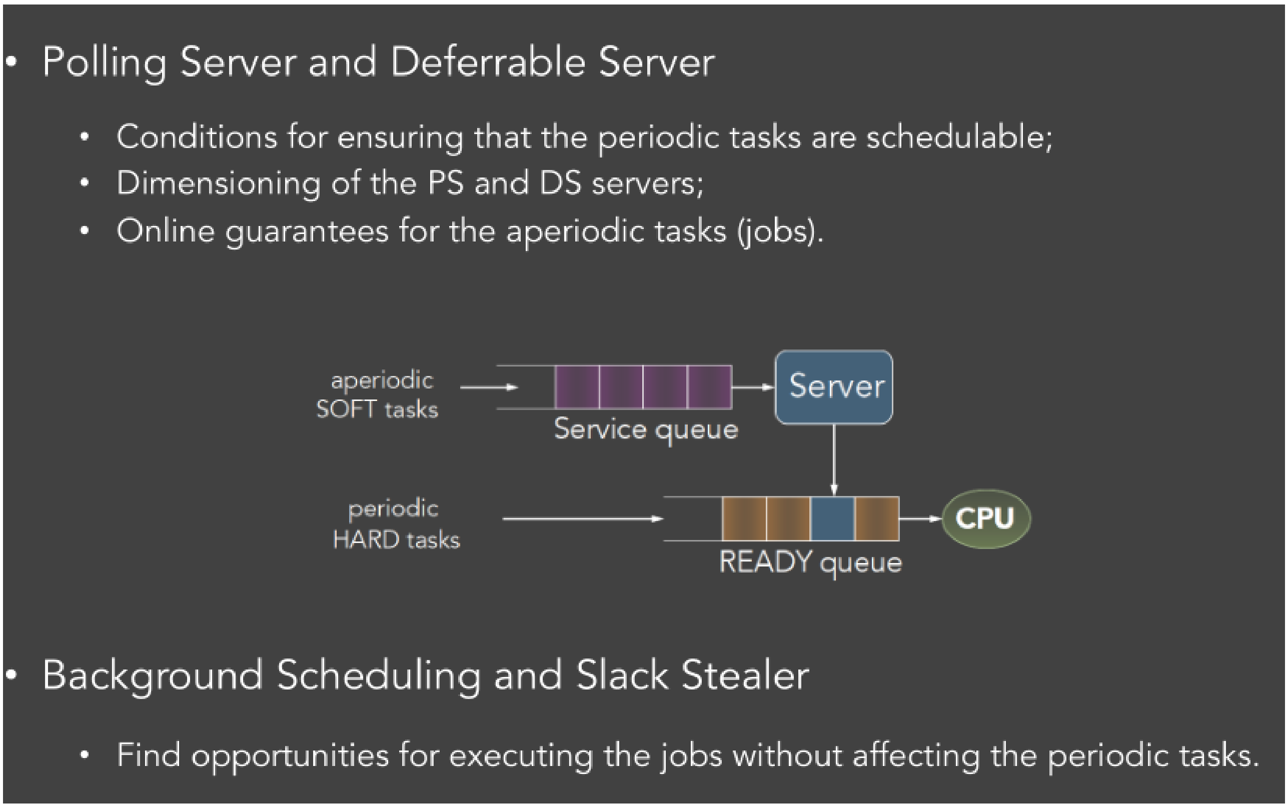
# 0_Hybrid Task Scheduling

---




# 1. Introduction

## 1.1. Background

Real-time systems often need to handle both periodic and aperiodic tasks

## 1.2. Hybrid Tasks Set

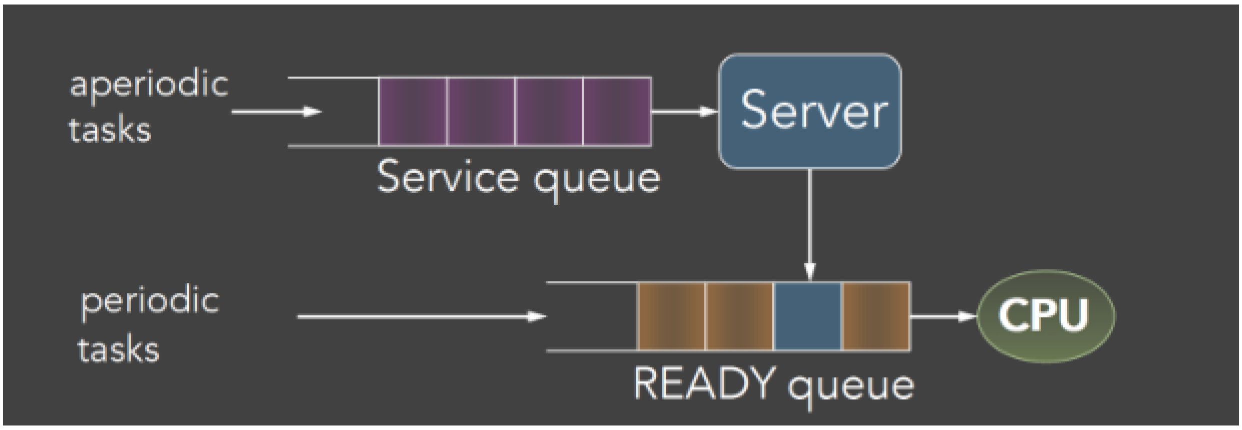
$$\Gamma = \{\tau_1, \tau_2, \ldots \tau_n\} \text{ and } J_a, J_b, J_c, \ldots$$

Solution can be found if they are sporadic tasks. (i.e., tasks with a priori known minimum interval times between their
successive arrivals.)

### 1.3. Goal

- **Minimize the response time of aperiodic tasks**.
    - Start processing them as soon as they arrive.
- Execute the periodic tasks **before their deadlines.**
    - Do not miss deadlines because the CPU serves aperiodic tasks.

### 1.4. The concept of Server




A **server** is a process that controls the execution of aperiodic tasks.

- It is a **periodic task itself** that, when active, handles the ready aperiodic tasks.
    - It has period $T_s$
    - It has capacity $C_s$ (budget, or computation time).

