Real-time Systems

Chapter 7: Fixed Priority Servers

Ngo Lam Trung

Dept. of Computer Engineering

Contents

Introduction

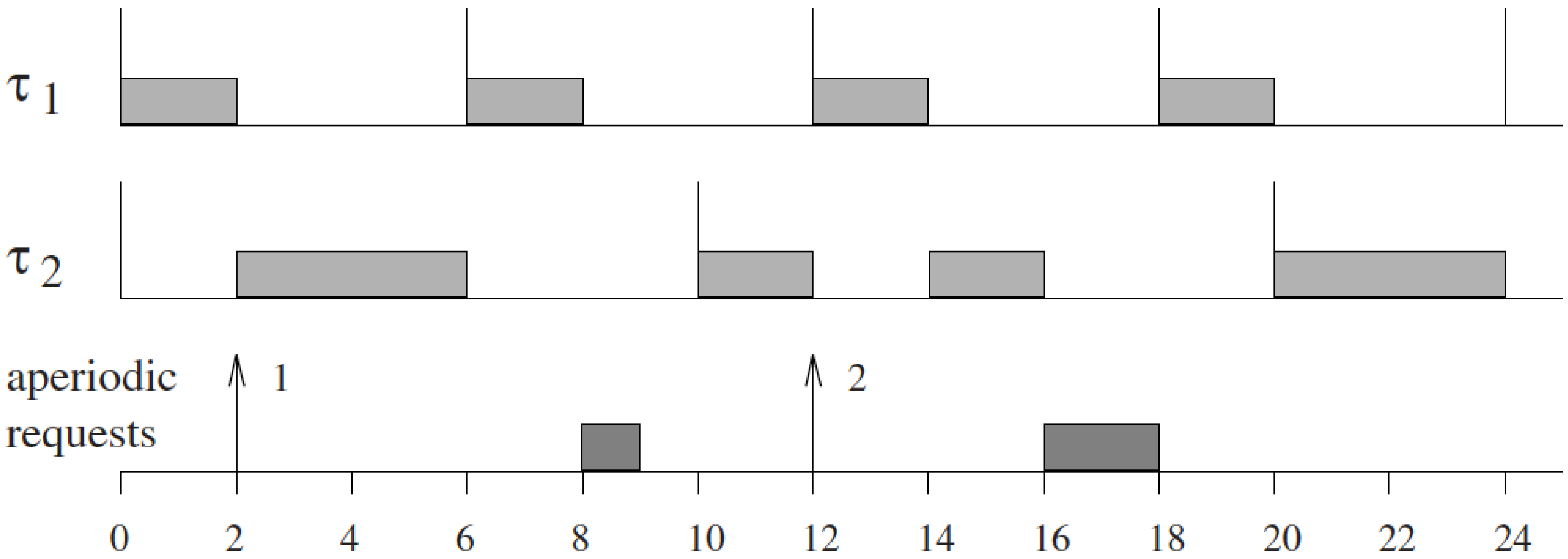
- Practical systems contain different types of task
  - Periodic tasks for critical activities: time driven, usually with hard timing constrain
  - Aperiodic tasks: event driven, may be hard/soft or non-real time.
  - ➔ Hybrid task set
- Problem: How to produce a schedule that
  - Guarantee the schedulability of critical (periodic) tasks
  - Provide acceptable response time of soft and non-real time tasks
- How about critical aperiodic tasks?
  - Assuming a maximum arrival rate → change to periodic task

Assumption

- All periodic task start at $t = 0$ and their deadline and period are equal.
- Periodic task are scheduled by RM (fixed priority).
- Arrival times of aperiodic requests are unknown.
- The minimum inter-arrival time of a sporadic task is assumed to be equal to its deadline.
- All tasks are fully preemptible

The simplest method: Background scheduling

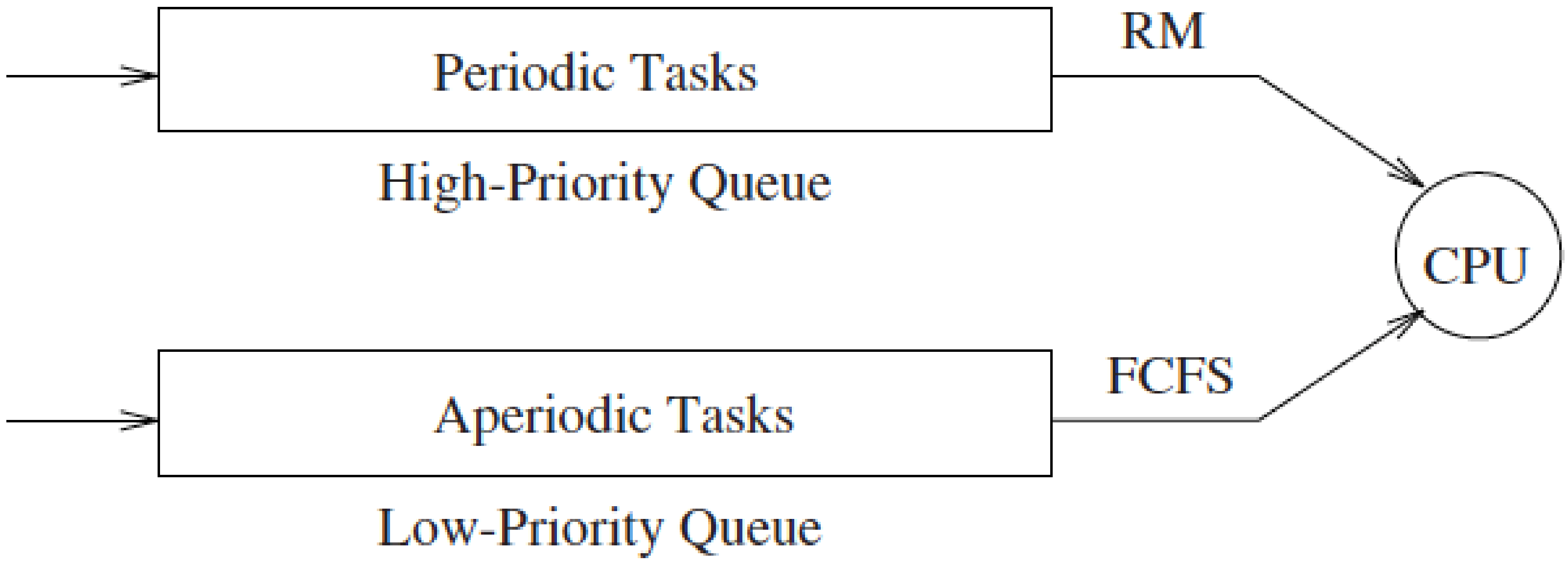
- Schedule periodic tasks with RM as usual
- Aperiodic tasks are scheduled at background: run when there is no periodic load.



$U_{periodic}$ = ?
Schedulability of periodic task will change of not?

Background scheduling

- Two task queues in background scheduling



- Pros: simple method
- Cons: response time of aperiodic tasks may be low in high periodic load

Polling Server

- Improve average aperiodic tasks response time
- Create an additional periodic task
  - Called Polling Server (PS)
  - PS serves aperiodic load asap upon request
- Server task parameter
  - Period T_S
  - Computation time C_S: server capacity
- PS is scheduled together with other periodic tasks
- When PS is activated
  - Select a waiting aperiodic task and execute it with server capacity
  - If there is no aperiodic task waiting, server suspends itself and gives up its capacity

