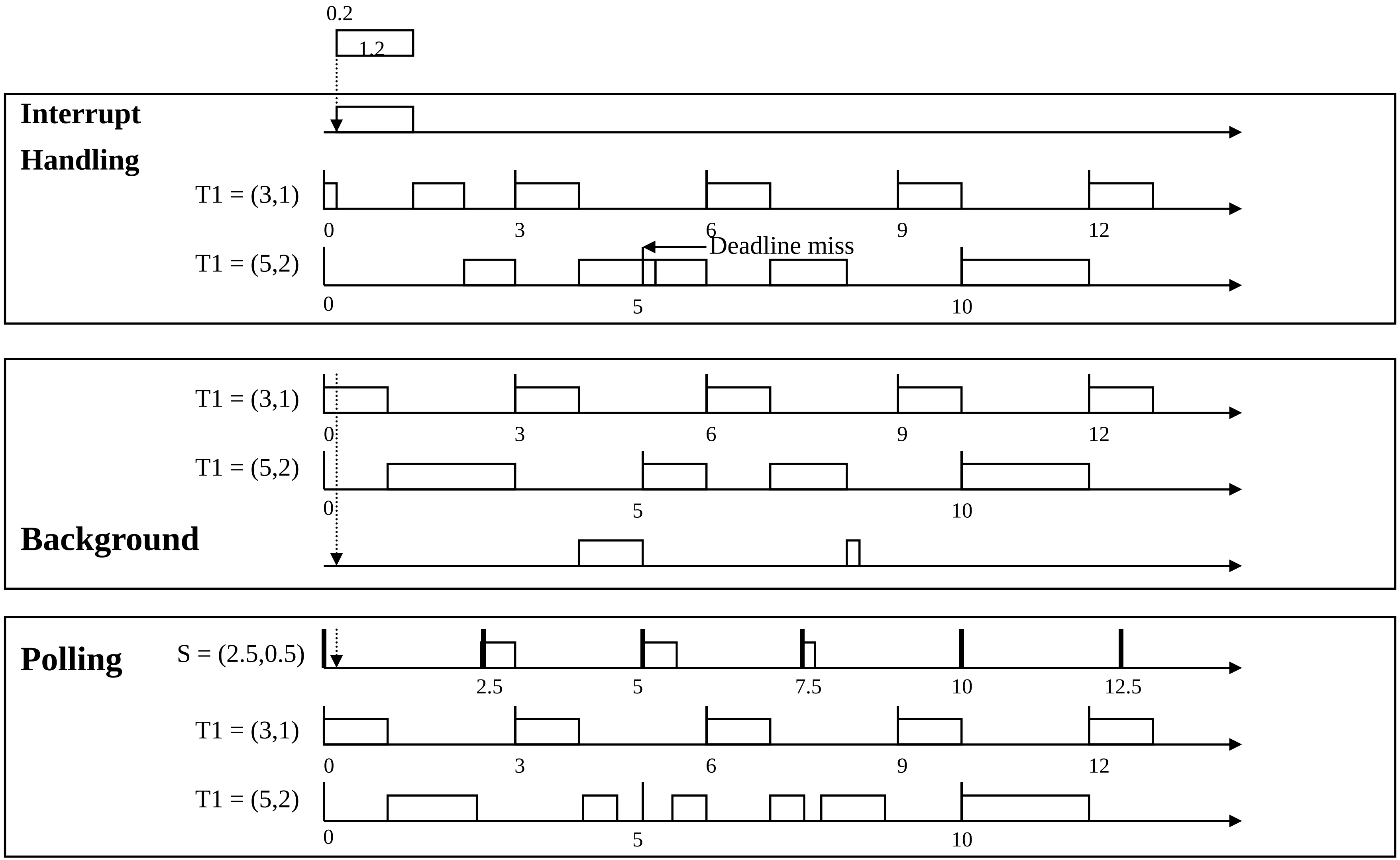
- For harmonic task sets, the utilization bound is U(n)=1.00 for all n. For large n, the bound converges to *ln 2* ~ 0.69.

- The L&L bound for rate monotonic algorithm is one of the most significant results in real-time scheduling theory. Its derivation also shows a wealth of analysis techniques that are useful in many new situations when considering static priority scheduling.

Handling Aperiodic Requests



Interrupt Handling, Background, Polling



Polling - 1

- The simplest form of integrated aperiodic and periodic service is polling.
  - For each aperiodic task, we assign a periodic service with budget e_s and period p_s. This creates a server (e_s, p_s)
  - The aperiodic requests are buffered into a queue
  - When polling server starts,
    - Resumes the existing job if it was suspended in last cycle.
    - it checks the queue.
  - The polling server runs until
    - All the requests are served
    - Or suspends itself when the budget is exhausted.