# 2. Fixed-Priority Server

---

### Assumption

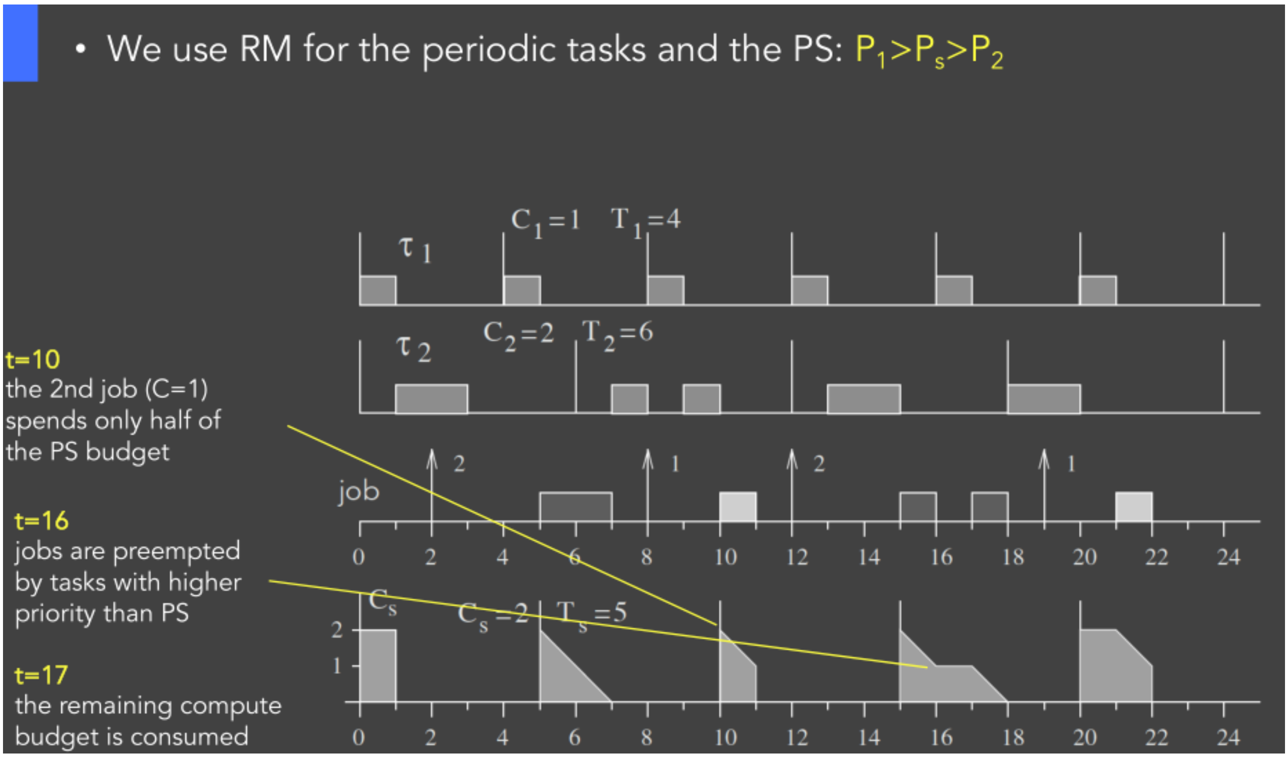
- The **periodic tasks arrive synchronously,** and are scheduled using RM (i.e., we assume $D_i = T_i$).
- Aperiodic tasks have **soft deadlines but unknown arrivals**.
- All tasks are **preemptable**.
- Priority ties are broken in favor of the job server.

## 2.1. Polling Server

### Rule

- It has period $T_s$ and computation budget $C_s$.
- At the beginning of each period, the budget is **initialized to** $C_s$.
- The budget is consumed as aperiodic tasks are executed; and the execution of jobs ends when $C_s$ becomes zero (consumed).
- **If there are no aperiodic tasks at some iteration;** $C_s$ **is set equal to zero and the token returns to periodic tasks.**

### Example




### Schedulability Analysis

In worst case, the PS **behaves as a periodic task** with $\{T_s, C_s\}$:

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

$$U_p = \sum_{i \in \Gamma} \frac{C_i}{T_i} \quad U_S = \frac{C_S}{T_S}$$

n+1 because n periodic tasks + the PS

**LL bound (neccessary)**

$$U_p + U_S \leq (n+1)\left(2^{\frac{1}{n+1}} - 1\right)$$

**Hyperbolic Bound (neccessary)**

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

**Response Time Calculate**

the response time of a periodic task i is the smallest integer satisfying the recurrence

$$R_i = C_i + \left\lceil \frac{R_i}{T_S} \right\rceil C_S + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

## Dimension a Polling Server

By using Hyperbolic bound

$$\prod_{i=1}^{n}(U_i+1) \leq \frac{2}{U_S+1} \Rightarrow U_S \leq \frac{2 - \Pi_{i=1}^{n}(U_i+1)}{\Pi_{i=1}^{n}(U_i+1)}$$

We can select any $C_s, T_s$ that satisfy this bound, e.g.

$$T_s = \min\{T_1, T_2, \ldots, T_n\}, \quad C_S = U_S^{\max} T_S$$

## Online Gurarntee for Aperiodic Tasks (Admission test)

**A Exact Test**

**Assume PS has the highest priority**


initial delay:

$$\Delta_\mathrm{a} = \left\lceil \frac{r_a}{T_s} \right\rceil T_s - r_a$$

full service periods

$$F_a = \left\lceil \frac{C_a}{C_s} \right\rceil - 1$$

final chunk

$$\delta_a = C_a - F_a C_s$$

Response Time

$$\mathrm{R_a} = \Delta_\mathrm{a} + \mathrm{F_a T_s} + \boldsymbol{\delta}$$

**Sufficient-only Test (PS does not have the highest priority)**

$$T_s + \left\lceil \frac{C_a}{C_s} \right\rceil T_s \leq D_a$$

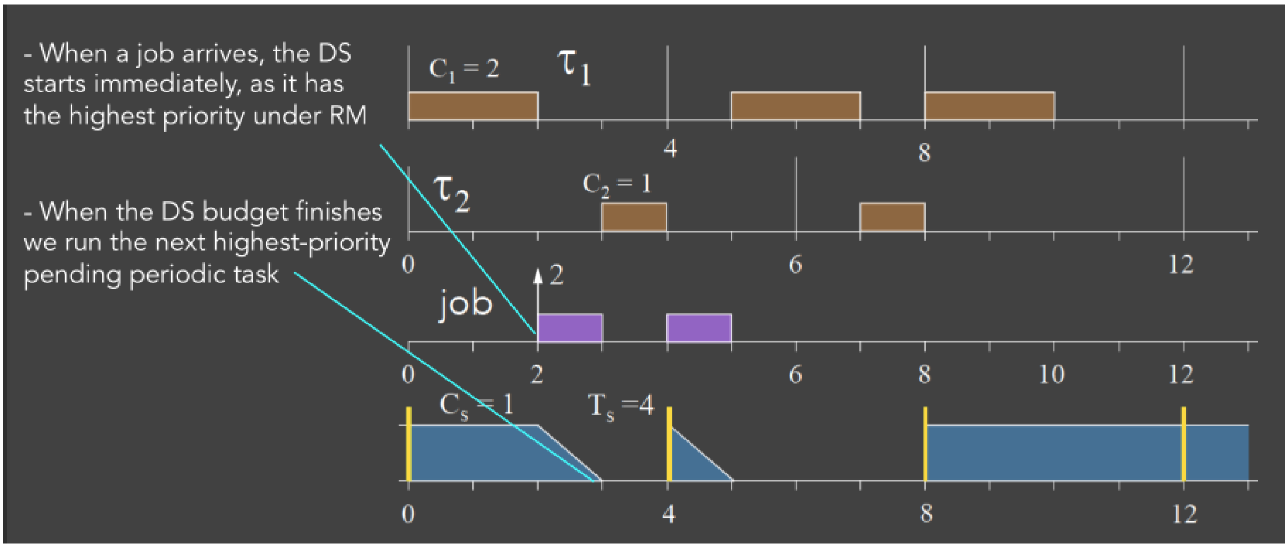
when $C_a < C_s$

$$2T_s \leq D_a$$

## 2.2. Deferrable Server

Same as PS, but the computation budget is **not discharged if there are no pending job requests.**

- This improves the responsiveness of aperiodic tasks (**no initial delay**).
- It benefits the response of jobs, but **induces task deadline misses**



## Schedulability Check

Due to this possibility of deferring the computation capacity, treating DS as a periodic task is not valid.