Polling Server

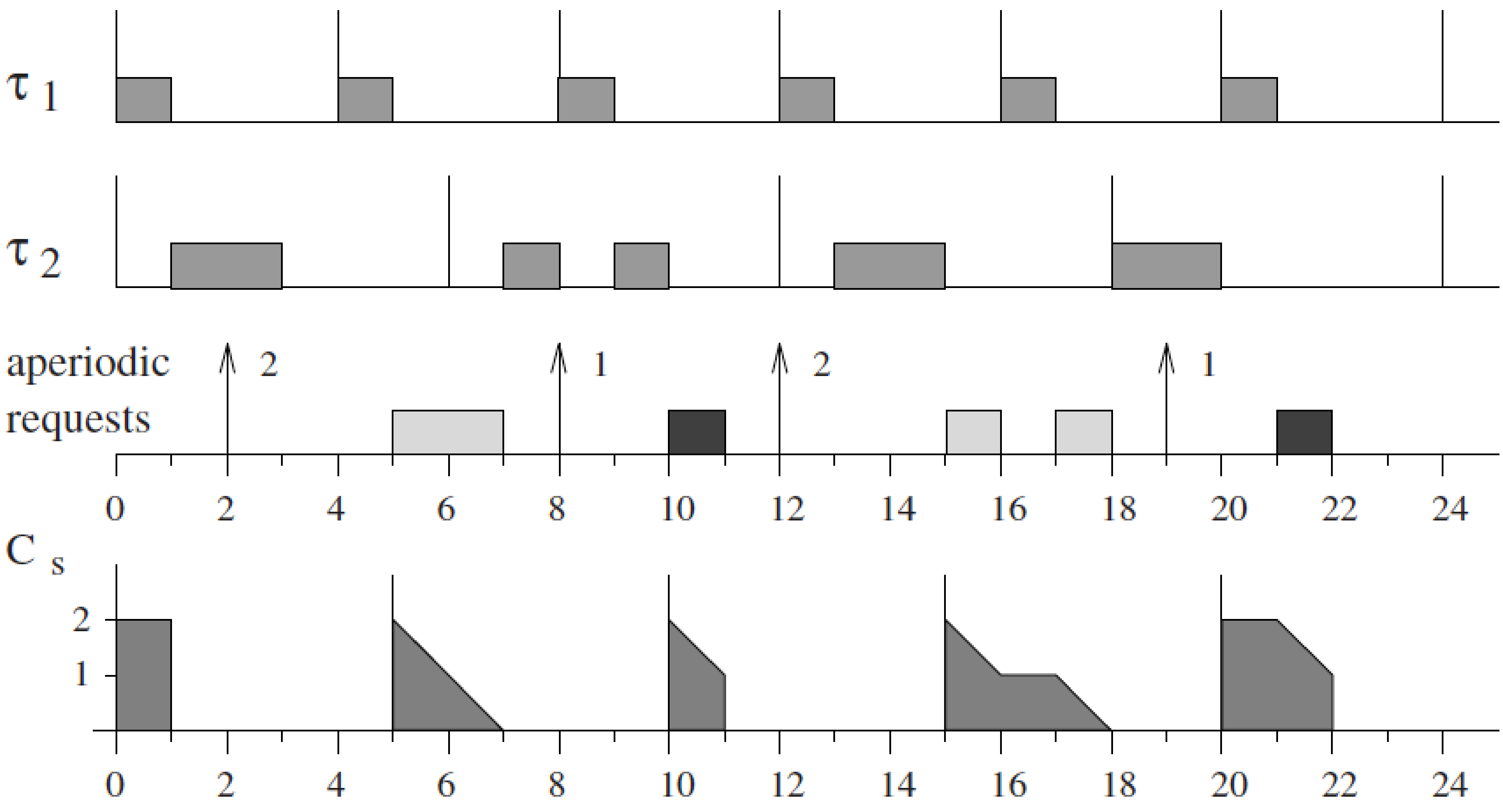
- Example

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

$C_s = 2$

$T_s = 5$



Polling Server: Schedulability analysis

- With RM, the task set including the server must be schedulable

$$U_p + Us \leq Ulub(n+1).$$

- Or

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

U_p = total CPU utilization of original periodic tasks

- Or

$$\prod_{i=1}^{n}(U_i+1) \leq \frac{2}{U_s+1}$$

Dimensioning the PS

- What are appropriate values of Ts, Ps that guarantee feasible schedule?
- Define

$$P \stackrel{\text{def}}{=} \prod_{i=1}^{n} (U_i + 1)$$

- From schedulability condition

$$P \leq \frac{2}{U_s + 1}$$

- So we need the server satisfies

$$U_s \leq \frac{2 - P}{P}$$

Dimensioning the PS

- Let

$$U_s^{max} = \frac{2 - P}{P}$$

- Server utilization can be selected so $U_s < Umax$
- Then select the smallest server period as possible

$$T_s = T_1$$

- Finally

$$C_s = U_s T_s$$

Polling Server

- Exercise 1

Consider two periodic tasks with computation times $C_1 = 1$, $C_2 = 2$ and periods $T_1 = 5$, $T_2 = 8$, handled by Rate Monotonic. Show the schedule produced by a Polling Server, having maximum utilization and intermediate priority, on the following aperiodic jobs:

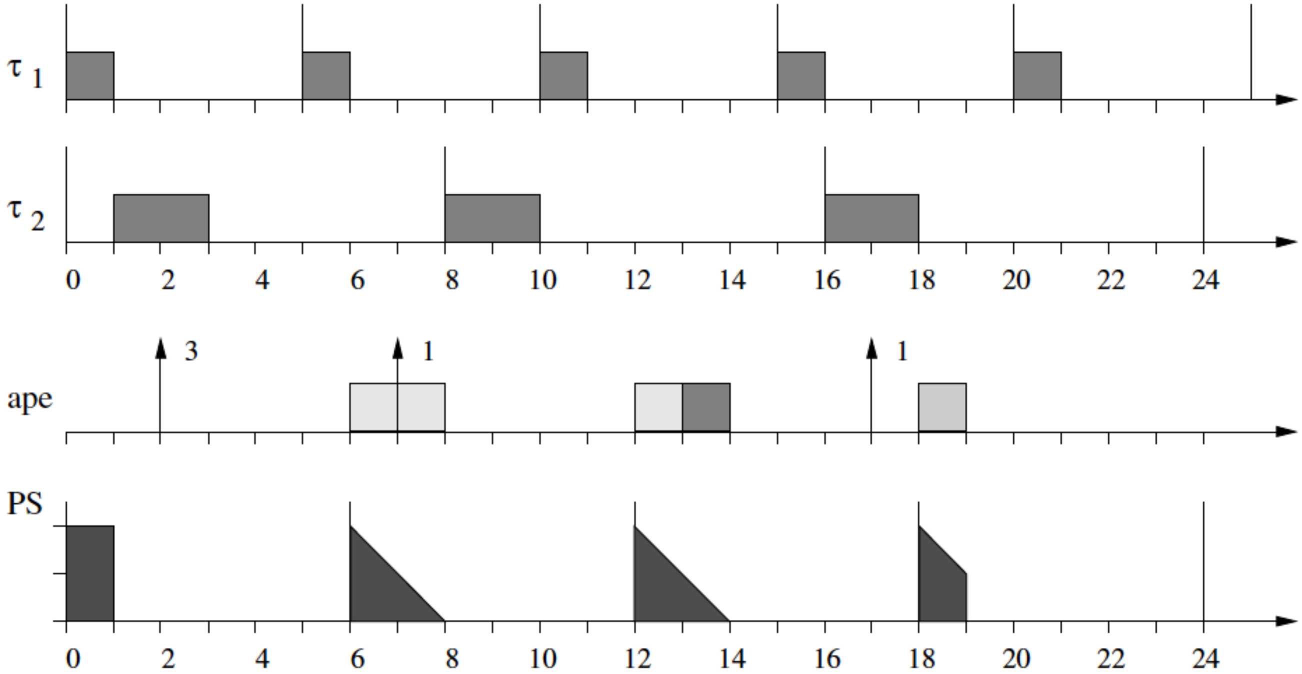
	a_i	C_i
J_1	2	3
J_2	7	1
J_3	17	1

Polling Server

- Sizing the PS

$$U_{PS}^{max} = \frac{2-P}{P} = \frac{1}{3}$$

- So we can set $T_s = 6$ (intermediate priority) and $C_s = 2$



Polling Server

- Problem: if an aperiodic request arrives after the server suspends itself, the request must wait until the next server period

  ➔ lowering average response time

- How to improve:
  - Server will not suspend
  - → Deferrable Server (DS)

Deferable Server (DS)

- Improves responsiveness of aperiodic tasks (compare to PS)
- Algorithm
  - Create high priority periodic task to serve aperiodic tasks
  - Server replenishes its capacity at the beginning of each period
  - If no aperiodic load are pending upon server invocation, the server preserves its capacity

    →aperiodic load can be served at anytime (as opposed to PS)

Example 1

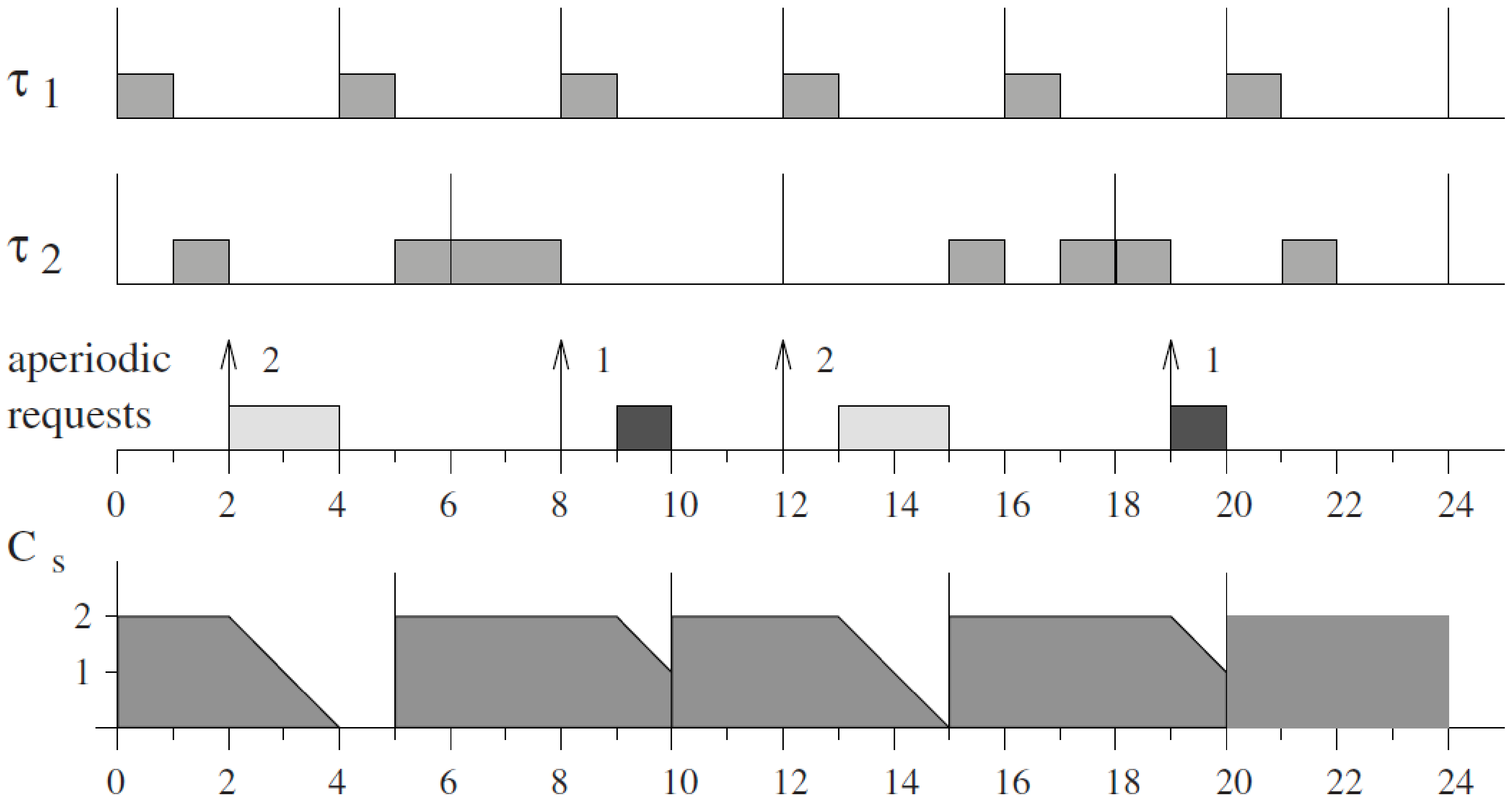
Periodic tasks' priorities are calculated by RM