  - Remark: a small improvement is to run the tasks in background priority instead of suspend. This background mode can be applied to all the servers discussed later.

Polling - 2

- A polling server is just a periodic task and thus the schedulability of periodic tasks is easy to analyze. For example, if we use L&L bound,

$$\sum_{i=1}^{n} \frac{e_i}{p_i} + \frac{e_s}{p_s} \leq (n+1)\left(2^{1/(n+1)} - 1\right)$$

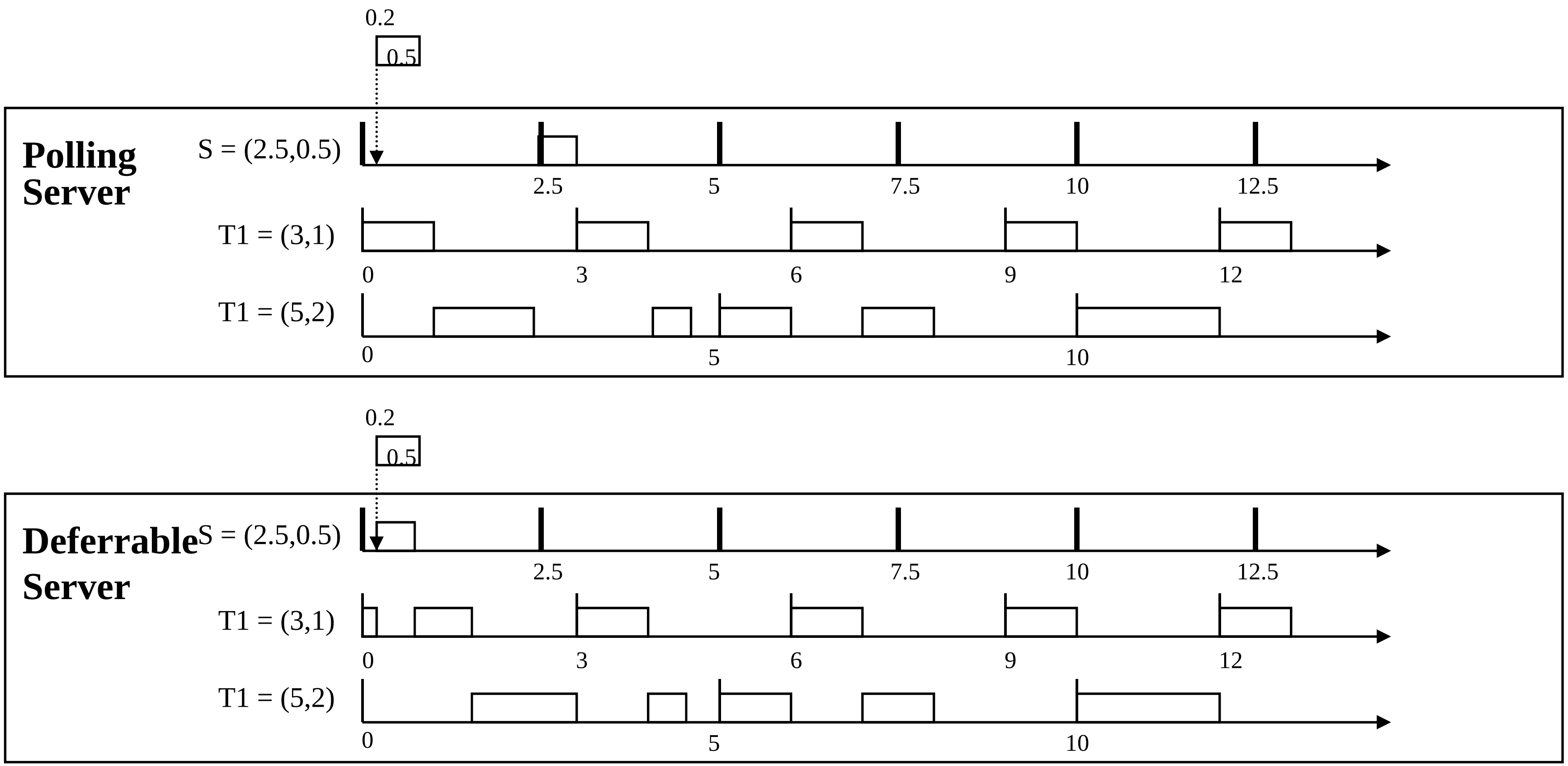
- Quiz: How can we analyze the aperiodic performance for each polling server?