**LUB**

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

**Hyperbolic Bound**

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

## Dimensioning the DS

Using the Hyperbolic bound

$$\prod_{i=1}^{n} (U_i + 1) \leq \frac{U_S + 2}{2U_S + 1} \Rightarrow U_S \leq \frac{2 - \Pi_{i=1}^{n} (U_i + 1)}{2\Pi_{i=1}^{n} (U_i + 1) - 1}$$

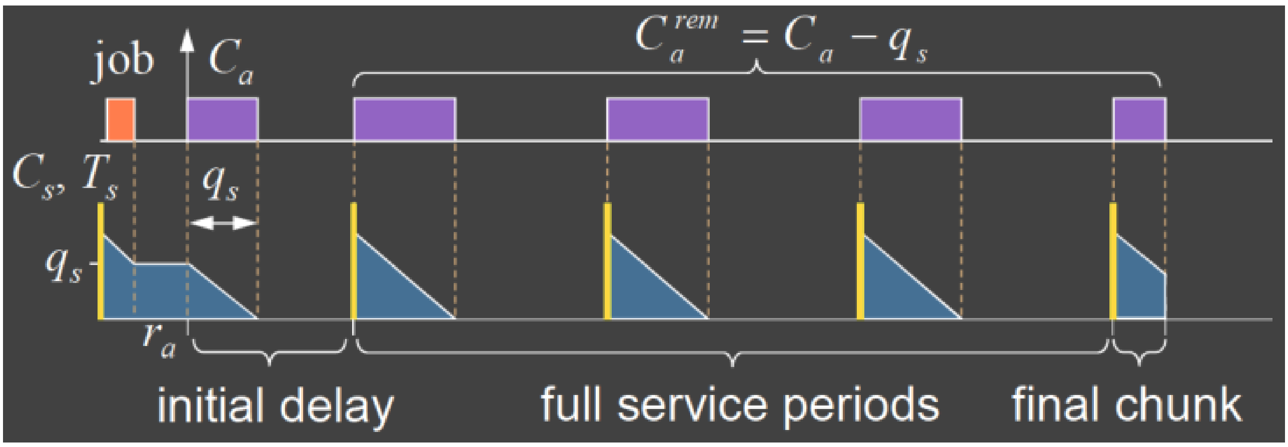
We can select any $C_s, T_s$ that satisfy this bound, e.g.,

$$T_s = \min \{T_1, T_2, \ldots, T_n\}, \quad C_S = U_S^{\max} T_S$$

## Online Guarantee for Aperiodic Tasks (Admission Tasks)

**Sufficient Test (DS has the highest priority)**

**Assume DS has the highest priority**



initial delay:

$$\Delta_\mathrm{a} = \left\lceil \frac{r_a}{T_s} \right\rceil T_s - r_a$$

full service periods

$$F_a = \left\lceil \frac{C_a^{rem}}{C_s} \right\rceil - 1$$

final chunk

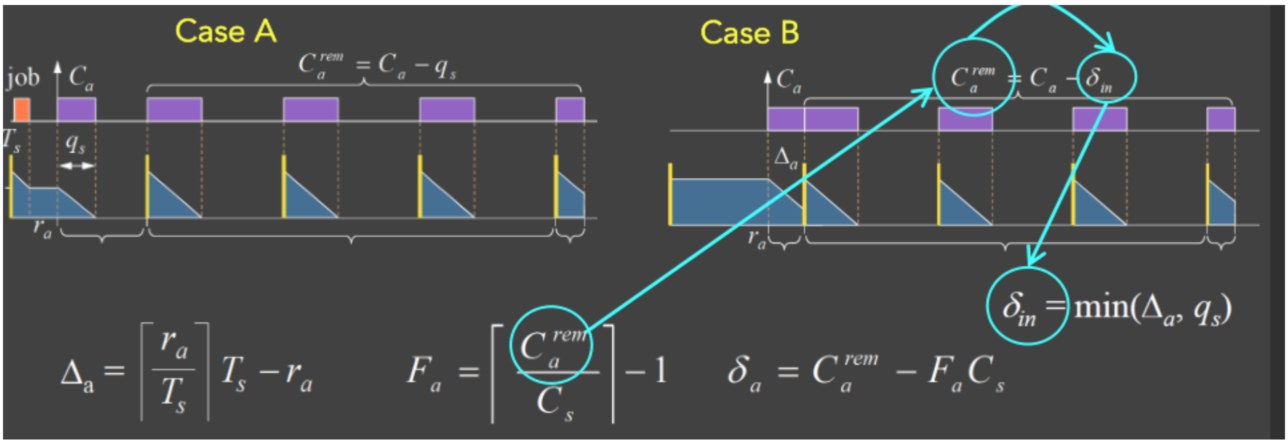
$$\delta_a = C_a^{rem} - F_a C_s$$

Response Time

$$\mathrm{R_a = \Delta_a + F_a T_s} + \boldsymbol{\delta}$$

Discussion of $C_a^{rem}$

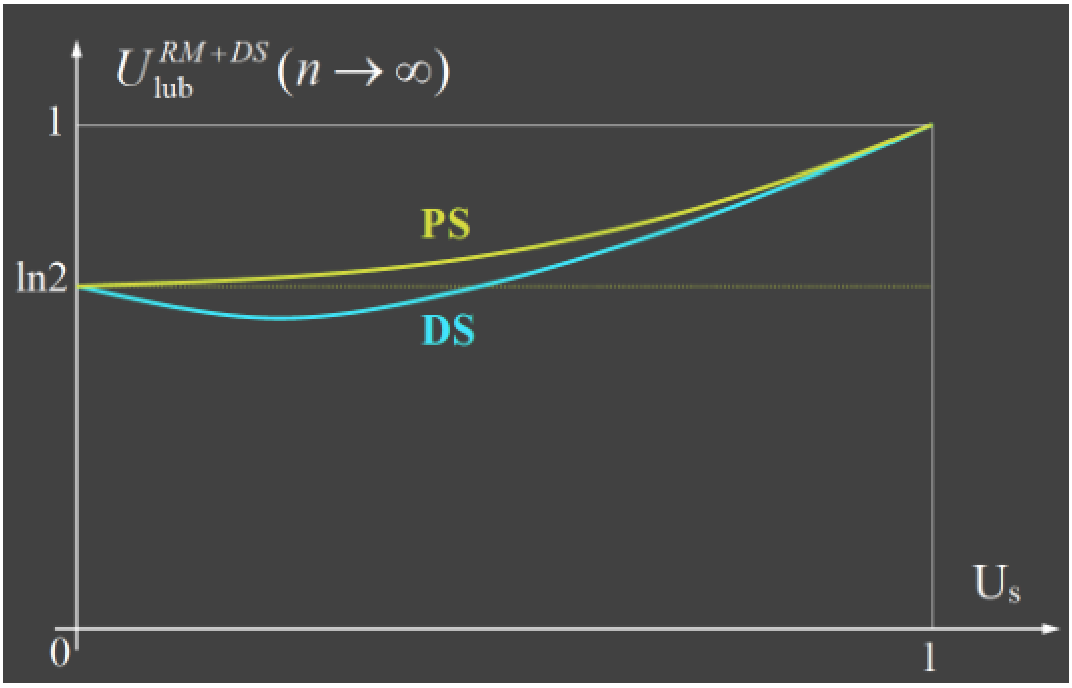
$$C_a^{rem} = C_a - \delta_{in}$$
$$\delta_{in} = \min(\Delta_a, q_s)$$




## PS VS DS

DS improves job responsiveness but shrinks the schedulability region

- For each task set, DS may has the lower $U_{ub}$, means the smaller region to high up utility for a given task set