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

$C_s = 2$

$T_s = 5$



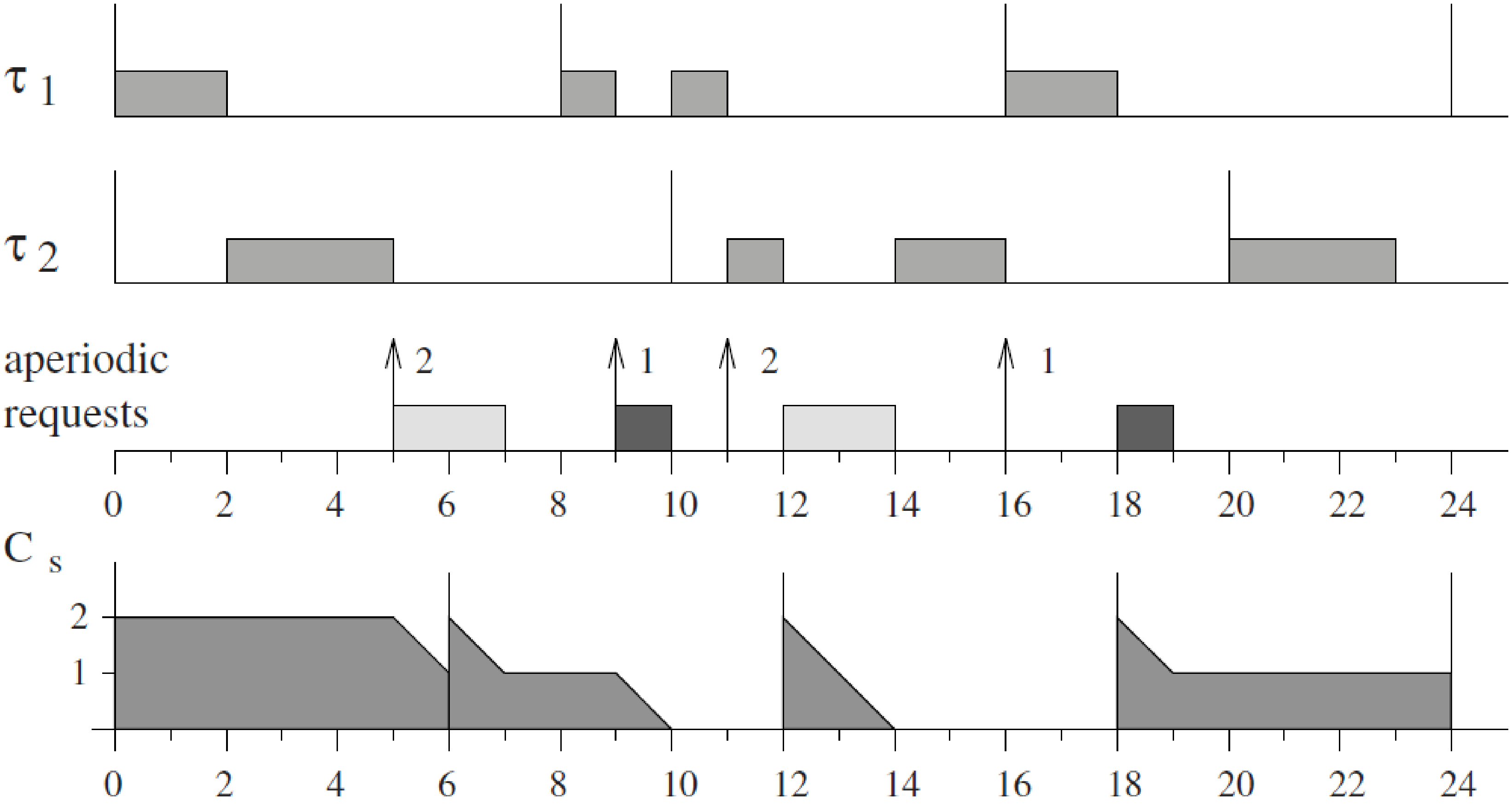
Example 2

DS is the highest priority task

	C_i	T_i
τ_1	2	8
τ_2	3	10

Server

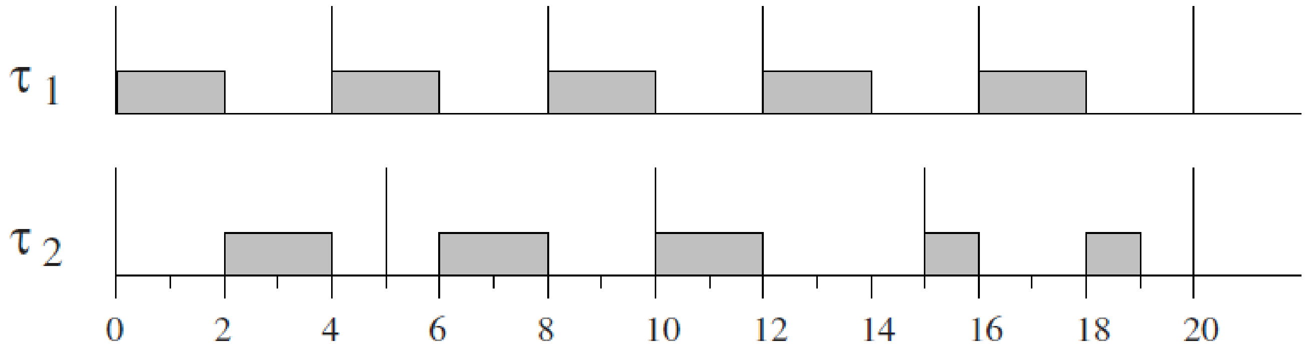
$C_s = 2$
$T_s = 6$



Example 3

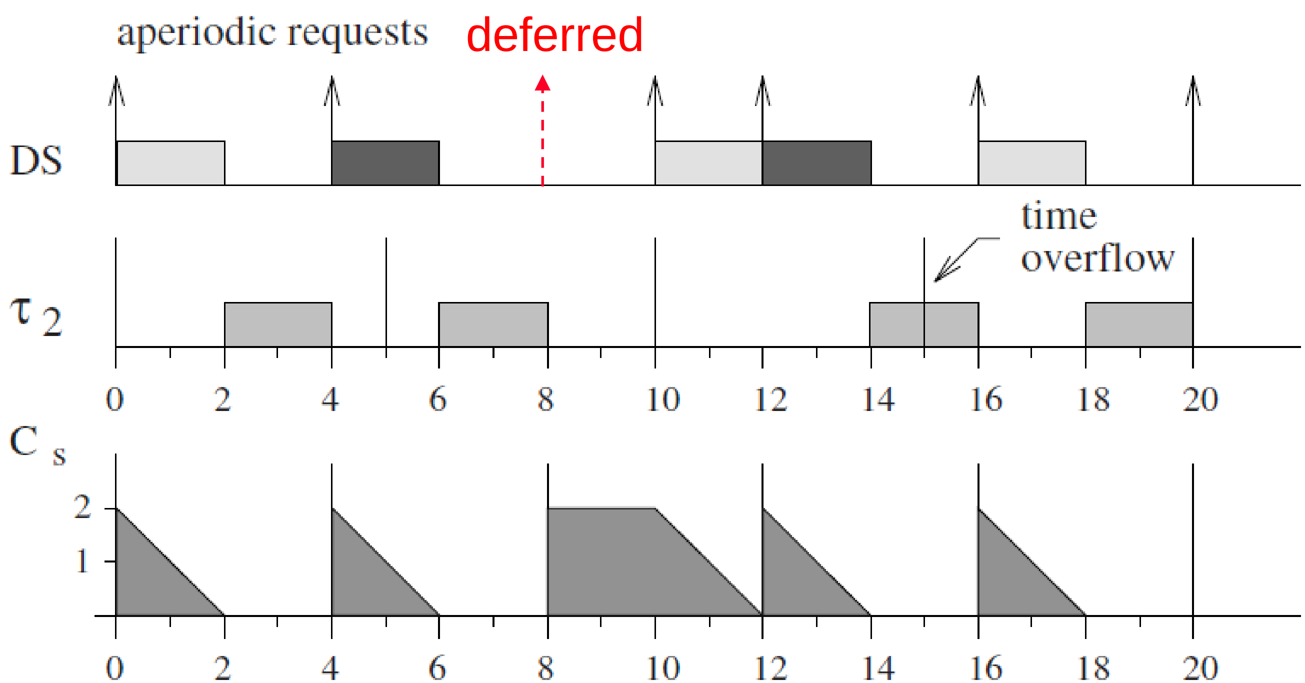
DS is not equivalent to a periodic task in RM → difficult schedulability analysis