- Answer: Simulation

Deferrable Server - 1

- Comparing polling with interrupt handling, interrupt handling serves aperiodic requests right away whereas the Polling Server creates an average of half a period waiting time.
- Deferrable Server is the 1st attempt to simulate interrupt handling service but bounds the service time of aperiodics so that it ensures periodic tasks are schedulable.
- The idea is to let the budget float, just like getting a monthly salary. The salary allocation is periodical, but one can spend it anytime he likes.

Deferrable Server - 2



Deferrable Server - 3

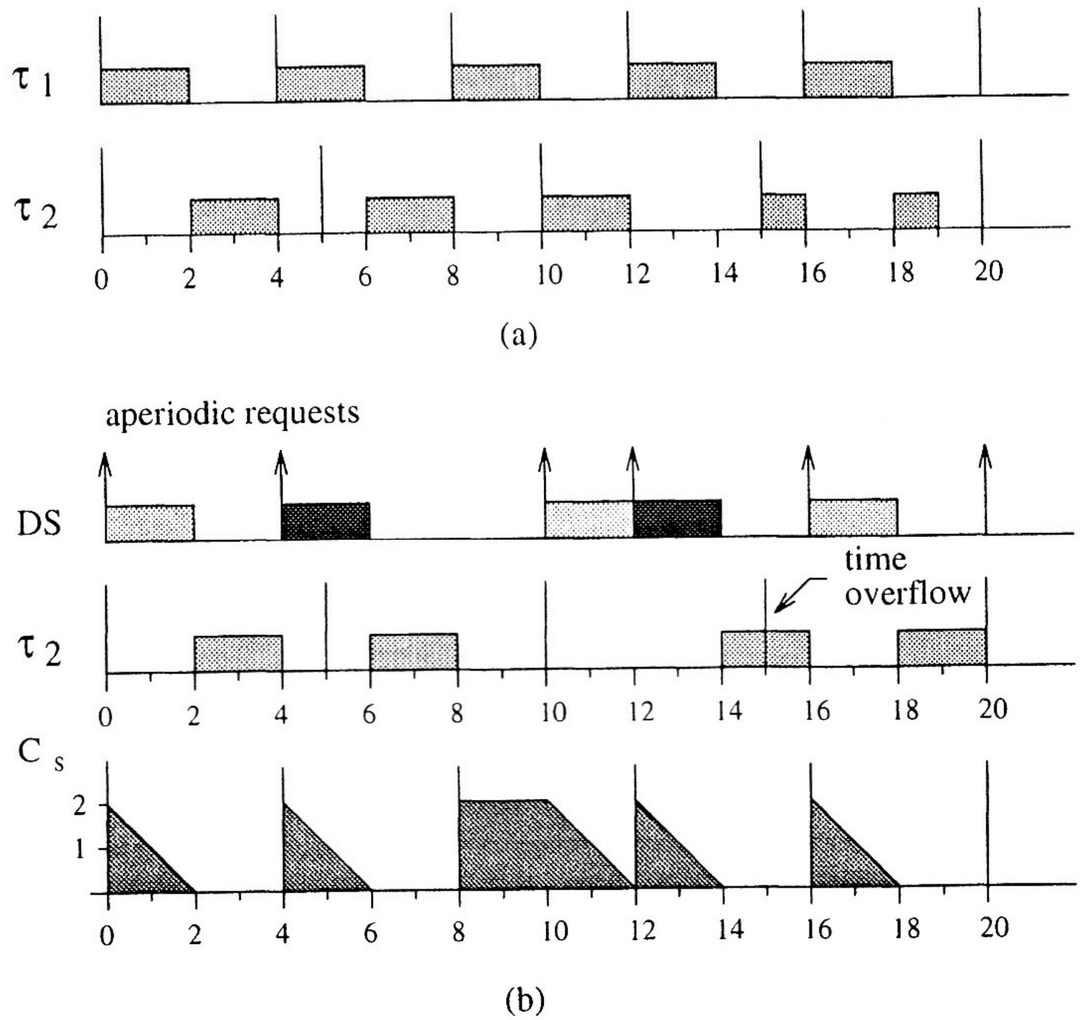
- Example: e = 50 ms; p = 250 ms.

- Every 250ms, the budget is RESET to 50 ms (<u>no savings of unused budget!</u>)
- Aperiodic requests arrive at a queue.
- The head of queue request checks if there is budget available.