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

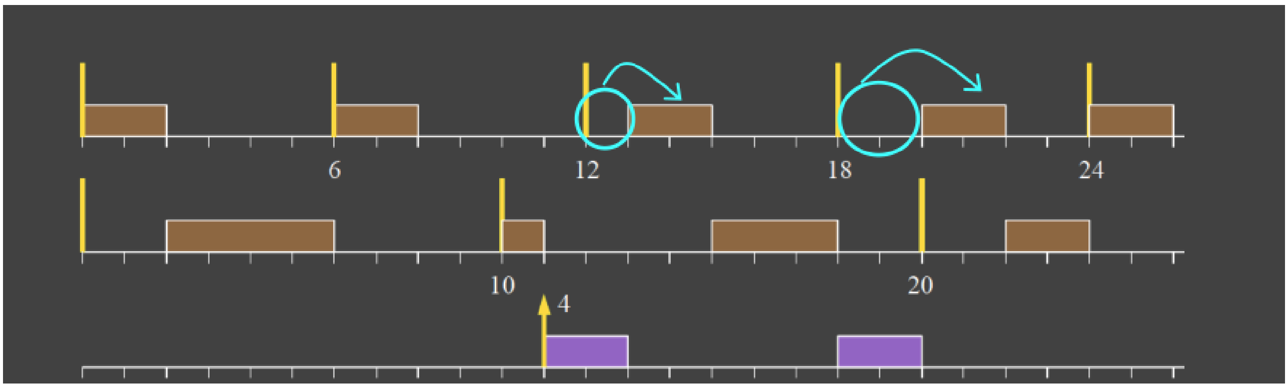
$$U_{lub}^{RM+DS}(n \to \infty) = U_S + \ln\left(\frac{U_S + 2}{2U_S + 1}\right)$$

## 2.3. Slack Stealer

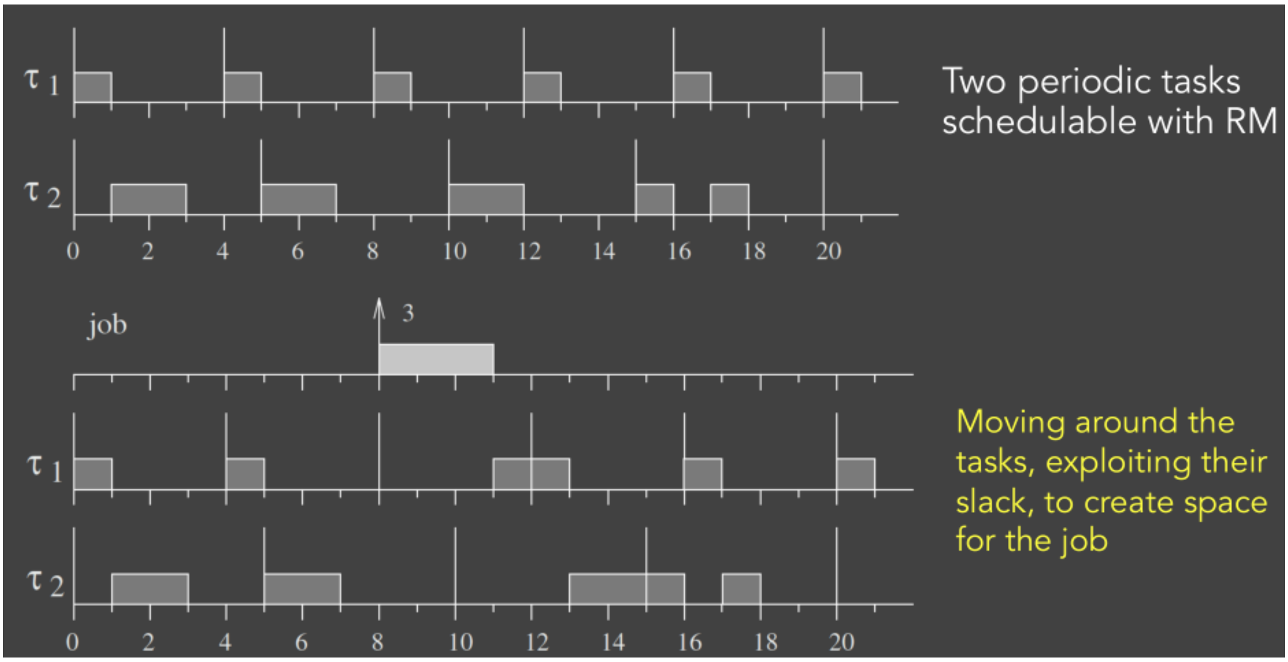
Not based on the concept of server.

A passive task (slack stealer) attempts to create a time budget by “stealing” time from the periodic tasks.

- there is no benefit when a periodic task finishes before its deadline (remember: $D_i = T_i$).




$$\text{Slack}_i(t) = d_i - t - c_i(t)$$




## Find the earlies possible slack

No involved

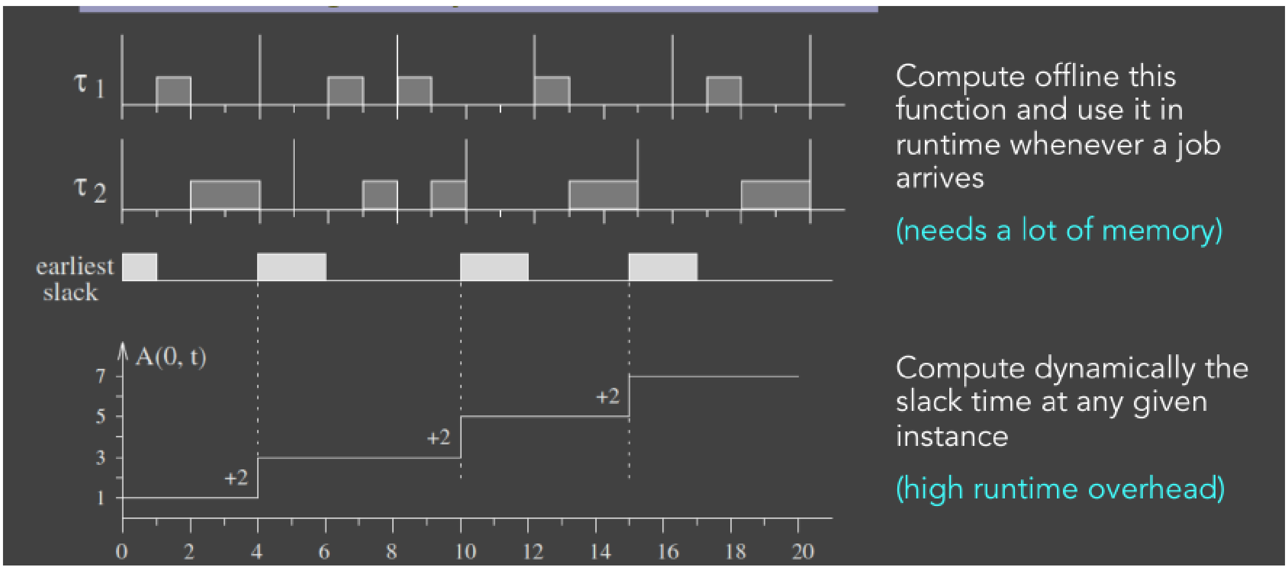
## Schedulability Analysis

**Model**

Assume there is a job $J_a$ arriving at $r_a$ with load $C_a$ units, What is the earliest time t that at least $C_a$ slack exists in $[ra, t]$