Schedulable original task set



(a)

Replace T1 by DS
→ Not schedulable



DS schedulability analysis

- Given a periodic task set with total utilization U_p and a DS with utilization U_s

- The schedulability is guaranteed if

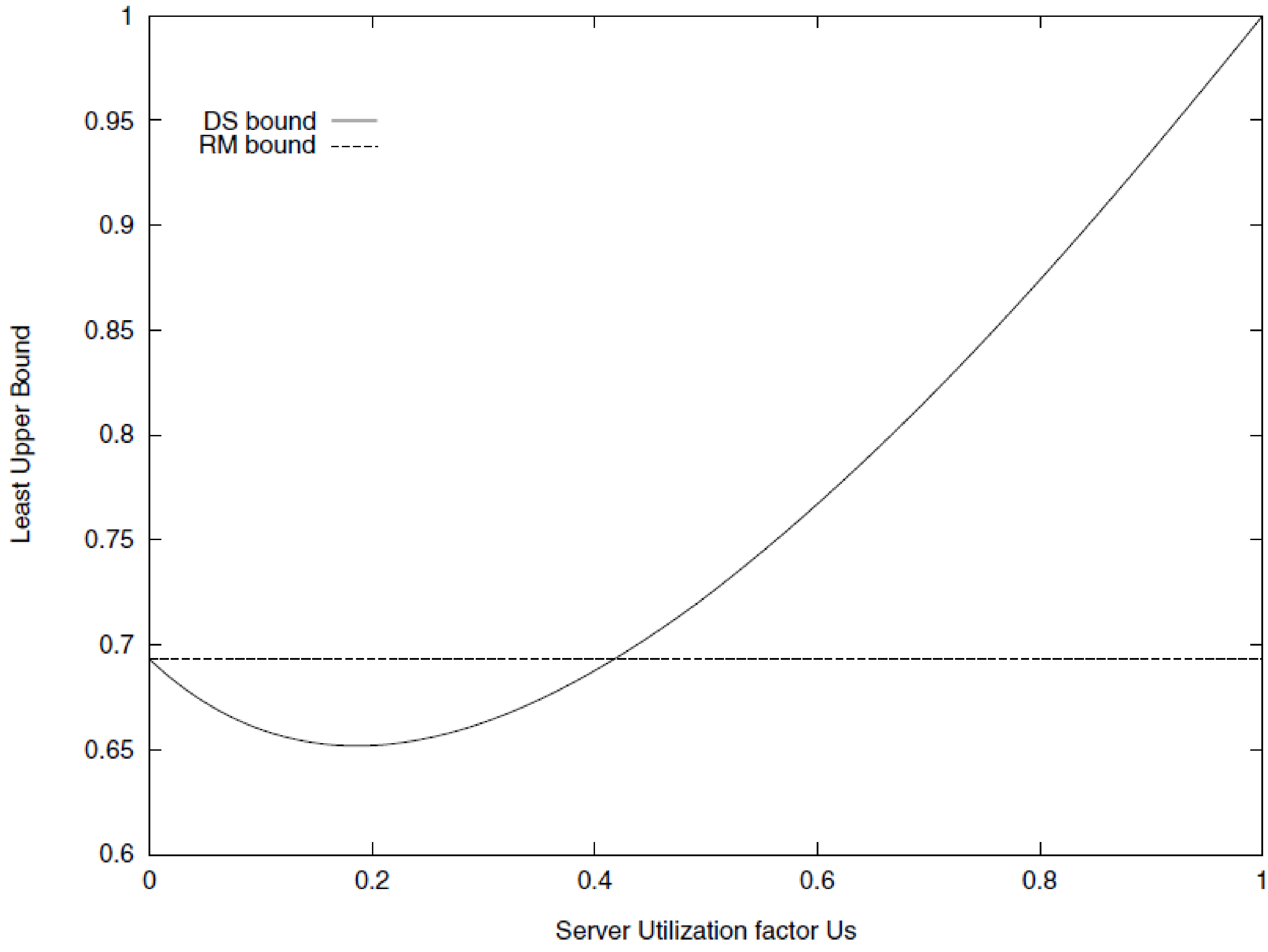
$$U_p \leq n\left[\left(\frac{U_s + 2}{2Us + 1}\right)^{\frac{1}{n}} - 1\right]$$

- Therefore the whole system bound is

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

$$\lim_{n\to\infty} U_{lub} = U_s + \ln\left(\frac{U_s + 2}{2U_s + 1}\right)$$

DS schedulability analysis



DS schedulability analysis

- Given a set of n periodic tasks with utilization U_i and a DS with utilization Us,
- The periodic task set is schedulable under RM if

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s + 2}{2U_s + 1}$$

Dimensioning a DS

- Find Us, Ts, Cs?
- Similar to PS, let

$$P \stackrel{\text{def}}{=} \prod_{i=1}^{n} (U_i + 1)$$

- Then from guarantee condition we have

$$U_s \leq \frac{2 - P}{2P - 1}$$

- So the max utilization for server is

$$U_s^{max} = \frac{2 - P}{2P - 1}$$

→choose $T_s = \min(T_i)$ and $C_s = Us * Ts$

Exercise 2

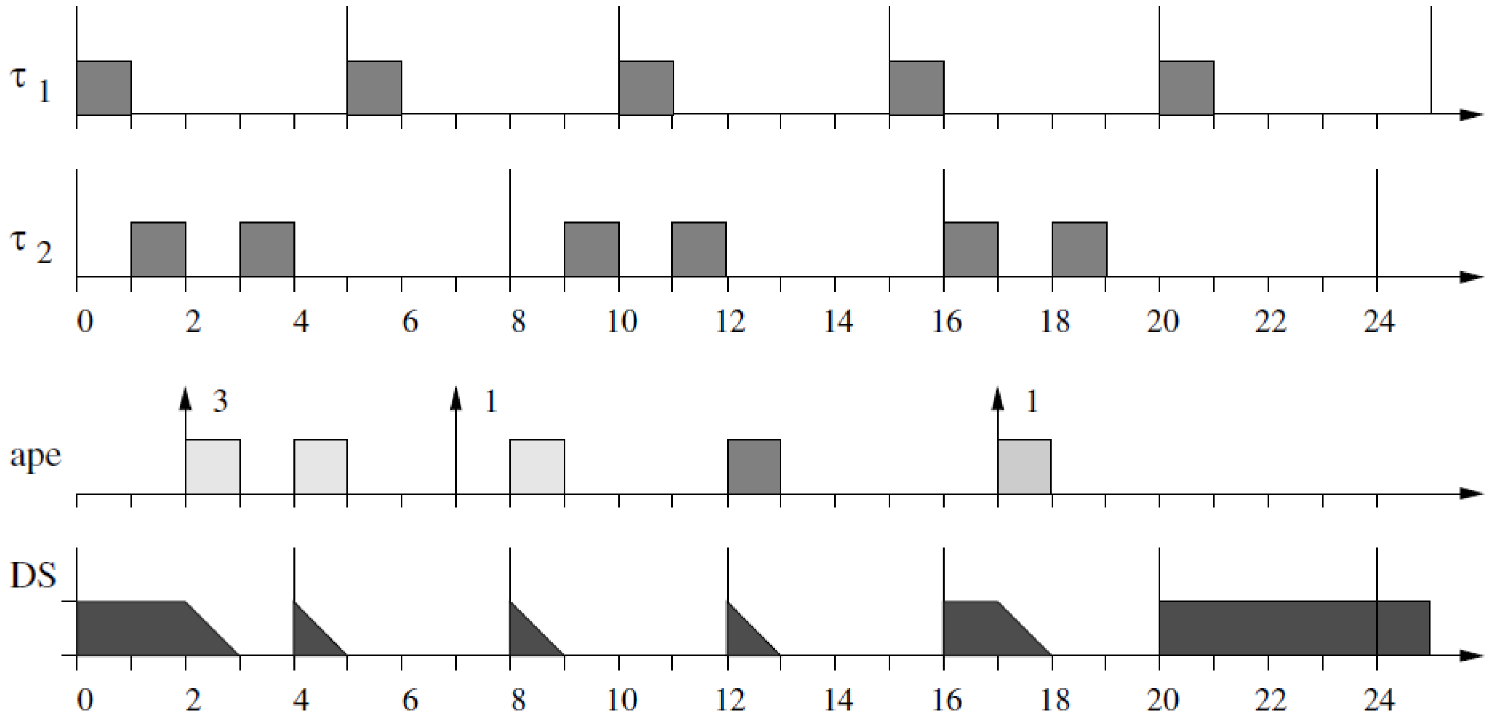
- From Ex1:
  - Periodic tasks: $C1 = 1$, T1=5, $C2 = 2$, T2=8 (scheduled by RM)
  - Aperiodic tasks:

	a_i	C_i
J_1	2	3
J_2	7	1
J_3	17	1

- Solve the scheduling problem based on DS, with highest possible priority and maximum utilization