- If there is budget left,
- the aperiodic request runs until either the request is served
- or the budget is exhausted
- and therefore the aperiodic request is suspended until there is new budget available
- else the aperiodic request is suspended and it waits until there is new budget available

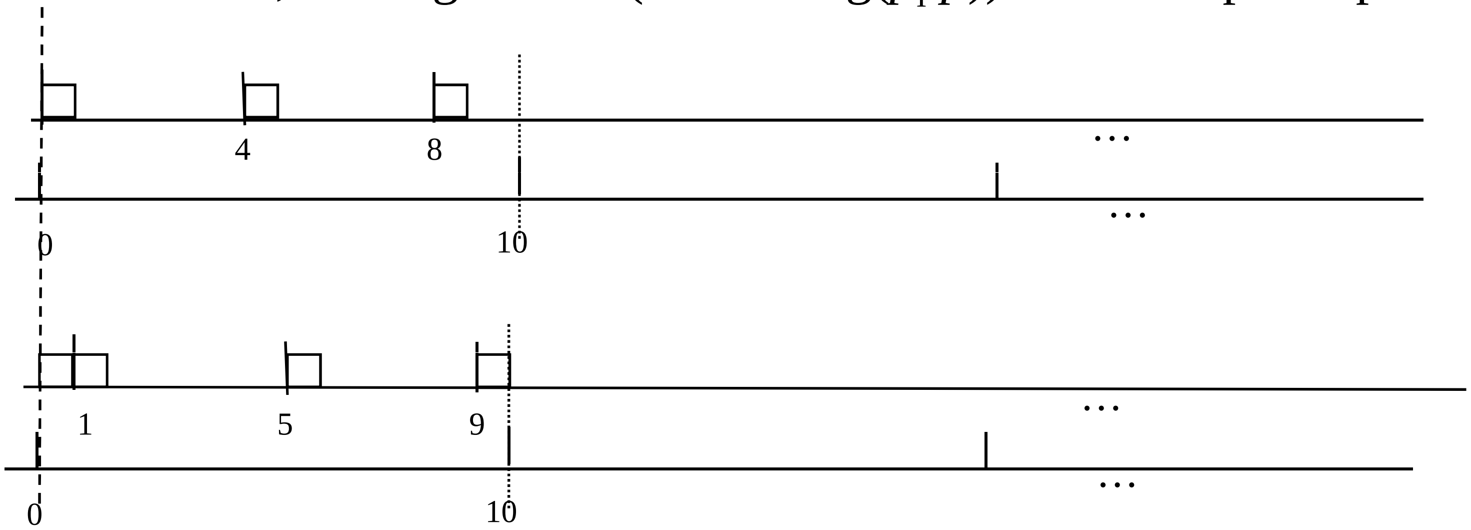
Deferrable Server vs. periodic task

- A Deferrable Server is not equivalent to a periodic task!



Deferrable Server - 5

- Schedulability of periodic tasks using RMS. Let the period of the sever be p. For any lower priority task with period p_i, it generates at most ceiling (p_i/p) times preemption, if it was a regular periodic task. However, it can generate (1 + ceiling(p_i/p)) times the preemption.



• Note that task 1 originally starts at t = 0 and the interval for the preemption is [0, 10]. In the second example, a 1 unit shifting lets the deferred unit to come in. The starting time is now 1 and the interval for preemption is still [0, 10]

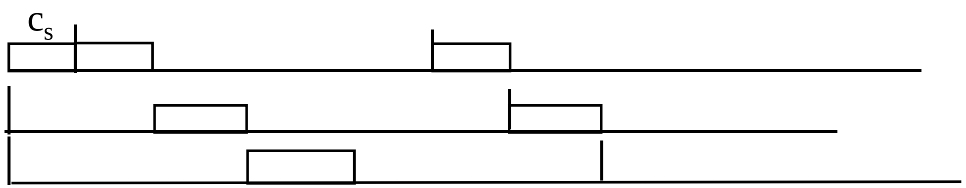
Deferrable Server - 6

- Scheduling bound under RMS: Considering the 1 additional unit of preemption, we will get the following bound

$$U_{\text{lub}} = n\left[\left(\frac{U_s + 2}{2U_s + 1}\right)^{1/n} - 1\right]$$

where U_s is the utilization of the Deferrable Server (e/p).

- It is worth noting that the tasks' pattern that provides the worst-case condition for the periodic tasks under the RM algorithm is:



Deferrable Server - 7

- Given a set of n periodic tasks and a Deferrable Server with utilization factors U_p and U_s, respectively, the schedulability of the periodic task set is guaranteed under RM if:

$$U_p \leq U_{\text{lub}} = n\left[\left(\frac{U_s + 2}{2U_s + 1}\right)^{1/n} - 1\right]$$

where U_s is the utilization of the Deferrable Server (e/p).