**Solution**

The answer is given by the slack function $A(s, t)$



Property

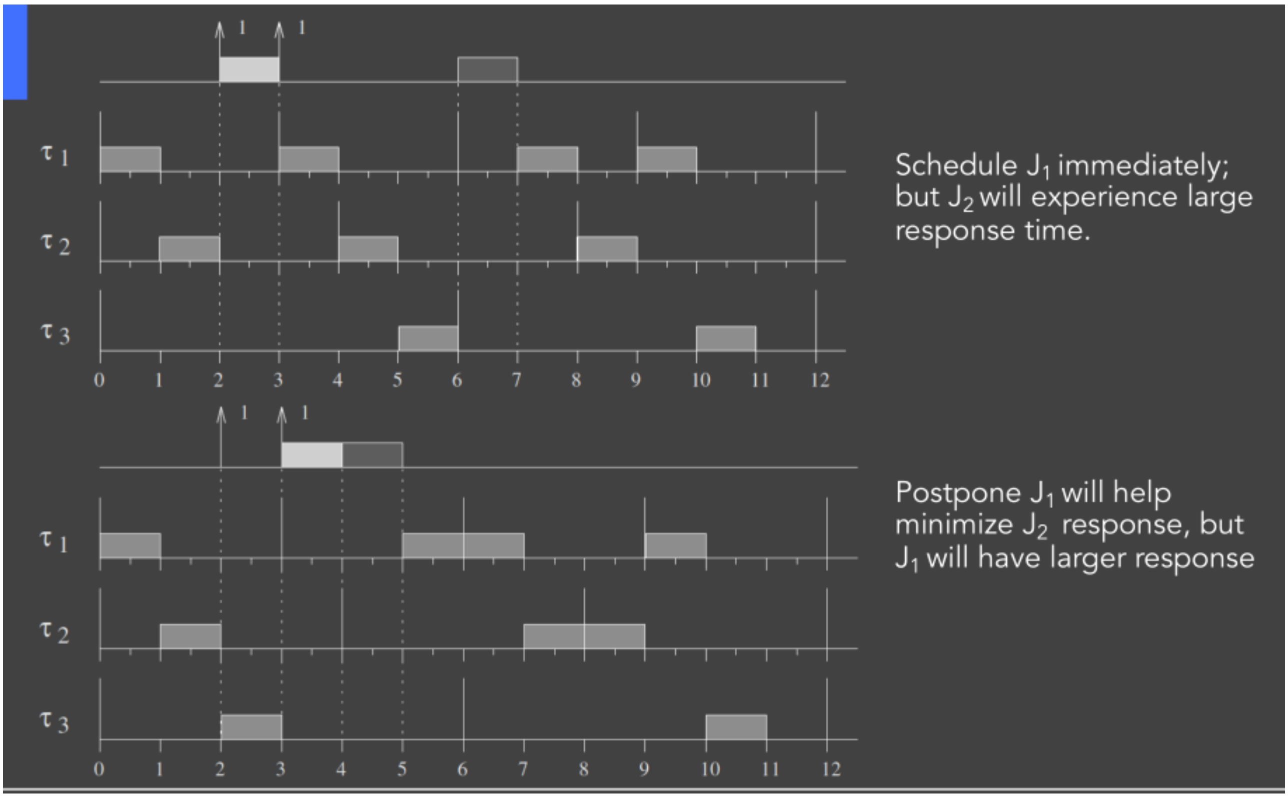
Slack Stealer is not Optimal

**Therorem 1**

For any set of **periodic tasks** ordered on some **fixed-priority scheme** and **aperiodic** requests ordered according to a **given queueing rule**, **no valid algorithm exists** that minimizes the response time of **every soft aperiodic job**.

**Theorem 2**

For any set of **periodic tasks** ordered on some **fixed-priority scheme** and **aperiodic** requests ordered according to a **given queueing rule**, **there does exist any online valid algorithm** that minimizes the **average response time of all soft aperiodic jobs.**




# 3. Dynamic Priority Servers

## 3.1. Total Bandwidth Server

Designed to be used with **EDF**.

Each aperiodic request is assigned a deadline.

- TBS load should not exceed a given utilization $U_s$ (or, **bandwidth**)
- Since we use EDF and $D_i = T_i$, the set $\{Tasks + Jobs\}$ is schedulable **iff** $U_p + U_s \leq 1$

Aperiodic jobs are inserted in the Ready queue.

- The Ready queue is served with EDF.

### Deadline Assignment

We know the required computation time $C_s$ ;

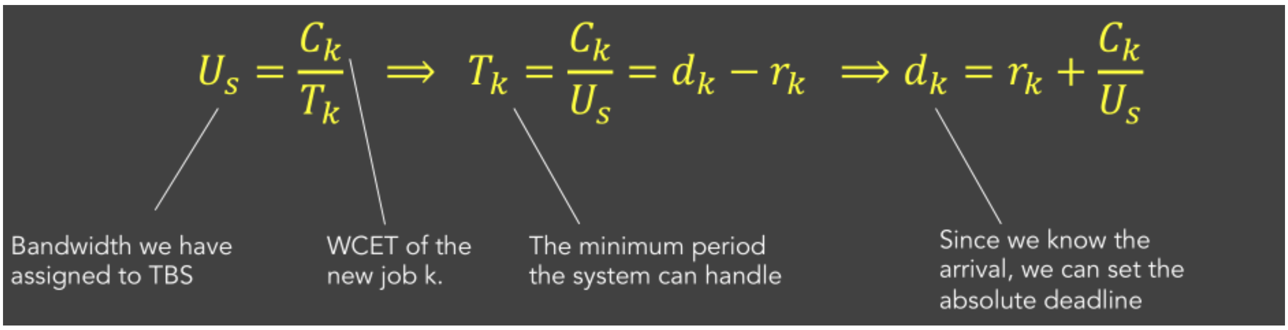
We know the maximum possible $U_s$ of TBS

$$U_s = \frac{C_k}{T_k} \Rightarrow T_k = \frac{C_k}{U_s} = d_k - r_k \Rightarrow d_k = r_k + \frac{C_k}{U_s}$$

For multiple task comes when the first is not finished:

$$d_2 = \max\{r_2, d_1\} + \frac{c_{J_2}}{U_S^{\max}}$$

**This ensures that at any time-interval the arrived jobs with deadlines in that interval do not impose utilization greater than $U_s$**



## Schedulability Analysis

Given a set of **n periodic tasks** with processor utilization $U_P$ and a **TBS** with processor utilization $U_s$, the whole set is schedulable by EDF **if and only if:**

$$U_p + U_s \leq 1$$

**Proof**

**Lemma**

In each time interval $[t_1, t_2]$, if $C_{job}$ is the total execution time requested by jobs arrived later than $t_1$ with deadlines sooner than $t_2$, then:

$$C_{job} = \sum_{\substack{k: r_k \geq t_1 \\ d_k \leq t_2}} C_k \leq (t_2 - t_1) U_s$$

$$C_{job} = \sum_{\substack{k: r_k \geq t_1, \\ d_k \leq t_2}} C_k \leq (t_2 - t_1) U_s \qquad d_k = \max\{r_k, d_{k-1}\} + \frac{C_k}{U_s}$$

$$\sum_{\substack{k: r_k \geq t_1, \\ d_k \leq t_2}} C_k = \sum_{k=k_a}^{k_b} C_k = \sum_{k=k_a}^{k_b} (d_k - \max\{r_k, d_{k-1}\}) U_s =$$

Assume 3 jobs in this interval so as to streamline our proof's exposition

$$= ( [d_3 - \max\{r_3, d_2\}] + [d_2 - \max\{r_2, d_1\}] + [d_1 - \max\{r_1, d_0\}] ) U_s$$

