

# Real-Time Scheduling

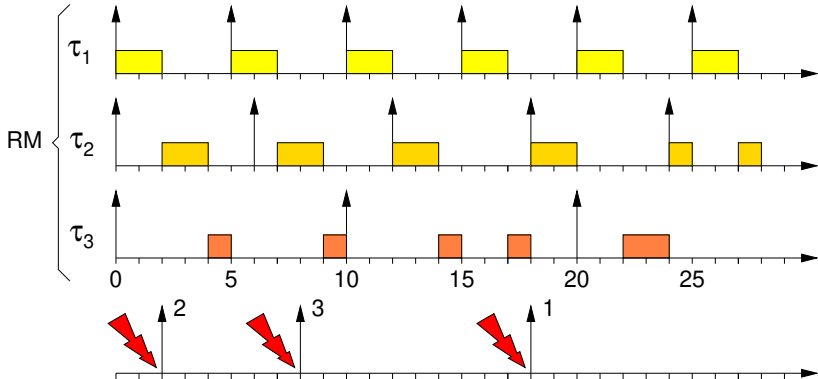
## Aperiodic tasks

Tullio Facchinetti  
<tullio.facchinetti@unipv.it>

16<sup>th</sup> October, 2025

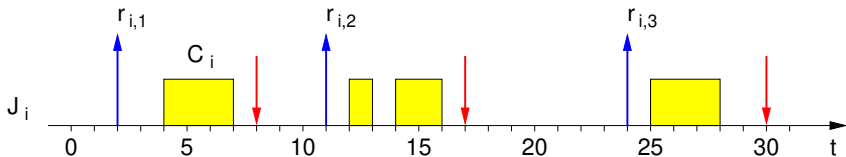
<http://robot.unipv.it/toolleeo>

## Aperiodic tasks: task model



- when aperiodic requests need to be scheduled
- guarantees on periodic tasks

## Aperiodic tasks



aperiodic tasks

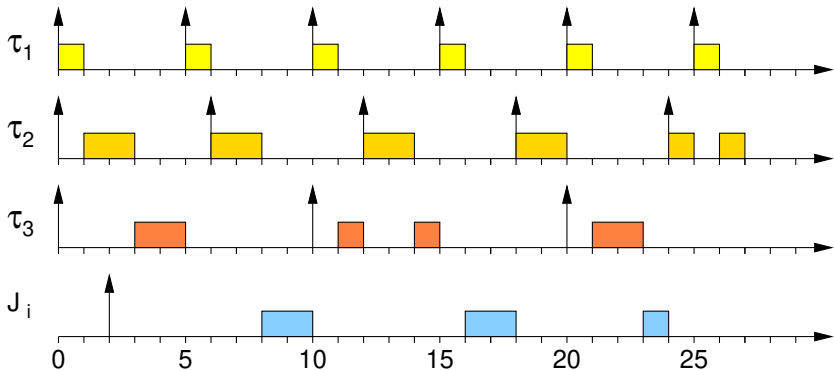
$$r_{i,k+1} > r_{i,k}$$

sporadic tasks

$$r_{i,k+1} \geq r_{i,k} + T_i$$

- $T_i$  is said *Minimum Interarrival Time*, being the **minimum time between two consecutive arrivals** of the same sporadic task.
- When sporadic tasks are generated at the maximum frequency, **they are equivalent to period tasks.**

# Background scheduling



- aperiodic tasks are scheduled when the processor is idle
- simple and easy-to-implement technique

## Dedicated methods

different methods have been proposed, distinguishing between **static** and **dynamic** priority assignment

considered algorithms:

### Static priorities

- Polling Server
- Sporadic Server

### Dynamic priorities

- Total Bandwidth Server
- Constant Bandwidth Server

## Polling Server

scheduling of soft aperiodic tasks concurrently with  
hard periodic tasks

### assumptions:

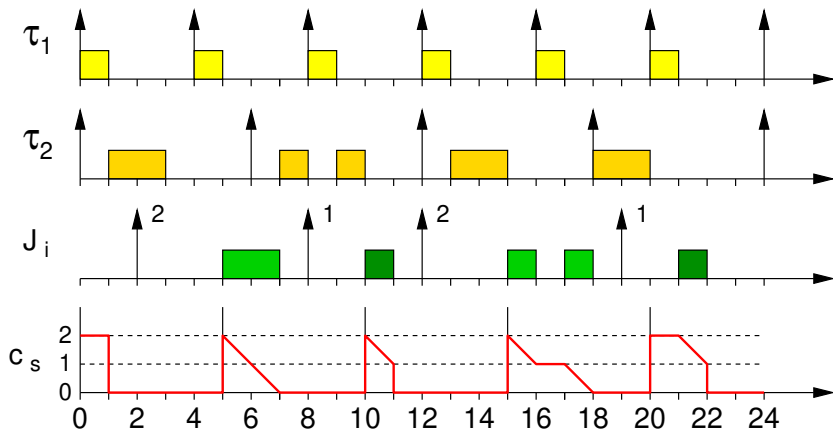
- full-preemption
- periodic tasks are scheduled by RM
- implicit deadlines
- aperiodic tasks have
  - unknown arrival time
  - known worst-case computation time