Deferrable Server - 8

- Time demand analysis: since there *could* be an additional preemption, a sufficient condition is to use the old time demand analysis and add 1 to the preemption of the deferrable task's term.

$$a_i(t) = e_i + (1 + \left\lceil \frac{t}{p} \right\rceil)e + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil e_j$$

An improvement can be made by noting that we can substract e out of t in the ceiling function for the deferrable server, since we shift the starting time to the right by e units to let the deferred units to come in.

$$a_i(t) = e_i + (1 + \left\lceil \frac{t-e}{p} \right\rceil)e + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil e_j$$

if we have m servers have shorter periods than task *i*

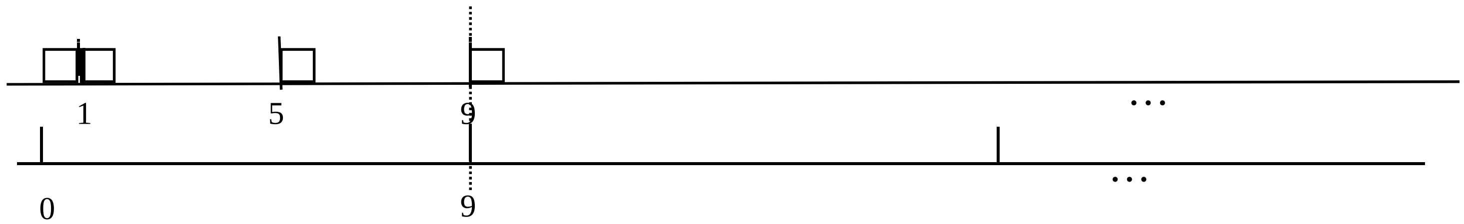
$$a_i(t) = e_i + \sum_{s=1}^{m} (1 + \left\lceil \frac{t-e_s}{p_s} \right\rceil)e_s + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil e_j$$

Remark: the textbook has an addition b term for blocking. We assume it is 0 for now

Deferrable Server - 9

- The effect of shifting to the right can best be illustrated as follows.

- 1 + ceiling(9/4) = 4; over count 1 unit
- 1 + ceiling ((9-1)/4) = 3; exact.



Class exercise (1)

- Consider the following task set
  - T_1 {e_1=1, p_1= 4}
  - T_2 {e_2=2, p_2= 6}
  - T_s {e_s=1, p_s= 5}

- Are the periodic task set and the deferrable server T_s schedulable?

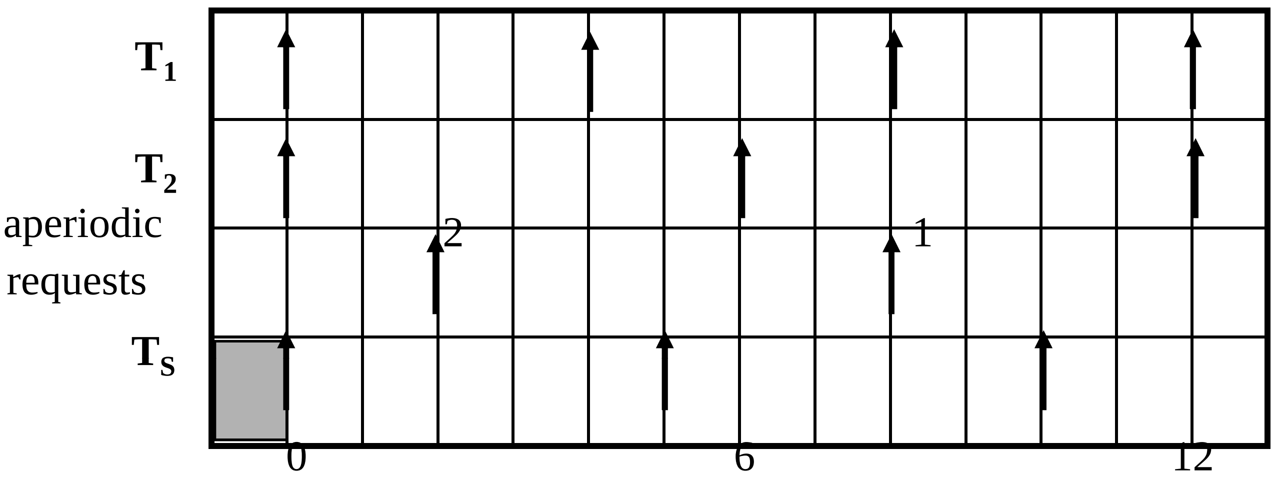
Use the time demand analysis ➔

$$a_i(t) = e_i + (1 + \left\lceil \frac{t - e}{p} \right\rceil) e + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil \cdot e_j$$

Class exercise (2)

- Consider the following task set
  - T_1 $\{e_1=1, p_1= 4\}$
  - T_2 $\{e_2=2, p_2= 6\}$
  - T_s $\{e_s=1, p_s= 5\}$

- Schedule the following aperiodic activities by using the deferrable server



Sporadic Server - 1

- The deferrable server has this one additional preemption and reduces the schedulability of periodic tasks. So we tried to get rid of this additional preemption.
- The SS differs from DS in the way it replenishes its capacity. Whereas DS periodically replenishes its capacity at the beginning of each server period, SS replenishes its capacity only after it has been consumed by aperiodic task execution.
- Idea: Spread the budget replenishment at least P time units
- We will see that Sporadic Server can be treated as if it is a periodic task.