Ex 2

- Maximum utilization: $U_{max} = 1/4$
- Highest priority with RM: $T_s = 4$
- ➔ $C_s = 1$



Priority Exchange (PE)

- Similar to DS with
  - Better schedulability bound
  - Worse aperiodic responsiveness
- PE algorithm
  - Create a periodic task (PE) with high priority for aperiodic load
  - PE preserves capacity by exchanging for lower priority tasks' execution time.
  - Upon PE activation: if there is no aperiodic load, lower priority tasks can execute and PE accumulates capacity at the corresponding priorities.
  - If there is no task waiting, PE capacity resolves to 0

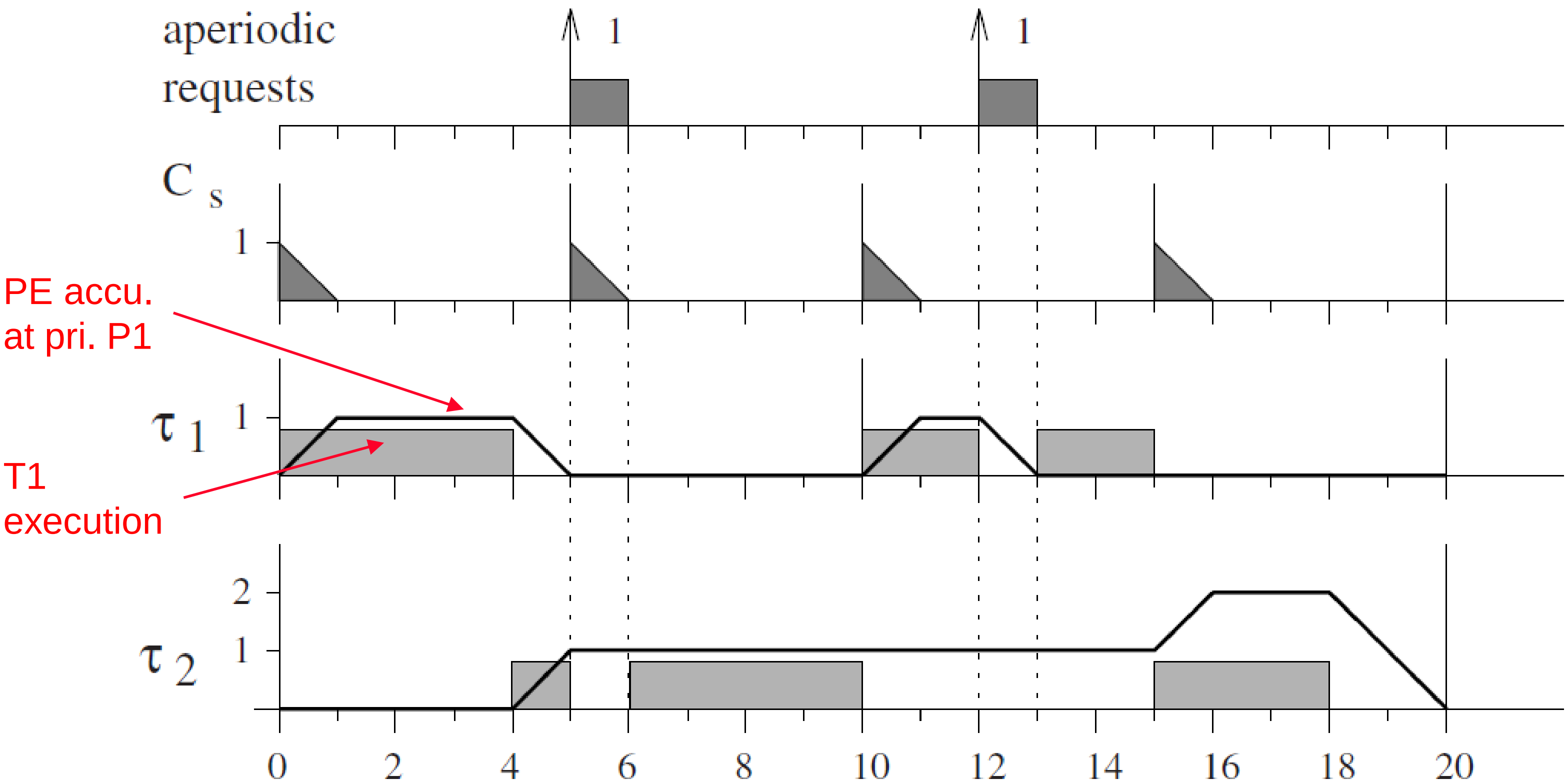
Example

	C_i	T_i
τ_1	4	10
τ_2	8	20

Server

$C_s = 1$

$T_s = 5$



Exercise 4

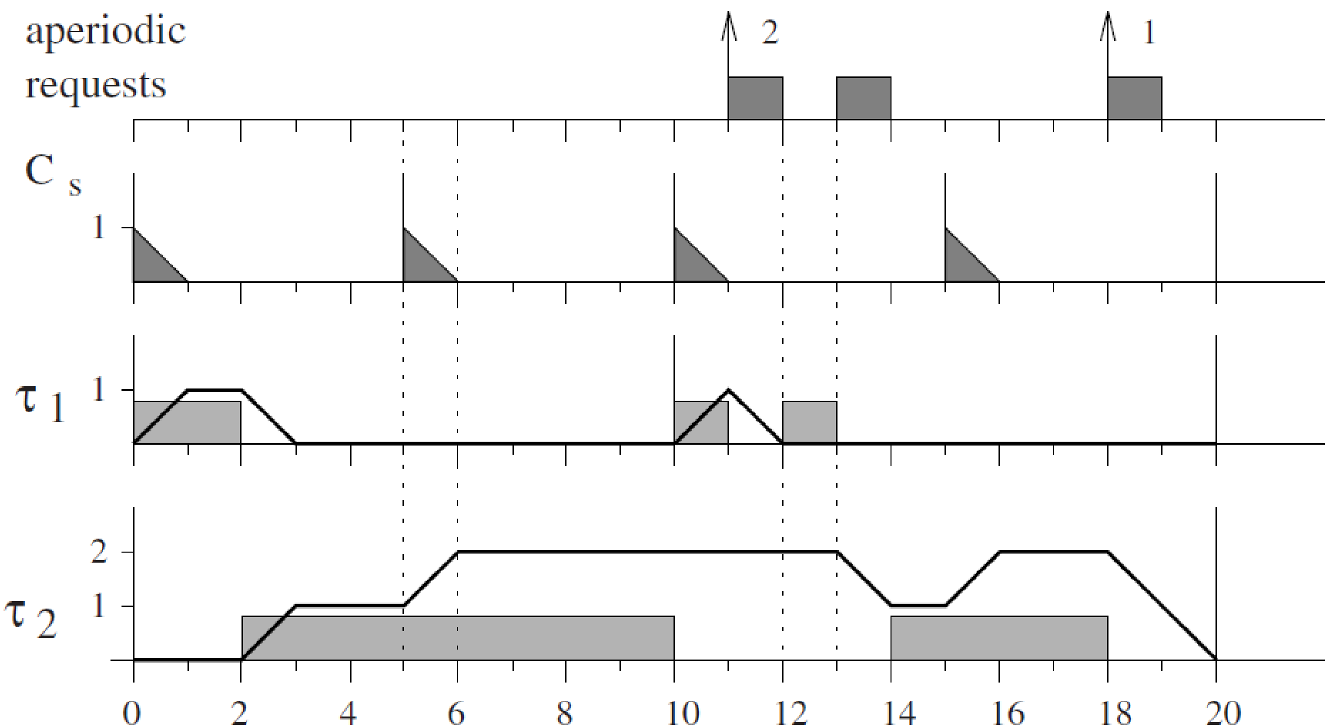
- Why in this case do the PE and T1 preempt each other?