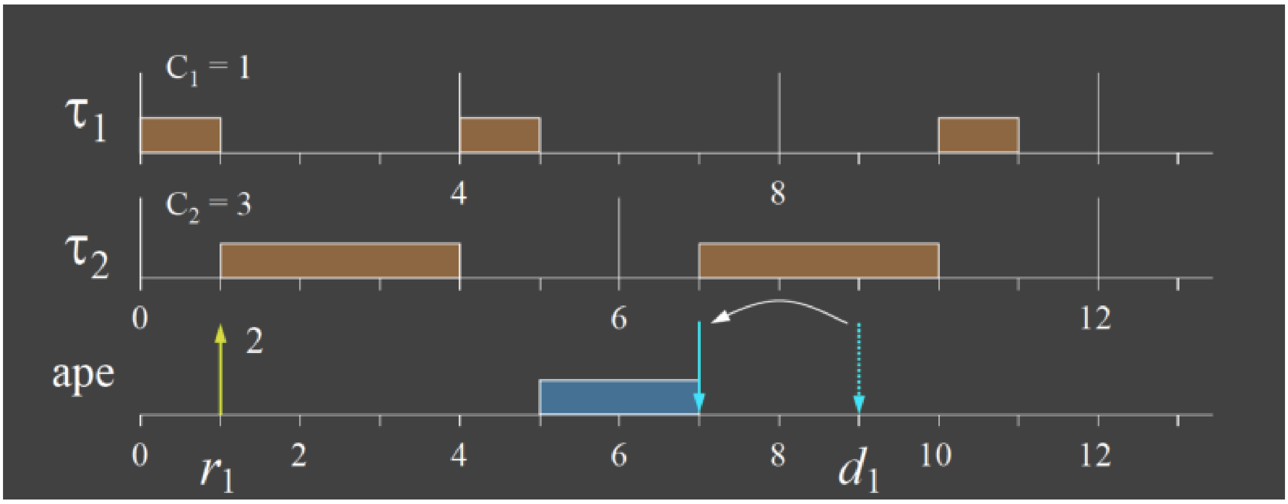
$$= ( d_3 + \underbrace{[d_2 - \max\{r_3, d_2\}]}_{\leq 0} + \underbrace{[d_1 - \max\{r_2, d_1\}]}_{\leq 0} - \max\{r_1, d_0\} ) U_s$$

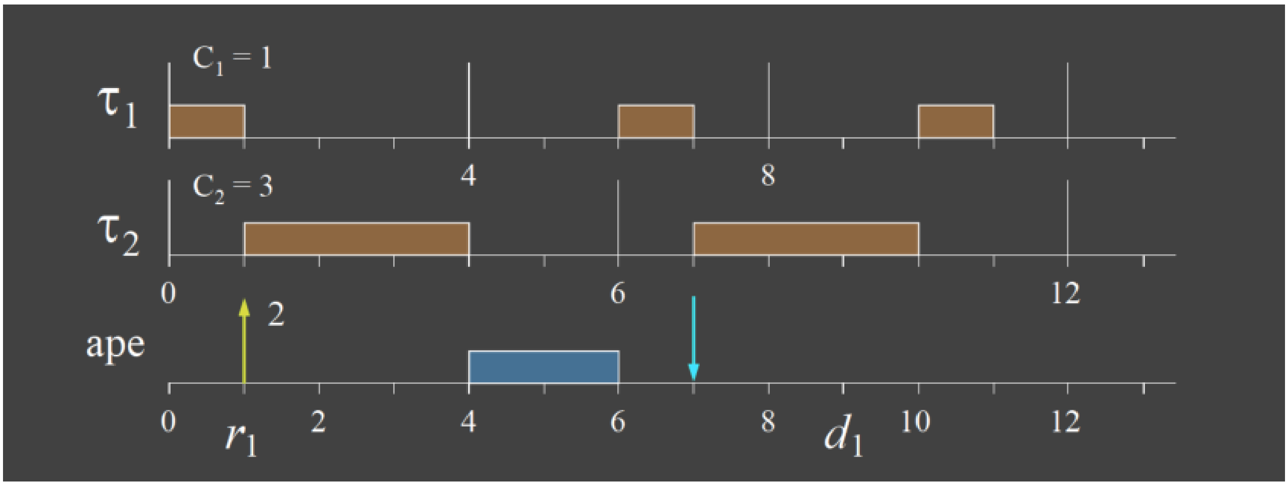
$$\leq ( d_3 - \max\{r_1, d_0\}) U_s \leq (t_2 - t_1) U_s$$

$t_2 \geq d_3 \qquad -t_1 \geq$

## Improving TBS

The deadline can be improved




$$d_k^0 = \max\left(r_k, d_{k-1}^0\right) + \frac{C_k}{U_s}$$

$$d_k^{s+1} = f_k^s = f_k\left(d_k^s\right)$$

$$f_k^s = r_k + C_k + I_p\left(r_k, d_k^s\right)$$

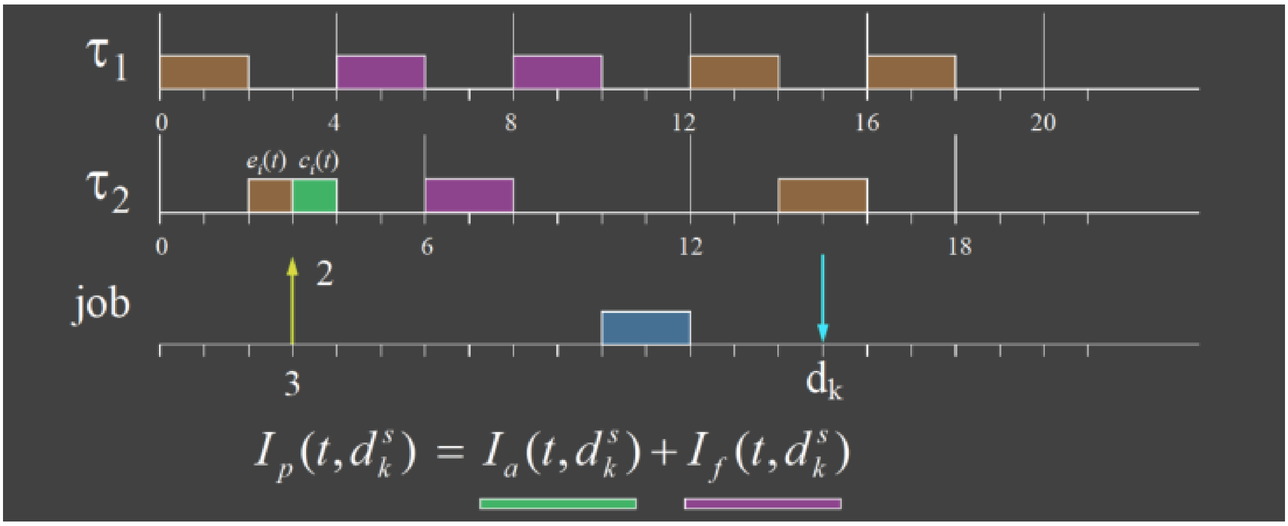
$$I_p\left(t, d_k^s\right) = I_a\left(t, d_k^s\right) + I_f\left(t, d_k^s\right)$$