Sporadic Server - 2

- A Sporadic Server with priority $Prio_s$ is said to be ***active*** when it is executing or another task with priority $Prio_T \geq Prio_s$ is executing. Hence, the server remains active even when it is preempted by a higher priority task.

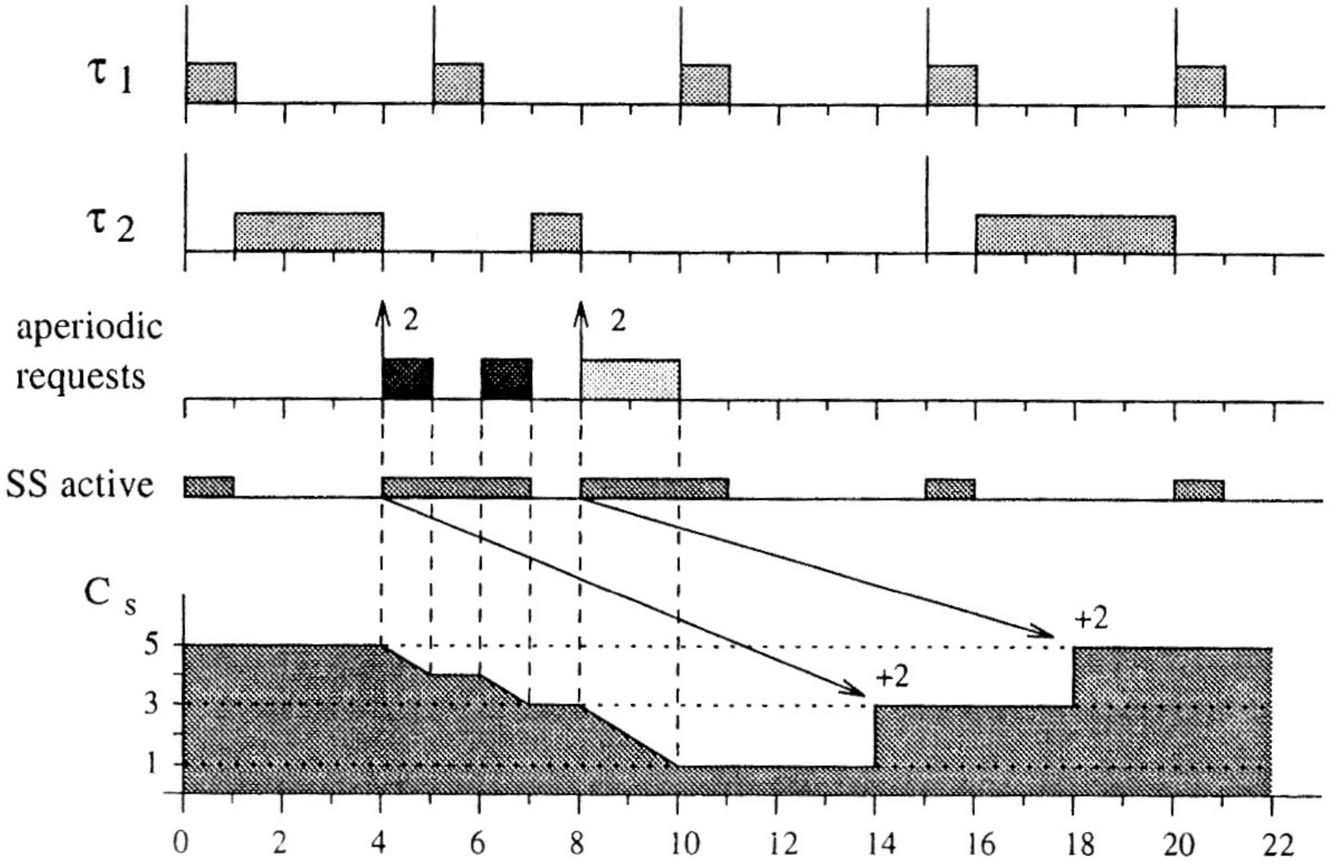
- If the server is not active, it is said to be ***idle***

- **Replenishment Time (RT):** it is set as soon as “SS becomes active and the server capacity $C_s > 0$”. Let T_A be such a time. The value of RT is set equal to T_A plus the server period ($RT = T_A + p_s$).