## Polling Server

- 1 **period**  $T_s$ , **nominal capacity**  $C_s$  and **current capacity**  $c_s$
- 2 every  $T_s$  time units the current capacity is recharged up to the nominal value  $C_s$  (i.e.,  $c_s = C_s$ )
- 3 one unit of  $c_s$  is consumed for each slot served to an aperiodic task
- 4 if there are no aperiodic tasks ready for execution, the server self-suspends and flushes its current capacity (i.e.,  $c_s = 0$ )

the flushing of the capacity may cause the presence of idle times that is not exploitable by aperiodic tasks ready for execution

## Polling Server



$$\tau_1 = (1, 4)$$

$$\tau_2 = (2, 6)$$

$$C_s = 2 \text{ e } T_s = 5$$

## Schedulability analysis

from the schedulability viewpoint, a Polling Server behaves like a periodic task having period  $T_s$  and WCET  $C_s$

$$U_p + U_s \leq U_{lub}(n + 1)$$

$$\sum_{i=1}^n \frac{C_i}{T_i} + \frac{C_s}{T_s} \leq (n + 1) \left[ 2^{1/(n+1)} - 1 \right]$$

in case there are  $m$  Polling Servers:

$$U_p + \sum_{j=1}^m U_{s_j} \leq U_{lub}(n + m)$$

## Sporadic Server

parameters for its definition:

- period  $T_s$
- maximum budget  $C_s$
- static priority  $P_s$  (e.g., set according to RM)

parameters used for its functioning:

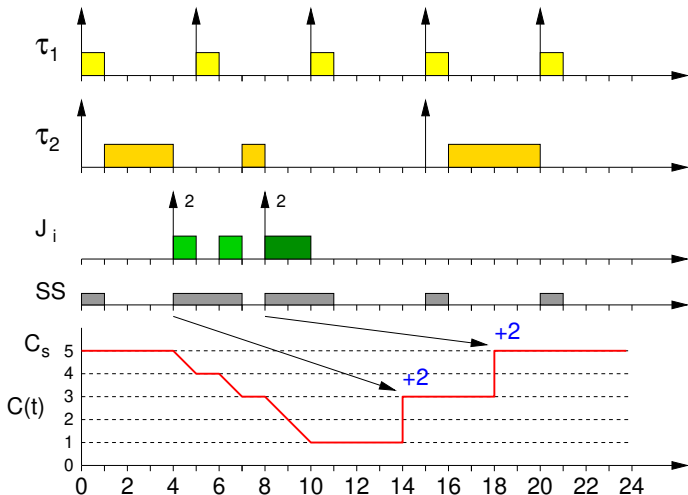
- $C(t)$  : current server capacity
- $P_{exe}$  : priority of the running task

## Sporadic Server: operating rules

the Sporadic Server works according to the following rules:

- 1 the server is said **active** at time  $t$  if  $P_{exe} \geq P_s$  and  $C(t) > 0$
- 2 the server is said **idle** at time  $t$  if  $P_{exe} < P_s$  or  $C(t) = 0$
- 3 at time  $t = 0$  the server is idle and  $C(0) = C_s$
- 4 when the server becomes active at time  $t_1$  a corresponding recharging time is set at time  $t_r = (t_1 + T_s)$
- 5 when the server becomes idle at time  $t_2 > t_1$  a recharge budget is set equal to the budget  $C_r$  consumed during the interval  $[t_1, t_2)$
- 6 at time  $t_r$  the capacity  $C_r$  is added to the current budget

## Sporadic Server: example



$$\tau_1 = (1, 5)$$

$$\tau_2 = (4, 15)$$

$$C_s = 5 \text{ e } T_s = 10$$

## SS: schedulability analysis

a Sporadic Server **does not behave like a periodic task**

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq n \left[ \left( \frac{2}{U_s + 1} \right)^{1/n} - 1 \right]$$