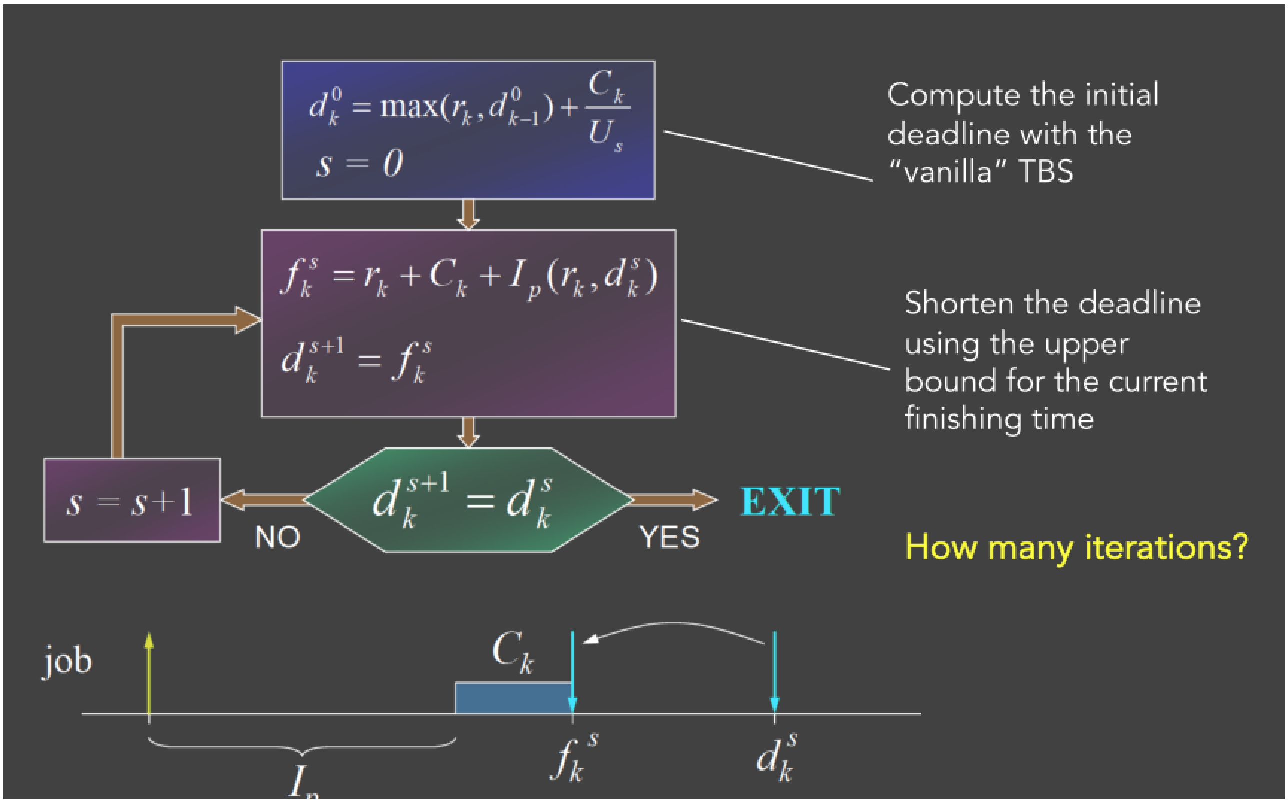
$$I_a\left(t, d_k^s\right) = \sum_{\substack{\tau_i \text{ active} \\ d_i < d_k^s}} c_i(t) = \sum_{\tau_i \text{ active}} \left[C_i - e_i(t)\right]$$

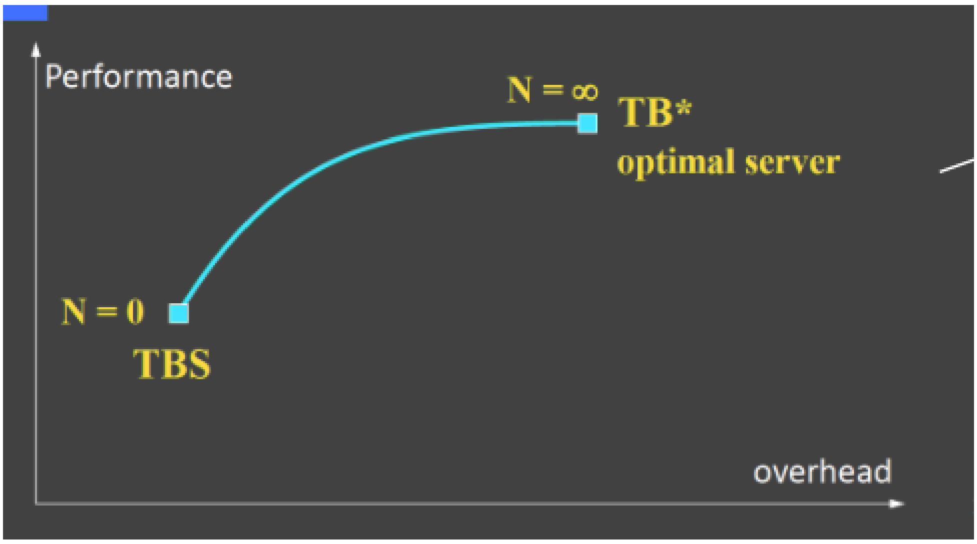
$$I_f\left(t, d_k^s\right) = \sum_{i=1}^{n} \left(\left\lceil \frac{d_k^s - n\,\mathrm{ext}_i(t)}{T_i} \right\rceil - 1\right) C_i$$

$$\mathrm{next}_i(t) = \left(\left\lfloor \frac{t}{T_i} \right\rfloor + 1\right) T_i$$

- $I_a$ is the Interference due to the **active periodic instances** with deadlines less than $d_k$
- $I_f$ Interference due to periodic tasks that will **start after t and need to end before** $d_k$




Improving TBS

- The more “shortening” iterations we do, the better is the response time.
- The more computing overheaad we need

## Drawbacks

**Job Overrun** may happen, we could use **Task Isolation** to improve (CBS method):

- Each task should not consume more time than its utilization, $Ui = \frac{C_i}{T_i}$ ;
- If a task overruns, its priority should be decreased, or its deadline postponed;

# 3.2. Constant Bandwidth Server

It assigns deadlines to tasks as TBS does; but keeps track of job execution through a **budget mechanism**

- When a job arrives, it is assigned a deadline and enters the **EDF queue.**
- If the job overruns (exceeds C), its **deadline is postponed**, and we reapply EDF.

## Parameters

**Fixed Parameters**

- Maximum Budget $\mathrm{Q_s}$
- Server Period $\mathrm{T_s}$
- Server Bandwidth $U_s = \mathrm{Q}_s/T_s$

**Changing Parameters**

- Current Budget $\mathrm{q_s}$ ( initialized to 0 )
- Server Deadline $\mathrm{d_s}$ ( initialized to 0 )
  Fixed Parameters

## Rule

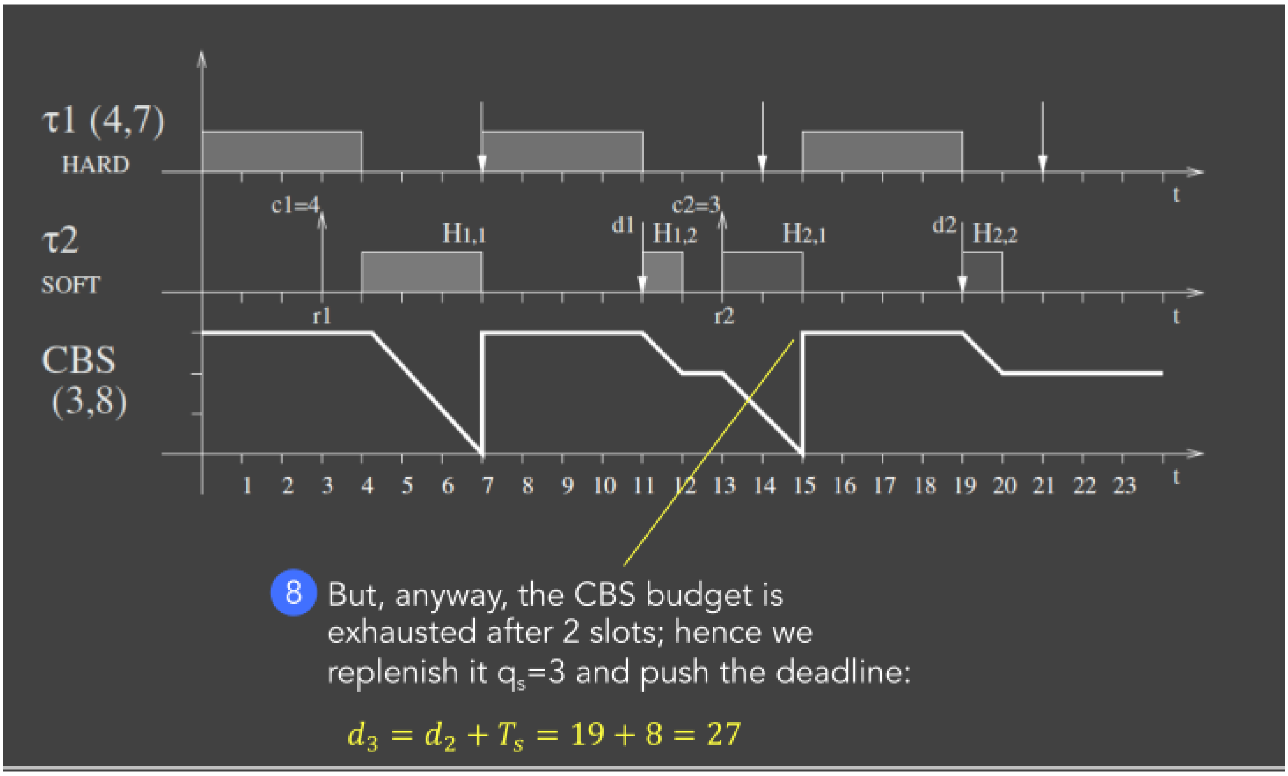
When a job $J_k$ **arrives** at time $r_k$ we assign it a proper deadline