- **Replenishment Amount (RA):** The RA to be done at time RT is computed when “SS becomes idle or the server capacity C_s has been exhausted”. Let T_I be such a time. The value of RA is set equal to the capacity consumed within the interval $[T_A, T_I]$.

Sporadic Server - 3

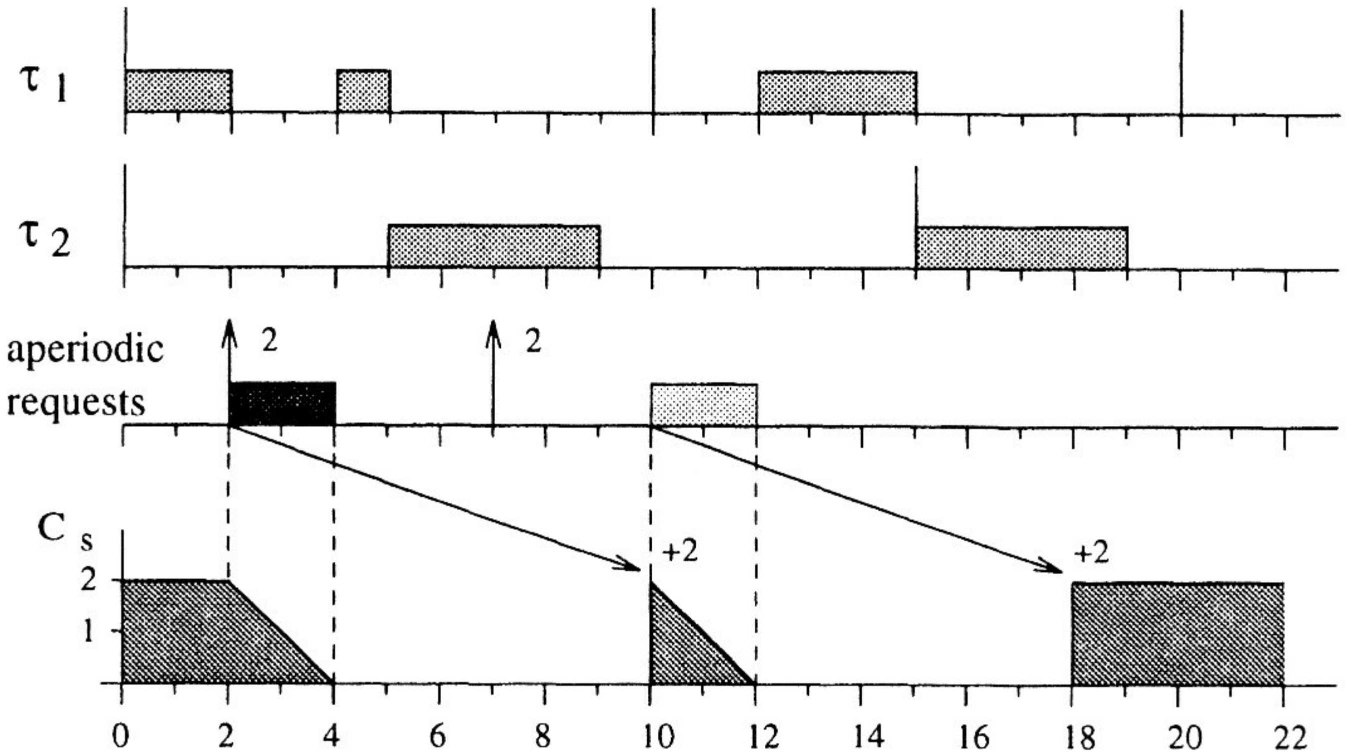
- Example of a medium-priority Sporadic Server.