- let  $U_p$  be the utilization of all periodic tasks
- the highest utilization of the sporadic server that guarantees the schedulability of periodic tasks is  $U_{SS}^*$

$$U_{SS}^* = 2 \left( \frac{U_p}{n} + 1 \right)^{-n} - 1$$

## Scheduling algorithms for dynamic priorities

many algorithms are adaptations of static priority scheduling algorithms

example:

- Dynamic Sporadic Server

some algorithms were born for dynamic priorities:

- Total Bandwidth Server
- Total Bandwidth Server\*
- Constant Bandwidth Server

## Assumptions

concurrent scheduling of soft aperiodic tasks and  
hard periodic tasks

### assuming that

- periodic tasks are scheduled by EDF
- implicit deadlines (deadlines are equal to periods)
- full preemption
- for aperiodic tasks:
  - unknown arrival times
  - known computation times

## Total Bandwidth Server

### server design parameter:

- bandwidth (utilization)  $U_s$

### operating rules:

- an aperiodic task  $J$  arrives at time  $r_k$
- the  $J$  task requires  $C_k$  time units to execute
- an absolute deadline  $d_k$  is calculated for  $J$

$$d_k = \max(r_k, d_{k-1}) + \frac{C_k}{U_s}$$

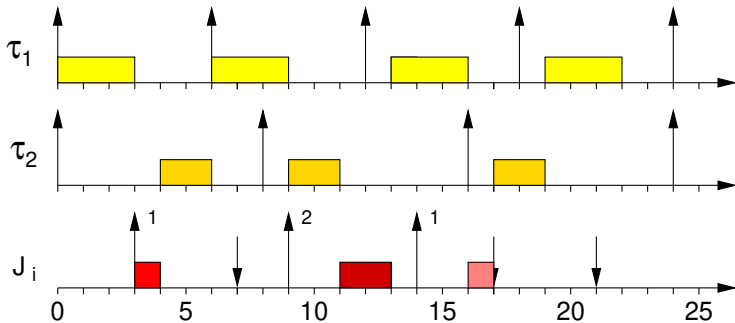
- being  $d_0 = 0$  by definition
- $J$  is scheduled by EDF considering the computed deadline  $d_k$

## TBS: example

$$\tau_1 = (3, 6)$$

$$\tau_2 = (2, 8)$$

$$U_s = 1 - U_p = 0.25$$



$$d_k = \max(r_k, d_{k-1}) + \frac{C_k}{U_s}$$

$$\max(3, 0) + \frac{1}{0.25} = 3 + 4 = 7$$

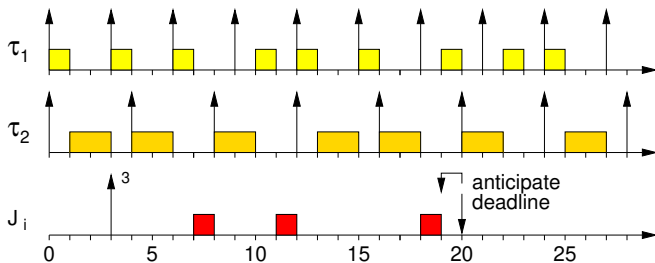
$$\max(9, 7) + \frac{2}{0.25} = 9 + 8 = 17$$

$$\max(14, 17) + \frac{1}{0.25} = 17 + 4 = 21$$

## Total Bandwidth Server\*

TBS\* shortens the deadline of the aperiodic task  $J$  as much as possible

- the first assignment is done as in TBS
- the deadline can be shortened to the actual finishing time



$$\tau_1 = (1, 3)$$

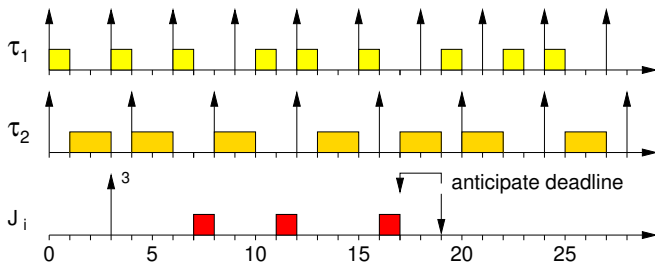
$$\tau_2 = (2, 4)$$

$$U_s = 1 - U_p = 0.1667$$

## Total Bandwidth Server\*

TBS\* shortens the deadline of the aperiodic task  $J$  as much as possible

- the first assignment is done as in TBS
- the deadline can be shortened to the actual finishing time