**If** ($\exists$ a pending aperiodic job) **then** <enqueue $J_k$>
**else if** ($r_k + (q_s/U_s) > d_s$) **then** {
$q_s \leftarrow Q_s$
$d_s \leftarrow r_k + T_s$
} **else** {use the current $q_s$ and $d_s$}

When the budget is **exhausted**, we replenish it and push the deadline:

$$q_s \leftarrow Q_s$$
$$d_s \leftarrow d_s + T_s$$

## Examples




## Properties

The processor utilization of a CBS with parameters $(Q_s, T_s)$ is $U_s = Q_s/T_s$ **independently of the computation times and arrival pattern of the aperiodic jobs.**

## Schedulability Analysis

**Lemma 1**

Given a set $\Gamma$ of n periodic tasks with total utilization $U_P$ and a set of m CBSs with utilization $U_1, \ldots U_m$ , the whole set is schedulable with EDF **iff**:

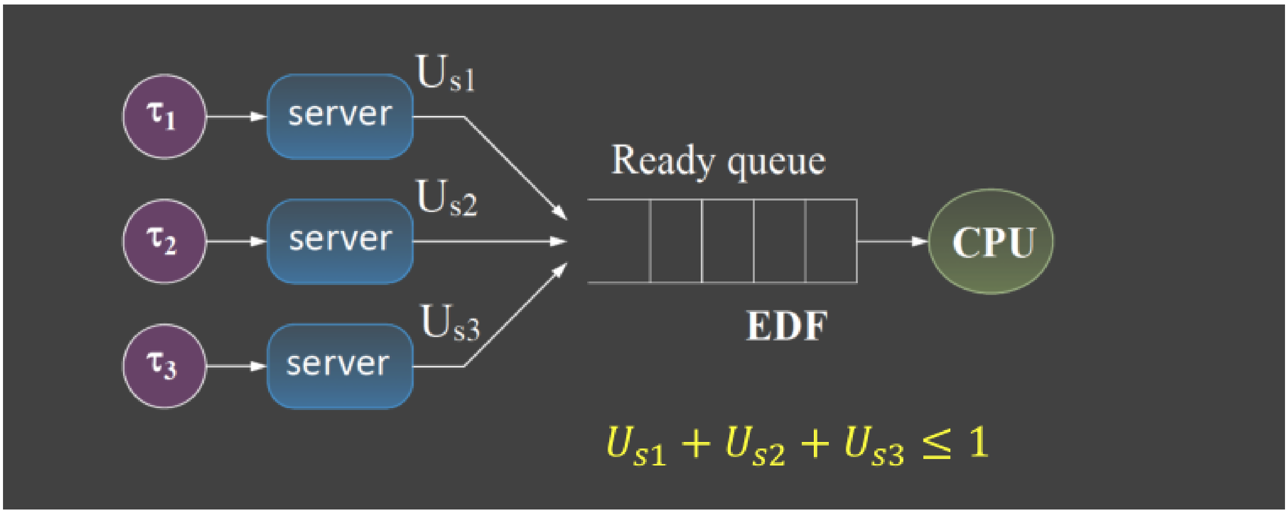
$$U_P + \sum_{i=1}^{m} U_i \leq 1$$

**Lemma 2**

A hard task i with parameters $(C_i, T_i)$ is schedulable by a CBS with parameters $Q_s$ larger than $C_i$ and $T_s = T_i$, iff

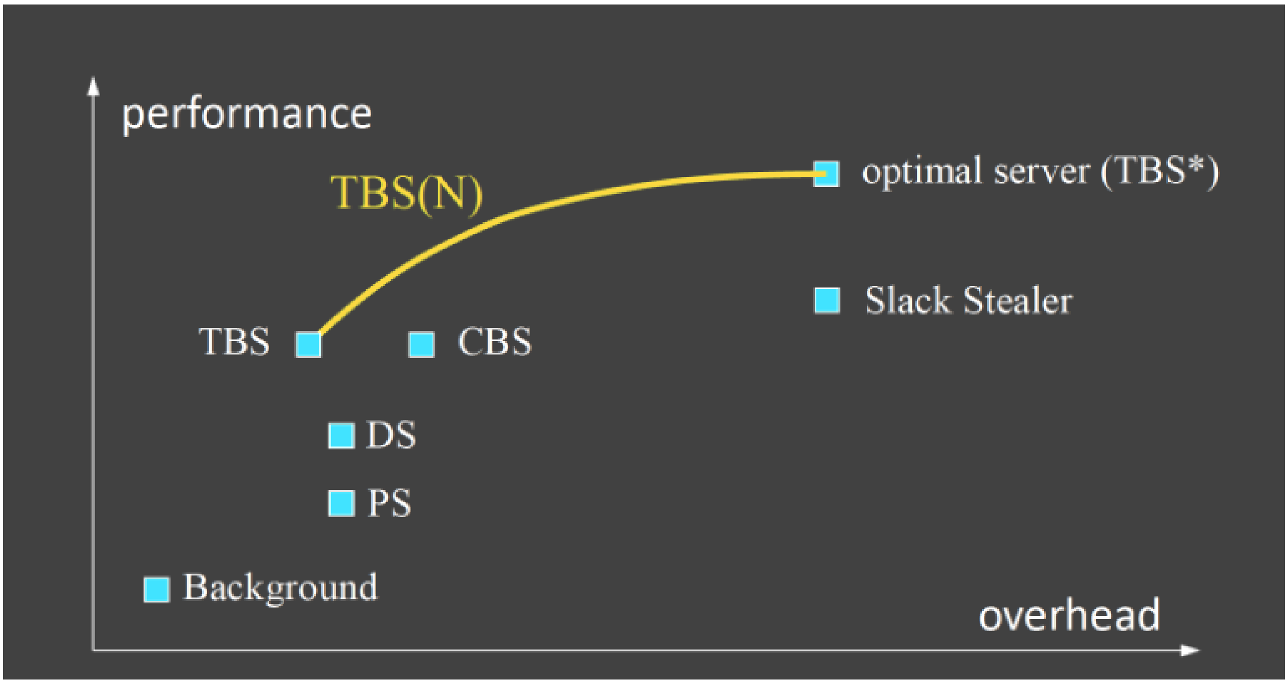
Task i is **schedulable with EDF**.

## Multiple CBSs




# 4. Comparison of Servers

---




Although It seems that TBS is pareto optimal than CBS, CBS has isolation property, which TBS does not have