	e	p
T_1	1	5
T_S	5	10
T_2	4	15

Sporadic Server - 4

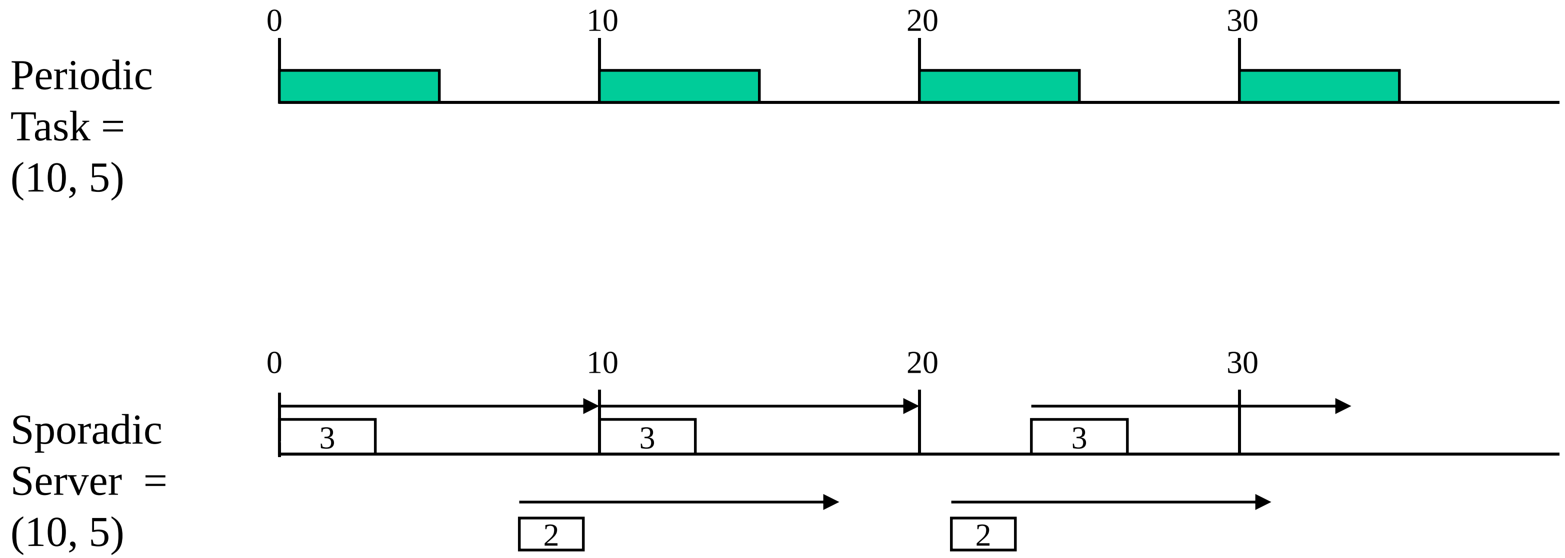
- Example of a high-priority Sporadic Server.



	e	p
T_S	2	8
T_1	3	10
T_2	4	15

Sporadic Server vs. periodic task

- A Sporadic Server ≤ a periodic task!



Sporadic Server - 5

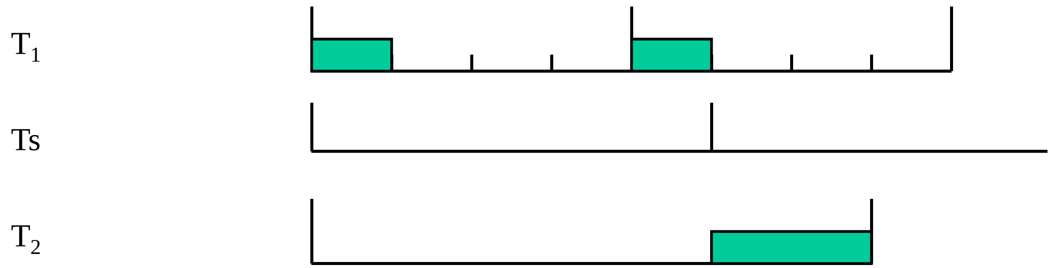
- *A periodic task set that is schedulable with a task T_i is also schedulable if T_i is replaced by a Sporadic Server with the same period and execution time.*

Proof: ???

Class exercise (3)

- Consider the following task set
  - T_1 {e_1=1, p_1= 4}
  - T_2 {e_2=2, p_2= 7}
  - T_s {e_s=?, p_s= 5}

- What is the maximum possible e_s if it is deferrable server?
  - $2*1+3*e_s + 2 <= 7$. Thus, $e_s < 1$
- What is the maximum possible e_s if it is sporadic server?
  - $2*1+2*e_s + 2 <= 7$. Thus, $e_s < 1.5$



Schedule Simulation for Mixed Tasks

- 2 periodic tasks
  - T_1 {e_1=1, p_1= 4}
  - T_2 {e_2=2, p_2= 10}
- Four methods to process aperiodic jobs
  - background
  - polling server
    - highest priority, period = 2, server budget = 1
  - deferrable server
    - highest priority, period = 2, server budget = 5/6=0.833333
  - sporadic server
    - highest priority, period = 2, server budget = 1

Homework 6

- Simulate the RM scheduler with aperiodic processing by
  - background
  - polling server (+ background if the server is idle)
  - deferrable server (+ background if the server is idle)
  - sporadic server (+ background if the server is idle)
- Compare the response time of aperiodic jobs for the above four cases as decreasing the average inter-arrival time from 10 to 0.4 while fixing the average execution time to 0.1
- Do the same as decreasing the average inter-arrival time from 1000 to 40 while fixing the average execution time to 10
- Explain your results

Modules and Connections