	C_i	T_i
τ_1	2	10
τ_2	12	20

Server

$C_s = 1$

$T_s = 5$



Schedulability bound

- Given a periodic task set with total utilization U_p and a PE server with utilization U_s

- The schedulability is guaranteed if

$$U_p \le n\left[\left(\frac{2}{U_s+1}\right)^{1/n} - 1\right]$$

- Sizing PE server

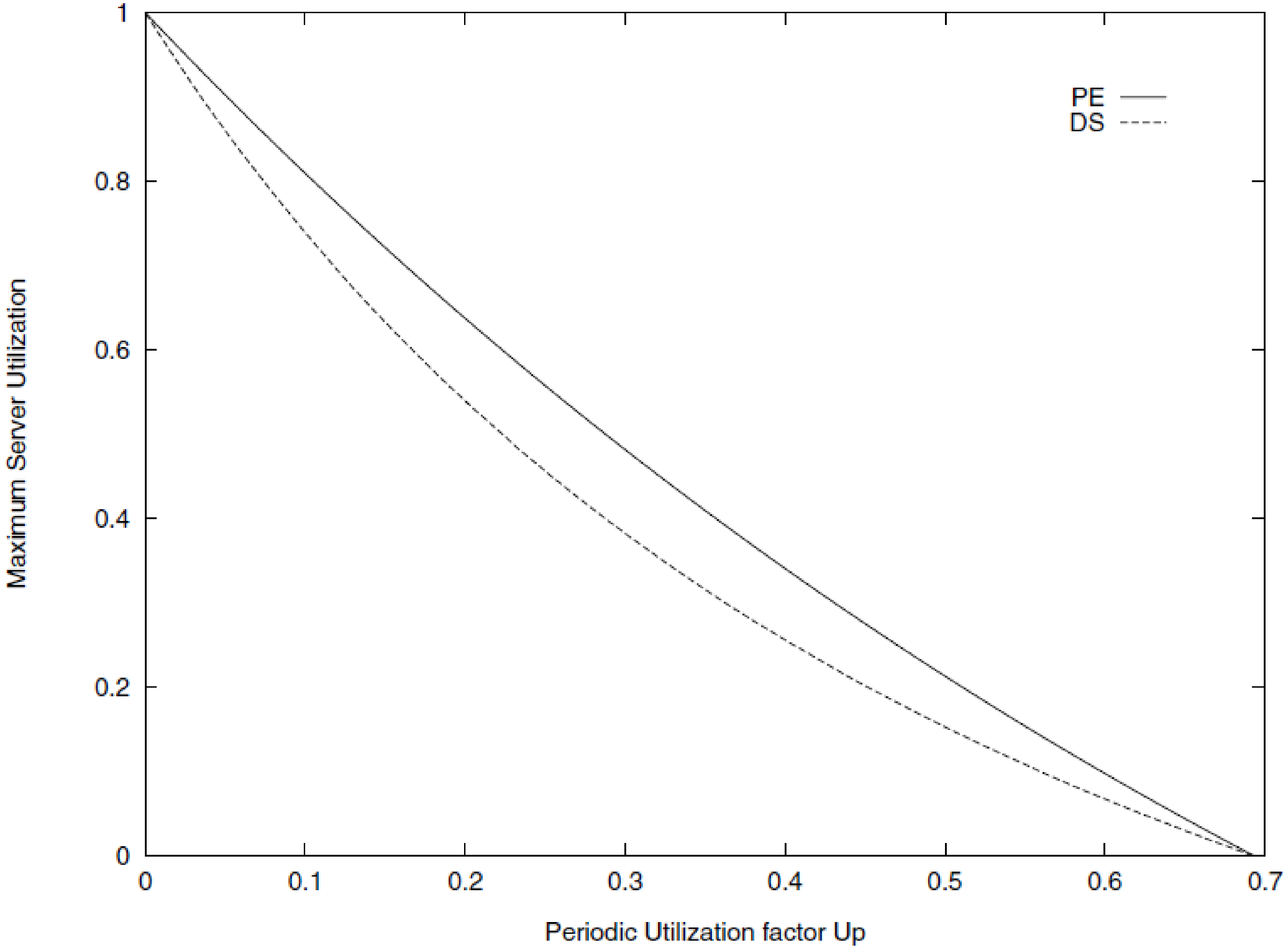
$$U_{PE}^{max} = \frac{2-P}{P}$$

where

$$P = \prod_{i=1}^{n}(U_i+1)$$

Comparing Up between DS and PE

Which is better in term of periodic utilization?



Sporadic Server

- Similar to DS
- Delay the replenishing time of server → server becomes equivalent to a normal periodic task
- Idea:
  - Divide the timeline of SS to active and inactive time slices
    - Active: server serves or may serve periodic task
    - Inactive: server does not serve periodic task
  - Start of active time slice: mark delayed replenishing time
  - End of active time slice: calculate replenishing amount

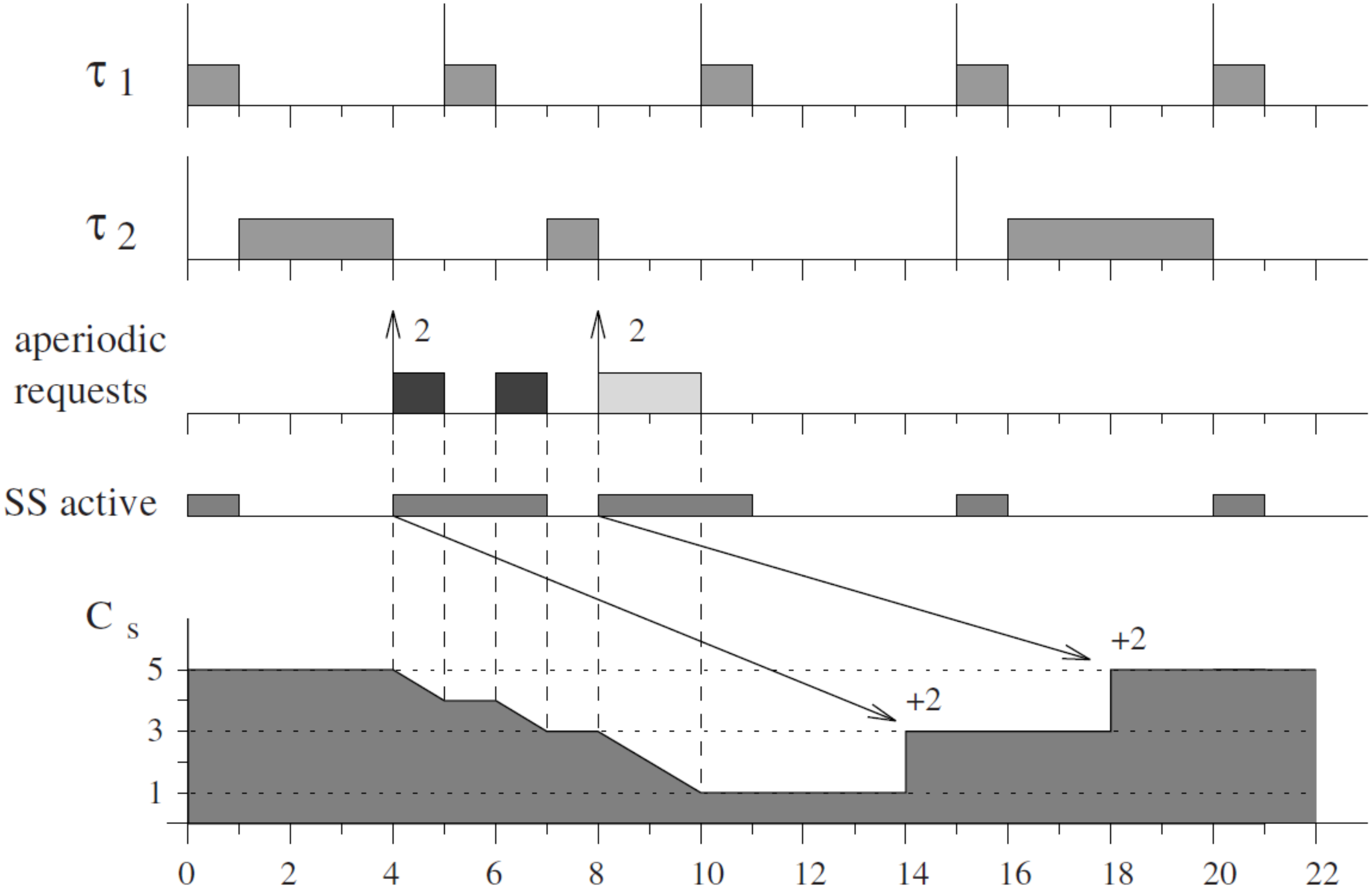
Example: intermediate SS

	C_i	T_i
τ_1	1	5
τ_2	4	15

Server

$C_s = 5$

$T_s = 10$



Sporadic Server

P_{exe} It denotes the priority level of the task that is currently executing.

P_s It denotes the priority level associated with SS.

Active SS is said to be *active* when $P_{exe} \geq P_s$.

Idle SS is said to be *idle* when $P_{exe} < P_s$.

RT It denotes the *replenishment time* at which the SS capacity will be replenished.

RA It denotes the *replenishment amount* that will be added to the capacity at time RT.

$$RT = Start\_of\_Active + Ts$$
$$RA = Consume\_capacity$$

Exercise 5:

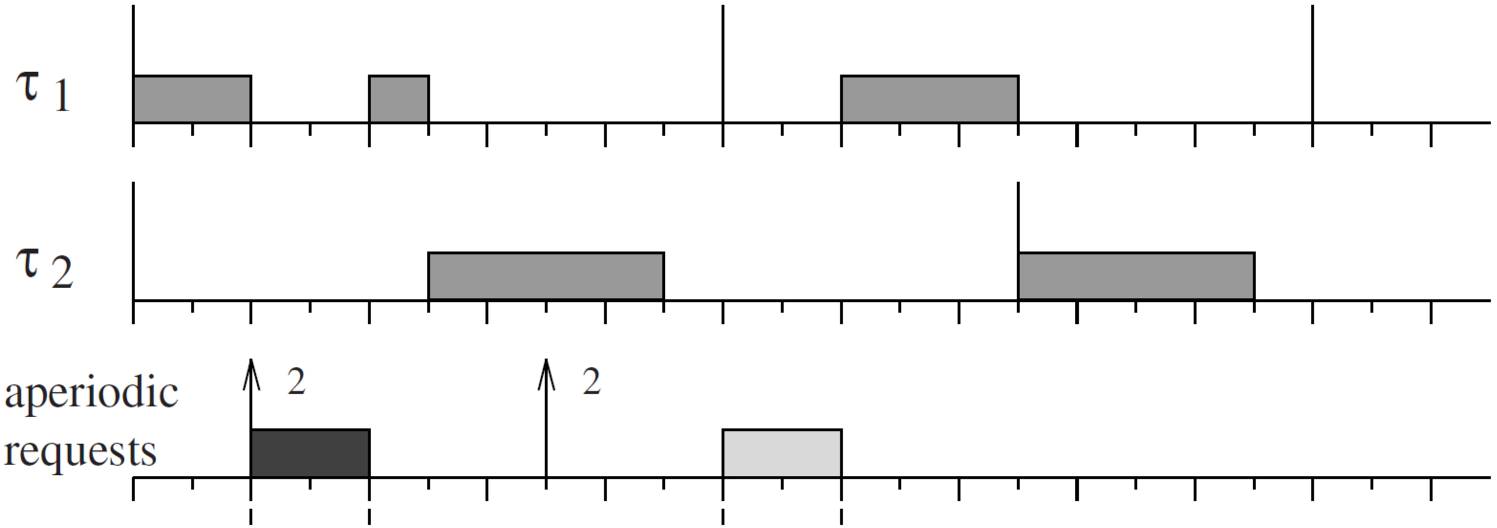
- Find response time of aperiodic tasks

	C_i	T_i
τ_1	3	10
τ_2	4	15

Server

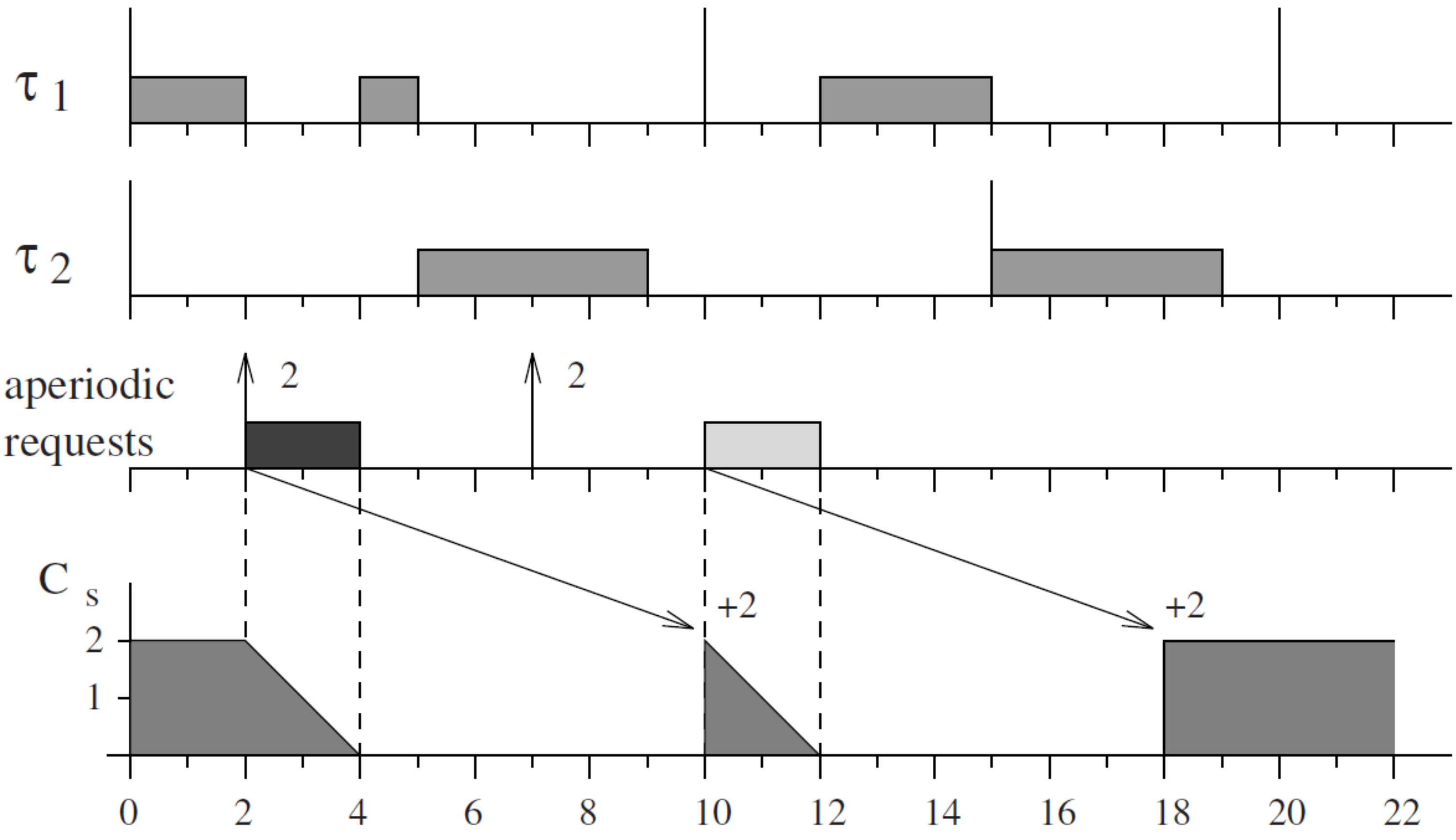
$C_s = 2$

$T_s = 8$



Exercise 5

- Server is with highest priority



Periodic task equivalent

- SS (like DS) violates assumption: task must not suspend itself and reactivate later

- SS is different from DS: replenishing time is delayed

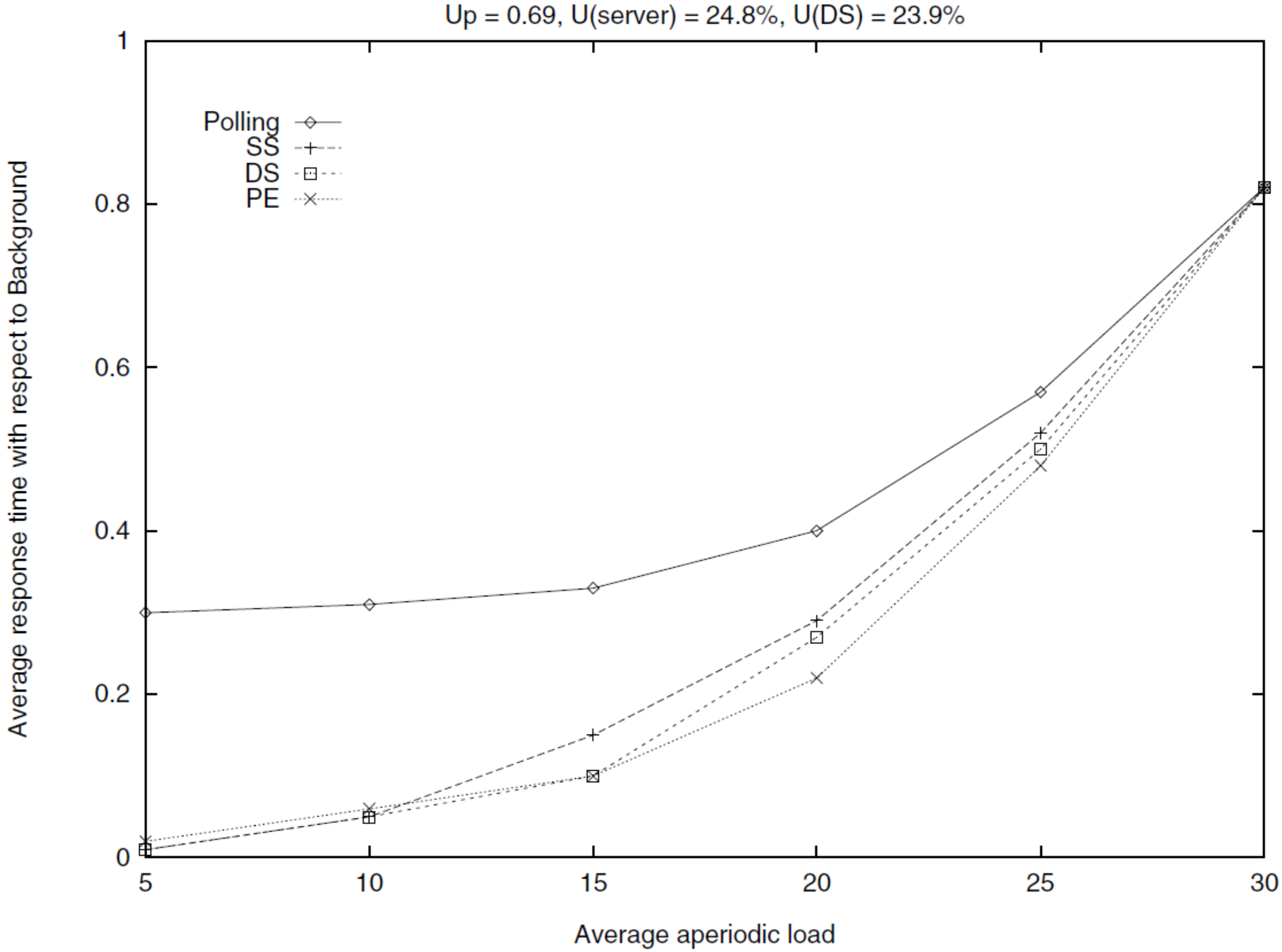
- **Theorem 5.1 (Sprunt, Sha, Lehoczky)** *A periodic task set that is schedulable with a task τi is also schedulable if τi is replaced by a Sporadic Server with the same period and execution time*