$$\tau_1 = (1, 3)$$

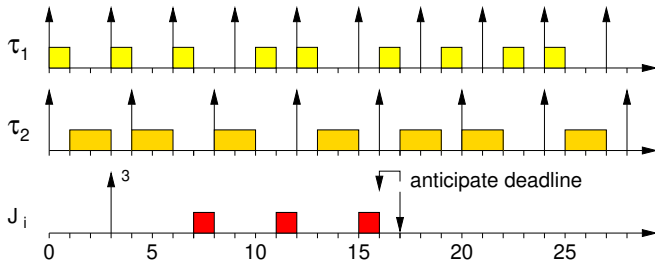
$$\tau_2 = (2, 4)$$

$$U_s = 1 - U_p = 0.1667$$

## Total Bandwidth Server\*

TBS\* shortens the deadline of the aperiodic task  $J$  as much as possible

- the first assignment is done as in TBS
- the deadline can be shortened to the actual finishing time



$$\tau_1 = (1, 3)$$

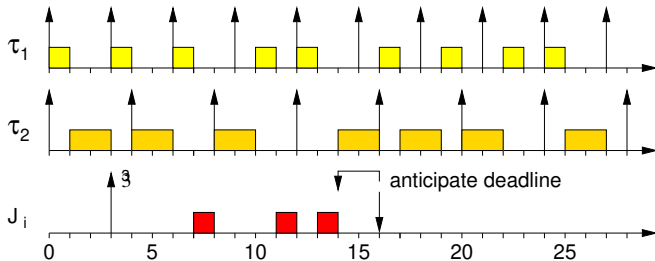
$$\tau_2 = (2, 4)$$

$$U_s = 1 - U_p = 0.1667$$

## Total Bandwidth Server\*

TBS\* shortens the deadline of the aperiodic task  $J$  as much as possible

- the first assignment is done as in TBS
- the deadline can be shortened to the actual finishing time



$$\tau_1 = (1, 3)$$

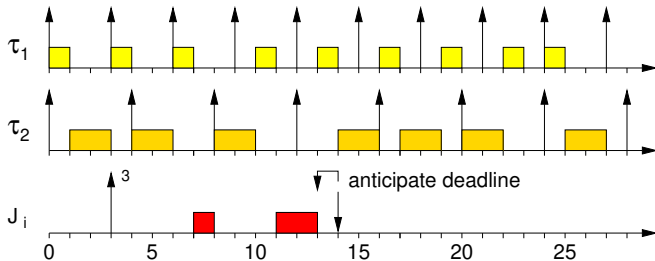
$$\tau_2 = (2, 4)$$

$$U_s = 1 - U_p = 0.1667$$

## Total Bandwidth Server\*

TBS\* shortens the deadline of the aperiodic task  $J$  as much as possible

- the first assignment is done as in TBS
- the deadline can be shortened to the actual finishing time



$$\tau_1 = (1, 3)$$

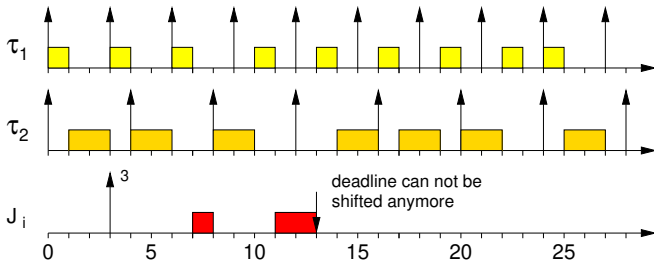
$$\tau_2 = (2, 4)$$

$$U_s = 1 - U_p = 0.1667$$

## Total Bandwidth Server\*

TBS\* shortens the deadline of the aperiodic task  $J$  as much as possible

- the first assignment is done as in TBS
- the deadline can be shortened to the actual finishing time



$$\tau_1 = (1, 3)$$

$$\tau_2 = (2, 4)$$

$$U_s = 1 - U_p = 0.1667$$

## Total Bandwidth Server\*

the deadline is shortened  
using an iterative process

being at iteration  $s$ :

- $d_k^s$  is the deadline assigned to  $J_k$
- $f_k^s$  is the finishing time of  $J_k$

the iterative shortening process is:

- at step  $s + 1$  it is set  $d_k^{s+1} = f_k^s$
- the process stops when  $d_k^{s+1} = d_k^s$

## Total Bandwidth Server\*

## computational issue

- the calculation of the worst-case finishing time may require to perform the schedule until the desired time
  - in many cases (e.g., high utilization of periodic tasks), this may lead to impractical computation times
- 
- the finishing time  $f_k^s$  can be approximated
  - an upper bound is proved to exist
  - its calculation is fast enough to be used online