- Therefore, schedulability analysis of SS is similar to Polling Server with RM

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

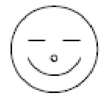
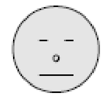
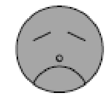
$$U_{SS}^{max} = \frac{2-P}{P}.$$

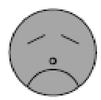
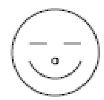
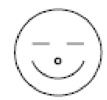
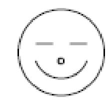
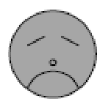
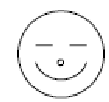
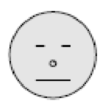
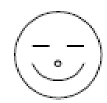
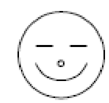
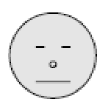
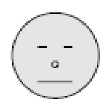
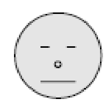
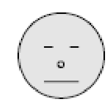
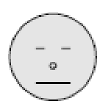
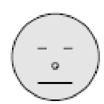
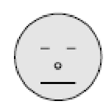
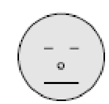
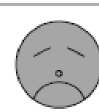
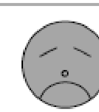
Performance evaluation



Performance results of PS, DS, PE, and SS

Comparison

		
excellent	good	poor

	performance	computational complexity	memory requirement	implementation complexity
Background Service				
Polling Server				
Deferrable Server				
Priority Exchange				
Sporadic Server				
Slack Stealer				

Note: Slack Stealer can be read from textbook

Real-time Systems

Week 8:
Dynamic Priority Servers

Ngo Lam Trung

Dept. of Computer Engineering

How to further increase U_{lub}?

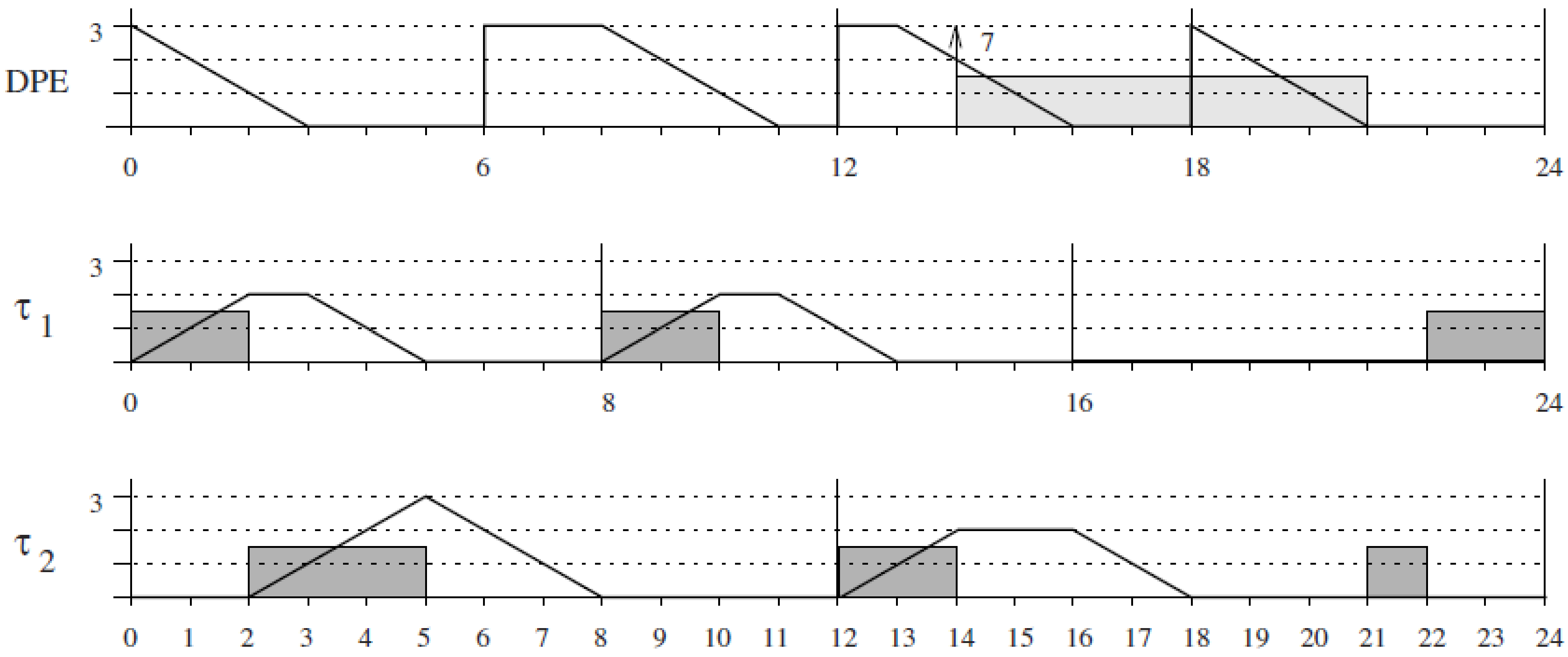
- Fixed priority server uses fixed priority algo.
  - Simple
  - Small U_{lub}
- How to increase U_{lub}?

➔uses the same approach: create periodic task to serve aperiodic task (the server)

→apply dynamic priority scheduling algorithm (EDF) to increase utilization bound

Dynamic Priority Exchange Server

- Similar to fixed priority exchange server
  - Server can exchange capacity with other tasks that have longer deadline at the scheduling time
  - Server accumulate capacity time with the new deadline
  - Server capacity will be consumed until it is exhausted

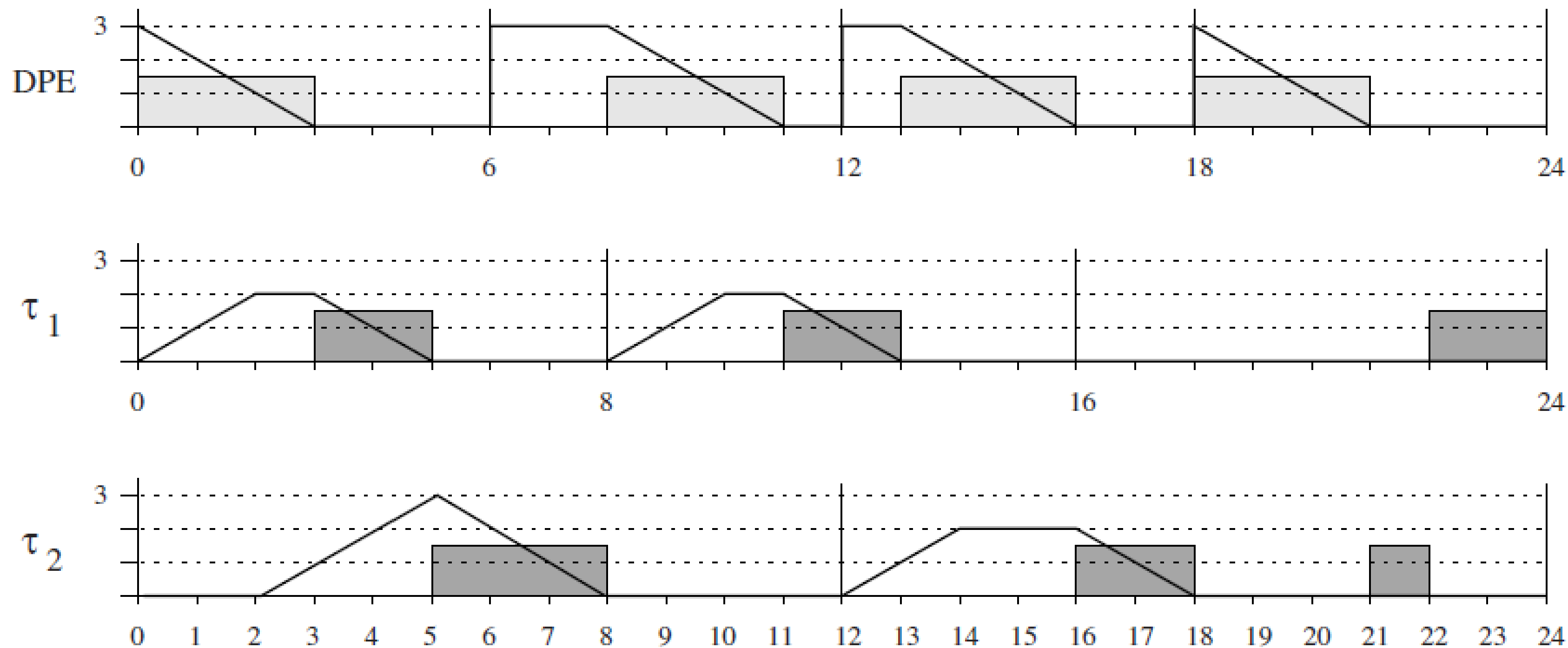


$\tau_1(2, 8)$, $\tau_2(3, 12)$, DPE(3,6)

Schedulability analysis

Theorem 6.1 (Spuri, Buttazzo) *Given a set of periodic tasks with processor utilization U_p and a DPE server with processor utilization U_s, the whole set is schedulable by EDF* if *and* only if

$$U_p + U_s \leq 1.$$



$U = \frac{3}{6} + \frac{2}{8} + \frac{3}{12} = 1$ ➔ schedulable task set

Reclaiming spare capacity

- What if the real C is smaller than worst case C?

→ Spare capacity from can be reclaimed and transfer to aperiodic capacity.