## TBS\*: optimality

### Theorem

TBS\* generates the absolute deadline of an aperiodic task such as its response time is minimized

therefore, TBS\* is optimal  
(in the sense of response time minimization)

## Constant Bandwidth Server

- the CBS implements a bandwidth reservation scheme
- let  $U_s$  be the bandwidth assigned to the CBS
- the CBS never requires more than  $U_s$  to work

the server absolute deadline is based on the server bandwidth

- ... even in case of overload (overrun of aperiodic tasks)

the absolute deadline is postponed to achieve the constraint on the bandwidth assigned to the server

- performance in terms of response time for aperiodic tasks is similar to those of TBS

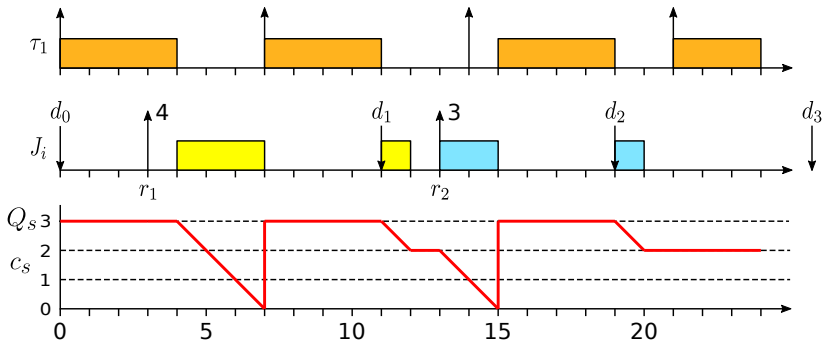
## CBS: operating rules

- 1 maximum budget  $Q_s$ , period  $T_s$  and current budget  $c_s$
- 2 server bandwidth:  $U_s = Q_s/T_s$
- 3 at every time, the  $k$ -th calculated absolute deadline  $d_{s,k}$  is the current deadline of the CBS
- 4 by definition  $d_{s,0} = 0$
- 5 at each job  $J_{i,j}$  it is assigned the absolute deadline  $d_{i,j} = d_{s,k}$
- 6 the current budget  $c_s$  decreases of one unit for each time unit of execution
- 7 when  $c_s = 0$  the budget is refilled (i.e., it is set  $c_s = Q_s$ ), and a new absolute deadline is computed as  $d_{s,k+1} = d_{s,k} + T_s$

## CBS: operating rules

- 1 the server is *active* when there are pending aperiodic jobs, *idle* otherwise
- 2 when the **server is active**, a new aperiodic job  $J_{i,j}$  is queued with arbitrary policy to the queue of pending requests
- 3 when the **server is idle** and a new aperiodic job  $J_{i,j}$  is released:
  - if  $c_s \geq (d_{s,k} - r_{i,j})U_s \Rightarrow$  compute a new absolute deadline  $d_{s,k+1} = r_{i,j} + T_s$  and set  $c_s = Q_s$
  - otherwise  $\Rightarrow$  schedule  $J_{i,j}$  with the current absolute deadline  $d_{s,k}$  and budget  $c_s$
- 4 when **a job finishes**, the next one is served with absolute deadline  $d_{s,k}$  and budget  $c_s$
- 5 if no more tasks are present in the queue, the server becomes idle

# CBS: example



$$c_s \geq (d_{s,0} - r_1)U_s \Rightarrow 3 \geq (0 - 3)0.375$$

$$c_s < (d_{s,2} - r_2)U_s \Rightarrow 2 < (19 - 13)0.375 = 2.25$$

$$\tau_1 = (4, 7)$$

$$Q_s = 3 \quad T_s = 8 \quad U_s = 0.375$$

## CBS: schedulability analysis

- let  $U_p$  be the utilization of periodic tasks
- the utilization of a CBS is always  $U_s = Q_s/T_s$  independently from the timing parameters of aperiodic jobs
- the system is schedulable iff  $U_p + U_s \leq 1$
- since the budget  $c_s$  is never null, the CBS performs an automatic reclaiming of unused computing time in case of earlier termination of an aperiodic job

in case of  $m$  CBSs where the  $i$ -th server has utilization  $U_{s_i}$ , the system is schedulable iff

$$U_p + U_s \leq 1 \quad U_s = \sum_{i=1}^m U_{s_i}$$

## Summary

### TBS

- trivial operating rules
- good performance
- does not tolerate overloads

### TBS\*

- optimal response time
- higher complexity w.r.t. TBS
- trade-off can be established between response time and computational overhead

### CBS

- bandwidth reservation in case of overload
- good performance (comparable with TBS)
- simple implementation